DDT86

ddt86.sub	gentab.pli
ins86.plm	dis86.plm
dislnk86.asm	dis86.mp2
ass86.plm	asmtab.plm
asslnk86.asm	ass86.mp2
ddt86.a86	

```
;
;create ddt86.cmd, on micro using ASAS interface
;
pli gentab $$d
link gentab
gentab op86.dat
;
isl4
plm86 ins86.plm debug pagewidth(100) xref
plm86 dis86.plm debug pagewidth(100) xref
asm86 dislnk86.asm debug
link86 dislnk86.obj, dis86.obj, ins86.obj,plm86.lib to dis86.lnk
loc86 dis86.lnk to dis86.abs ad(sm(dats(0),code(0h))) od(sm(dats,code,const,data))
oh86 dis86.abs to dis86.h86
;
plm86 ass86.plm debug xref pagewidth(100)
plm86 asmtab.plm debug xref pagewidth(100)
asm86 asslnk86.asm debug
link86 asslnk86.obj, asmtab.obj, ass86.obj, po(dis86.abs), plm86.lib to ass86.lnk
loc86 ass86.lnk to ass86.abs ad(sm(dats(0),code(0h))) od(sm(dats,code,const,data))
oh86 ass86.abs to ass86.h86
;
cpm
asm86 ddt86 $$fi
pip ddt86.h86 = ddt86.h86, dis86.h86, ass86.h86
;  the value for the minimum code group required comes from the start of
;  the ??seg as listed in ass86.mp2
gencmd ddt86 8080 code[m366]
```

PL/I-80 V1.3 COMPILATION OF: GENTAB

D: Disk Print
L: List Source Program

NO ERROR(S) IN PASS 1

NO ERROR(S) IN PASS 2

PL/I-80 V1.3 COMPILATION OF: GENTAB

```
   1 a 0000 gentab: proc options (main);
   2 a 0006
   3 a 0006     /* generate tables for 8086 disassembler  12/23/80 */
   4 a 0006     /* modified  5/14/81 */
   5 a 0006
   6 c 0006     declare
   7 c 0006         opcnt (2:6) fixed (7) static initial (0,0,0,0,0),
   8 c 0006         sum fixed (7),
   9 c 0006         len fixed (7),
  10 c 0006         line char (100) varying,
  11 c 0006         infile file,
  12 c 0006         outfile file,
  13 c 0006         litfile file,
  14 c 0006         opcode char (10) varying,
  15 c 0006         i fixed (7),
  16 c 0006         j fixed (15),
  17 c 0006         n fixed (7),
  18 c 0006         count fixed (15),
  19 c 0006         chars (200) char (6) varying;
  20 c 0006
  21 c 0006     open file (infile) input stream title ('OP86.DAT');
  22 c 0022     open file (outfile) print title ('OPTAB.DAT');
  23 c 003E     open file (litfile) print title ('OPTAB.LIT');
  24 c 005A
  25 c 005A     on endpage (outfile) begin; end;
  26 c 0065     on endpage (litfile) begin; end;
  27 d 0071
  28 c 0071     count = 0;
  29 c 0078
  30 c 0078     /* read op86.dat file into chars array */
  31 c 0078
  32 c 0078     get file (infile) list (opcode);
  33 c 0095     do while (opcode ^= '$');
  34 c 00A3         count = count + 1;
  35 c 00AA         chars (count) = opcode;
  36 c 00C1         get file (infile) list (opcode);
  37 c 00E1     end;
  38 c 00E1
  39 c 00E1     /* create ascii opcode tables, 1 for each character length */
  40 c 00E1
  41 c 00E1     do i = 2 to 6;
  42 c 00F6         line = 'declare ops' || deblank (i) || ' (*) byte initial (';
  43 c 0118         n = 0;
  44 c 011C         do j = 1 to count;
```

```
 45 c 0135              if length (chars (j)) = i then
 46 c 0158                  do;
 47 c 0158                      if n > 0 then line = line || ', ';
 48 c 016F                      if divide (n, 5, 7) * 5 = n then
 49 c 0190                          do;
 50 c 0190                              put file (outfile) skip list (line);
 51 c 018A                              line = '^I';
 52 c 01C7                          end;
 53 c 01C7                      n = n + 1;
 54 c 01CB                      line = line || '''' || chars (j) || '''';
 55 c 0203                      opcnt (i) = opcnt (i) + 1;
 56 c 021B                  end;
 57 c 021B          end;
 58 c 021B          line = line || ');';
 59 c 022A          put file (outfile) skip list (line);
 60 c 0247          put file (outfile) skip;
 61 c 0262      end;
 62 c 0262
 63 c 0262      /* create array containing # of opcodes of each length */
 64 c 0262
 65 c 0262      line = 'declare nops (5) byte public initial (';
 66 c 026F      do i = 2 to 6;
 67 c 0284          line = line || deblank (opcnt (i));
 68 c 02A2          if i < 6 then line = line || ', ';
 69 c 02C8      end;
 70 c 02C8      put file (outfile) skip list (line || ');');
 71 c 02ED      put file (outfile) skip;
 72 c 0301
 73 c 0301      /* create array containing starting index for each opcode length */
 74 c 0301
 75 c 0301      line = 'declare opn$in (*) byte public initial (';
 76 c 030E      sum = 0;
 77 c 0312      do i = 2 to 6;
 78 c 0327          line = line || deblank (sum) || ', ';
 79 c 0347          sum = sum + opcnt (i);
 80 c 0363      end;
 81 c 0363      put file (outfile) skip list (line || '255);');
 82 c 0388
 83 c 0388      /* create literals for starting indexes for each opcode length */
 84 c 0388
 85 c 0388      sum = 0;
 86 c 038C      put file (litfile) skip list ('declare');
 87 c 03AB      do i = 2 to 6;
 88 c 03C0          put skip list (deblank (opcnt (i)), deblank (i) || '-character opcodes');
 89 c 03FF          line = '^I' || 'op' || deblank (i) ||
 90 c 0443            '$in literally ''' || deblank (sum) || '''';
 91 c 0443          if i = 6 then line = line || ';';
 92 c 045D          else line = line || ',';
 93 c 046C          put file (litfile) skip list (line);
 94 c 0489          sum = sum + opcnt (i);
 95 c 049E          opcnt (i) = 0;
 96 c 04B4      end;
 97 c 04B4
 98 c 04B4      /* create literals for position in opcode tables of each opcode */
 99 c 04B4
100 c 04B4      put file (litfile) skip;
101 c 04C8      put file (litfile) skip list ('declare');
102 c 04E7      do j = 1 to count;
```

```
 103 c 0500          len = length (chars (j));
 104 c 0516          if index (chars (j), ':') > 0 then
 105 c 0534              chars (j) = substr (chars (j), 1, len-1);
 106 c 0568          line = '^T' || chars (j) || '$in literally '''
 107 c 05E1            || 'op' || deblank (len) || '$in + '
 108 c 05E1            || deblank (opcnt (len)) || '''';
 109 c 05E1          if j = count then line = line || ';';
 110 c 0603          else line = line || ',';
 111 c 0612          put file (litfile) skip list (line);
 112 c 062F          opcnt (len) = opcnt (len) + 1;
 113 c 0647      end;
 114 c 0647
 115 c 0647      deblank: proc (i) returns (char (10) varying);
 116 e 0647          declare i fixed (7);
 117 e 0651          declare temp char (10) varying;
 118 e 0651          temp = char (i);
 119 e 0666          return (substr (temp, verify (temp, ' ')));
 120 c 0687          end deblank;
 121 c 0687
 122 a 0687      end gentab;

CODE SIZE = 068A
DATA AREA = 06DE
FREE SYMS = 1262
END  COMPILATION
```

```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE INSTR86
OBJECT MODULE PLACED IN INS86.OBJ
COMPILER INVOKED BY:  PLM86 INS86.PLM DEBUG PAGEWIDTH(100) XREF


            $title('instruction table for 8086 disassembler')
            $date(10/5/80)

  1         instr86: do;

  2   1     declare
                qq$in literally '0',
                alt1 literally '0',
                alt2 literally '1',
                alt3 literally '2',
                alt4 literally '3',
                alt5 literally '4',
                alt6 literally '5',
                alt7 literally '6',
                alt8 literally '7';

            $include(optab.lit)
        =
  3   1 =   declare
        =       op2$in literally '0',
        =       op3$in literally '12',
        =       op4$in literally '81',
        =       op5$in literally '106',
        =       op6$in literally '122';
        =
  4   1 =   declare
        =       AAA$in literally 'op3$in + 0',
        =       AAD$in literally 'op3$in + 1',
        =       AAM$in literally 'op3$in + 2',
        =       AAS$in literally 'op3$in + 3',
        =       ADC$in literally 'op3$in + 4',
        =       ADD$in literally 'op3$in + 5',
        =       AND$in literally 'op3$in + 6',
        =       CALL$in literally 'op4$in + 0',
        =       CALLF$in literally 'op5$in + 0',
        =       CBW$in literally 'op3$in + 7',
        =       CLC$in literally 'op3$in + 8',
        =       CLD$in literally 'op3$in + 9',
        =       CLI$in literally 'op3$in + 10',
        =       CMC$in literally 'op3$in + 11',
        =       CMP$in literally 'op3$in + 12',
        =       CMPSB$in literally 'op5$in + 1',
        =       CMPSW$in literally 'op5$in + 2',
        =       CS$in literally 'op3$in + 13',
        =       CWD$in literally 'op3$in + 14',
        =       DAA$in literally 'op3$in + 15',
        =       DAS$in literally 'op3$in + 16',
        =       DEC$in literally 'op3$in + 17',
        =       DIV$in literally 'op3$in + 18',
        =       DS$in literally 'op3$in + 19',
        =       ES$in literally 'op3$in + 20',
```

```
=       ESC$in literally 'op3$in + 21',
=       HLT$in literally 'op3$in + 22',
=       IDIV$in literally 'op4$in + 1',
=       IMUL$in literally 'op4$in + 2',
=       IN$in literally 'op2$in + 0',
=       INC$in literally 'op3$in + 23',
=       INT$in literally 'op3$in + 24',
=       INTO$in literally 'op4$in + 3',
=       IRET$in literally 'op4$in + 4',
=       JA$in literally 'op2$in + 1',
=       JAE$in literally 'op3$in + 25',
=       JB$in literally 'op2$in + 2',
=       JBE$in literally 'op3$in + 26',
=       JC$in literally 'op2$in + 3',
=       JCXZ$in literally 'op4$in + 5',
=       JE$in literally 'op2$in + 4',
=       JG$in literally 'op2$in + 5',
=       JGE$in literally 'op3$in + 27',
=       JL$in literally 'op2$in + 6',
=       JLE$in literally 'op3$in + 28',
=       JMP$in literally 'op3$in + 29',
=       JMPF$in literally 'op4$in + 6',
=       JMPS$in literally 'op4$in + 7',
=       JNA$in literally 'op3$in + 30',
=       JNAE$in literally 'op4$in + 8',
=       JNB$in literally 'op3$in + 31',
=       JNBE$in literally 'op4$in + 9',
=       JNC$in literally 'op3$in + 32',
=       JNE$in literally 'op3$in + 33',
=       JNG$in literally 'op3$in + 34',
=       JNGE$in literally 'op4$in + 10',
=       JNL$in literally 'op3$in + 35',
=       JNLE$in literally 'op4$in + 11',
=       JNO$in literally 'op3$in + 36',
=       JNP$in literally 'op3$in + 37',
=       JNS$in literally 'op3$in + 38',
=       JNZ$in literally 'op3$in + 39',
=       JO$in literally 'op2$in + 7',
=       JP$in literally 'op2$in + 8',
=       JPE$in literally 'op3$in + 40',
=       JPO$in literally 'op3$in + 41',
=       JS$in literally 'op2$in + 9',
=       JZ$in literally 'op2$in + 10',
=       LAHF$in literally 'op4$in + 12',
=       LDS$in literally 'op3$in + 42',
=       LEA$in literally 'op3$in + 43',
=       LES$in literally 'op3$in + 44',
=       LOCK$in literally 'op4$in + 13',
=       LODSB$in literally 'op5$in + 3',
=       LODSW$in literally 'op5$in + 4',
=       LOOP$in literally 'op4$in + 14',
=       LOOPE$in literally 'op5$in + 5',
=       LOOPNE$in literally 'op6$in + 0',
=       LOOPNZ$in literally 'op6$in + 1',
=       LOOPZ$in literally 'op5$in + 6',
=       MOV$in literally 'op3$in + 45',
=       MOVSB$in literally 'op5$in + 7',
```

```
=       MOVSW$in literally 'op5$in + 8',
=       MUL$in literally 'op3$in + 46',
=       NEG$in literally 'op3$in + 47',
=       NOP$in literally 'op3$in + 48',
=       NOT$in literally 'op3$in + 49',
=       OR$in literally 'op2$in + 11',
=       OUT$in literally 'op3$in + 50',
=       POP$in literally 'op3$in + 51',
=       POPF$in literally 'op4$in + 15',
=       PUSH$in literally 'op4$in + 16',
=       PUSHF$in literally 'op5$in + 9',
=       RCL$in literally 'op3$in + 52',
=       RCR$in literally 'op3$in + 53',
=       REP$in literally 'op3$in + 54',
=       REPE$in literally 'op4$in + 17',
=       REPNE$in literally 'op5$in + 10',
=       REPNZ$in literally 'op5$in + 11',
=       REPZ$in literally 'op4$in + 18',
=       RET$in literally 'op3$in + 55',
=       RETF$in literally 'op4$in + 19',
=       ROL$in literally 'op3$in + 56',
=       ROR$in literally 'op3$in + 57',
=       SAHF$in literally 'op4$in + 20',
=       SAL$in literally 'op3$in + 58',
=       SAR$in literally 'op3$in + 59',
=       SBB$in literally 'op3$in + 60',
=       SCASB$in literally 'op5$in + 12',
=       SCASW$in literally 'op5$in + 13',
=       SHL$in literally 'op3$in + 61',
=       SHR$in literally 'op3$in + 62',
=       SS$in literally 'op3$in + 63',
=       STC$in literally 'op3$in + 64',
=       STD$in literally 'op3$in + 65',
=       STI$in literally 'op3$in + 66',
=       STOSB$in literally 'op5$in + 14',
=       STOSW$in literally 'op5$in + 15',
=       SUB$in literally 'op3$in + 67',
=       TEST$in literally 'op4$in + 21',
=       WAIT$in literally 'op4$in + 22',
=       XCHG$in literally 'op4$in + 23',
=       XLAT$in literally 'op4$in + 24',
=       XOR$in literally 'op3$in + 68';

5   1   declare
            type$0 literally '0',
            type$1 literally '1',
            type$2 literally '2',
            type$3 literally '3',
            type$4 literally '4',
            type$5 literally '5',
            type$6 literally '6',
            type$7 literally '7',
            type$8 literally '7',
            type$9 literally '7',
            type$10 literally '8',
            type$11 literally '9',
            type$12 literally '10',
```

```
                type$13 literally '11',
                type$14 literally '12',
                type$15 literally '12',
                type$16 literally '12',
                type$17 literally '12',
                type$18 literally '13',
                type$19 literally '13',
                type$20 literally '13',
                type$21 literally '13',
                type$22 literally '14',
                type$23 literally '15',
                type$24 literally '15',
                type$25 literally '16',
                type$26 literally '16',
                type$27 literally '17',
                type$28 literally '17',
                type$29 literally '17',
                type$30 literally '17',
                type$31 literally '18',
                type$32 literally '18',
                type$33 literally '19',
                type$34 literally '20',
                type$35 literally '20',
                type$36 literally '21',
                type$37 literally '21',
                type$38 literally '22',
                type$39 literally '23',
                type$40 literally '24',
                type$41 literally '24',
                type$42 literally '25',
                type$43 literally '26',
                type$44 literally '27',
                type$45 literally '28';

  6   1     declare
                prefix$type literally '0ffh';

  7   1     declare alt$table$ptrs (8) address public data (
                .alt$1$tab,
                .alt$2$tab,
                .alt$3$tab,
                .alt$4$tab,
                .alt$5$tab,
                .alt$6$tab,
                .alt$7$tab,
                .alt$8$tab);

  8   1     declare alt$1$tab (*) byte data (
                add$in, type$44,
                or$in,  type$34,
                adc$in, type$44,
                sbb$in, type$44,
                and$in, type$34,
                sub$in, type$44,
                xor$in, type$34,
                cmp$in, type$44);
```

```
 9   1      declare alt$2$tab (*) byte data (
                add$in, type$44,
                qq$in,  type$0,
                adc$in, type$44,
                sbb$in, type$44,
                qq$in,  type$0,
                sub$in, type$44,
                qq$in,  type$0,
                cmp$in, type$44);

10   1      declare alt$3$tab (*) byte data (
                add$in, type$38,
                qq$in,  type$0,
                adc$in, type$38,
                sbb$in, type$38,
                qq$in,  type$0,
                sub$in, type$38,
                qq$in,  type$0,
                cmp$in, type$38);

11   1      declare alt$4$tab (*) byte data (
                rol$in, type$29,
                ror$in, type$29,
                rcl$in, type$29,
                rcr$in, type$29,
                shl$in, type$29,
                shr$in, type$29,
                qq$in,  type$0,
                sar$in, type$29);

12   1      declare alt$5$tab (*) byte data (
                rol$in, type$27,
                ror$in, type$27,
                rcl$in, type$27,
                rcr$in, type$27,
                shl$in, type$27,
                shr$in, type$27,
                qq$in,  type$0,
                sar$in, type$27);

13   1      declare alt$6$tab (*) byte data (
                test$in,    type$34,
                qq$in,  type$0,
                not$in, type$45,
                neg$in, type$45,
                mul$in, type$45,
                imul$in,    type$45,
                div$in, type$45,
                idiv$in,    type$45);

14   1      declare alt$7$tab (*) byte data (
                inc$in, type$45,
                dec$in, type$45,
                qq$in,  type$0,
                qq$in,  type$0,
                qq$in,  type$0,
                qq$in,  type$0,
```

```
            qq$in,  type$0,
            qq$in,  type$0);

15  1       declare alt$8$tab (*) byte data (
            inc$in, type$45,
            dec$in, type$45,
            call$in,    type$9,
            callf$in,   type$7,
            jmp$in, type$9,
            jmpf$in,    type$7,
            push$in,    type$7,
            qq$in,  type$0);

        /*
            instruction table for 8086 disassembler
            instruction is index into table
            there are 2 bytes per instruction:
                1.  index into ascii opcode table
                2.  instruction type (how many operands of what type, etc.)
        */

16  1       declare instr$table (512) byte public data (
            add$in, type$14,    /* 0 */
            add$in, type$15,
            add$in, type$16,
            add$in, type$17,
            add$in, type$23,
            add$in, type$24,
            push$in,    type$4,
            pop$in, type$4,
            or$in,  type$14,
            or$in,  type$15,
            or$in,  type$16,
            or$in,  type$17,
            or$in,  type$23,
            or$in,  type$24,
            push$in,    type$4,
            qq$in,  type$0,
            adc$in, type$14,    /* 10 */
            adc$in, type$15,
            adc$in, type$16,
            adc$in, type$17,
            adc$in, type$23,
            adc$in, type$24,
            push$in,    type$4,
            pop$in, type$4,
            sbb$in, type$14,
            sbb$in, type$15,
            sbb$in, type$16,
            sbb$in, type$17,
            sbb$in, type$23,
            sbb$in, type$24,
            push$in,    type$4,
            pop$in, type$4,
            and$in, type$14,    /* 20 */
            and$in, type$15,
            and$in, type$16,
```

```
        and$in, type$17,
        and$in, type$23,
        and$in, type$24,
        es$in,  prefix$type,
        daa$in, type$1,
        sub$in, type$14,
        sub$in, type$15,
        sub$in, type$16,
        sub$in, type$17,
        sub$in, type$23,
        sub$in, type$24,
        cs$in,  prefix$type,
        das$in, type$1,
        xor$in, type$14,    /* 30 */
        xor$in, type$15,
        xor$in, type$16,
        xor$in, type$17,
        xor$in, type$23,
        xor$in, type$24,
        ss$in,  prefix$type,
        aaa$in, type$1,
        cmp$in, type$14,
        cmp$in, type$15,
        cmp$in, type$16,
        cmp$in, type$17,
        cmp$in, type$23,
        cmp$in, type$24,
        ds$in,  prefix$type,
        aas$in, type$1,
        inc$in, type$3,     /* 40 */
        inc$in, type$3,
        inc$in, type$3,
        inc$in, type$3,
        inc$in, type$3,
        inc$in, type$3,
        inc$in, type$3,
        inc$in, type$3,
        dec$in, type$3,
        dec$in, type$3,
        dec$in, type$3,
        dec$in, type$3,
        dec$in, type$3,
        dec$in, type$3,
        dec$in, type$3,
        dec$in, type$3,
        push$in,    type$3,     /* 50 */
        push$in,    type$3,
        push$in,    type$3,
        push$in,    type$3,
        push$in,    type$3,
        push$in,    type$3,
        push$in,    type$3,
        push$in,    type$3,
        pop$in, type$3,
        pop$in, type$3,
        pop$in, type$3,
        pop$in, type$3,
```

```
pop$in, type$3,
pop$in, type$3,
pop$in, type$3,
pop$in, type$3,
qq$in,  type$0,     /* 60 */
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
qq$in,  type$0,
jo$in,  type$5,     /* 70 */
jno$in, type$5,
jb$in,  type$5,
jnb$in, type$5,
jz$in,  type$5,
jnz$in, type$5,
jbe$in, type$5,
ja$in,  type$5,
js$in,  type$5,
jns$in, type$5,
jp$in,  type$5,
jnp$in, type$5,
jl$in,  type$5,
jnl$in, type$5,
jle$in, type$5,
jg$in,  type$5,
alt1,   type$43,    /* 80 */
alt1,   type$43,
alt2,   type$43,
alt2,   type$43,
test$in,    type$14,
test$in,    type$15,
xchg$in,    type$16,
xchg$in,    type$17,
mov$in, type$14,
mov$in, type$15,
mov$in, type$16,
mov$in, type$17,
mov$in, type$36,
lea$in, type$39,
mov$in, type$37,
pop$in, type$9,
nop$in, type$1,     /* 90 */
xchg$in,    type$33,
xchg$in,    type$33,
xchg$in,    type$33,
xchg$in,    type$33,
```

```
xchg$in,    type$33,
xchg$in,    type$33,
xchg$in,    type$33,
cbw$in, type$1,
cwd$in, type$1,
callf$in,   type$13,
wait$in,    type$1,
pushf$in,   type$1,
popf$in,    type$1,
sahf$in,    type$1,
lahf$in,    type$1,
mov$in, type$18,    /* A0 */
mov$in, type$19,
mov$in, type$20,
mov$in, type$21,
movsb$in,   type$1,
movsw$in,   type$1,
cmpsb$in,   type$1,
cmpsw$in,   type$1,
test$in,    type$23,
test$in,    type$24,
stosb$in,   type$1,
stosw$in,   type$1,
lodsb$in,   type$1,
lodsw$in,   type$1,
scasb$in,   type$1,
scasw$in,   type$1,
mov$in, type$31,    /* B0 */
mov$in, type$31,
mov$in, type$31,
mov$in, type$31,
mov$in, type$31,
mov$in, type$31,
mov$in, type$31,
mov$in, type$31,
mov$in, type$32,
mov$in, type$32,
mov$in, type$32,
mov$in, type$32,
mov$in, type$32,
mov$in, type$32,
mov$in, type$32,
mov$in, type$32,
qq$in,  type$0,     /* C0 */
qq$in,  type$0,
ret$in, type$11,
ret$in, type$1,
les$in, type$39,
lds$in, type$39,
mov$in, type$40,
mov$in, type$41,
qq$in,  type$0,
qq$in,  type$0,
retf$in,    type$11,
retf$in,    type$1,
int$in, type$12,
int$in, type$10,
```

```
                into$in,    type$1,
                iret$in,    type$1,
                alt4,   type$43,    /* D0 */
                alt4,   type$43,
                alt5,   type$43,
                alt5,   type$43,
                aam$in, type$2,
                aad$in, type$2,
                qq$in,  type$0,
                xlat$in,    type$1,
                esc$in, type$42,
                esc$in, type$42,
                esc$in, type$42,
                esc$in, type$42,
                esc$in, type$42,
                esc$in, type$42,
                esc$in, type$42,
                esc$in, type$42,
                loopne$in,  type$5,     /* E0 */
                loope$in,   type$5,
                loop$in,    type$5,
                jcxz$in,    type$5,
                in$in,  type$22,
                in$in,  type$22,
                out$in, type$22,
                out$in, type$22,
                call$in,    type$6,
                jmp$in, type$6,
                jmpf$in,    type$13,
                jmps$in,    type$5,
                in$in,  type$25,
                in$in,  type$26,
                out$in, type$25,
                out$in, type$26,
                lock$in,    prefix$type,    /* F0 */
                qq$in,  type$0,
                repne$in,   prefix$type,
                rep$in, prefix$type,
                hlt$in, type$1,
                cmc$in, type$1,
                alt6,   type$43,
                alt6,   type$43,
                clc$in, type$1,
                stc$in, type$1,
                cli$in, type$1,
                sti$in, type$1,
                cld$in, type$1,
                std$in, type$1,
                alt7,   type$43,
                alt8,   type$43);

17  1       end instr86;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES					
4			AAAIN.	LITERALLY	16			
4			AADIN.	LITERALLY	16			
4			AAMIN.	LITERALLY	16			
4			AASIN.	LITERALLY	16			
4			ADCIN.	LITERALLY	8	9	10	16
4			ADDIN.	LITERALLY	8	9	10	16
2			ALT1	LITERALLY	16			
8	0010H	16	ALT1TAB.	BYTE ARRAY(16) DATA		7		
2			ALT2	LITERALLY	16			
9	0020H	16	ALT2TAB.	BYTE ARRAY(16) DATA		7		
2			ALT3	LITERALLY				
10	0030H	16	ALT3TAB.	BYTE ARRAY(16) DATA		7		
2			ALT4	LITERALLY	16			
11	0040H	16	ALT4TAB.	BYTE ARRAY(16) DATA		7		
2			ALT5	LITERALLY	16			
12	0050H	16	ALT5TAB.	BYTE ARRAY(16) DATA		7		
2			ALT6	LITERALLY	16			
13	0060H	16	ALT6TAB.	BYTE ARRAY(16) DATA		7		
2			ALT7	LITERALLY	16			
14	0070H	16	ALT7TAB.	BYTE ARRAY(16) DATA		7		
2			ALT8	LITERALLY	16			
15	0080H	16	ALT8TAB.	BYTE ARRAY(16) DATA		7		
7	0000H	16	ALTTABLEPTRS . . .	WORD ARRAY(8) PUBLIC DATA				
4			ANDIN.	LITERALLY	8	16		
4			CALLFIN.	LITERALLY	15	16		
4			CALLIN	LITERALLY	15	16		
4			CBWIN.	LITERALLY	16			
4			CLCIN.	LITERALLY	16			
4			CLDIN.	LITERALLY	16			
4			CLIIN.	LITERALLY	16			
4			CMCIN.	LITERALLY	16			
4			CMPIN.	LITERALLY	8	9	10	16
4			CMPSBIN.	LITERALLY	16			
4			CMPSWIN.	LITERALLY	16			
4			CSIN	LITERALLY	16			
4			CWDIN.	LITERALLY	16			
4			DAAIN.	LITERALLY	16			
4			DASIN.	LITERALLY	16			
4			DECIN.	LITERALLY	14	15	16	
4			DIVIN.	LITERALLY	13			
4			DSIN	LITERALLY	16			
4			ESCIN.	LITERALLY	16			
4			ESIN	LITERALLY	16			
4			HLTIN.	LITERALLY	16			
4			IDIVIN	LITERALLY	13			
4			IMULIN	LITERALLY	13			
4			INCIN.	LITERALLY	14	15	16	
4			ININ	LITERALLY	16			
1	0000H		INSTR86.	PROCEDURE STACK=0000H				

```
 16  0090H   512  INSTRTABLE . . . .   BYTE ARRAY(512) PUBLIC DATA
  4               INTIN. . . . . .    LITERALLY        16
  4               INTOIN . . . . . .  LITERALLY        16
  4               IRETIN . . . . . .  LITERALLY        16
  4               JAEIN. . . . . .    LITERALLY
  4               JAIN . . . . . .    LITERALLY        16
  4               JBEIN. . . . . .    LITERALLY        16
  4               JBIN . . . . . .    LITERALLY        16
  4               JCIN . . . . . .    LITERALLY
  4               JCXZIN . . . . . .  LITERALLY        16
  4               JEIN . . . . . .    LITERALLY
  4               JGEIN. . . . . .    LITERALLY
  4               JGIN . . . . . .    LITERALLY        16
  4               JLEIN. . . . . .    LITERALLY        16
  4               JLIN . . . . . .    LITERALLY        16
  4               JMPFIN . . . . . .  LITERALLY        15   16
  4               JMPIN. . . . . .    LITERALLY        15   16
  4               JMPSIN . . . . . .  LITERALLY        16
  4               JNAEIN . . . . . .  LITERALLY
  4               JNAIN. . . . . .    LITERALLY
  4               JNBEIN . . . . . .  LITERALLY
  4               JNBIN. . . . . .    LITERALLY        16
  4               JNCIN. . . . . .    LITERALLY
  4               JNEIN. . . . . .    LITERALLY
  4               JNGEIN . . . . . .  LITERALLY
  4               JNGIN. . . . . .    LITERALLY
  4               JNLEIN . . . . . .  LITERALLY
  4               JNLIN. . . . . .    LITERALLY        16
  4               JNOIN. . . . . .    LITERALLY        16
  4               JNPIN. . . . . .    LITERALLY        16
  4               JNSIN. . . . . .    LITERALLY        16
  4               JNZIN. . . . . .    LITERALLY        16
  4               JOIN . . . . . .    LITERALLY        16
  4               JPEIN. . . . . .    LITERALLY
  4               JPIN . . . . . .    LITERALLY        16
  4               JPOIN. . . . . .    LITERALLY
  4               JSIN . . . . . .    LITERALLY        16
  4               JZIN . . . . . .    LITERALLY        16
  4               LAHFIN . . . . . .  LITERALLY        16
  4               LDSIN. . . . . .    LITERALLY        16
  4               LEAIN. . . . . .    LITERALLY        16
  4               LESIN. . . . . .    LITERALLY        16
  4               LOCKIN . . . . . .  LITERALLY        16
  4               LODSBIN. . . . . .  LITERALLY        16
  4               LODSWIN. . . . . .  LITERALLY        16
  4               LOOPEIN. . . . . .  LITERALLY        16
  4               LOOPIN . . . . . .  LITERALLY        16
  4               LOOPNEIN . . . . .  LITERALLY        16
  4               LOOPNZIN . . . . .  LITERALLY
  4               LOOPZIN. . . . . .  LITERALLY
  4               MOVIN. . . . . .    LITERALLY        16
  4               MOVSBIN. . . . . .  LITERALLY        16
  4               MOVSWIN. . . . . .  LITERALLY        16
  4               MULIN. . . . . .    LITERALLY        13
  4               NEGIN. . . . . .    LITERALLY        13
  4               NOPIN. . . . . .    LITERALLY        16
  4               NOTIN. . . . . .    LITERALLY        13
```

```
3       OP2IN. . . . . LITERALLY        3   16
3       OP3IN. . . . . LITERALLY        8    9   10   11   12   13   14
                         15   16
3       OP4IN. . . . . LITERALLY       13   15   16
3       OP5IN. . . . . LITERALLY       15   16
3       OP6IN. . . . . LITERALLY       16
4       ORIN . . . . . LITERALLY        8   16
4       OUTIN. . . . . LITERALLY       16
4       POPFIN . . . . . LITERALLY     16
4       POPIN. . . . . LITERALLY       16
6       PREFIXTYPE . . . . LITERALLY   16
4       PUSHFIN. . . . . LITERALLY     16
4       PUSHIN . . . . . LITERALLY     15   16
2       QQIN . . . . . LITERALLY        9   10   11   12   13   14   15
                         16
4       RCLIN. . . . . LITERALLY       11   12
4       RCRIN. . . . . LITERALLY       11   12
4       REPEIN . . . . . LITERALLY
4       REPIN. . . . . LITERALLY       16
4       REPNEIN. . . . . LITERALLY     16
4       REPNZIN. . . . . LITERALLY
4       REPZIN . . . . . LITERALLY
4       RETFIN . . . . . LITERALLY     16
4       RETIN. . . . . LITERALLY       16
4       ROLIN. . . . . LITERALLY       11   12
4       RORIN. . . . . LITERALLY       11   12
4       SAHFIN . . . . . LITERALLY     16
4       SALIN. . . . . LITERALLY
4       SARIN. . . . . LITERALLY       11   12
4       SBBIN. . . . . LITERALLY        8    9   10   16
4       SCASBIN. . . . . LITERALLY     16
4       SCASWIN. . . . . LITERALLY     16
4       SHLIN. . . . . LITERALLY       11   12
4       SHRIN. . . . . LITERALLY       11   12
4       SSIN . . . . . LITERALLY       16
4       STCIN. . . . . LITERALLY       16
4       STDIN. . . . . LITERALLY       16
4       STIIN. . . . . LITERALLY       16
4       STOSBIN. . . . . LITERALLY     16
4       STOSWIN. . . . . LITERALLY     16
4       SUBIN. . . . . LITERALLY        8    9   10   16
4       TESTIN . . . . . LITERALLY     13   16
5       TYPE0. . . . . LITERALLY        9   10   11   12   13   14   15
                         16
5       TYPE1. . . . . LITERALLY       16
5       TYPE10 . . . . . LITERALLY     16
5       TYPE11 . . . . . LITERALLY     16
5       TYPE12 . . . . . LITERALLY     16
5       TYPE13 . . . . . LITERALLY     16
5       TYPE14 . . . . . LITERALLY     16
5       TYPE15 . . . . . LITERALLY     16
5       TYPE16 . . . . . LITERALLY     16
5       TYPE17 . . . . . LITERALLY     16
5       TYPE18 . . . . . LITERALLY     16
5       TYPE19 . . . . . LITERALLY     16
5       TYPE2. . . . . LITERALLY       16
5       TYPE20 . . . . . LITERALLY     16
```

```
      5          TYPE21 . . . . . LITERALLY        16
      5          TYPE22 . . . . . LITERALLY        16
      5          TYPE23 . . . . . LITERALLY        16
      5          TYPE24 . . . . . LITERALLY        16
      5          TYPE25 . . . . . LITERALLY        16
      5          TYPE26 . . . . . LITERALLY        16
      5          TYPE27 . . . . . LITERALLY        12
      5          TYPE28 . . . . . LITERALLY
      5          TYPE29 . . . . . LITERALLY        11
      5          TYPE3. . . . . . LITERALLY        16
      5          TYPE30 . . . . . LITERALLY
      5          TYPE31 . . . . . LITERALLY        16
      5          TYPE32 . . . . . LITERALLY        16
      5          TYPE33 . . . . . LITERALLY        16
      5          TYPE34 . . . . . LITERALLY         8    13
      5          TYPE35 . . . . . LITERALLY
      5          TYPE36 . . . . . LITERALLY        16
      5          TYPE37 . . . . . LITERALLY        16
      5          TYPE38 . . . . . LITERALLY        10
      5          TYPE39 . . . . . LITERALLY        16
      5          TYPE4. . . . . . LITERALLY        16
      5          TYPE40 . . . . . LITERALLY        16
      5          TYPE41 . . . . . LITERALLY        16
      5          TYPE42 . . . . . LITERALLY        16
      5          TYPE43 . . . . . LITERALLY        16
      5          TYPE44 . . . . . LITERALLY         8     9
      5          TYPE45 . . . . . LITERALLY        13    14    15
      5          TYPE5. . . . . . LITERALLY        16
      5          TYPE6. . . . . . LITERALLY        16
      5          TYPE7. . . . . . LITERALLY        15
      5          TYPE8. . . . . . LITERALLY
      5          TYPE9. . . . . . LITERALLY        15    16
      4          WAITIN . . . . . LITERALLY        16
      4          XCHGIN . . . . . LITERALLY        16
      4          XLATIN . . . . . LITERALLY        16
      4          XORIN. . . . . . LITERALLY         8    16
```


MODULE INFORMATION:

```
     CODE AREA SIZE     = 0000H      0D
     CONSTANT AREA SIZE = 0290H    656D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 0000H      0D
     559 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE DISEM86
OBJECT MODULE PLACED IN DIS86.OBJ
COMPILER INVOKED BY:  PLM86 DIS86.PLM DEBUG PAGEWIDTH(100) XREF
```



```
                $title('8086 disassembler')
                $date(5/14/81)
                $compact
                $optimize(2)

1               disem86: do;

2   1           declare
                    cr literally '0dh',
                    lf literally '0ah',
                    true literally '1',
                    false literally '0';

                $include(optab.dat)
      =
3   1   =       declare ops2 (*) byte initial (
      =             'IN', 'JA', 'JB', 'JC', 'JE',
      =             'JG', 'JL', 'JO', 'JP', 'JS',
      =             'JZ', 'OR');
      =
4   1   =       declare ops3 (*) byte initial (
      =             'AAA', 'AAD', 'AAM', 'AAS', 'ADC',
      =             'ADD', 'AND', 'CBW', 'CLC', 'CLD',
      =             'CLI', 'CMC', 'CMP', 'CS:', 'CWD',
      =             'DAA', 'DAS', 'DEC', 'DIV', 'DS:',
      =             'ES:', 'ESC', 'HLT', 'INC', 'INT',
      =             'JAE', 'JBE', 'JGE', 'JLE', 'JMP',
      =             'JNA', 'JNB', 'JNC', 'JNE', 'JNG',
      =             'JNL', 'JNO', 'JNP', 'JNS', 'JNZ',
      =             'JPE', 'JPO', 'LDS', 'LEA', 'LES',
      =             'MOV', 'MUL', 'NEG', 'NOP', 'NOT',
      =             'OUT', 'POP', 'RCL', 'RCR', 'REP',
      =             'RET', 'ROL', 'ROR', 'SAL', 'SAR',
      =             'SBB', 'SHL', 'SHR', 'SS:', 'STC',
      =             'STD', 'STI', 'SUB', 'XOR');
      =
5   1   =       declare ops4 (*) byte initial (
      =             'CALL', 'IDIV', 'IMUL', 'INTO', 'IRET',
      =             'JCXZ', 'JMPF', 'JMPS', 'JNAE', 'JNBE',
      =             'JNGE', 'JNLE', 'LAHF', 'LOCK', 'LOOP',
      =             'POPF', 'PUSH', 'REPE', 'REPZ', 'RETF',
      =             'SAHF', 'TEST', 'WAIT', 'XCHG', 'XLAT');
      =
6   1   =       declare ops5 (*) byte initial (
      =             'CALLF', 'CMPSB', 'CMPSW', 'LODSB', 'LODSW',
      =             'LOOPE', 'LOOPZ', 'MOVSB', 'MOVSW', 'PUSHF',
      =             'REPNE', 'REPNZ', 'SCASB', 'SCASW', 'STOSB',
      =             'STOSW');
      =
7   1   =       declare ops6 (*) byte initial (
      =             'LOOPNE', 'LOOPNZ');
```

```
      =
  8   1   =   declare nops (5) byte public initial (12, 69, 25, 16, 2);
      =
  9   1   =   declare opn$in (*) byte public initial (0, 12, 81, 106, 122, 255);

 10   1       declare
                 tab$ptrs (5) address public initial (.ops2, .ops3, .ops4, .ops5, .ops6);

 11   1       declare
                 left$bracket byte data ('[');
                 right$bracket byte data (']');

 12   1       declare
                 alt$table$base address,
                 alt$table based alt$table$base (16) byte,
                 alt$table$ptrs (8) address external;

 13   1       declare
                 mod$bits byte,
                 reg$bits byte,
                 rm$bits byte,
                 byte1$reg$bits byte,
                 d$bit byte,
                 s$bit byte,
                 v$bit byte,
                 w$bit byte,
                 z$bit byte;

 14   1       declare
                 mnemonic$index byte,    /* index into opcodes */
                 instr$type byte,
                 table$ptr address,
                 table$char based table$ptr byte,
                 disem$ptr pointer,
                 disem$offset address at (.disem$ptr),
                 disem$end address,
                 disem$byte based disem$ptr (1) byte,
                 disem$word based disem$ptr (1) address,
                 b$or$w$flag byte,
                 error$flag byte;

 15   1       declare instr$table (512) byte external;

 16   1       declare
                 ax$reg literally '0',
                 cx$reg literally '1',
                 dx$reg literally '2',
                 bx$reg literally '3',
                 sp$reg literally '4',
                 bp$reg literally '5',
                 si$reg literally '6',
                 di$reg literally '7';

 17   1       declare
                 al$reg literally '0',
                 cl$reg literally '1',
                 dl$reg literally '2',
```

```
            bl$reg literally '3',
            ah$reg literally '4',
            ch$reg literally '5',
            dh$reg literally '6',
            bh$reg literally '7';

18   1      declare
            es$reg literally '0',
            cs$reg literally '1',
            ss$reg literally '2',
            ds$reg literally '3';

19   1      declare
            reg16 (*) byte public initial ('AX', 'CX', 'DX', 'BX', 'SP', 'BP', 'SI', 'DI'),
            reg8 (*) byte public initial ('AL', 'CL', 'DL', 'BL', 'AH', 'CH', 'DH', 'BH'),
            segreg (*) byte public initial ('ES', 'CS', 'SS', 'DS');

20   1      conout: procedure (c) external;
21   2          declare c byte;
22   2          end conout;

23   1      comma: procedure;
24   2          call conout (',');
25   2          end comma;

26   1      printm: procedure (a) PUBLIC;
27   2          declare a address;
28   2          declare b based a byte;
29   2          do while b <> '$';
30   3              call conout (b);
31   3              a = a + 1;
32   3          end;
33   2          end printm;

34   1      print$nibble: procedure (b);
35   2          declare b byte;
36   2          if b > 9 then call conout (b - 10 + 'A');
38   2          else call conout (b + '0');
39   2          end print$nibble;

40   1      print$byte: procedure (b);
41   2          declare b byte;
42   2          call print$nibble (shr (b, 4));
43   2          call print$nibble (b and 0fh);
44   2          end print$byte;

45   1      print$word: procedure (a) public;
46   2          declare a address;
47   2          call print$byte (high (a));
48   2          call print$byte (low (a));
49   2          end print$word;

50   1      error: procedure;
51   2          call printm (.('??=    $'));
52   2          call print$byte (disem$byte (0));
53   2          disem$offset = disem$offset + 1;
54   2          end error;
```

```
 55   1      set$bits: procedure;
 56   2          byte1$reg$bits = disem$byte (0) and 7;
 57   2          mod$bits = shr (disem$byte (1), 6);
 58   2          reg$bits = shr (disem$byte (1), 3) and 7;
 59   2          rm$bits = disem$byte (1) and 7;
 60   2          w$bit, z$bit = disem$byte (0) and 1;
 61   2          d$bit, s$bit, v$bit = shr (disem$byte (0), 1) and 1;
 62   2          end set$bits;

 63   1      print$b$or$w: procedure;
 64   2          if w$bit then call printm (.('WORD $'));
 66   2          else call printm (.('BYTE $'));
 67   2          end print$b$or$w;

 68   1      print$reg: procedure (reg$add, reg);
 69   2          declare reg$add address, reg byte;
 70   2          table$ptr = reg$add + shl (reg, 1);
 71   2          call conout (table$char);
 72   2          table$ptr = table$ptr + 1;
 73   2          call conout (table$char);
 74   2          end print$reg;

 75   1      print$reg8: procedure (reg);
 76   2          declare reg byte;
 77   2          call print$reg (.reg8, reg);
 78   2          end print$reg8;

 79   1      print$reg16: procedure (reg);
 80   2          declare reg byte;
 81   2          call print$reg (.reg16, reg);
 82   2          end print$reg16;

 83   1      print$reg$8$or$16: procedure (reg$num);
 84   2          declare reg$num byte;
 85   2          if w$bit then call print$reg$16 (reg$num);
 87   2          else call print$reg$8 (reg$num);
 88   2          end print$reg$8$or$16;

 89   1      print$2$reg$16: procedure (r1, r2);
 90   2          declare (r1, r2) byte;
 91   2          call print$reg$16 (r1);
 92   2          call conout ('+');
 93   2          call print$reg$16 (r2);
 94   2          end print$2$reg$16;

 95   1      print$A$reg: procedure;
 96   2          if w$bit then call print$reg$16 (ax$reg);
 98   2          else call print$reg$8 (al$reg);
 99   2          end print$A$reg;

100   1      print$seg$reg: procedure (reg);
101   2          declare reg byte;
102   2          call print$reg (.seg$reg, reg);
103   2          end print$seg$reg;

104   1      print$data$8: procedure;
```

```
105   2           call print$byte (disem$byte (0));
106   2           disem$offset = disem$offset + 1;
107   2           end print$data$8;

108   1       print$data$16: procedure;
109   2           call print$word (disem$word (0));
110   2           disem$offset = disem$offset + 2;
111   2           end print$data$16;

112   1       print$data$8$or$16: procedure;
113   2           if w$bit then call print$data$16;
115   2           else call print$data$8;
116   2           end print$data$8$or$16;

117   1       print$data$sw: procedure;
118   2           if rol (disem$byte (0), 1) then call print$word (0ff00h or disem$byte (0));
120   2           else call print$word (disem$byte (0));
121   2           disem$offset = disem$offset + 1;
122   2           end print$data$sw;

123   1       print$signed$8: procedure;
124   2           declare a address;
125   2           a = disem$byte (0);
126   2           if low (a) >= 80h then a = a or 0ff00h;  /* sign extend to 16 bits */
128   2           call print$word (disem$offset + a + 1);
129   2           disem$offset = disem$offset + 1;
130   2           end print$signed$8;

131   1       print$signed$16: procedure;
132   2           call print$word (disem$offset + disem$word (0) + 2);
133   2           disem$offset = disem$offset + 2;
134   2           end print$signed$16;

135   1       print$direct$addr: procedure;
136   2           call conout (left$bracket);
137   2           call print$word (disem$word (0));
138   2           call conout (right$bracket);
139   2           disem$offset = disem$offset + 2;
140   2           end print$direct$addr;

141   1       print$mod$rm: procedure;
142   2           disem$offset = disem$offset + 1;     /* point past mod/reg/rm byte */
143   2           if mod$bits = 3 then
144   2               do;
145   3                   call print$reg$8$or$16 (rm$bits);
146   3                   return;
147   3               end;
148   2           if b$or$w$flag then call print$b$or$w;
150   2           if rm$bits = 6 and mod$bits = 0 then
151   2               do;
152   3                   call print$direct$addr;
153   3                   return;
154   3               end;
155   2           if mod$bits = 1 then
156   2               do;
157   3                   if (rm$bits <> 6) or (disem$byte (0) <> 0)
                            then call print$byte (disem$byte (0));
```

```
159   3                 disem$offset = disem$offset + 1;
160   3             end;
161   2         else if mod$bits = 2 then
162   2             do;
163   3                 call print$word (disem$word (0));
164   3                 disem$offset = disem$offset + 2;
165   3             end;
              call conout (left$bracket);
167   2         do case rm$bits;
168   3             call print$2$reg$16 (3, 6);
169   3             call print$2$reg$16 (3, 7);
170   3             call print$2$reg$16 (5, 6);
171   3             call print$2$reg$16 (5, 7);
172   3             call print$reg$16 (6);
173   3             call print$reg$16 (7);
174   3             call print$reg$16 (5);
175   3             call print$reg$16 (3);
176   3         end;
177   2         call conout (right$bracket);
178   2         end print$mod$rm;

179   1     print$mod$reg$rm: procedure;
180   2         if d$bit then
181   2             do;
182   3                 call print$reg$8$or$16 (reg$bits);
183   3                 call conout (',');
184   3                 call print$mod$rm;
185   3             end;
              else
186   2             do;
187   3                 call print$mod$rm;
188   3                 call conout (',');
189   3                 call print$reg$8$or$16 (reg$bits);
190   3             end;
191   2         end print$mod$reg$rm;

192   1     print$mnemonic: procedure;
193   2         declare (len, i) byte;
194   2         len = 2;
195   2         do while mnemonic$index >= opn$in (len - 1);
196   3             len = len + 1;
197   3         end;
198   2         table$ptr = tab$ptrs (len - 2) + (mnemonic$index - opn$in (len - 2))
                * len;
199   2         do i = 1 to 7;
200   3             if i <= len then
201   3                 do;
202   4                     call conout (table$char);
203   4                     table$ptr = table$ptr + 1;
204   4                 end;
205   3             else call conout (' ');
206   3         end;
207   2         disem$offset = disem$offset + 1;
208   2         end print$mnemonic;

209   1     type1: procedure;
210   2         call print$mnemonic;
```

```
211   2          end type1;

212   1      type2: procedure;
213   2          if disem$byte (1) = 0ah then
214   2              do;
215   3                  call print$mnemonic;
216   3                  disem$offset = disem$offset + 1;
217   3              end;
218   2          else error$flag = true;
219   2          end type2;

220   1      type3: procedure;
221   2          call print$mnemonic;
222   2          call print$reg$16 (byte1$reg$bits);
223   2          end type3;

224   1      type4: procedure;
225   2          declare temp byte;
226   2          temp = shr (disem$byte (0), 3) and 3;
227   2          call print$mnemonic;
228   2          call print$segreg (temp);
229   2          end type4;

230   1      type5: procedure;
231   2          call print$mnemonic;
232   2          call print$signed$8;
233   2          end type5;

234   1      type6: procedure;
235   2          call print$mnemonic;
236   2          call print$signed$16;
237   2          end type6;

238   1      type8: procedure;   /* 7, 9 */
239   2          call print$mnemonic;
240   2          call print$mod$rm;
241   2          end type8;

242   1      type10: procedure;
243   2          call print$mnemonic;
244   2          call print$data$8;
245   2          end type10;

246   1      type11: procedure;
247   2          call print$mnemonic;
248   2          call print$data$16;
249   2          end type11;

250   1      type12: procedure;
251   2          call print$mnemonic;
252   2          call conout ('J');
253   2          end type12;

254   1      type13: procedure;
255   2          declare temp address;
256   2          call print$mnemonic;
257   2          temp = disem$word (0);
```

```
258   2          disem$offset = disem$offset + 2;
259   2          call print$data$16;
260   2          call conout (':');
261   2          call print$word (temp);
262   2          end type13;

263   1      type14: procedure;    /* 15, 16, 17 */
264   2          call print$mnemonic;
265   2          call print$mod$reg$rm;
266   2          end type14;

267   1      type18: procedure;    /* 19, 20, 21 */
268   2          call print$mnemonic;
269   2          if d$bit then
270   2              do;
271   3                  call print$direct$addr;
272   3                  call comma;
273   3                  call print$A$reg;
274   3              end;
            else
275   2              do;
276   3                  call print$A$reg;
277   3                  call comma;
278   3                  call print$direct$addr;
279   3              end;
280   2          end type18;

281   1      type22: procedure;
282   2          call print$mnemonic;
283   2          if d$bit then
284   2              do;
285   3                  call print$data$8;
286   3                  call comma;
287   3                  call print$A$reg;
288   3              end;
            else
289   2              do;
290   3                  call print$A$reg;
291   3                  call comma;
292   3                  call print$data$8;
293   3              end;
294   2          end type22;

295   1      type23: procedure;    /* 24 */
296   2          call print$mnemonic;
297   2          call print$A$reg;
298   2          call comma;
299   2          call print$data$8$or$16;
300   2          end type23;

301   1      type25: procedure;   /* 26 */
302   2          call print$mnemonic;
303   2          if d$bit then
304   2              do;
305   3                  call print$reg$16 (dx$reg);
306   3                  call comma;
307   3                  call print$A$reg;
```

```
308   3               end;
                  else
309   2               do;
310   3                   call print$A$reg;
311   3                   call comma;
312   3                   call print$reg$16 (dx$reg);
313   3               end;
314   2           end type25;

315   1       type27: procedure;   /* 28, 29, 30 */
316   2           call print$mnemonic;
317   2           b$or$w$flag = true;
318   2           call print$mod$rm;
319   2           call comma;
320   2           if v$bit then call print$reg$8 (cl$reg);
322   2           else call conout ('1');
323   2           end type27;

324   1       type31: procedure;    /* 32 */
325   2           call setbits;
326   2           reg$bits = byte1$reg$bits;
327   2           w$bit = shr (disem$byte (0), 3) and 1;
328   2           call print$mnemonic;
329   2           call print$reg$8$or$16 (reg$bits);
330   2           call comma;
331   2           call print$data$8$or$16;
332   2           end type31;

333   1       type33: procedure;
334   2           call print$mnemonic;
335   2           call print$reg$16 (ax$reg);
336   2           call comma;
337   2           call print$reg$16 (byte1$reg$bits);
338   2           end type33;

339   1       type34: procedure;   /* 35 */
340   2           call print$mnemonic;
341   2           b$or$w$flag = true;
342   2           call print$mod$rm;
343   2           call comma;
344   2           call print$data$8$or$16;
345   2           end type34;

346   1       type36: procedure;   /* 37 */
347   2           w$bit = true;   /* force 16 bit reg, mem */
348   2           if reg$bits > 3 then
349   2               do;
350   3                   error$flag = true;
351   3                   return;
352   3               end;
353   2           call print$mnemonic;
354   2           if d$bit then
355   2               do;
356   3                   call print$seg$reg (reg$bits);
357   3                   call comma;
358   3                   call print$mod$rm;
359   3               end;
```

```
                    else
360   2                 do;
361   3                     call print$mod$rm;
362   3                     call comma;
363   3                     call print$seg$reg (reg$bits);
364   3                 end;
365   2             end type36;

366   1         type38: procedure;
367   2             call print$mnemonic;
368   2             call print$mod$rm;
369   2             call comma;
370   2             call print$data$8;
371   2             end type38;

372   1         type39: procedure;
373   2             if mod$bits = 3 then
374   2                 do;
375   3                     error$flag = true;
376   3                     return;
377   3                 end;
378   2             call print$mnemonic;
379   2             call print$reg$16 (reg$bits);
380   2             call comma;
381   2             call print$mod$rm;
382   2             end type39;

383   1         type40: procedure;   /* 41 */
384   2             if mod$bits = 3 then
385   2                 do;
386   3                     error$flag = true;
387   3                     return;
388   3                 end;
389   2             call print$mnemonic;
390   2             b$or$w$flag = true;
391   2             call print$mod$rm;
392   2             call comma;
393   2             call print$data$8$or$16;
394   2             end type40;

395   1         type42: procedure;
396   2             call print$mnemonic;
397   2             call print$byte (shl (byte1$reg$bits, 3) or reg$bits);
398   2             call comma;
399   2             call print$mod$rm;
400   2             end type42;

401   1         type44: procedure;
402   2             call print$mnemonic;
403   2             b$or$w$flag = true;
404   2             call print$modrm;
405   2             call comma;
406   2             if s$bit = 1 and w$bit = 1 then call print$data$sw;
408   2             else call print$data$8$or$16;
409   2             end type44;

410   1         type45: procedure;
```

```
411   2          b$or$w$flag = true;
412   2          call type8;
413   2          end type45;

414   1      dis: procedure;
415   2          error$flag, b$or$w$flag = false;
416   2          call set$bits;
417   2          if instr$type = 26 then
418   2              do;
419   3                  alt$table$base = alt$table$ptrs (mnemonic$index);
420   3                  mnemonic$index = alt$table (reg$bits * 2);
421   3                  instr$type = alt$table (reg$bits * 2 + 1);
422   3              end;
423   2          if instr$type > 28 then error$flag = true;
            else
425   2          do case instr$type;
426   3              error$flag = true;
427   3              call type1;
428   3              call type2;
429   3              call type3;
430   3              call type4;
431   3              call type5;
432   3              call type6;
433   3              call type8;
434   3              call type10;
435   3              call type11;
436   3              call type12;
437   3              call type13;
438   3              call type14;
439   3              call type18;
440   3              call type22;
441   3              call type23;
442   3              call type25;
443   3              call type27;
444   3              call type31;
445   3              call type33;
446   3              call type34;
447   3              call type36;
448   3              call type38;
449   3              call type39;
450   3              call type40;
451   3              call type42;
452   3              ;
453   3              call type44;
454   3              call type45;
455   3          end;
456   2          if error$flag then call error;
458   2          end dis;

459   1      disem: procedure (disloc) address public;
460   2          declare disloc pointer;
461   2          declare nprefix byte;
462   2          disem$ptr = disloc;
463   2          nprefix = 0;
464   2          do while true;
465   3              mnemonic$index = instr$table (disem$byte (0) * 2);
466   3              instr$type = instr$table (disem$byte (0) * 2 + 1);
```

```
467   3             if instr$type = 0ffh and nprefix < 3 then
468   3                 do;
469   4                     call print$mnemonic;
470   4                     nprefix = nprefix + 1;
471   4                 end;
                    else
472   3                 do;
473   4                     if instr$type = 0ffh then instr$type = 1;
475   4                     call dis;
476   4                     return disem$offset;
477   4                 end;
478   3         end;
479   2         end disem;

480   1     end disem86;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
124	0014H	2	A. . . . . WORD 125 126 127 128
45	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 46 47 48
26	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 27 28 29 30 31
17			AHREG. . . . . LITERALLY
17			ALREG. . . . . LITERALLY 98
12	0000H	16	ALTTABLE . . . . . BYTE BASED(ALTTABLEBASE) ARRAY(16) 420 421
12	000AH	2	ALTTABLEBASE . . . WORD 12 419 420 421
12	0000H	16	ALTTABLEPTRS . . . WORD ARRAY(8) EXTERNAL(0) 419
16			AXREG. . . . . LITERALLY 97 335
40	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 41 42 43
34	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 35 36 37 38
28	0000H	1	B. . . . . BYTE BASED(A) 29 30
17			BHREG. . . . . LITERALLY
17			BLREG. . . . . LITERALLY
14	01D5H	1	BORWFLAG . . . . . BYTE 148 317 341 390 403 411 415
16			BPREG. . . . . LITERALLY
16			BXREG. . . . . LITERALLY
13	01CDH	1	BYTE1REGBITS . . . BYTE 56 222 326 337 397
20	0000H	1	C. . . . . BYTE PARAMETER 21
17			CHREG. . . . . LITERALLY
17			CLREG. . . . . LITERALLY 321
23	0002H	11	COMMA. . . . . PROCEDURE STACK=0006H 272 277 286 291 298 306 311 319 330 336 343 357 362 369 380 392 398 405
20	0000H		CONOUT . . . . . PROCEDURE EXTERNAL(2) STACK=0000H 24 30 37 38 71 73 92 136 138 166 177 183 188 202 205 252 260 322
2			CR . . . . . LITERALLY
18			CSREG. . . . . LITERALLY
16			CXREG. . . . . LITERALLY
13	01CEH	1	DBIT . . . . . BYTE 61 180 269 283 303 354
17			DHREG. . . . . LITERALLY
16			DIREG. . . . . LITERALLY
414	0694H	295	DIS. . . . . PROCEDURE STACK=0028H 475
459	078BH	90	DISEM. . . . . PROCEDURE WORD PUBLIC STACK=0030H
1	0002H		DISEM86. . . . . PROCEDURE STACK=0000H
14	0000H	1	DISEMBYTE. . . . . BYTE BASED(DISEMPTR) ARRAY(1) 52 56 57 58 59 60 61 105 118 119 120 125 157 158 213 226 327 465 466
14	0012H	2	DISEMEND . . . . . WORD
14	000EH	2	DISEMOFFSET. . . . WORD AT(53 106 110 121 128 129 132 133 139 142 159 164 207 216 258 476
14	000EH	4	DISEMPTR . . . . . POINTER 14 52 56 57 58 59 60 61 105 109 118 119 120 125 132 137 157 158 163 213 226 257 327 462 465 466
14	0000H	2	DISEMWORD. . . . . WORD BASED(DISEMPTR) ARRAY(1) 109 132 137 163 257
459	0004H	4	DISLOC . . . . . POINTER PARAMETER AUTOMATIC 460 462

```
  17                    DLREG. . . . . .  LITERALLY
  18                    DSREG. . . . . .  LITERALLY
  16                    DXREG. . . . . .  LITERALLY       305  312
  50  0077H     26      ERROR. . . . . .  PROCEDURE STACK=0012H          457
  14  01D6H      1      ERRORFLAG. . . . .  BYTE        218  350  375  386  415  424  426  456
  18                    ESREG. . . . . .  LITERALLY
   2                    FALSE. . . . . .  LITERALLY       415
                        HIGH . . . . . .  BUILTIN          47
 193  0200H      1      I. . . . . .  BYTE        199  200
  15  0000H    512      INSTRTABLE . . . .  BYTE ARRAY(512) EXTERNAL(1)          465  466
  14  01D4H      1      INSTRTYPE. . . . .  BYTE        417  421  423  425  466  467  473  474
  11  0000H      1      LEFTBRACKET. . . .  BYTE DATA       136  166
 193  01FFH      1      LEN. . . . . .  BYTE        194  195  196  198  200
   2                    LF . . . . . .  LITERALLY
                        LOW. . . . . .  BUILTIN          48  126
  14  01D3H      1      MNEMONICINDEX. . .  BYTE        195  198  419  420  465
  13  01CAH      1      MODBITS. . . . . .  BYTE         57  143  150  155  161  373  384
   8  018FH      5      NOPS . . . . . .  BYTE ARRAY(5) PUBLIC INITIAL
 461  0202H      1      NPREFIX. . . . . .  BYTE        463  467  470
   9  01C4H      6      OPNIN. . . . . .  BYTE ARRAY(6) PUBLIC INITIAL          195  198
   3  0018H     24      OPS2 . . . . . .  BYTE ARRAY(24) INITIAL          10
   4  0030H    207      OPS3 . . . . . .  BYTE ARRAY(207) INITIAL          10
   5  00FFH    100      OPS4 . . . . . .  BYTE ARRAY(100) INITIAL          10
   6  0163H     80      OPS5 . . . . . .  BYTE ARRAY(80) INITIAL          10
   7  01B3H     12      OPS6 . . . . . .  BYTE ARRAY(12) INITIAL          10
  89  0159H     25      PRINT2REG16. . . .  PROCEDURE STACK=0018H          168  169  170  171
  95  0172H     26      PRINTAREG. . . . .  PROCEDURE STACK=0014H          273  276  287  290
                          297  307  310
  63  00DAH     24      PRINTBORW. . . . .  PROCEDURE STACK=000CH          149
  40  0045H     27      PRINTBYTE. . . . .  PROCEDURE STACK=000EH           47   48   52  105
                          158  397
 108  0180H     20      PRINTDATA16. . . .  PROCEDURE STACK=0018H          114  248  259
 104  019DH     19      PRINTDATA8 . . . .  PROCEDURE STACK=0012H          115  244  285  292
                          370
 112  01C4H     20      PRINTDATA8OR16 . .  PROCEDURE STACK=001CH          299  331  344  393
                          408
 117  01D8H     45      PRINTDATASW. . . .  PROCEDURE STACK=0018H          407
 135  024BH     34      PRINTDIRECTADOR. .  PROCEDURE STACK=0018H          152  271  278
  26  000DH     25      PRINTM . . . . . .  PROCEDURE PUBLIC STACK=0008H           51   65   66
 192  0368H    116      PRINTMNEMONIC. . .  PROCEDURE STACK=0006H          210  215  221  227
                          231  235  239  243  247  251  256  264  268  282
                          296  302  316  328  334  340  353  367  378  389
                          396  402  469
 179  033AH     46      PRINTMODREGRM. . .  PROCEDURE STACK=0020H          265
 141  026DH    205      PRINTMODRM . . . .  PROCEDURE STACK=001CH          184  187  240  318
                          342  358  361  368  381  391  399  404
  34  0026H     31      PRINTNIBBLE. . . .  PROCEDURE STACK=0008H           42   43
  68  00F2H     41      PRINTREG . . . . .  PROCEDURE STACK=000AH           77   81  102
  79  012CH     17      PRINTREG16 . . . .  PROCEDURE STACK=0010H           86   91   93   97
                          172  173  174  175  222  305  312  335  337  379
  75  0118H     17      PRINTREG8. . . . .  PROCEDURE STACK=0010H           87   98  321
  83  013DH     28      PRINTREG8OR16. . .  PROCEDURE STACK=0016H          145  182  189  329
 100  018CH     17      PRINTSEGREG. . . .  PROCEDURE STACK=0010H          228  356  363
 131  0230H     27      PRINTSIGNED16. . .  PROCEDURE STACK=0018H          236
 123  0205H     43      PRINTSIGNED8 . . .  PROCEDURE STACK=0018H          232
  45  0060H     23      PRINTWORD. . . . .  PROCEDURE PUBLIC STACK=0014H          109  119  120
                          128  132  137  163  261
```

```
  89  0006H     1  R1 . . . . . .   BYTE PARAMETER AUTOMATIC          90   91
  89  0004H     1  R2 . . . . . .   BYTE PARAMETER AUTOMATIC          90   93
 100  0004H     1  REG. . . . . .   BYTE PARAMETER AUTOMATIC         101  102
  79  0004H     1  REG. . . . . .   BYTE PARAMETER AUTOMATIC          80   81
  75  0004H     1  REG. . . . . .   BYTE PARAMETER AUTOMATIC          76   77
  68  0004H     1  REG. . . . . .   BYTE PARAMETER AUTOMATIC          69   70
  19  01D7H    16  REG16. . . . . .  BYTE ARRAY(16) PUBLIC INITIAL          81
  19  01E7H    16  REG8 . . . . . .  BYTE ARRAY(16) PUBLIC INITIAL          77
  68  0006H     2  REGADD . . . . .  WORD PARAMETER AUTOMATIC          69   70
  13  01CBH     1  REGBITS. . . . .  BYTE          58  182  189  326  329  348  356  363
                                      379  397  420  421
  83  0004H     1  REGNUM . . . . .  BYTE PARAMETER AUTOMATIC          84   86   87
  11  0001H     1  RIGHTBRACKET . .  BYTE DATA          138  177
  13  01CCH     1  RMBITS . . . . .  BYTE          59  145  150  157  167
                   ROL. . . . . .   BUILTIN         118
  13  01CFH     1  SBIT . . . . . .  BYTE          61  406
  19  01F7H     8  SEGREG . . . . .  BYTE ARRAY(8) PUBLIC INITIAL          102
  55  0091H    73  SETBITS. . . . .  PROCEDURE STACK=0006H          325  416
                   SHL. . . . . .   BUILTIN          70  397
                   SHR. . . . . .   BUILTIN          42   57   58   61  226  327
  16               SIREG. . . . . .  LITERALLY
  16               SPREG. . . . . .  LITERALLY
  18               SSREG. . . . . .  LITERALLY
  14  0000H     1  TABLECHAR. . . .  BYTE BASED(TABLEPTR)          71   73  202
  14  000CH     2  TABLEPTR . . . .  WORD          14   70   71   72   73  198  202  203

  10  0000H    10  TABPTRS. . . . .  WORD ARRAY(5) PUBLIC INITIAL          198
 255  0016H     2  TEMP . . . . . .  WORD        257  261
 225  0201H     1  TEMP . . . . . .  BYTE        226  228
   2               TRUE . . . . . .  LITERALLY         218  317  341  347  350  375  386
                                      390  403  411  424  426  464
 209  03DCH     8  TYPE1. . . . . .  PROCEDURE STACK=000AH          427
 242  0451H    11  TYPE10 . . . . .  PROCEDURE STACK=0016H          434
 246  045CH    11  TYPE11 . . . . .  PROCEDURE STACK=001CH          435
 250  0467H    14  TYPE12 . . . . .  PROCEDURE STACK=000AH          436
 254  0475H    39  TYPE13 . . . . .  PROCEDURE STACK=001CH          437
 263  049CH    11  TYPE14 . . . . .  PROCEDURE STACK=0024H          438
 267  04A7H    35  TYPE18 . . . . .  PROCEDURE STACK=001CH          439
 212  03E4H    30  TYPE2. . . . . .  PROCEDURE STACK=000AH          428
 281  04CAH    35  TYPE22 . . . . .  PROCEDURE STACK=0018H          440
 295  04EDH    17  TYPE23 . . . . .  PROCEDURE STACK=0020H          441
 301  04FEH    41  TYPE25 . . . . .  PROCEDURE STACK=0018H          442
 315  0527H    40  TYPE27 . . . . .  PROCEDURE STACK=0020H          443
 220  0402H    15  TYPE3. . . . . .  PROCEDURE STACK=0014H          429
 324  054FH    46  TYPE31 . . . . .  PROCEDURE STACK=0020H          444
 333  057DH    24  TYPE33 . . . . .  PROCEDURE STACK=0014H          445
 339  0595H    22  TYPE34 . . . . .  PROCEDURE STACK=0020H          446
 346  05ABH    62  TYPE36 . . . . .  PROCEDURE STACK=0020H          447
 366  05E9H    17  TYPE38 . . . . .  PROCEDURE STACK=0020H          448
 372  05FAH    35  TYPE39 . . . . .  PROCEDURE STACK=0020H          449
 224  0411H    31  TYPE4. . . . . .  PROCEDURE STACK=0014H          430
 383  061DH    36  TYPE40 . . . . .  PROCEDURE STACK=0020H          450
 395  0641H    29  TYPE42 . . . . .  PROCEDURE STACK=0020H          451
 401  065EH    41  TYPE44 . . . . .  PROCEDURE STACK=0020H          453
 410  0687H    13  TYPE45 . . . . .  PROCEDURE STACK=0024H          454
 230  0430H    11  TYPE5. . . . . .  PROCEDURE STACK=001CH          431
 234  043BH    11  TYPE6. . . . . .  PROCEDURE STACK=001CH          432
```

```
  238  0446H    11  TYPE8. . . . . .   PROCEDURE STACK=0020H          412  433
   13  01D0H     1  VBIT . . . . . .   BYTE          61  320
   13  01D1H     1  WBIT . . . . . .   BYTE          60   64   85   96  113  327  347  406
   13  01D2H     1  ZBIT . . . . . .   BYTE          60
```

MODULE INFORMATION:

```
    CODE AREA SIZE     = 0815H   2069D
    CONSTANT AREA SIZE = 0016H     22D
    VARIABLE AREA SIZE = 0203H    515D
    MAXIMUM STACK SIZE = 0030H     48D
    636 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION
```

```
ISIS-II MCS-86 ASSEMBLER V1.0 ASSEMBLY OF MODULE DISLNK86
OBJECT MODULE PLACED IN DISLNK86.OBJ
ASSEMBLER INVOKED BY: ASM86 DISLNK86.ASM DEBUG

LOC  OBJ                  LINE   SOURCE

                             1   ;       transient interface module for 8086
                             2   ;        JULY, 1980
                             3   dgroup  group   dats
----                         4   dats    segment public 'DATS'
0000 3F                      5           db      '?'
----                         6   dats    ends
                             7   cgroup  group   code
                             8           extrn   disem:near
                             9           public  boot
                            10           public  conin,conout
----                        11   code    segment public 'CODE'
                            12           assume cs:cgroup
0000                        13   disent  proc
1400                        14           org     1400h
1400 E90000          E      15           jmp     disem
                            16   disent  endp
1403                        17   boot    proc
1403 C3                     18           ret
                            19   boot    endp
1404                        20   conin   proc
1404 E9FCEC                 21           jmp     $-1301h
                            22   conin   endp
1407                        23   conout  proc
1407 E9FCEC                 24           jmp     $-1301h
                            25   conout  endp
----                        26   code    ends
                            27           end
```

SYMBOL TABLE LISTING
------ ----- -------

NAME	TYPE	VALUE	ATTRIBUTES
BOOT.	L NEAR	1403H	CODE PUBLIC
CGROUP.	GROUP		CODE
CODE.	SEGMENT		SIZE=140AH PARA PUBLIC 'CODE'
CONIN .	L NEAR	1404H	CODE PUBLIC
CONOUT.	L NEAR	1407H	CODE PUBLIC
DATS.	SEGMENT		SIZE=0001H PARA PUBLIC 'DATS'
DGROUP.	GROUP		DATS
DISEM .	L NEAR	0000H	EXTRN
DISENT.	L NEAR	0000H	CODE

ASSEMBLY COMPLETE, NO ERRORS FOUND

```
ISIS-II MCS-86 LOCATER, V1.2 INVOKED BY:
LOC86 DIS86.LNK TO DIS86.ABS AD(SM(DATS(0),CODE(0H))) CO(SM(DATS,CODE,CONST,DATA))
WARNING 36:  SEGMENTS OVERLAP
  SEGMENT:  DATS
  SEGMENT:  CODE
  LOW OVERLAP ADDRESS :  00000H
  HIGH OVERLAP ADDRESS:  00000H

SYMBOL TABLE OF MODULE DISLNK86
READ FROM FILE DIS86.LNK
WRITTEN TO FILE DIS86.ABS

BASE   OFFSET TYPE SYMBOL              BASE   OFFSET TYPE SYMBOL

0000H  1BC5H  PUB  DISEM               0000H  146AH  PUB  PRINTWORD
0000H  1417H  PUB  PRINTM              0000H  1403H  PUB  BOOT
0000H  1404H  PUB  CONIN               0000H  1407H  PUB  CONOUT
0000H  20BDH  PUB  SEGREG              0000H  20ADH  PUB  REG8
0000H  209DH  PUB  REG16               0000H  1EC6H  PUB  TABPTRS
0000H  208AH  PUB  OPNIN               0000H  2085H  PUB  NOPS
0000H  1CC6H  PUB  INSTRTABLE          0000H  1C36H  PUB  ALTTABLEPTRS

DISLNK86:  SYMBOLS AND LINES
0000H  1403H  SYM  BOOT                0000H  1404H  SYM  CONIN
0000H  1407H  SYM  CONOUT              0000H  0000H  SYM  DISENT

DISEM86:  SYMBOLS AND LINES
0000H  2136H  SYM  MEMORY              0000H  1EDEH  SYM  OPS2
0000H  1EF6H  SYM  OPS3                0000H  1FC5H  SYM  OPS4
0000H  2029H  SYM  OPS5                0000H  2079H  SYM  OPS6
0000H  2085H  SYM  NOPS                0000H  208AH  SYM  OPNIN
0000H  1EC6H  SYM  TABPTRS             0000H  1C20H  SYM  LEFTBRACKET
0000H  1C21H  SYM  RIGHTBRACKET        0000H  1ED0H  SYM  ALTTABLEBASE
0000H  1ED0H  BAS  ALTTABLE            0000H  2090H  SYM  MODBITS
0000H  2091H  SYM  REGBITS             0000H  2092H  SYM  RMBITS
0000H  2093H  SYM  BYTE1REGBITS        0000H  2094H  SYM  DBIT
0000H  2095H  SYM  SBIT                0000H  2096H  SYM  VBIT
0000H  2097H  SYM  WBIT                0000H  2098H  SYM  ZBIT
0000H  2099H  SYM  MNEMONICINDEX       0000H  209AH  SYM  INSTRTYPE
0000H  1ED2H  SYM  TABLEPTR            0000H  1ED2H  BAS  TABLECHAR
0000H  1ED4H  SYM  DISEMPTR            0000H  1ED4H  SYM  DISEMOFFSET
0000H  1ED8H  SYM  DISEMEND            0000H  1ED4H  BAS  DISEMBYTE
0000H  1ED4H  BAS  DISEMWORD           0000H  209BH  SYM  BORWFLAG
0000H  209CH  SYM  ERRORFLAG           0000H  209DH  SYM  REG16
0000H  20ADH  SYM  REG8                0000H  20BDH  SYM  SEGREG
0000H  140CH  SYM  COMMA               0000H  1417H  SYM  PRINTM
STACK  0004H  SYM  A                   STACK  0004H  BAS  B
0000H  1430H  SYM  PRINTNIBBLE         STACK  0004H  SYM  B
0000H  144FH  SYM  PRINTBYTE           STACK  0004H  SYM  B
0000H  146AH  SYM  PRINTWORD           STACK  0004H  SYM  A
0000H  1481H  SYM  ERROR               0000H  149BH  SYM  SETBITS
0000H  14E4H  SYM  PRINTBORW           0000H  14FCH  SYM  PRINTREG
STACK  0006H  SYM  REGADD              STACK  0004H  SYM  REG
0000H  1525H  SYM  PRINTREG8           STACK  0004H  SYM  REG
0000H  1536H  SYM  PRINTREG16          STACK  0004H  SYM  REG
0000H  1547H  SYM  PRINTREG8OR16       STACK  0004H  SYM  REGNUM
0000H  1563H  SYM  PRINT2REG16         STACK  0006H  SYM  R1
STACK  0004H  SYM  R2                  0000H  157CH  SYM  PRINTAREG
0000H  1596H  SYM  PRINTSEGREG         STACK  0004H  SYM  REG
0000H  15A7H  SYM  PRINTDATA8          0000H  15BAH  SYM  PRINTDATA16
```

```
0000H  15CEH  SYM  PRINTDATA8OR16
0000H  160FH  SYM  PRINTSIGNED8
0000H  163AH  SYM  PRINTSIGNED16
0000H  1677H  SYM  PRINTMODRM
0000H  1772H  SYM  PRINTMNEMONIC
0000H  20C6H  SYM  I
0000H  17EEH  SYM  TYPE2
0000H  181BH  SYM  TYPE4
0000H  183AH  SYM  TYPE5
0000H  1850H  SYM  TYPE8
0000H  1866H  SYM  TYPE11
0000H  187FH  SYM  TYPE13
0000H  18A6H  SYM  TYPE14
0000H  18D4H  SYM  TYPE22
0000H  1908H  SYM  TYPE25
0000H  1959H  SYM  TYPE31
0000H  199FH  SYM  TYPE34
0000H  19F3H  SYM  TYPE38
0000H  1A27H  SYM  TYPE40
0000H  1A68H  SYM  TYPE44
0000H  1A9EH  SYM  DIS
STACK  0004H  SYM  DISLOC
0000H  140CH  LIN     23
0000H  1415H  LIN     25
0000H  141AH  LIN     29
0000H  1427H  LIN     31
0000H  142CH  LIN     33
0000H  1433H  LIN     36
0000H  1442H  LIN     38
0000H  144FH  LIN     40
0000H  145DH  LIN     43
0000H  146AH  LIN     45
0000H  1476H  LIN     48
0000H  1481H  LIN     50
0000H  148BH  LIN     52
0000H  1499H  LIN     54
0000H  149EH  LIN     56
0000H  1489H  LIN     58
0000H  14CAH  LIN     60
0000H  14E2H  LIN     62
0000H  14E7H  LIN     64
0000H  14F3H  LIN     66
0000H  14FCH  LIN     68
0000H  150CH  LIN     71
0000H  151AH  LIN     73
0000H  1525H  LIN     75
0000H  1532H  LIN     78
0000H  1539H  LIN     81
0000H  1547H  LIN     83
0000H  1551H  LIN     86
0000H  155FH  LIN     88
0000H  1566H  LIN     91
0000H  1572H  LIN     93
0000H  157CH  LIN     95
0000H  1586H  LIN     97
0000H  1594H  LIN     99
0000H  1599H  LIN    102
0000H  15A7H  LIN    104
0000H  15B4H  LIN    106
0000H  15BAH  LIN    108
0000H  15E2H  SYM  PRINTDATA8N
0000H  1EDAH  SYM  A
0000H  1655H  SYM  PRINTDIRECTADDR
0000H  1744H  SYM  PRINTMODREGRM
0000H  20C5H  SYM  LEN
0000H  17E6H  SYM  TYPE1
0000H  180CH  SYM  TYPE3
0000H  20C7H  SYM  TEMP
0000H  1845H  SYM  TYPE6
0000H  185BH  SYM  TYPE10
0000H  1871H  SYM  TYPE12
0000H  1EDCH  SYM  TEMP
0000H  18B1H  SYM  TYPE18
0000H  18F7H  SYM  TYPE23
0000H  1931H  SYM  TYPE27
0000H  1987H  SYM  TYPE33
0000H  19B5H  SYM  TYPE36
0000H  1AD4H  SYM  TYPE39
0000H  1A4BH  SYM  TYPE42
0000H  1A91H  SYM  TYPE45
0000H  1BC5H  SYM  DISEM
0000H  20C8H  SYM  NPREFIX
0000H  140FH  LIN     24
0000H  1417H  LIN     26
0000H  1422H  LIN     30
0000H  142AH  LIN     32
0000H  1430H  LIN     34
0000H  1439H  LIN     37
0000H  144BH  LIN     39
0000H  1452H  LIN     42
0000H  1466H  LIN     44
0000H  146DH  LIN     47
0000H  147DH  LIN     49
0000H  1484H  LIN     51
0000H  1495H  LIN     53
0000H  149BH  LIN     55
0000H  14ADH  LIN     57
0000H  14C4H  LIN     59
0000H  14D4H  LIN     61
0000H  14E4H  LIN     63
0000H  14EEH  LIN     65
0000H  14FAH  LIN     67
0000H  14FFH  LIN     70
0000H  1513H  LIN     72
0000H  1521H  LIN     74
0000H  1528H  LIN     77
0000H  1536H  LIN     79
0000H  1543H  LIN     82
0000H  154AH  LIN     85
0000H  1559H  LIN     87
0000H  1563H  LIN     89
0000H  156CH  LIN     92
0000H  1578H  LIN     94
0000H  157FH  LIN     96
0000H  158EH  LIN     98
0000H  1596H  LIN    100
0000H  15A3H  LIN    103
0000H  15AAH  LIN    105
0000H  15B8H  LIN    107
0000H  15BDH  LIN    109
```

```
0000H  15C7H  LIN   110
0000H  15CEH  LIN   112
0000H  15D8H  LIN   114
0000H  15E0H  LIN   116
0000H  15E5H  LIN   118
0000H  15FCH  LIN   120
0000H  160DH  LIN   122
0000H  1612H  LIN   125
0000H  1622H  LIN   127
0000H  1634H  LIN   129
0000H  163AH  LIN   131
0000H  164EH  LIN   133
0000H  1655H  LIN   135
0000H  165FH  LIN   137
0000H  1670H  LIN   139
0000H  1677H  LIN   141
0000H  167EH  LIN   143
0000H  168CH  LIN   146
0000H  1695H  LIN   149
0000H  16A6H  LIN   152
0000H  16ABH  LIN   155
0000H  16C3H  LIN   158
0000H  16D1H  LIN   160
0000H  16DAH  LIN   163
0000H  16E9H  LIN   166
0000H  16FDH  LIN   168
0000H  1708H  LIN   170
0000H  1717H  LIN   172
0000H  171FH  LIN   174
0000H  172BH  LIN   176
0000H  1742H  LIN   178
0000H  1747H  LIN   180
0000H  1755H  LIN   183
0000H  175EH  LIN   185
0000H  1763H  LIN   188
0000H  1770H  LIN   191
0000H  1775H  LIN   194
0000H  178BH  LIN   196
0000H  1791H  LIN   198
0000H  17BCH  LIN   200
0000H  17CEH  LIN   203
0000H  17D4H  LIN   205
0000H  17E0H  LIN   207
0000H  17E6H  LIN   209
0000H  17ECH  LIN   211
0000H  17F1H  LIN   213
0000H  17FFH  LIN   216
0000H  1805H  LIN   218
0000H  180CH  LIN   220
0000H  1812H  LIN   222
0000H  181BH  LIN   224
0000H  182EH  LIN   227
0000H  1838H  LIN   229
0000H  183DH  LIN   231
0000H  1843H  LIN   233
0000H  1848H  LIN   235
0000H  184EH  LIN   237
0000H  1853H  LIN   239
0000H  1859H  LIN   241
0000H  185EH  LIN   243

0000H  15CCH  LIN   111
0000H  15D1H  LIN   113
0000H  15DDH  LIN   115
0000H  15E2H  LIN   117
0000H  15F2H  LIN   119
0000H  1609H  LIN   121
0000H  160FH  LIN   123
0000H  161EH  LIN   126
0000H  1628H  LIN   128
0000H  1638H  LIN   130
0000H  163DH  LIN   132
0000H  1653H  LIN   134
0000H  1658H  LIN   136
0000H  1669H  LIN   138
0000H  1675H  LIN   140
0000H  167AH  LIN   142
0000H  1685H  LIN   145
0000H  168EH  LIN   148
0000H  1698H  LIN   150
0000H  16A9H  LIN   153
0000H  16B2H  LIN   157
0000H  16CDH  LIN   159
0000H  16D3H  LIN   161
0000H  16E4H  LIN   164
0000H  16F0H  LIN   167
0000H  1704H  LIN   169
0000H  170CH  LIN   171
0000H  171BH  LIN   173
0000H  1727H  LIN   175
0000H  173BH  LIN   177
0000H  1744H  LIN   179
0000H  174EH  LIN   182
0000H  175BH  LIN   184
0000H  1760H  LIN   187
0000H  1769H  LIN   189
0000H  1772H  LIN   192
0000H  177AH  LIN   195
0000H  178FH  LIN   197
0000H  17B0H  LIN   199
0000H  17C5H  LIN   202
0000H  17D2H  LIN   204
0000H  17DAH  LIN   206
0000H  17E4H  LIN   208
0000H  17E9H  LIN   210
0000H  17EEH  LIN   212
0000H  17FCH  LIN   215
0000H  1803H  LIN   217
0000H  180AH  LIN   219
0000H  180FH  LIN   221
0000H  1819H  LIN   223
0000H  181EH  LIN   226
0000H  1831H  LIN   228
0000H  183AH  LIN   230
0000H  1840H  LIN   232
0000H  1845H  LIN   234
0000H  1848H  LIN   236
0000H  1850H  LIN   238
0000H  1856H  LIN   240
0000H  185BH  LIN   242
0000H  1861H  LIN   244
```

```
0000H  1864H  LIN    245
0000H  1869H  LIN    247
0000H  186FH  LIN    249
0000H  1874H  LIN    251
0000H  187DH  LIN    253
0000H  1882H  LIN    256
0000H  188FH  LIN    258
0000H  1897H  LIN    260
0000H  18A4H  LIN    262
0000H  18A9H  LIN    264
0000H  18AFH  LIN    266
0000H  18B4H  LIN    268
0000H  18BEH  LIN    271
0000H  18C4H  LIN    273
0000H  18C9H  LIN    276
0000H  18CFH  LIN    278
0000H  18D4H  LIN    281
0000H  18DAH  LIN    283
0000H  18E4H  LIN    286
0000H  18EAH  LIN    288
0000H  18EFH  LIN    291
0000H  18F5H  LIN    294
0000H  18FAH  LIN    296
0000H  1900H  LIN    298
0000H  1906H  LIN    300
0000H  1908H  LIN    302
0000H  1915H  LIN    305
0000H  191EH  LIN    307
0000H  1923H  LIN    310
0000H  1929H  LIN    312
0000H  1931H  LIN    315
0000H  1937H  LIN    317
0000H  193FH  LIN    319
0000H  1949H  LIN    321
0000H  1957H  LIN    323
0000H  195CH  LIN    325
0000H  1965H  LIN    327
0000H  1978H  LIN    329
0000H  1982H  LIN    331
0000H  1987H  LIN    333
0000H  198DH  LIN    335
0000H  1996H  LIN    337
0000H  199FH  LIN    339
0000H  19A5H  LIN    341
0000H  19ADH  LIN    343
0000H  19B3H  LIN    345
0000H  19B8H  LIN    347
0000H  19C4H  LIN    350
0000H  19CBH  LIN    353
0000H  19D5H  LIN    356
0000H  19DFH  LIN    358
0000H  19E4H  LIN    361
0000H  19EAH  LIN    363
0000H  19F3H  LIN    366
0000H  19F9H  LIN    368
0000H  19FFH  LIN    370
0000H  1A04H  LIN    372
0000H  1A0EH  LIN    375
0000H  1A15H  LIN    378
0000H  1A1FH  LIN    380
0000H  1856H  LIN    246
0000H  185CH  LIN    248
0000H  1871H  LIN    250
0000H  1877H  LIN    252
0000H  187FH  LIN    254
0000H  1885H  LIN    257
0000H  1894H  LIN    259
0000H  189DH  LIN    261
0000H  18A6H  LIN    263
0000H  18ACH  LIN    265
0000H  18B1H  LIN    267
0000H  18B7H  LIN    269
0000H  18C1H  LIN    272
0000H  18C7H  LIN    274
0000H  18CCH  LIN    277
0000H  18D2H  LIN    280
0000H  18D7H  LIN    282
0000H  18E1H  LIN    285
0000H  18E7H  LIN    287
0000H  18ECH  LIN    290
0000H  18F2H  LIN    292
0000H  18F7H  LIN    295
0000H  18FDH  LIN    297
0000H  1903H  LIN    299
0000H  1908H  LIN    301
0000H  190EH  LIN    303
0000H  191BH  LIN    306
0000H  1921H  LIN    308
0000H  1926H  LIN    311
0000H  192FH  LIN    314
0000H  1934H  LIN    316
0000H  193CH  LIN    318
0000H  1942H  LIN    320
0000H  1951H  LIN    322
0000H  1959H  LIN    324
0000H  195FH  LIN    326
0000H  1975H  LIN    328
0000H  197FH  LIN    330
0000H  1985H  LIN    332
0000H  198AH  LIN    334
0000H  1993H  LIN    336
0000H  199DH  LIN    338
0000H  19A2H  LIN    340
0000H  19AAH  LIN    342
0000H  19B0H  LIN    344
0000H  19B5H  LIN    346
0000H  19BDH  LIN    348
0000H  19C9H  LIN    351
0000H  19CEH  LIN    354
0000H  19DCH  LIN    357
0000H  19E2H  LIN    359
0000H  19E7H  LIN    362
0000H  19F1H  LIN    365
0000H  19F6H  LIN    367
0000H  19FCH  LIN    369
0000H  1A02H  LIN    371
0000H  1A07H  LIN    373
0000H  1A13H  LIN    376
0000H  1A18H  LIN    379
0000H  1A22H  LIN    381
```

```
0000H  1A25H  LIN    382        0000H  1A27H  LIN    383
0000H  1A2AH  LIN    384        0000H  1A31H  LIN    386
0000H  1A36H  LIN    387        0000H  1A38H  LIN    389
0000H  1A3BH  LIN    390        0000H  1A40H  LIN    391
0000H  1A43H  LIN    392        0000H  1A46H  LIN    393
0000H  1A49H  LIN    394        0000H  1A4BH  LIN    395
0000H  1A4EH  LIN    396        0000H  1A51H  LIN    397
0000H  1A60H  LIN    398        0000H  1A63H  LIN    399
0000H  1A66H  LIN    400        0000H  1A68H  LIN    401
0000H  1A6BH  LIN    402        0000H  1A6EH  LIN    403
0000H  1A73H  LIN    404        0000H  1A76H  LIN    405
0000H  1A79H  LIN    406        0000H  1A87H  LIN    407
0000H  1A8CH  LIN    408        0000H  1A8FH  LIN    409
0000H  1A91H  LIN    410        0000H  1A94H  LIN    411
0000H  1A99H  LIN    412        0000H  1A9CH  LIN    413
0000H  1A9EH  LIN    414        0000H  1AA1H  LIN    415
0000H  1AABH  LIN    416        0000H  1AAEH  LIN    417
0000H  1AB5H  LIN    419        0000H  1AC4H  LIN    420
0000H  1AD7H  LIN    421        0000H  1ADDH  LIN    423
0000H  1AE4H  LIN    425        0000H  1AF1H  LIN    426
0000H  1AF8H  LIN    427        0000H  1AFDH  LIN    428
0000H  1B02H  LIN    429        0000H  1B07H  LIN    430
0000H  1B0CH  LIN    431        0000H  1B11H  LIN    432
0000H  1B16H  LIN    433        0000H  1B18H  LIN    434
0000H  1B20H  LIN    435        0000H  1B25H  LIN    436
0000H  1B2AH  LIN    437        0000H  1B2FH  LIN    438
0000H  1B34H  LIN    439        0000H  1B39H  LIN    440
0000H  1B3EH  LIN    441        0000H  1B43H  LIN    442
0000H  1B48H  LIN    443        0000H  1B4DH  LIN    444
0000H  1B52H  LIN    445        0000H  1B57H  LIN    446
0000H  1B5CH  LIN    447        0000H  1B61H  LIN    448
0000H  1B66H  LIN    449        0000H  1B6BH  LIN    450
0000H  1B70H  LIN    451        0000H  1B73H  LIN    452
0000H  1B75H  LIN    453        0000H  1B7AH  LIN    454
0000H  1B7FH  LIN    455        0000H  1BB9H  LIN    456
0000H  1BC0H  LIN    457        0000H  1BC3H  LIN    458
0000H  1BC5H  LIN    459        0000H  1BC8H  LIN    462
0000H  1BD2H  LIN    463        0000H  1BD7H  LIN    464
0000H  1BD7H  LIN    465        0000H  1BECH  LIN    466
0000H  1BF3H  LIN    467        0000H  1BFEH  LIN    469
0000H  1C01H  LIN    470        0000H  1C05H  LIN    471
0000H  1C07H  LIN    473        0000H  1C0EH  LIN    474
0000H  1C13H  LIN    475        0000H  1C16H  LIN    476
0000H  1C1BH  LIN    478        0000H  1C1BH  LIN    479
0000H  140CH  LIN    480

INSTR86:  SYMBOLS AND LINES
0000H  2136H  SYM   MEMORY      0000H  1C36H  SYM   ALTTABLEPTRS
0000H  1C46H  SYM   ALT1TAB     0000H  1C56H  SYM   ALT2TAB
0000H  1C66H  SYM   ALT3TAB     0000H  1C76H  SYM   ALT4TAB
0000H  1C86H  SYM   ALT5TAB     0000H  1C96H  SYM   ALT6TAB
0000H  1CA6H  SYM   ALT7TAB     0000H  1CB6H  SYM   ALT8TAB
0000H  1CC6H  SYM   INSTRTABLE  0000H  1C20H  LIN     17

MEMORY MAP OF MODULE DISLNK86
READ FROM FILE DIS86.LNK
WRITTEN TO FILE DIS86.ABS

SEGMENT MAP
```

```
START     STOP      LENGTH ALIGN NAME              CLASS

00000H    00000H  C   0001H   G   DATS             DATS
00000H    01C1FH      1C20H   G   CODE             CODE
01C20H    01EC5H      02A6H   W   CONST            CONST
01EC6H    020C9H      0204H   W   DATA             DATA
020D0H    02105H      0036H   G   ??SEG
02106H    02135H      0030H   W   STACK            STACK
02136H    02136H      0000H   W   MEMORY           MEMORY


GROUP MAP

ADDRESS   GROUP OR SEGMENT NAME
00000H    CGROUP
          CODE
00000H    DGROUP
          DATS
          CONST
          DATA
          STACK
          MEMORY
```

```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE DDTASM
OBJECT MODULE PLACED IN ASS86.OBJ
COMPILER INVOKED BY:  PLM86 ASS86.PLM DEBUG XREF PAGEWIDTH(100)
```

```
            $title('assembler for ddt86')
            $date(6/15/81)
            $compact
            $optimize(2)

1           ddtasm: do;

2   1       declare
                logical literally 'byte',
                true literally '1',
                false literally '0',
                tab literally '09h',
                cr literally '0dh',
                lf literally '0ah';

3   1       declare
                tab$ptrs (5) address external,
                nops (5) byte external,
                opn$in (6) byte external,
                optab (480) byte external;

4   1       declare
                token (32) byte,        /* ASCII token */
                token$word address at (.token),    /* used for word compares */
                token$len byte,         /* number of chars in token */
                token$type byte,        /* number, string, or delimiter */
                    string$type literally '0',
                    number$type literally '1',
                    delim$type literally '2',
                token$value address,    /* value if token$type = number */
                next$ch byte;           /* next char of input stream */

5   1       declare
                assem$ptr pointer,
                assem$offset address at (.assem$ptr),
                assem$segment address at (.assem$ptr+2),
                assem$byte based assem$ptr byte;

6   1       declare
                seg$reg (8) byte EXTERNAL,
                reg16 (16) byte EXTERNAL,
                reg8 (16) byte EXTERNAL,
                base$reg (*) byte data ('BX', 'BP'),
                index$reg (*) byte data ('SI', 'DI');

7   1       declare
                op$base address,
                (op1, op2, op based op$base) structure (
                    seg$reg$num byte,
                    reg$num byte,
                    base$reg$num byte,
```

```
                index$reg$num byte,
                seg$reg$type logical,
                reg$16$type logical,
                reg$8$type logical,
                num$type logical,
                acc$type logical,
                null$type logical,
                far$type logical,
                mod$rm$type logical,
                w$bit byte,
                bw$prefix byte,
                sub$type byte,
                seg$value address,
                value address);

        /*
            op.sub$types

            1   reg8
            2   reg16
            3   [num]
            4   [pointer reg]
            5   [index reg]
            6   [pointer reg + index reg]
            7   num [pointer reg]
            8   num [index reg]
            9   num [pointer reg + index reg]
        */

 8  1      declare
            prefix (3) byte,        /* up to 3 prefixes */
            nprefix byte,           /* number of prefixes */
            seg$prefix$flag byte,
            lock$flag byte,
            rep$flag byte;

 9  1      declare
            mod$bits byte,
            reg$bits byte,
            rm$bits byte,
            op$type byte,
            mnemonic$index byte,
            instr$l byte,
            disp$len byte,          /* number of bytes in optional disp field */
            disp$word address;      /* value of optional disp field */

10  1      declare
            delimiters (*) byte data (',', ':', '+', cr, ' ', tab),
            left$bracket byte data ('['),
            right$bracket byte data (']'),
            num$delims literally '(length (delimiters) + 2)';

11  1      go$ddt: procedure external;     /* return to assem loop in ddt on error */
12  2          end go$ddt;

13  1      ddt$set: procedure (assem$off, b) external;
14  2          declare assem$off address, b byte;
```

```
15  2       end ddt$set;

16  1   ddt$getline: procedure external;
17  2       end ddt$getline;

18  1   conin: procedure byte external;
19  2       end conin;

20  1   conout: procedure (b) external;
21  2       declare b byte;
22  2       end conout;

23  1   printm: procedure (a) EXTERNAL;
24  2       declare a address;
25  2       end printm;

26  1   crlf: procedure;
27  2       call conout (cr);
28  2       call conout (lf);
29  2       end crlf;

30  1   getline: procedure;
31  2       call ddt$getline;
32  2       next$ch = 0;
33  2       end getline;

34  1   print$word: procedure (a) EXTERNAL;
35  2       declare a address;
36  2       end print$word;

37  1   error: procedure;
38  2       call crlf;
39  2       call conout ('?');
40  2       call go$ddt;
41  2       end error;

42  1   ambig: procedure;
43  2       call printm (.(cr,lf,'ambiguous operand$'));
44  2       call go$ddt;
45  2       end ambig;

46  1   check$ambig: procedure;
47  2       if (op1.w$bit and op2.w$bit) = 0ffh then call ambig;
49  2       if (op1.w$bit xor op2.w$bit) = 1 then call error;
51  2       end check$ambig;

52  1   compare: procedure (a, b, c) logical;
         /* should be able to replace this with a PL/M builtin */
53  2       declare (a, b) address, c byte;
54  2       declare a1 based a byte, b1 based b byte;
55  2       do while (c := c - 1) <> 255;
56  3           if a1 <> b1 then return false;
58  3           a = a + 1;
59  3           b = b + 1;
60  3       end;
61  2       return true;
62  2       end compare;
```

```
                /**********************************/
                /*                                */
                /*   token scanning procedures    */
                /*                                */
                /**********************************/

 63   1         delimiter: procedure (b) logical;
 64   2             declare b byte;
 65   2             declare i byte;
 66   2             do i = 0 to num$delims - 1;
 67   3                 if delimiters (i) = b then return true;
 69   3             end;
 70   2             return false;
 71   2             end delimiter;

 72   1         number: procedure (b) logical;
 73   2             declare b byte;
 74   2             return (b - '0' <= 9) or (b - 'A' <= 5);
 75   2             end number;

 76   1         hex: procedure (b) byte;
 77   2             declare b byte;
 78   2             if b - '0' <= 9 then return b - '0';
 80   2             else return b - 'A' + 10;
 81   2             end hex;

 82   1         get$token: procedure;
 83   2             declare numeric logical;
 84   2             token$value = 0;
 85   2             numeric = true;
 86   2             token$len = 0;
 87   2             if next$ch = 0 then next$ch = conin;
 89   2             do while true;
 90   3                 token (token$len) = next$ch;
 91   3                 if (token$len := token$len + 1) >= length (token) then
 92   3                   call error;
 93   3                 if delimiter (next$ch) then
 94   3                     do;
 95   4                         if token$len = 1 then
 96   4                             do;
 97   5                                 token$type = delim$type;
 98   5                                 next$ch = 0;
 99   5                             end;
100   4                         else
101   5                             do;
102   5                                 token$len = token$len - 1;
104   5                                 if numeric then token$type = number$type;
105   5                                 else token$type = string$type;
106   4                             end;
107   4                         return;
108   3                     end;
109   3                 if numeric then
110   4                     do;
                                if number (next$ch) then token$value =
                                  shl (token$value,4) or hex (next$ch);
112   4                         else numeric = false;
```

```
113   4                 end;
114   3               next$ch = conin;
115   3           end;
116   2           end get$token;

117   1       get$real$token: procedure;
118   2           do while true;
119   3               call get$token;
120   3               if token (0) <> ' ' and token (0) <> tab then return;
122   3           end;
123   2           end get$real$token;

            /*************************************/
            /*                                   */
            /*   operand scanning procedures     */
            /*                                   */
            /*************************************/

124   1       look$up$reg: procedure (reg$tab$ptr, tab$len, reg$num$ptr) logical;
125   2           declare (reg$tab$ptr, reg$num$ptr) address;
126   2           declare reg$word based reg$tab$ptr address;
127   2           declare reg$num based reg$num$ptr byte;
128   2           declare tab$len byte;
129   2           declare i byte;
130   2           do i = 0 to tab$len - 1;
131   3               if token$word = reg$word then
132   3                   do;
133   4                       reg$num = i;
134   4                       return true;
135   4                   end;
136   3               reg$tab$ptr = reg$tab$ptr + 2;
137   3           end;
138   2           return false;
139   2           end look$up$reg;

140   1       check$base$reg: procedure logical;
141   2           return look$up$reg (.base$reg, 2, .op.base$reg$num);
142   2           end check$base$reg;

143   1       check$index$reg: procedure logical;
144   2           return look$up$reg (.index$reg, 2, .op.index$reg$num);
145   2           end check$index$reg;

146   1       check$seg$reg: procedure logical;
147   2           if look$up$reg (.seg$reg, 4, .op.seg$reg$num) then
148   2               do;
149   3                   op.seg$reg$type = true;
150   3                   return true;
151   3               end;
152   2           return false;
153   2           end check$seg$reg;

154   1       check$reg$8: procedure logical;
155   2           if look$up$reg (.reg$8, 8, .op.reg$num) then
156   2               do;
157   3                   op.reg$8$type, op.modrm$type = true;
158   3                   if op.reg$num = 0 then op.acc$type = true;
```

```
160   3                 op.sub$type = 1;
161   3                 op.w$bit = 0;
162   3                 return true;
163   3             end;
164   2         return false;
165   2         end check$reg$8;

166   1     check$reg$16: procedure logical;
167   2         if look$up$reg (.reg$16, 8, .op.reg$num) then
168   2             do;
169   3                 op.reg$16$type, op.modrm$type = true;
170   3                 if op.reg$num = 0 then op.acc$type = true;
172   3                 op.sub$type = 2;
173   3                 op.w$bit = 1;
174   3                 return true;
175   3             end;
176   2         return false;
177   2         end check$reg$16;

178   1     check$bw$prefix: procedure logical;
179   2         if token$type = string$type then
180   2             do;
181   3                 if token$word = 'YB' then op.bw$prefix = 0;
183   3                 else if token$word = 'OW' then op.bw$prefix = 1;
                        if op.bw$prefix <> 0ffh then return true;
187   3             end;
188   2         return false;
189   2         end check$bw$prefix;

190   1     check$seg$prefix: procedure logical;
191   2         return check$seg$reg and (next$ch = ':');
192   2         end check$seg$prefix;

193   1     set$prefix: procedure (b);
194   2         declare b byte;
195   2         prefix (nprefix) = b;
196   2         nprefix = nprefix + 1;
197   2         end set$prefix;

198   1     set$seg$prefix: procedure;
199   2         if seg$prefix$flag then call error;
201   2         call set$prefix (26h or shl (op.seg$reg$num, 3));
202   2         seg$prefix$flag = true;
203   2         call get$token;   /* eat ':' */
204   2         op.seg$reg$type = false; /* operand is not a seg reg */
205   2         end set$seg$prefix;

206   1     get$prefix: procedure;
207   2         do while true;
208   3             call get$real$token;
209   3             if token$type = delim$type then
210   3                 do;
211   4                     if token (0) = cr then op.null$type = true;
213   4                     return;
214   4                 end;
215   3             if check$seg$prefix then call set$seg$prefix;
217   3             else if check$bw$prefix then
```

```
218   3                 do;
219   4                     if op.w$bit <> 0ffh then call error;
221   4                     else op.w$bit = op.bw$prefix;
222   4                 end;
223   3             else return;   /* must not be a prefix */
224   3         end;
225   2         end get$prefix;

226   1     reg$indirect: procedure (x);
227   2         declare x byte;
228   2         op.modrm$type = true;
229   2         if check$base$reg then
230   2             do;
231   3                 call get$token;
232   3                 if token (0) = right$bracket then
233   3                     do;
234   4                         op.modrm$type = true;
235   4                         op.sub$type = 7 - x;
236   4                         return;
237   4                     end;
238   3                 else if token (0) = '+' then
239   3                     do;
240   4                         op.sub$type = 9 - x;
241   4                         call get$token;
242   4                     end;
243   3                 else call error;
244   3             end;
245   2         else op.sub$type = 8 - x;
246   2         if check$index$reg then
247   2             do;
248   3                 call get$token;
249   3                 if token (0) = right$bracket then return;
251   3             end;
252   2         call error;
253   2         end reg$indirect;

254   1     get$operand: procedure;
255   2         op.seg$reg$type, op.reg$16$type, op.reg$8$type, op.num$type,
              op.acc$type, op.null$type, op.far$type, op.modrm$type = false;
256   2         op.bw$prefix, op.w$bit = 0ffh;   /* indeterminate */
257   2         call get$prefix;
258   2         if op.null$type then return;
260   2         if check$seg$reg then return;
262   2         else if check$reg$8 then return;
264   2         else if check$reg$16 then return;
266   2         else if token$type = number$type then
267   2             do;
268   3                 op.value = token$value;
269   3                 if next$ch = left$bracket then
270   3                     do;
271   4                         call get$token;   /* eat '[' */
272   4                         call get$token;
273   4                         call reg$indirect (0);
274   4                     end;
275   3                 else if next$ch = ':' then
276   3                     do;
277   4                         op.seg$value = token$value;
```

```
278   4                       call get$token;   /* eat ':' */
279   4                       call get$token;
280   4                       if token$type <> number$type then call error;
282   4                       op.value = token$value;
283   4                       op.far$type = true;
284   4                   end;
285   3               else op.num$type = true;
286   3           end;
            else
287   2           do;
288   3               if token (0) <> left$bracket then call error;
290   3               call get$token;
291   3               if token$type = number$type then
292   3                   do;
293   4                       op.value = token$value;
294   4                       call get$token;
295   4                       if token (0) = right$bracket then
296   4                           do;
297   5                               op.sub$type = 3;
298   5                               op.modrm$type = true;
299   5                           end;
300   4                       else call error;
301   4                   end;
302   3               else call reg$indirect (3);
303   3           end;
304   2       end get$operand;

305   1   get$operand$1: procedure;
306   2       op$base = .op1;
307   2       call get$operand;
308   2       call get$token;
309   2       if token (0) <> ',' then call error;
311   2       end get$operand$1;

312   1   get$operand$2: procedure;
313   2       op$base = .op2;
314   2       call get$operand;
315   2       if op.null$type then return;
317   2       call get$token;
318   2       if token (0) <> cr then call error;
320   2       end get$operand$2;

        /************************************/
        /*                                  */
        /*   opcode scanning procedures     */
        /*                                  */
        /************************************/

321   1   lookup$optab: procedure;
322   2       declare i address;
323   2       do i = 0 to last (optab) by 4;
324   3           if optab (i) = mnemonic$index then
325   3               do;
326   4                   op$type = op$tab (i+1);
327   4                   instr$1 = op$tab (i+2);
328   4                   reg$bits = op$tab (i+3);
329   4                   return;
```

```
330   4                 end;
331   3             end;
332   2             call error;   /* should never get here */
333   2             end lookup$optab;

334   1         valid$mnemonic: procedure logical;
335   2             declare
                      i byte,
                      table$base address,
                      mnemonic based table$base byte;
336   2             if token$len - 2 > 4 then call error;
338   2             table$base = tab$ptrs (token$len - 2);
339   2             do i = 0 to nops (token$len - 2) - 1;
340   3                 if compare (.token, .mnemonic, token$len) then
341   3                     do;
342   4                         mnemonic$index = i + opn$in (token$len - 2);
343   4                         return true;
344   4                     end;
345   3                 table$base = table$base + token$len;
346   3             end;
347   2             return false;
348   2             end valid$mnemonic;

349   1         get$opcode: procedure logical;
350   2             nprefix = 0;
351   2             rep$flag, lock$flag, seg$prefix$flag = false;
352   2             do while true;
353   3                 call get$real$token;
354   3                 if token (0) = cr or token (0) = '.' then return false;   /* no opcode */
356   3                 if check$seg$prefix then call set$seg$prefix;
358   3                 else if valid$mnemonic then
359   3                     do;
360   4                         call look$up$op$tab;
361   4                         if op$type = 0ffh then
362   4                             do;
363   5                                 if lock$flag then call error;
365   5                                 lock$flag = true;
366   5                                 call set$prefix (instr$1);
367   5                             end;
368   4                         else if op$type = 0feh then    /* repeat */
369   4                             do;
370   5                                 if rep$flag then call error;
372   5                                 rep$flag = true;
373   5                                 call set$prefix (instr$1);
374   5                             end;
375   4                         else return true;   /* opcode present */
376   4                     end;
377   3                 else call error;
378   3             end;
379   2             end get$opcode;

            /*****************************************/
            /*                                       */
            /*   forming and output of instructions  */
            /*                                       */
            /*****************************************/
```

```
380   1       put$mem: procedure (b);
381   2           declare b byte;
382   2           call ddt$set (assem$offset, b);
383   2           assem$offset = assem$offset + 1;
384   2           end put$mem;

385   1       put$prefix: procedure;
386   2           declare i byte;
387   2           i = 0;
388   2           do while (nprefix := nprefix - 1) <> 0ffh;
389   3               call put$mem (prefix (i));
390   3               i = i + 1;
391   3           end;
392   2           end put$prefix;

393   1       put$instr: procedure;
394   2           call put$prefix;
395   2           call put$mem (instr$1);
396   2           end put$instr;

397   1       set$instr: procedure (b);
398   2           declare b byte;
399   2           instr$1 = b;
400   2           call put$instr;
401   2           end set$instr;

402   1       put$word: procedure (a);
403   2           declare a address;
404   2           call put$mem (low (a));
405   2           call put$mem (high (a));
406   2           end put$word;

407   1       set$w$bit: procedure;
408   2           if (op1.w$bit and op2.w$bit) = 1 then instr$1 = instr$1 or 1;
410   2           end set$w$bit;

411   1       set$mod: procedure;
412   2           if op.value > 7FH then mod$bits, disp$len = 2;
414   2           else mod$bits, disp$len = 1;
415   2           end set$mod;

416   1       set$modrm: procedure;
417   2           disp$len = 0;
418   2           disp$word = op.value;
419   2           do case op.sub$type - 1;
420   3               do;   /* 1 */
421   4                   mod$bits = 3;
422   4                   rm$bits = op.reg$num;
423   4               end;
424   3               do;   /* 2 */
425   4                   mod$bits = 3;
426   4                   rm$bits = op.reg$num;
427   4               end;
428   3               do;   /* 3 */
429   4                   mod$bits = 0;
430   4                   rm$bits = 6;
431   4                   disp$len = 2;
```

```
 432   4              end;
 433   3              do;   /* 4 */
 434   4                  if op.base$reg$num = 1 then   /* BP */
 435   4                      do;
 436   5                          mod$bits = 1;
 437   5                          rm$bits = 6;
 438   5                          disp$len = 1;
 439   5                          disp$word = 0;
 440   5                      end;
                           else
 441   4                      do;
 442   5                          mod$bits = 0;
 443   5                          rm$bits = 7;
 444   5                      end;
 445   4              end;
 446   3              do;   /* 5 */
 447   4                  mod$bits = 0;
 448   4                  rm$bits = op.index$reg$num + 4;
 449   4              end;
 450   3              do;   /* 6 */
 451   4                  mod$bits = 0;
 452   4                  rm$bits = op.base$reg$num * 2 + op.index$reg$num;
 453   4              end;
 454   3              do;   /* 7 */
 455   4                  call set$mod;
 456   4                  rm$bits = 7 - op.base$reg$num;
 457   4              end;
 458   3              do;   /* 8 */
 459   4                  call set$mod;
 460   4                  rm$bits = op.index$reg$num + 4;
 461   4              end;
 462   3              do;   /* 9 */
 463   4                  call set$mod;
 464   4                  rm$bits = op.base$reg$num * 2 + op.index$reg$num;
 465   4              end;
 466   3          end;
 467   2          end set$modrm;

 468   1      do$modrm: procedure (op$ptr);
 469   2          declare op$ptr address;
 470   2          op$base = op$ptr;
 471   2          call set$modrm;
 472   2          call put$instr;
 473   2          call put$mem (shl (mod$bits, 6) or shl (reg$bits, 3) or rm$bits);
 474   2          if disp$len > 0 then call put$mem (low (disp$word));
 476   2          if disp$len > 1 then call put$mem (high (disp$word));
 478   2          end do$modrm;

 479   1      modrm$w: procedure (ins, reg, op$ptr);
 480   2          declare (ins, reg) byte;
 481   2          declare op$ptr address;
 482   2          op$base = op$ptr;
 483   2          if op.modrm$type then
 484   2              do;
 485   3                  if op.w$bit = 0ffh then call ambig;
 487   3                  instr$1 = ins or op.w$bit;
 488   3                  reg$bits = reg;
```

```
489   3                 call do$modrm (op$ptr);
490   3             end;
491   2         else call error;
492   2         end modrm$w;

493   1     modrm$16: procedure (ins, reg);
494   2         declare (ins, reg) byte;
495   2         if op2.modrm$type and not op2.reg$8$type then
496   2             do;
497   3                 instr$1 = ins;
498   3                 reg$bits = reg;
499   3                 call do$modrm (.op2);
500   3             end;
501   2         else call error;
502   2         end modrm$16;

503   1     modregrm: procedure;
504   2         call check$ambig;
505   2         if op1.sub$type <= 2 then   /* first op is register */
506   2             do;
507   3                 reg$bits = op1.reg$num;
508   3                 call do$modrm (.op2);
509   3             end;
              else
510   2             do;
511   3                 instr$1 = instr$1 or op2.w$bit;
512   3                 reg$bits = op2.reg$num;
513   3                 call do$modrm (.op1);
514   3             end;
515   2         end modregrm;

516   1     modregrm$w: procedure;
517   2         call set$w$bit;
518   2         call modregrm;
519   2         end modregrm$w;

520   1     modregrm$dw: procedure;
521   2         if op1.modrm$type and (op1.sub$type <= 2) then instr$1 = instr$1 or 2;   /* set d
          -  bit */
523   2         call modregrm$w;
524   2         end modregrm$dw;

525   1     put$immed: procedure;
526   2         call put$mem (low (op2.value));
527   2         if op1.w$bit then call put$mem (high (op2.value));
529   2         end put$immed;

530   1     put$immed$s: procedure;
531   2         call put$mem (low (op2.value));
532   2         if (instr$1 and 3) = 1 then call put$mem (high (op2.value));
534   2         end put$immed$s;

535   1     check$w$and$val: procedure;
536   2         if op1.w$bit = 0 and op2.value > 255 then call error;
538   2         if op1.w$bit = 0ffh then
539   2             do;
540   3                 if op2.value >= 256 then op1.w$bit = 1;
```

```
542   3                 else call ambig;
543   3              end;
544   2          end check$w$and$val;

545   1      acc$immed: procedure;
546   2          call check$w$and$val;
547   2          call set$w$bit;
548   2          call put$instr;
549   2          call put$immed;
550   2          end acc$immed;

551   1      modrm$immed: procedure;
552   2          call check$w$and$val;
553   2          call set$w$bit;
554   2          call do$modrm (.op1);
555   2          call put$immed;
556   2          end modrm$immed;

557   1      set$instr$w: procedure (b);
558   2          declare b byte;
559   2          if op2.sub$type <= 2 then instr$1 = b or op2.w$bit;
561   2          else instr$1 = b or op1.w$bit;
562   2          end set$instr$w;

563   1      check$acc$immed: procedure logical;
564   2          return op1.acc$type and op2.num$type;
565   2          end check$acc$immed;

566   1      check$modrm$immed: procedure logical;
567   2          return op1.modrm$type and op2.num$type;
568   2          end check$modrm$immed;

569   1      check$modrm$modrm: procedure logical;
570   2          return op1.modrm$type and op2.modrm$type and not
                    (op1.sub$type > 2 and op2.sub$type > 2);
571   2          end check$modrm$modrm;

            /*************************************/
            /*                                   */
            /*   mnemonic processing procedures  */
            /*                                   */
            /*************************************/

572   1      type1: procedure;
               /*
                   mnemonics:  simple one-byte instructions
                   operands:   none
                   forms:      xxxxxxxx
               */
573   2          call get$operand$2;
574   2          if op2.null$type then call put$instr;
576   2          else call error;
577   2          end type1;

578   1      type2: procedure;
               /*
                   mnemonics:  aad  aam
```

```
                    operands:   none
                    forms:      xxxxxxxx    00001010
                */
579   2         call get$operand$2;
580   2         if op2.null$type then
581   2             do;
582   3                 call put$instr;
583   3                 call put$mem (0ah);
584   3             end;
585   2         else call error;
586   2         end type2;

587   1     type3: procedure;
                /*
                    mnemonics:  conditional jumps  loopxx  jmps
                    operands:   1   (number)
                    forms:      xxxxxxxx    IP-INC8
                */
588   2         call get$operand$2;
589   2         if op2.num$type and (op2.value - (assem$offset - 7eh) <= 255) then
590   2             do;
591   3                 call put$instr;
592   3                 call put$mem (low (op2.value - (assem$offset + 1)));
593   3             end;
594   2         else call error;
595   2         end type3;

596   1     type4: procedure;
                /*
                    mnemonics:  logical shifts and rotates
                    operands:   2
                    forms:      xxxxxxvw  mmxxxr/m  [disp-lo]  [disp-hi]
                */
597   2         call get$operand$1;
598   2         call get$operand$2;
599   2         if op1.modrm$type then
600   2             do;
601   3                 if op2.reg$8$type and (op2.reg$num = 1) then
602   3                   instr$1 = instr$1 or 2;   /* set v bit */
603   3                 else if not (op2.num$type and (op2.value = 1)) then call error;
                      call modrm$w (instr1, reg$bits, .op1);
606   3             end;
607   2         else call error;
608   2         end type4;

609   1     type5: procedure;
                /*
                    mnemonics:  div  idiv  mul  imul  not  neg
                    operands:   1
                    forms:      xxxxxxxw  mmxxxr/m  [disp-lo]  [disp-hi]
                */
610   2         call get$operand$2;
611   2         call modrm$w (instr$1, reg$bits, .op2);
612   2         end type5;

613   1     type6: procedure;
                /*
```

```
                    mnemonics:  les  lds  lea
                    operands:   2   (reg16, memory)
                    forms:      xxxxxxxx  mmregr/m  [disp-lo]  [disp-hi]
                */
614   2         call get$operand$1;
615   2         call get$operand$2;
616   2         if op1.reg$16$type and op2.modrm$type and (op2.sub$type > 2) then
617   2           call mod$reg$rm;
618   2         else call error;
619   2         end type6;

620   1     declare
            type$7$struc (2) structure (op$val (2) byte) initial (
                048h,   001h,   /* dec */
                040h,   000h);  /* inc */

621   1     type7: procedure;
                /*
                    mnemonics:  inc  dec
                    operands:   1
                    forms:      xxxxxreg
                                xxxxxxxw  mmxxxr/m  [disp-lo]  [disp-hi]
                */
622   2         call get$operand$2;
623   2         if op2.reg$16$type then call set$instr (type$7$struc (instr$1).op$val (0) or op2.r
          -   eg$num);
625   2         else call modrm$w (0feh, type$7$struc (instr$1).op$val (1), .op2);
626   2         end type7;

627   1     type8or8a: procedure;
628   2         if op2.seg$reg$type then call set$instr (type$8$struc (instr$1).op$val (0) or
                        shl (op2.seg$reg$num, 3));
630   2         else if op2.reg$16$type then call set$instr (type$8$struc (instr$1).op$val (1) or
          -   op2.reg$num);
632   2         else call modrm$16 (type$8$struc (instr$1).op$val (2),
                  type$8$struc (instr$1).op$val (3));
633   2         end type8or8a;

634   1     declare
            type$8$struc (2) structure (op$val (4) byte) initial (
                07h,    58h,    8fh,    00h,    /* pop */
                06h,    50h,    0ffh,   06h);   /* push */

635   1     type8: procedure;
                /*
                    mnemonics:  push
                    operands:   1
                    forms:      xxx$Rxxx
                                xxxxxreg
                                xxxxxxxx  mmxxxr/m  [disp-lo]  [disp-hi]
                */
636   2         call get$operand2;
637   2         call type8or8a;
638   2         end type8;

639   1     type8a: procedure;
                /*
```

```
                    mnemonics:  pop
                    operands:   1
                    forms:      xxxSRxxx
                                xxxxxreg
                                xxxxxxxx  mmxxxr/m  [disp-lo]  [disp-hi]
                */
640   2             call get$operand2;
641   2             if op2.seg$reg$type and (op2.seg$reg$num = 1) then call error;
643   2             else call type8or8a;
644   2             end type8a;

645   1         type9or10: procedure (type);
646   2             declare type byte;
647   2             call get$operand$1;
648   2             call get$operand$2;
649   2             if check$acc$immed then
650   2                 do;
651   3                     instr$1 = instr$1 or 4;
652   3                     call acc$immed;
653   3                 end;
654   2             else if check$modrm$immed then
655   2                 do;
656   3                     instr$1 = 80h;
657   3                     if type = 9 then
658   3                         do;
659   4                             if op2.value < 80h or op2.value >= 0ff80h then
660   4                               instr$1 = instr$1 or 2;
661   4                             call check$w$and$val;
662   4                             call set$w$bit;
663   4                             call do$modrm (.op1);
664   4                             call put$immed$s;
665   4                         end;
666   3                     else call modrm$immed;
667   3                 end;
668   2             else if check$modrm$modrm then call mod$reg$rm$dw;
670   2             else call error;
671   2             end type9or10;

672   1         type9: procedure;
                    /*
                        mnemonics:  add  adc  cmp  sub  sbb
                        operands:   2
                        forms:      xxxxxxxw  data-lo   data-hi (if w=1)
                                    xxxxxxsw  mmxxxr/m  [disp-lo]  [disp-hi]  data-lo  data-hi (if sw=
        -   01)
                                    xxxxxxdw  mmregr/m  [disp-lo]  [disp-hi]
                    */
673   2             call type9or10 (9);
674   2             end type9;

675   1         type10: procedure;
                    /*
                        mnemonics:  and  or  xor
                        operands:   2
                        forms:      xxxxxxxw  data-lo   data-hi (if w=1)
                                    xxxxxxxw  mmxxxr/m  [disp-lo]  [disp-hi]  data-lo  data-hi (if w=1
        -   )
```

```
                            xxxxxxdw  mmregr/m  [disp-lo]  [disp-hi]
        */
676   2         call type9or10 (10);
677   2         end type10;

678   1       declare type11$struc (2) structure (op$val (2) byte) initial (
                0e8h,   2,  /* call */
                0e9h,   4); /* jmp */

679   1       type11: procedure;
        /*
            mnemonics:  call  jmp
            operands:   1
            forms:      xxxxxxxx  IP-INC16
                        xxxxxxxx  mmxxxr/m  [disp-lo]  [disp-hi]
        */
680   2         call get$operand$2;
681   2         if op2.num$type then
682   2             do;
683   3               call set$instr (type11$struc (instr$1).op$val (0));
684   3               call put$word (op2.value - (assem$offset + 2));
685   3             end;
686   2         else call modrm$16 (0ffh, type$11$struc (instr$1).op$val (1));
687   2         end type11;

688   1       type$int: procedure;
689   2         call get$operand$2;
690   2         if op2.num$type and (op2.value <= 255) then
691   2             do;
692   3               if op2.value = 3 then call set$instr (0cch);
                    else
694   3                   do;
695   4                     call set$instr (0cdh);
696   4                     call put$mem (low (op2.value));
697   4                   end;
698   3             end;
699   2         else call error;
700   2         end type$int;

701   1       type$esc: procedure;
702   2         call get$operand$1;
703   2         call get$operand$2;
704   2         if op1.num$type and (op1.value < 64) and op2.modrm$type then
705   2             do;
706   3               instr$1 = instr$1 or shr (low (op1.value), 3);
707   3               reg$bits = low (op1.value) and 7;
708   3               call do$modrm (.op2);
709   3             end;
710   2         else call error;
711   2         end type$esc;

712   1       type$ret$retf: procedure;
713   2         op$base = .op2;
714   2         call get$operand;
715   2         if op2.null$type then call set$instr (instr$1 or 1);
717   2         else if op2.num$type then
718   2             do;
```

```
719   3                 call put$instr;
720   3                 call put$word (op2.value);
721   3             end;
722   2         else call error;
723   2         end type$ret$retf;

724   1     type$in: procedure;
725   2         call get$operand$1;
726   2         if not op1.acc$type then call error;
728   2         call get$operand$2;
729   2         if op2.reg$16$type and (op2.reg$num = 2) /* DX */ then
730   2           call set$instr (instr$1 or 8 or op1.w$bit);
731   2         else if op2.num$type and (op2.value <= 255) then
732   2             do;
733   3                 call put$instr;
734   3                 call put$mem (low (op2.value));
735   3             end;
736   2         else call error;
737   2         end type$in;

738   1     type$out: procedure;
739   2         declare temp byte;
740   2         call get$operand$1;
741   2         if op1.reg$16$type and (op1.reg$num = 2) /* DX */ then temp = 1;
743   2         else if op1.num$type and (op1.value <= 255) then temp = 0;
745   2         else call error;
746   2         call get$operand$2;
747   2         if not op2.acc$type then call error;
749   2         call put$mem (instr$1 or shl (temp,3) or op2.w$bit);
750   2         if temp = 0 then call put$mem (low (op1.value));
752   2         end type$out;

753   1     declare cjstruc (2) structure (op$val (2) byte) initial (
              09ah,   3,  /* callf */
              0eah,   5); /* jmpf */

754   1     type$callf$jmpf: procedure;
755   2         call get$operand$2;
756   2         if op2.far$type then
757   2             do;
758   3                 call set$instr (cjstruc (instr$1).op$val (0));
759   3                 call put$word (op2.value);
760   3                 call put$word (op2.seg$value);
761   3             end;
762   2         else call modrm$16 (0ffh, cj$struc (instr$1).op$val (1));
763   2         end type$callf$jmpf;

764   1     type$test: procedure;
765   2         call get$operand$1;
766   2         call get$operand$2;
767   2         if check$acc$immed then
768   2             do;
769   3                 instr$1 = 0a8h;
770   3                 call acc$immed;
771   3             end;
772   2         else if check$modrm$immed then
773   2             do;
```

```
774   3                 instr$1 = 0f6h;
775   3                 call modrm$immed;
776   3             end;
777   2         else if check$modrm$modrm then
778   2             do;
779   3                 call set$instr$w (84h);
780   3                 call mod$reg$rm;
781   3             end;
782   2         else call error;
783   2         end type$test;

784   1     type$xchg: procedure;
785   2         call get$operand$1;
786   2         call get$operand$2;
787   2         if op1.acc$type and op1.w$bit and op2.reg$16$type then
788   2           call set$instr (90h + op2.reg$num);
789   2         else if check$modrm$modrm then
790   2             do;
791   3                 call set$instr$w (86h);
792   3                 call modregrm;
793   3             end;
794   2         else call error;
795   2         end type$xchg;

796   1     type$mov: procedure;
797   2         call get$operand$1;
798   2         call get$operand$2;
799   2         if op1.seg$reg$type and op2.modrm$type and not op2.reg$8$type then
800   2             do;
801   3                 instr$1 = 8eh;
802   3                 reg$bits = op1.seg$reg$num;
803   3                 call do$modrm (.op2);
804   3             end;
805   2         else if op1.modrm$type and not (op1.sub$type = 1) and op2.seg$reg$type then
806   2             do;
807   3                 instr$1 = 8ch;
808   3                 reg$bits = op2.seg$reg$num;
809   3                 call do$modrm (.op1);
810   3             end;
811   2         else if op1.acc$type and op2.modrm$type and (op2.subtype = 3) then
812   2             do;
813   3                 instr$1 = 0a0h or op1.w$bit;
814   3                 call put$instr;
815   3                 call put$word (op2.value);
816   3             end;
817   2         else if op2.acc$type and op1.modrm$type and (op1.sub$type = 3) then
818   2             do;
819   3                 instr$1 = 0a2h or op2.w$bit;
820   3                 call put$instr;
821   3                 call put$word (op1.value);
822   3             end;
823   2         else if check$modrm$immed then
824   2             do;
825   3                 if op1.sub$type <= 2 then
826   3                     do;
827   4                         call set$instr (0b0h or shl (op1.w$bit, 3) or op1.reg$num);
828   4                         call put$immed;
```

```
829   4                     end;
                        else
830   3                     do;
831   4                         instr$1 = 0c6h;
832   4                         call modrm$immed;
833   4                     end;
834   3               end;
835   2           else if check$modrm$modrm then
836   2               do;
837   3                   instr$1 = 88h;
838   3                   call mod$reg$rm$dw;
839   3               end;
840   2           else call error;
841   2           end type$mov;

842   1       asm: procedure (loc) address public;
843   2           declare loc pointer;
844   2           assem$ptr = loc;
845   2           call crlf;
846   2           call print$word (assem$segment);
847   2           call conout (":");
848   2           call print$word (assem$offset);
849   2           call conout (" ");
850   2           call get$line;
851   2           op$base = .op1;   /* must be set for prefix scan */
852   2           if get$opcode then
853   2               do;
854   3                   if op$type > 21 then call error;
856   3                   do case optype;
857   4                       call error;
858   4                       call type1;
859   4                       call type2;
860   4                       call type3;
861   4                       call type4;
862   4                       call type5;
863   4                       call type6;
864   4                       call type7;
865   4                       call type8;
866   4                       call type9;
867   4                       call type10;
868   4                       call type11;
869   4                       call type$int;
870   4                       call type$esc;
871   4                       call type$ret$retf;
872   4                       call type$in;
873   4                       call type$out;
874   4                       call type$callf$jmpf;
875   4                       call type$test;
876   4                       call type$xchg;
877   4                       call type$mov;
878   4                       call type8a;
879   4                   end;
880   3               end;
881   2           else call put$prefix;   /* may be prefix without opcode */
882   2           return assem$offset;
883   2           end asm;
```

```
884   1       end ddtasm;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES							
34	0000H	2	A.	WORD PARAMETER	35					
52	0008H	2	A.	WORD PARAMETER AUTOMATIC	53	54	56	58		
23	0000H	2	A.	WORD PARAMETER	24					
402	0004H	2	A.	WORD PARAMETER AUTOMATIC	403	404	405			
54	0000H	1	A1	BYTE BASED(A)	56					
545	09CAH	17	ACCIMMED	PROCEDURE STACK=0016H	652	770				
7	0008H	1	ACCTYPE.	BYTE MEMBER(OP)	159	171	255			
7	0008H	1	ACCTYPE.	BYTE MEMBER(OP2)	747	817				
7	0008H	1	ACCTYPE.	BYTE MEMBER(OP1)	564	726	787	811		
42	0031H	15	AMBIG.	PROCEDURE STACK=0006H	48	486	542			
842	0FCBH	248	ASM.	PROCEDURE WORD PUBLIC STACK=0036H						
5	0000H	1	ASSEMBYTE.	BYTE BASED(ASSEMPTR)						
13	0000H	2	ASSEMOFF	WORD PARAMETER	14					
5	0002H	2	ASSEMOFFSET. . . .	WORD AT	382	383	589	592	684	848	882
5	0002H	4	ASSEMPTR	POINTER	5	844					
5	0004H	2	ASSEMSEGMENT . . .	WORD AT	846						
72	0004H	1	B.	BYTE PARAMETER AUTOMATIC	73	74					
52	0006H	2	B.	WORD PARAMETER AUTOMATIC	53	54	56	59			
13	0000H	1	B.	BYTE PARAMETER	14						
557	0004H	1	B.	BYTE PARAMETER AUTOMATIC	558	560	561				
63	0004H	1	B.	BYTE PARAMETER AUTOMATIC	64	67					
397	0004H	1	B.	BYTE PARAMETER AUTOMATIC	398	399					
20	0000H	1	B.	BYTE PARAMETER	21						
380	0004H	1	B.	BYTE PARAMETER AUTOMATIC	381	382					
76	0004H	1	B.	BYTE PARAMETER AUTOMATIC	77	78	79	80			
193	0004H	1	B.	BYTE PARAMETER AUTOMATIC	194	195					
54	0000H	1	B1	BYTE BASED(B)	56						
6	0000H	4	BASEREG.	BYTE ARRAY(4) DATA	141						
7	0002H	1	BASEREGNUM	BYTE MEMBER(OP2)							
7	0002H	1	BASEREGNUM	BYTE MEMBER(OP1)							
7	0002H	1	BASEREGNUM	BYTE MEMBER(OP)	141	434	452	456	464		
7	000DH	1	BWPREFIX	BYTE MEMBER(OP2)							
7	000DH	1	BWPREFIX	BYTE MEMBER(OP1)							
7	000DH	1	BWPREFIX	BYTE MEMBER(OP)	182	184	185	221	256		
52	0004H	1	C.	BYTE PARAMETER AUTOMATIC	53	55					
563	0A11H	12	CHECKACCIMMED. . .	PROCEDURE BYTE STACK=0002H	649	767					
46	0040H	33	CHECKAMBIG	PROCEDURE STACK=000EH	504						
140	01F1H	23	CHECKBASEREG . . .	PROCEDURE BYTE STACK=000CH	229						
178	02E3H	62	CHECKBWPREFIX. . .	PROCEDURE BYTE STACK=0002H	217						
143	0208H	23	CHECKINDEXREG. . .	PROCEDURE BYTE STACK=000CH	246						
566	0A1DH	12	CHECKMODRMIMMED. .	PROCEDURE BYTE STACK=0002H	654	772	823				
569	0A29H	42	CHECKMODRMMODRM. .	PROCEDURE BYTE STACK=0006H	668	777	789				
			835								
166	0295H	78	CHECKREG16	PROCEDURE BYTE STACK=000CH	264						
154	0247H	78	CHECKREG8.	PROCEDURE BYTE STACK=000CH	262						
190	0321H	22	CHECKSEGPREFIX . .	PROCEDURE BYTE STACK=0010H	215	356					
146	021FH	40	CHECKSEGREG. . . .	PROCEDURE BYTE STACK=000CH	191	260					

535	099AH	48	CHECKWANDVAL . . .	PROCEDURE STACK=000EH	546	552	661					
753	001EH	4	CJSTRUC.	STRUCTURE ARRAY(2) INITIAL	758	762						
52	0061H	45	COMPARE.	PROCEDURE BYTE STACK=0008H	340							
18	0000H		CONIN.	PROCEDURE BYTE EXTERNAL(10) STACK=0000H								88
				114								
20	0000H		CONOUT	PROCEDURE EXTERNAL(11) STACK=0000H							27	28
				39 847 849								
2			CR	LITERALLY	10	27	43	211	318	354		
26	0002H	17	CRLF	PROCEDURE STACK=0006H	38	845						
1	0002H		DDTASM	PROCEDURE STACK=0000H								
16	0000H		DDTGETLINE	PROCEDURE EXTERNAL(9) STACK=0000H							31	
13	0000H		DDTSET	PROCEDURE EXTERNAL(8) STACK=0000H							382	
63	008EH	46	DELIMITER.	PROCEDURE BYTE STACK=0004H	93							
10	0008H	6	DELIMITERS	BYTE ARRAY(6) DATA	66	67						
4			DELIMTYPE.	LITERALLY	97	209						
9	0078H	1	DISPLEN.	BYTE	413	414	417	431	438	474	476	
9	0008H	2	DISPWORD	WORD	418	439	475	477				
468	0859H	74	DOMODRM.	PROCEDURE STACK=0018H	489	499	508	513				
				554 663 708 803 809								
37	0020H	17	ERROR.	PROCEDURE STACK=000AH	50	92	200	220				
				243 252 281 289 300 310 319 332 337 364								
				371 377 491 501 537 576 585 594 604 607								
				618 642 670 699 710 722 727 736 745 748								
				782 794 840 855 857								
2			FALSE.	LITERALLY	57	70	112	138	152	164	176	
				188 204 255 347 351 355								
7	000AH	1	FARTYPE.	BYTE MEMBER(OP1)								
7	000AH	1	FARTYPE.	BYTE MEMBER(OP2)	756							
7	000AH	1	FARTYPE.	BYTE MEMBER(OP)	255	283						
30	0013H	13	GETLINE.	PROCEDURE STACK=0004H	850							
349	066EH	126	GETOPCODE.	PROCEDURE BYTE STACK=0016H	852							
254	0445H	295	GETOPERAND	PROCEDURE STACK=001AH	307	314	714					
305	056CH	27	GETOPERAND1. . . .	PROCEDURE STACK=001EH	597	614	647	702				
				725 740 765 785 797								
312	0587H	40	GETOPERAND2. . . .	PROCEDURE STACK=001EH	573	579	588	598				
				610 615 622 636 640 648 680 689 703 728								
				746 755 766 786 798								
206	037EH	80	GETPREFIX.	PROCEDURE STACK=0016H	257							
117	01A0H	24	GETREALTOKEN . . .	PROCEDURE STACK=0012H	208	353						
82	00F8H	168	GETTOKEN	PROCEDURE STACK=000EH	119	203	231	241				
				248 271 272 278 279 290 294 308 317								
11	0000H		GOODT.	PROCEDURE EXTERNAL(7) STACK=0000H							40	44
76	00E1H	23	HEX.	PROCEDURE BYTE STACK=0004H	111							
			HIGH	BUILTIN	405	477	528	533				
335	007CH	1	I.	BYTE	339	342						
129	007BH	1	I.	BYTE	130	133						
65	0079H	1	I.	BYTE	66	67						
322	000AH	2	I.	WORD	323	324	326	327	328			
386	007DH	1	I.	BYTE	387	389	390					
6	0004H	4	INDEXREG	BYTE ARRAY(4) DATA	144							
7	0003H	1	INDEXREGNUM. . . .	BYTE MEMBER(OP2)								
7	0003H	1	INDEXREGNUM. . . .	BYTE MEMBER(OP1)								
7	0003H	1	INDEXREGNUM. . . .	BYTE MEMBER(OP)	144	448	452	460	464			
479	0008H	1	INS.	BYTE PARAMETER AUTOMATIC	480	487						
493	0006H	1	INS.	BYTE PARAMETER AUTOMATIC	494	497						
9	0077H	1	INSTR1	BYTE	327	366	373	395	399	409	487	497
				511 522 532 560 561 602 605 611 624 625								

```
                                        629  631  632  651  656  660  683  686  706  716
                                        730  749  758  762  769  774  801  807  813  819
                                        831  837
                    LAST . . . . . .    BUILTIN          323
   10   000EH       1   LEFTBRACKET. . . .  BYTE DATA        269  288
                    LENGTH . . . . . .  BUILTIN            66   91
    2                   LF . . . . . .  LITERALLY          28   43
  842   0004H       4   LOC. . . . . .  POINTER PARAMETER AUTOMATIC        843  844
    8   0070H       1   LOCKFLAG . . . . .  BYTE          351  363  365
    2                   LOGICAL. . . . . .  LITERALLY        7   52   63   72   83  124  140
                                        143  146  154  166  178  190  334  349  563  566
                                        569
  321   05AFH      68   LOOKUPOPTAB. . . .  PROCEDURE STACK=000EH          360
  124   0188H      57   LOOKUPREG. . . . .  PROCEDURE BYTE STACK=0008H      141  144  147
                                        155  167
                    LOW. . . . . .      BUILTIN          404  475  526  531  592  696  706
                                        707  734  751
  335   0000H       1   MNEMONIC . . . . .  BYTE BASED(TABLEBASE)          340
    9   0076H       1   MNEMONICINDEX. . .  BYTE          324  342
    9   0072H       1   MODBITS. . . . . .  BYTE          413  414  421  425  429  436  442  447
                                        451  473
  503   090CH      46   MODREGRM . . . . .  PROCEDURE STACK=001CH          518  617  780  792
  520   0945H      27   MODREGRMDW . . . .  PROCEDURE STACK=0024H          669  838
  516   093AH      11   MODREGRMW. . . . .  PROCEDURE STACK=0020H          523
  493   08E0H      44   MODRM16. . . . . .  PROCEDURE STACK=0020H          632  686  762
  551   09D8H      21   MODRMIMMED . . . .  PROCEDURE STACK=001CH          666  775  832
    7   0008H       1   MODRMTYPE. . . . .  BYTE MEMBER(OP2)          495  570  616  704  799
                                        811
    7   0008H       1   MODRMTYPE. . . . .  BYTE MEMBER(OP1)          521  567  570  599  805
                                        817
    7   0008H       1   MODRMTYPE. . . . .  BYTE MEMBER(OP)       157  169  228  234  255  298
                                        483
  479   08A3H      61   MODRMW . . . . . .  PROCEDURE STACK=0022H          605  611  625
    4   0044H       1   NEXTCH . . . . . .  BYTE           32   87   88   90   93   98  110  111
                                        114  191  269  275
    3   0000H       5   NOPS . . . . . .    BYTE ARRAY(5) EXTERNAL(1)     339
    8   006EH       1   NPREFIX. . . . . .  BYTE          195  196  350  388
    7   0009H       1   NULLTYPE . . . . .  BYTE MEMBER(OP1)
    7   0009H       1   NULLTYPE . . . . .  BYTE MEMBER(OP2)          574  580  715
    7   0009H       1   NULLTYPE . . . . .  BYTE MEMBER(OP)       212  255  258  315
   72   00BCH      37   NUMBER . . . . . .  PROCEDURE BYTE STACK=0006H      110
    4                   NUMBERTYPE . . . .  LITERALLY        103  266  280  291
   10                   NUMDELIMS. . . . .  LITERALLY         66
   83   007AH       1   NUMERIC. . . . . .  BYTE           85  102  108  112
    7   0007H       1   NUMTYPE. . . . . .  BYTE MEMBER(OP2)          564  567  589  603  681
                                        690  717  731
    7   0007H       1   NUMTYPE. . . . . .  BYTE MEMBER(OP1)          704  743
    7   0007H       1   NUMTYPE. . . . . .  BYTE MEMBER(OP)       255  285
    7   0000H      19   OP . . . . . . . .  STRUCTURE BASED(OPBASE)      141  144  147  149
                                        155  157  158  159  160  161  167  169  170  171
                                        172  173  182  184  185  201  204  212  219  221
                                        228  234  235  240  245  255  256  258  268  277
                                        282  283  285  293  297  298  315  412  418  419
                                        422  426  434  448  452  456  460  464  483  485
                                        487
    7   0045H      19   OP1. . . . . . . .  STRUCTURE         47   49  306  408  505  507  513
                                        521  527  536  538  541  554  561  564  567  570
```

```
                                        599  605  616  663  704  706  707  726  730  741
                                        743  751  787  799  802  805  809  811  813  817
                                        821  825  827  851
    7  0058H    19  OP2. . . . . .    STRUCTURE          47   49  313  408  495  499  508
                                        511  512  526  528  531  533  536  540  559  560
                                        564  567  570  574  580  589  592  601  603  611
                                        616  623  624  625  628  629  630  631  641  659
                                        681  684  690  692  696  704  708  713  715  717
                                        720  729  731  734  747  749  756  759  760  787
                                        788  799  803  305  808  811  815  817  819
    7  0006H     2  OPBASE . . . . .  WORD             7  141  144  147  149  155  157  158
                                        159  160  161  167  169  170  171  172  173  182
                                        184  185  201  204  212  219  221  228  234  235
                                        240  245  255  256  258  268  277  282  283  285
                                        293  297  298  306  313  315  412  418  419  422
                                        426  434  448  452  456  460  464  470  482  483
                                        485  487  713  851
    3  0000H     6  OPNIN. . . . . .  BYTE ARRAY(6) EXTERNAL(2)                342
  479  0004H     2  OPPTR. . . . . .  WORD PARAMETER AUTOMATIC                 481  482  489
  468  0004H     2  OPPTR. . . . . .  WORD PARAMETER AUTOMATIC                 469  470
    3  0000H   480  OPTAB. . . . . .  BYTE ARRAY(480) EXTERNAL(3)                    323  324  326
                                        327  328
    9  0075H     1  OPTYPE . . . . .  BYTE          326  361  368  854  856
  753  0000H     2  OPVAL. . . . . .  BYTE ARRAY(2) MEMBER(CJSTRUC)             758  762
  634  0000H     4  OPVAL. . . . . .  BYTE ARRAY(4) MEMBER(TYPE8STRUC)                629  631
                                        632
  620  0000H     2  OPVAL. . . . . .  BYTE ARRAY(2) MEMBER(TYPE7STRUC)                624  625
  678  0000H     2  OPVAL. . . . . .  BYTE ARRAY(2) MEMBER(TYPE11STRUC)               683  686
    8  0068H     3  PREFIX . . . . .  BYTE ARRAY(3)          195  389
   23  0000H        PRINTM . . . . .  PROCEDURE EXTERNAL(12) STACK=0000H             43
   34  0000H        PRINTWORD. . . . .  PROCEDURE EXTERNAL(13) STACK=0000H           846  848
  525  0960H    28  PUTIMMED . . . . .  PROCEDURE STACK=000EH              549  555  828
  530  097CH    30  PUTIMMEDS. . . . .  PROCEDURE STACK=000EH              664
  393  072AH    15  PUTINSTR . . . . .  PROCEDURE STACK=0012H              400  472  548  575
                                        582  591  719  733  814  820
  380  06ECH    21  PUTMEM . . . . . .  PROCEDURE STACK=000AH              389  395  404  405
                                        473  475  477  526  528  531  533  583  592  696
                                        734  749  751
  385  0701H    41  PUTPREFIX. . . . .  PROCEDURE STACK=000EH              394  881
  402  0749H    23  PUTWORD. . . . . .  PROCEDURE STACK=0010H              684  720  759  760
                                        815  821
  493  0004H     1  REG. . . . . . .  BYTE PARAMETER AUTOMATIC              494  498
  479  0006H     1  REG. . . . . . .  BYTE PARAMETER AUTOMATIC              480  488
    6  0000H    16  REG16. . . . . .  BYTE ARRAY(16) EXTERNAL(5)                 167
    7  0005H     1  REG16TYPE. . . . .  BYTE MEMBER(OP2)              623  630  729  787
    7  0005H     1  REG16TYPE. . . . .  BYTE MEMBER(OP1)              616  741
    7  0005H     1  REG16TYPE. . . . .  BYTE MEMBER(OP)          169  255
    6  0000H    16  REG8 . . . . . .  BYTE ARRAY(16) EXTERNAL(6)                 155
    7  0006H     1  REG8TYPE . . . . .  BYTE MEMBER(OP2)              495  601  799
    7  0006H     1  REG8TYPE . . . . .  BYTE MEMBER(OP1)
    7  0006H     1  REG8TYPE . . . . .  BYTE MEMBER(OP)          157  255
    9  0073H     1  REGBITS. . . . . .  BYTE          328  473  488  498  507  512  605  611
                                        707  802  808
  226  03CEH   119  REGINDIRECT. . . .  PROCEDURE STACK=0014H              273  302
    7  0001H     1  REGNUM . . . . . .  BYTE MEMBER(OP2)              512  601  624  631  729
                                        788
  127  0000H     1  REGNUM . . . . . .  BYTE BASED(REGNUMPTR)              133
```

```
   7  0001H     1  REGNUM . . . . . .  BYTE MEMBER(OP1)          507  741  827
   7  0001H     1  REGNUM . . . . . .  BYTE MEMBER(OP)      155  158  167  170  422  426
 124  0004H     2  REGNUMPTR. . . . .  WORD PARAMETER AUTOMATIC      125  127  133
 124  0008H     2  REGTABPTR. . . . .  WORD PARAMETER AUTOMATIC      125  126  131  136

 126  0000H     2  REGWORD. . . . . .  WORD BASED(REGTABPTR)        131
   8  0071H     1  REPFLAG. . . . . .  BYTE        351  370  372
  10  000FH     1  RIGHTBRACKET . . .  BYTE DATA        232  249  295
   9  0074H     1  RMBITS . . . . . .  BYTE        422  426  430  437  443  448  452  456
                                        460  464  473
   8  006FH     1  SEGPREFIXFLAG. . .  BYTE        199  202  351
   6  0000H     8  SEGREG . . . . . .  BYTE ARRAY(8) EXTERNAL(4)      147
   7  0000H     1  SEGREGNUM. . . . .  BYTE MEMBER(OP2)          629  641  808
   7  0000H     1  SEGREGNUM. . . . .  BYTE MEMBER(OP1)          802
   7  0000H     1  SEGREGNUM. . . . .  BYTE MEMBER(OP)      147  201
   7  0004H     1  SEGREGTYPE . . . .  BYTE MEMBER(OP2)          628  641  805
   7  0004H     1  SEGREGTYPE . . . .  BYTE MEMBER(OP1)          799
   7  0004H     1  SEGREGTYPE . . . .  BYTE MEMBER(OP)      149  204  255
   7  000FH     2  SEGVALUE . . . . .  WORD MEMBER(OP2)          760
   7  000FH     2  SEGVALUE . . . . .  WORD MEMBER(OP1)
   7  000FH     2  SEGVALUE . . . . .  WORD MEMBER(OP)      277
 397  0739H    16  SETINSTR . . . . .  PROCEDURE STACK=0018H      624  629  631  683
                                        693  695  716  730  758  788  827
 557  09F0H    33  SETINSTRW. . . . .  PROCEDURE STACK=0004H      779  791
 411  0775H    37  SETMOD . . . . . .  PROCEDURE STACK=0002H      455  459  463
 416  079AH   191  SETMODRM . . . . .  PROCEDURE STACK=0006H      471
 193  0337H    24  SETPREFIX. . . . .  PROCEDURE STACK=0004H      201  366  373
 198  034FH    47  SETSEGPREFIX . . .  PROCEDURE STACK=0012H      216  357
 407  0760H    21  SETWBIT. . . . . .  PROCEDURE STACK=0002H      517  547  553  662
              SHL. . . . . .  BUILTIN          111  201  473  629  749  827
              SHR. . . . . .  BUILTIN          706
   4          STRINGTYPE . . . .  LITERALLY        104  179
   7  000EH     1  SUBTYPE. . . . . .  BYTE MEMBER(OP2)          559  570  616  811
   7  000EH     1  SUBTYPE. . . . . .  BYTE MEMBER(OP1)          505  521  570  805  817
                                        825
   7  000EH     1  SUBTYPE. . . . . .  BYTE MEMBER(OP)      160  172  235  240  245  297
                                        419
   2          TAB. . . . . .  LITERALLY         10  120
 335  000CH     2  TABLEBASE. . . . .  WORD        335  338  340  345
 124  0006H     1  TABLEN . . . . . .  BYTE PARAMETER AUTOMATIC      128  130
   3  0000H    10  TABPTRS. . . . . .  WORD ARRAY(5) EXTERNAL(0)     338
 739  007EH     1  TEMP . . . . . .  BYTE        742  744  749  750
   4  0022H    32  TOKEN. . . . . .  BYTE ARRAY(32)          4   90   91  120  211  232
                                        238  249  288  295  309  318  340  354
   4  0042H     1  TOKENLEN . . . . .  BYTE         86   90   91   95  101  336  338  339
                                        340  342  345
   4  0043H     1  TOKENTYPE. . . . .  BYTE         97  103  104  179  209  266  280  291
   4  0000H     2  TOKENVALUE . . . .  WORD         84  111  268  277  282  293
   4  0022H     2  TOKENWORD. . . . .  WORD AT          131  181  183
   2          TRUE . . . . . .  LITERALLY          61   68   85   89  118  134  149
                                        150  157  159  162  169  171  174  186  202  207
                                        212  228  234  283  285  298  343  352  365  372
                                        375
 645  0004H     1  TYPE . . . . . .  BYTE PARAMETER AUTOMATIC      646  657
 572  0A53H    23  TYPE1. . . . . .  PROCEDURE STACK=0022H      858
 675  0C7EH    11  TYPE10 . . . . .  PROCEDURE STACK=002EH      867
 679  0C89H    68  TYPE11 . . . . .  PROCEDURE STACK=0024H      868
```

```
 678  001AH    4  TYPE11STRUC. . . .  STRUCTURE ARRAY(2) INITIAL          683  686
 578  0A6AH   29  TYPE2. . . . . .    PROCEDURE STACK=0022H          859
 587  0A87H   55  TYPE3. . . . . .    PROCEDURE STACK=0022H          860
 596  0ABEH   82  TYPE4. . . . . .    PROCEDURE STACK=0026H          861
 609  0B10H   23  TYPE5. . . . . .    PROCEDURE STACK=0026H          862
 613  0B27H   37  TYPE6. . . . . .    PROCEDURE STACK=0022H          863
 621  0B4CH   61  TYPE7. . . . . .    PROCEDURE STACK=0026H          864
 620  000EH    4  TYPE7STRUC . . . .  STRUCTURE ARRAY(2) INITIAL          624  625
 635  0BDFH   11  TYPE8. . . . . .    PROCEDURE STACK=0028H          865
 639  0BEAH   30  TYPE8A . . . . . .  PROCEDURE STACK=0028H          878
 627  0B89H   86  TYPE8OR8A. . . . .  PROCEDURE STACK=0024H          637  643
 634  0012H    8  TYPE8STRUC . . . .  STRUCTURE ARRAY(2) INITIAL          629  631  632

 672  0C73H   11  TYPE9. . . . . .    PROCEDURE STACK=002EH          866
 645  0C08H  107  TYPE9OR10. . . . .  PROCEDURE STACK=002AH          673  676
 754  0E2EH   66  TYPECALLFJMPF. . .  PROCEDURE STACK=0024H          874
 701  0D05H   59  TYPEESC. . . . . .  PROCEDURE STACK=0022H          870
 724  0D76H   82  TYPEIN . . . . . .  PROCEDURE STACK=0022H          872
 688  0CCDH   56  TYPEINT. . . . . .  PROCEDURE STACK=0022H          869
 796  0EECH  223  TYPEMOV. . . . . .  PROCEDURE STACK=0028H          877
 738  0DC8H  102  TYPEOUT. . . . . .  PROCEDURE STACK=0022H          873
 712  0D40H   54  TYPERETRETF. . . .  PROCEDURE STACK=001EH          871
 764  0E70H   66  TYPETEST . . . . .  PROCEDURE STACK=0022H          875
 784  0EB2H   58  TYPEXCHG . . . . .  PROCEDURE STACK=0022H          876
 334  05F3H  123  VALIDMNEMONIC. . .  PROCEDURE BYTE STACK=000EH          358
   7  0011H    2  VALUE. . . . . .   WORD MEMBER(OP2)          526  528  531  533  536
                                       540  589  592  603  659  684  690  692  696  720
                                       731  734  759  815
   7  0011H    2  VALUE. . . . . .   WORD MEMBER(OP1)          704  706  707  743  751
                                       821
   7  0011H    2  VALUE. . . . . .   WORD MEMBER(OP)      268  282  293  412  418
   7  000CH    1  WBIT . . . . . .   BYTE MEMBER(OP2)          47   49  408  511  560
                                       749  819
   7  000CH    1  WBIT . . . . . .   BYTE MEMBER(OP1)          47   49  408  527  536
                                       538  541  561  730  787  813  827
   7  000CH    1  WBIT . . . . . .   BYTE MEMBER(OP)      161  173  219  221  256  485
                                       487
 226  0004H    1  X. . . . . . . .   BYTE PARAMETER AUTOMATIC       227  235  240  245
```


MODULE INFORMATION:

```
  CODE AREA SIZE     = 10C3H   4291D
  CONSTANT AREA SIZE = 0024H     36D
  VARIABLE AREA SIZE = 007FH    127D
  MAXIMUM STACK SIZE = 0036H     54D
  1130 LINES READ
  0 PROGRAM ERROR(S)
```

END OF PL/M-86 COMPILATION

```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE ASMTAB
OBJECT MODULE PLACED IN ASMTAB.OBJ
COMPILER INVOKED BY:  PLM86 ASMTAB.PLM DEBUG XREF PAGEWIDTH(100)


            $title('tables for ddt86 assembler')
            $date(6/15/81)

  1         asmtab: do;

            $nolist

  4  1      declare op$tab (*) byte public data (
                aaa$in, 1,  037h,   0,
                aad$in, 2,  0d5h,   0ah,
                aam$in, 2,  0d4h,   0ah,
                aas$in, 1,  03fh,   0,
                adc$in, 9,  10h,    2,
                add$in, 9,  0,  0,
                and$in, 10, 20h,    4,
                call$in,    11, 0,  0,
                callf$in,   17, 0,  0,
                cbw$in, 1,  098h,   0,
                clc$in, 1,  0f8h,   0,
                cld$in, 1,  0fch,   0,
                cli$in, 1,  0fah,   0,
                cmc$in, 1,  0f5h,   0,
                cmp$in, 9,  38h,    7,
                cmpsb$in,   1,  0a6h,   0,
                cmpsw$in,   1,  0a7h,   0,
                cwd$in, 1,  099h,   0,
                daa$in, 1,  027h,   0,
                das$in, 1,  02fh,   0,
                dec$in, 7,  0,  0,
                div$in, 5,  0f6h,   6,
                esc$in, 13, 0d8h,   0,
                hlt$in, 1,  0f4h,   0,
                idiv$in,    5,  0f6h,   7,
                imul$in,    5,  0f6h,   5,
                in$in,  15, 0e4h,   0,
                inc$in, 7,  1,  0,
                int$in, 12, 0,  0,
                into$in,    1,  0ceh,   0,
                iret$in,    1,  0cfh,   0,
                ja$in,  3,  077h,   0,
                jae$in, 3,  073h,   0,
                jb$in,  3,  072h,   0,
                jbe$in, 3,  076h,   0,
                jc$in,  3,  072h,   0,
                jcxz$in,    3,  0e3h,   0,
                je$in,  3,  074h,   0,
                jg$in,  3,  07fh,   0,
                jge$in, 3,  07dh,   0,
                jl$in,  3,  07ch,   0,
                jle$in, 3,  07eh,   0,
                jmp$in, 11, 1,  0,
```

```
jmpf$in,     17, 1,  0,
jmps$in,     3,  0ebh,    0,
jna$in, 3,   076h,   0,
jnae$in,     3,  072h,    0,
jnb$in, 3,   073h,   0,
jnbe$in,     3,  077h,    0,
jnc$in, 3,   073h,   0,
jne$in, 3,   075h,   0,
jng$in, 3,   07eh,   0,
jnge$in,     3,  07ch,    0,
jnl$in, 3,   07dh,   0,
jnle$in,     3,  07fh,    0,
jno$in, 3,   071h,   0,
jnp$in, 3,   07bh,   0,
jns$in, 3,   079h,   0,
jnz$in, 3,   075h,   0,
jo$in,  3,   070h,   0,
jp$in,  3,   07ah,   0,
jpe$in, 3,   07ah,   0,
jpo$in, 3,   07bh,   0,
js$in,  3,   078h,   0,
jz$in,  3,   074h,   0,
lahf$in,     1,  09fh,    0,
lds$in, 6,   0c5h,   0,
lea$in, 6,   08dh,   0,
les$in, 6,   0c4h,   0,
lock$in,     0ffh,   0f0h,    0,
lodsb$in,    1,  0ach,    0,
lodsw$in,    1,  0adh,    0,
loop$in,     3,  0e2h,    0,
loope$in,    3,  0e1h,    0,
loopne$in,   3,  0e0h,    0,
loopnz$in,   3,  0e0h,    0,
loopz$in,    3,  0e1h,    0,
mov$in, 20,  0,  0,
movsb$in,    1,  0a4h,    0,
movsw$in,    1,  0a5h,    0,
mul$in, 5,   0f6h,   4,
neg$in, 5,   0f6h,   3,
nop$in, 1,   090h,   0,
not$in, 5,   0f6h,   2,
or$in,  10,  8,  1,
out$in, 16,  0e6h,   0,
pop$in, 21,  0,  0,
popf$in,     1,  09dh,    0,
push$in,     8,  1,  0,
pushf$in,    1,  09ch,    0,
rcl$in, 4,   0d0h,   2,
rcr$in, 4,   0d0h,   3,
rep$in, 0feh,    0f3h,    0,
repe$in,     0feh,   0f3h,    0,
repne$in,    0feh,   0f2h,    0,
repnz$in,    0feh,   0f2h,    0,
repz$in,     0feh,   0f3h,    0,
ret$in, 14,  0c2h,   0,
retf$in,     14,  0cah,   0,
rol$in, 4,   0d0h,   0,
```

```
                ror$in, 4,  0d0h,   1,
                sahf$in,    1,  09eh,   0,
                sal$in, 4,  0d0h,   4,
                sar$in, 4,  0d0h,   7,
                sbb$in, 9,  18h,    3,
                scasb$in,   1,  0aeh,   0,
                scasw$in,   1,  0afh,   0,
                shl$in, 4,  0d0h,   4,
                shr$in, 4,  0d0h,   5,
                stc$in, 1,  0f9h,   0,
                std$in, 1,  0fdh,   0,
                sti$in, 1,  0fbh,   0,
                stosb$in,   1,  0aah,   0,
                stosw$in,   1,  0abh,   0,
                sub$in, 9,  28h,    5,
                test$in,    18, 0,  0,
                wait$in,    1,  09bh,   0,
                xchg$in,    19, 0,  0,
                xlat$in,    1,  0d7h,   0,
                xor$in, 10, 30h,    6
                );

5   1           end asm$tab;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES		
3			AAAIN.	LITERALLY	4
3			AADIN.	LITERALLY	4
3			AAMIN.	LITERALLY	4
3			AASIN.	LITERALLY	4
3			ADCIN.	LITERALLY	4
3			ADDIN.	LITERALLY	4
3			ANDIN.	LITERALLY	4
1	0000H		ASMTAB	PROCEDURE	STACK=0000H
3			CALLFIN.	LITERALLY	4
3			CALLIN	LITERALLY	4
3			CBWIN.	LITERALLY	4
3			CLCIN.	LITERALLY	4
3			CLDIN.	LITERALLY	4
3			CLIIN.	LITERALLY	4
3			CMCIN.	LITERALLY	4
3			CMPIN.	LITERALLY	4
3			CMPSBIN.	LITERALLY	4
3			CMPSWIN.	LITERALLY	4
3			CSIN	LITERALLY	
3			CWDIN.	LITERALLY	4
3			DAAIN.	LITERALLY	4
3			DASIN.	LITERALLY	4
3			DECIN.	LITERALLY	4
3			DIVIN.	LITERALLY	4
3			DSIN	LITERALLY	
3			ESCIN.	LITERALLY	4
3			ESIN	LITERALLY	
3			HLTIN.	LITERALLY	4
3			IDIVIN	LITERALLY	4
3			IMULIN	LITERALLY	4
3			INCIN.	LITERALLY	4
3			ININ	LITERALLY	4
3			INTIN.	LITERALLY	4
3			INTOIN	LITERALLY	4
3			IRETIN	LITERALLY	4
3			JAEIN.	LITERALLY	4
3			JAIN	LITERALLY	4
3			JBEIN.	LITERALLY	4
3			JBIN	LITERALLY	4
3			JCIN	LITERALLY	4
3			JCXZIN	LITERALLY	4
3			JEIN	LITERALLY	4
3			JGEIN.	LITERALLY	4
3			JGIN	LITERALLY	4
3			JLEIN.	LITERALLY	4
3			JLIN	LITERALLY	4
3			JMPFIN	LITERALLY	4
3			JMPIN.	LITERALLY	4
3			JMPSIN	LITERALLY	4

```
3                 JNAEIN . . . . . .   LITERALLY          4
3                 JNAIN. . . . . .     LITERALLY          4
3                 JNBEIN . . . . . .   LITERALLY          4
3                 JNBIN. . . . . .     LITERALLY          4
3                 JNCIN. . . . . .     LITERALLY          4
3                 JNEIN. . . . . .     LITERALLY          4
3                 JNGEIN . . . . . .   LITERALLY          4
3                 JNGIN. . . . . .     LITERALLY          4
3                 JNLEIN . . . . . .   LITERALLY          4
3                 JNLIN. . . . . .     LITERALLY          4
3                 JNOIN. . . . . .     LITERALLY          4
3                 JNPIN. . . . . .     LITERALLY          4
3                 JNSIN. . . . . .     LITERALLY          4
3                 JNZIN. . . . . .     LITERALLY          4
3                 JOIN . . . . . .     LITERALLY          4
3                 JPEIN. . . . . .     LITERALLY          4
3                 JPIN . . . . . .     LITERALLY          4
3                 JPOIN. . . . . .     LITERALLY          4
3                 JSIN . . . . . .     LITERALLY          4
3                 JZIN . . . . . .     LITERALLY          4
3                 LAHFIN . . . . . .   LITERALLY          4
3                 LDSIN. . . . . .     LITERALLY          4
3                 LEAIN. . . . . .     LITERALLY          4
3                 LESIN. . . . . .     LITERALLY          4
3                 LOCKIN . . . . . .   LITERALLY          4
3                 LODSBIN. . . . . .   LITERALLY          4
3                 LODSWIN. . . . . .   LITERALLY          4
3                 LOOPEIN. . . . . .   LITERALLY          4
3                 LOOPIN . . . . . .   LITERALLY          4
3                 LOOPNEIN . . . . . .  LITERALLY         4
3                 LOOPNZIN . . . . . .  LITERALLY         4
3                 LOOPZIN. . . . . .   LITERALLY          4
3                 MOVIN. . . . . .     LITERALLY          4
3                 MOVSBIN. . . . . .   LITERALLY          4
3                 MOVSWIN. . . . . .   LITERALLY          4
3                 MULIN. . . . . .     LITERALLY          4
3                 NEGIN. . . . . .     LITERALLY          4
3                 NOPIN. . . . . .     LITERALLY          4
3                 NOTIN. . . . . .     LITERALLY          4
2                 OP2IN. . . . . .     LITERALLY          4
2                 OP3IN. . . . . .     LITERALLY          4
2                 OP4IN. . . . . .     LITERALLY          4
2                 OP5IN. . . . . .     LITERALLY          4
2                 OP6IN. . . . . .     LITERALLY          4
4  0000H   480    OPTAB. . . . . .     BYTE ARRAY(480) PUBLIC DATA
3                 ORIN . . . . . .     LITERALLY          4
3                 OUTIN. . . . . .     LITERALLY          4
3                 POPFIN . . . . . .   LITERALLY          4
3                 POPIN. . . . . .     LITERALLY          4
3                 PUSHFIN. . . . . .   LITERALLY          4
3                 PUSHIN . . . . . .   LITERALLY          4
3                 RCLIN. . . . . .     LITERALLY          4
3                 RCRIN. . . . . .     LITERALLY          4
3                 REPEIN . . . . . .   LITERALLY          4
3                 REPIN. . . . . .     LITERALLY          4
3                 REPNEIN. . . . . .   LITERALLY          4
3                 REPNZIN. . . . . .   LITERALLY          4
```

```
   3          REPZIN . . . . . .  LITERALLY       4
   3          RETFIN . . . . . .  LITERALLY       4
   3          RETIN. . . . . .    LITERALLY       4
   3          ROLIN. . . . . .    LITERALLY       4
   3          RORIN. . . . . .    LITERALLY       4
   3          SAHFIN . . . . . .  LITERALLY       4
   3          SALIN. . . . . .    LITERALLY       4
   3          SARIN. . . . . .    LITERALLY       4
   3          SBBIN. . . . . .    LITERALLY       4
   3          SCASBIN. . . . . .  LITERALLY       4
   3          SCASWIN. . . . . .  LITERALLY       4
   3          SHLIN. . . . . .    LITERALLY       4
   3          SHRIN. . . . . .    LITERALLY       4
   3          SSIN . . . . . .    LITERALLY
   3          STCIN. . . . . .    LITERALLY       4
   3          STDIN. . . . . .    LITERALLY       4
   3          STIIN. . . . . .    LITERALLY       4
   3          STOSBIN. . . . . .  LITERALLY       4
   3          STOSWIN. . . . . .  LITERALLY       4
   3          SUBIN. . . . . .    LITERALLY       4
   3          TESTIN . . . . . .  LITERALLY       4
   3          WAITIN . . . . . .  LITERALLY       4
   3          XCHGIN . . . . . .  LITERALLY       4
   3          XLATIN . . . . . .  LITERALLY       4
   3          XORIN. . . . . .    LITERALLY       4
```


MODULE INFORMATION:

```
     CODE AREA SIZE     = 0000H      0D
     CONSTANT AREA SIZE = 01E0H    480D
     VARIABLE AREA SIZE = 0000H      0D
     MAXIMUM STACK SIZE = 0000H      0D
     267 LINES READ
     0 PROGRAM ERROR(S)
```

END OF PL/M-86 COMPILATION

```
ISIS-II MCS-86 ASSEMBLER V1.0 ASSEMBLY OF MODULE ASSLNK86
OBJECT MODULE PLACED IN ASSLNK86.OBJ
ASSEMBLER INVOKED BY: ASM86 ASSLNK86.ASM DEBUG

LOC  OBJ                  LINE   SOURCE

                             1   ;
                             2   ;       interface module for ddt86
                             3   ;       6/15/81
                             4   ;
                             5   dgroup  group   dats
----                         6   dats    segment public  'DATS'
0000 3F                      7           db      '?'
----                         8   dats    ends
                             9   ;
                            10   cgroup  group   code
                            11           extrn   asm:near
                            12           public  ddtgetline
                            13           public  goddt
                            14           public  ddtset
                            15   ;
----                        16   code    segment public 'CODE'
                            17           assume cs:cgroup
                            18   ;
0000                        19   assent  proc
2300                        20           org     2300h
2300 E90000          E      21           jmp     asm
                            22   assent  endp
                            23   ;
2303                        24   ddtgetline      proc
2303 E903DE                 25           jmp     $-21fah
                            26   ddtgetline      endp
                            27   ;
2306                        28   goddt   proc
2306 E903DE                 29           jmp     $-21fah
                            30   goddt   endp
                            31   ;
2309                        32   ddtset  proc
2309 E903DE                 33           jmp     $-21fah
                            34   ddtset  endp
                            35   ;
----                        36   code    ends
                            37           end
```

SYMBOL TABLE LISTING

NAME	TYPE	VALUE	ATTRIBUTES
ASM	L NEAR	0000H	EXTRN
ASSENT	L NEAR	0000H	CODE
CGROUP	GROUP		CODE
CODE	SEGMENT		SIZE=230CH PARA PUBLIC 'CODE'
DATS	SEGMENT		SIZE=0001H PARA PUBLIC 'DATS'
DDIGETLINE	L NEAR	2303H	CODE PUBLIC
DDISET	L NEAR	2309H	CODE PUBLIC
DGROUP	GROUP		DATS
GODDT	L NEAR	2306H	CODE PUBLIC

ASSEMBLY COMPLETE, NO ERRORS FOUND

```
ISIS-II MCS-86 LOCATER, V1.2 INVOKED BY:
LOC86 ASS86.LNK TO ASS86.ABS AD(SM(DATS(0),CODE(0H))) DO(SM(DATS,CODE,CONST,DATA))
WARNING 36:  SEGMENTS OVERLAP
  SEGMENT:  DATS
  SEGMENT:  CODE
  LOW OVERLAP ADDRESS :  00000H
  HIGH OVERLAP ADDRESS:  00000H

SYMBOL TABLE OF MODULE ASSLNK86
READ FROM FILE ASS86.LNK
WRITTEN TO FILE ASS86.ABS

BASE   OFFSET TYPE SYMBOL              BASE   OFFSET TYPE SYMBOL

0000H  32D7H  PUB  ASM                 0000H  2303H  PUB  DDTGETLINE
0000H  2306H  PUB  GODDT               0000H  2309H  PUB  DDTSET
0000H  33D0H  PUB  OPTAB               0000H  1C36H  PUB  ALTTABLEPTRS
0000H  1CC6H  PUB  INSTRTABLE          0000H  2085H  PUB  NOPS
0000H  208AH  PUB  OPNIN               0000H  1EC6H  PUB  TABPTRS
0000H  209DH  PUB  REG16               0000H  20ADH  PUB  REG8
0000H  208DH  PUB  SEGREG              0000H  1407H  PUB  CONOUT
0000H  1404H  PUB  CONIN               0000H  1403H  PUB  BOOT
0000H  1417H  PUB  PRINTM              0000H  146AH  PUB  PRINTWORD
0000H  18C5H  PUB  DISEM

ASSLNK86:  SYMBOLS AND LINES
0000H  0000H  SYM  ASSENT              0000H  2303H  SYM  DDTGETLINE
0000H  2309H  SYM  DDTSET              0000H  2306H  SYM  GODDT

ASMTAB:  SYMBOLS AND LINES
0000H  36CCH  SYM  MEMORY              0000H  33D0H  SYM  OPTAB
0000H  230CH  LIN       5

DDTASM:  SYMBOLS AND LINES
0000H  36CCH  SYM  MEMORY              0000H  35F6H  SYM  TOKEN
0000H  35F6H  SYM  TOKENWORD           0000H  3616H  SYM  TOKENLEN
0000H  3617H  SYM  TOKENTYPE          0000H  35D4H  SYM  TOKENVALUE
0000H  3618H  SYM  NEXTCH             0000H  35D6H  SYM  ASSEMPTR
0000H  35D6H  SYM  ASSEMOFFSET         0000H  35D8H  SYM  ASSEMSEGMENT
0000H  35D6H  BAS  ASSEMBYTE           0000H  35B0H  SYM  BASEREG
0000H  35B4H  SYM  INDEXREG            0000H  35DAH  SYM  OPBASE
0000H  3619H  SYM  OP1                 0000H  362CH  SYM  OP2
0000H  35DAH  BAS  OP                  0000H  363FH  SYM  PREFIX
0000H  3642H  SYM  NPREFIX             0000H  3643H  SYM  SEGPREFIXFLAG
0000H  3644H  SYM  LOCKFLAG            0000H  3645H  SYM  REPFLAG
0000H  3646H  SYM  MODBITS             0000H  3647H  SYM  REGBITS
0000H  3648H  SYM  RMBITS              0000H  3649H  SYM  OPTYPE
0000H  364AH  SYM  MNEMONICINDEX       0000H  364BH  SYM  INSTR1
0000H  364CH  SYM  DISPLEN             0000H  35DCH  SYM  DISPWORD
0000H  35B8H  SYM  DELIMITERS          0000H  35BEH  SYM  LEFTBRACKET
0000H  35BFH  SYM  RIGHTBRACKET        0000H  230EH  SYM  CRLF
0000H  231FH  SYM  GETLINE             0000H  232CH  SYM  ERROR
0000H  233DH  SYM  AMBIG               0000H  234CH  SYM  CHECKAMBIG
0000H  236DH  SYM  COMPARE             STACK  0008H  SYM  A
STACK  0006H  SYM  B                   STACK  0004H  SYM  C
STACK  0008H  BAS  A1                  STACK  0006H  BAS  B1
0000H  239AH  SYM  DELIMITER           STACK  0004H  SYM  B
0000H  364DH  SYM  I                   0000H  23C8H  SYM  NUMBER
STACK  0004H  SYM  B                   0000H  23EDH  SYM  HEX
STACK  0004H  SYM  B                   0000H  2404H  SYM  GETTOKEN
```

```
0000H  364EH  SYM  NUMERIC
0000H  24C4H  SYM  LOOKUPREG
STACK  0006H  SYM  TABLEN
STACK  0008H  BAS  REGWORD
0000H  364FH  SYM  I
0000H  2514H  SYM  CHECKINDEXREG
0000H  2553H  SYM  CHECKREG8
0000H  25EFH  SYM  CHECKBWPREFIX
0000H  2643H  SYM  SETPREFIX
0000H  265BH  SYM  SETSEGPREFIX
0000H  26DAH  SYM  REGINDIRECT
0000H  2751H  SYM  GETOPERAND
0000H  2893H  SYM  GETOPERAND2
0000H  35DEH  SYM  I
0000H  3650H  SYM  I
0000H  35E0H  BAS  MNEMONIC
0000H  29F8H  SYM  PUTMEM
0000H  2A0DH  SYM  PUTPREFIX
0000H  2A36H  SYM  PUTINSTR
STACK  0004H  SYM  B
STACK  0004H  SYM  A
0000H  2A81H  SYM  SETMOD
0000H  2B65H  SYM  DOMODRM
0000H  2BAFH  SYM  MODRMW
STACK  0006H  SYM  REG
0000H  2BECH  SYM  MODRM16
STACK  0004H  SYM  REG
0000H  2C46H  SYM  MODREGRMW
0000H  2C6CH  SYM  PUTIMMED
0000H  2CA6H  SYM  CHECKWANDVAL
0000H  2CE7H  SYM  MODRMIMMED
STACK  0004H  SYM  B
0000H  2D29H  SYM  CHECKMODRMIMMED
0000H  2D5FH  SYM  TYPE1
0000H  2D93H  SYM  TYPE3
0000H  2E1CH  SYM  TYPE5
0000H  35E2H  SYM  TYPE7STRUC
0000H  2E95H  SYM  TYPE8OR8A
0000H  2EE8H  SYM  TYPE8
0000H  2F14H  SYM  TYPE9OR10
0000H  2F7FH  SYM  TYPE9
0000H  35EEH  SYM  TYPE11STRUC
0000H  2FD9H  SYM  TYPEINT
0000H  304CH  SYM  TYPERETRETF
0000H  30D4H  SYM  TYPEOUT
0000H  35F2H  SYM  CJSTRUC
0000H  317CH  SYM  TYPETEST
0000H  31F8H  SYM  TYPEMOV
STACK  0004H  SYM  LOC
0000H  2311H  LIN     27
0000H  231DH  LIN     29
0000H  2322H  LIN     31
0000H  232AH  LIN     33
0000H  232FH  LIN     38
0000H  2338H  LIN     40
0000H  233DH  LIN     42
0000H  2347H  LIN     44
0000H  234CH  LIN     46
0000H  235AH  LIN     48
0000H  2368H  LIN     50
0000H  24ACH  SYM  GETREALTOKEN
STACK  0008H  SYM  REGTABPTR
STACK  0004H  SYM  REGNUMPTR
STACK  0004H  BAS  REGNUM
0000H  24FDH  SYM  CHECKBASEREG
0000H  252BH  SYM  CHECKSEGREG
0000H  25A1H  SYM  CHECKREG16
0000H  262DH  SYM  CHECKSEGPREFIX
STACK  0004H  SYM  B
0000H  268AH  SYM  GETPREFIX
STACK  0004H  SYM  X
0000H  2878H  SYM  GETOPERAND1
0000H  28BBH  SYM  LOOKUPOPTAB
0000H  28FFH  SYM  VALIDMNEMONIC
0000H  35E0H  SYM  TABLEBASE
0000H  297AH  SYM  GETOPCODE
STACK  0004H  SYM  B
0000H  3651H  SYM  T
0000H  2A45H  SYM  SETINSTR
0000H  2A55H  SYM  PUTWORD
0000H  2A6CH  SYM  SETWBIT
0000H  2AA6H  SYM  SETMODRM
STACK  0004H  SYM  OPPTR
STACK  0008H  SYM  INS
STACK  0004H  SYM  OPPTR
STACK  0006H  SYM  INS
0000H  2C18H  SYM  MODREGRM
0000H  2C51H  SYM  MODREGRMDW
0000H  2C88H  SYM  PUTIMMEDS
0000H  2CD6H  SYM  ACCIMMED
0000H  2CFCH  SYM  SETINSTRW
0000H  2D1DH  SYM  CHECKACCIMMED
0000H  2D35H  SYM  CHECKMODRMMODRM
0000H  2D76H  SYM  TYPE2
0000H  2DCAH  SYM  TYPE4
0000H  2E33H  SYM  TYPE6
0000H  2E58H  SYM  TYPE7
0000H  35E6H  SYM  TYPE8STRUC
0000H  2EF6H  SYM  TYPE8A
STACK  0004H  SYM  TYPE
0000H  2F8AH  SYM  TYPE10
0000H  2F95H  SYM  TYPE11
0000H  3011H  SYM  TYPEESC
0000H  3082H  SYM  TYPEIN
0000H  3652H  SYM  TEMP
0000H  313AH  SYM  TYPECALLFJMPF
0000H  318EH  SYM  TYPEXCHG
0000H  32D7H  SYM  ASM
0000H  230EH  LIN     26
0000H  2317H  LIN     28
0000H  231FH  LIN     30
0000H  2325H  LIN     32
0000H  232CH  LIN     37
0000H  2332H  LIN     39
0000H  233BH  LIN     41
0000H  2340H  LIN     43
0000H  234AH  LIN     45
0000H  234FH  LIN     47
0000H  2350H  LIN     49
0000H  236BH  LIN     51
```

```
0000H  236DH  LIN    52
0000H  237CH  LIN    56
0000H  238CH  LIN    58
0000H  2392H  LIN    60
0000H  239AH  LIN    62
0000H  239DH  LIN    66
0000H  23B8H  LIN    68
0000H  23C2H  LIN    70
0000H  23C8H  LIN    72
0000H  23EDH  LIN    75
0000H  23F0H  LIN    78
0000H  2404H  LIN    82
0000H  240DH  LIN    85
0000H  2417H  LIN    87
0000H  2424H  LIN    89
0000H  2431H  LIN    91
0000H  243FH  LIN    93
0000H  2451H  LIN    97
0000H  245BH  LIN    99
0000H  2461H  LIN   102
0000H  246FH  LIN   104
0000H  2476H  LIN   108
0000H  2488H  LIN   111
0000H  24A7H  LIN   114
0000H  24AAH  LIN   116
0000H  24AFH  LIN   118
0000H  24B2H  LIN   120
0000H  24C2H  LIN   122
0000H  24C4H  LIN   124
0000H  24D7H  LIN   131
0000H  24E9H  LIN   134
0000H  24F1H  LIN   137
0000H  24F0H  LIN   139
0000H  2500H  LIN   141
0000H  2514H  LIN   143
0000H  252BH  LIN   145
0000H  252EH  LIN   147
0000H  2548H  LIN   150
0000H  2553H  LIN   153
0000H  2556H  LIN   155
0000H  257CH  LIN   158
0000H  2589H  LIN   160
0000H  2599H  LIN   162
0000H  25A1H  LIN   165
0000H  25A4H  LIN   167
0000H  25CAH  LIN   170
0000H  25D7H  LIN   172
0000H  25E7H  LIN   174
0000H  25EFH  LIN   177
0000H  25F2H  LIN   179
0000H  2601H  LIN   182
0000H  2613H  LIN   184
0000H  2625H  LIN   186
0000H  262DH  LIN   189
0000H  2630H  LIN   191
0000H  2643H  LIN   193
0000H  2653H  LIN   196
0000H  2658H  LIN   198
0000H  2665H  LIN   200
0000H  2678H  LIN   202

0000H  2370H  LIN    55
0000H  2388H  LIN    57
0000H  238FH  LIN    59
0000H  2394H  LIN    61
0000H  239AH  LIN    63
0000H  23A9H  LIN    67
0000H  23BCH  LIN    69
0000H  23C8H  LIN    71
0000H  23CBH  LIN    74
0000H  23EDH  LIN    76
0000H  23F9H  LIN    80
0000H  2407H  LIN    84
0000H  2412H  LIN    86
0000H  241EH  LIN    88
0000H  2424H  LIN    90
0000H  243CH  LIN    92
0000H  244AH  LIN    95
0000H  2456H  LIN    98
0000H  245DH  LIN   101
0000H  2468H  LIN   103
0000H  2474H  LIN   106
0000H  247DH  LIN   110
0000H  24A2H  LIN   112
0000H  24AAH  LIN   115
0000H  24ACH  LIN   117
0000H  24AFH  LIN   119
0000H  24C0H  LIN   121
0000H  24C2H  LIN   123
0000H  24C7H  LIN   130
0000H  24E1H  LIN   133
0000H  24EDH  LIN   136
0000H  24F7H  LIN   138
0000H  24FDH  LIN   140
0000H  2514H  LIN   142
0000H  2517H  LIN   144
0000H  252BH  LIN   146
0000H  2543H  LIN   149
0000H  254FH  LIN   152
0000H  2553H  LIN   154
0000H  256CH  LIN   157
0000H  2586H  LIN   159
0000H  2591H  LIN   161
0000H  259DH  LIN   164
0000H  25A1H  LIN   166
0000H  25BAH  LIN   169
0000H  25D4H  LIN   171
0000H  25DFH  LIN   173
0000H  25EBH  LIN   176
0000H  25EFH  LIN   178
0000H  25F9H  LIN   181
0000H  260BH  LIN   183
0000H  261BH  LIN   185
0000H  2629H  LIN   188
0000H  262DH  LIN   190
0000H  2643H  LIN   192
0000H  2646H  LIN   195
0000H  2657H  LIN   197
0000H  265EH  LIN   199
0000H  2668H  LIN   201
0000H  267DH  LIN   203
```

```
0000H  2680H  LIN    204
0000H  268AH  LIN    206
0000H  268DH  LIN    208
0000H  2697H  LIN    211
0000H  26A6H  LIN    213
0000H  26AFH  LIN    216
0000H  26B8H  LIN    219
0000H  26CAH  LIN    221
0000H  26D6H  LIN    223
0000H  26D8H  LIN    225
0000H  26DDH  LIN    228
0000H  26ECH  LIN    231
0000H  26F8H  LIN    234
0000H  270CH  LIN    236
0000H  270EH  LIN    238
0000H  2721H  LIN    241
0000H  2726H  LIN    243
0000H  272BH  LIN    245
0000H  273EH  LIN    248
0000H  274AH  LIN    250
0000H  274DH  LIN    253
0000H  2754H  LIN    255
0000H  279EH  LIN    257
0000H  27ACH  LIN    259
0000H  27B5H  LIN    261
0000H  27BEH  LIN    263
0000H  27C7H  LIN    265
0000H  27D0H  LIN    268
0000H  27E3H  LIN    271
0000H  27E9H  LIN    273
0000H  27EEH  LIN    275
0000H  27FFH  LIN    278
0000H  2805H  LIN    280
0000H  280FH  LIN    282
0000H  2821H  LIN    284
0000H  282BH  LIN    286
0000H  2836H  LIN    289
0000H  283CH  LIN    291
0000H  284DH  LIN    294
0000H  2859H  LIN    297
0000H  2869H  LIN    299
0000H  286EH  LIN    301
0000H  2876H  LIN    304
0000H  287BH  LIN    306
0000H  2884H  LIN    308
0000H  288EH  LIN    310
0000H  2893H  LIN    312
0000H  289CH  LIN    314
0000H  28AAH  LIN    316
0000H  28AFH  LIN    318
0000H  28B9H  LIN    320
0000H  28BEH  LIN    323
0000H  28E1H  LIN    326
0000H  28F1H  LIN    328
0000H  28FAH  LIN    331
0000H  28FDH  LIN    333
0000H  2902H  LIN    336
0000H  290EH  LIN    338
0000H  293AH  LIN    340
0000H  2963H  LIN    343

0000H  2688H  LIN    205
0000H  268DH  LIN    207
0000H  2690H  LIN    209
0000H  269EH  LIN    212
0000H  26A8H  LIN    215
0000H  26B4H  LIN    217
0000H  26C5H  LIN    220
0000H  26D4H  LIN    222
0000H  26D8H  LIN    224
0000H  26DAH  LIN    226
0000H  26E5H  LIN    229
0000H  26EFH  LIN    232
0000H  2700H  LIN    235
0000H  270EH  LIN    237
0000H  2715H  LIN    240
0000H  2724H  LIN    242
0000H  2729H  LIN    244
0000H  2737H  LIN    246
0000H  2741H  LIN    249
0000H  274AH  LIN    252
0000H  2751H  LIN    254
0000H  278EH  LIN    256
0000H  27A1H  LIN    258
0000H  27AEH  LIN    260
0000H  27B7H  LIN    262
0000H  27C0H  LIN    264
0000H  27C9H  LIN    266
0000H  27DAH  LIN    269
0000H  27E6H  LIN    272
0000H  27EEH  LIN    274
0000H  27F5H  LIN    277
0000H  2802H  LIN    279
0000H  280CH  LIN    281
0000H  2819H  LIN    283
0000H  2823H  LIN    285
0000H  282DH  LIN    288
0000H  2839H  LIN    290
0000H  2843H  LIN    293
0000H  2850H  LIN    295
0000H  2861H  LIN    298
0000H  286BH  LIN    300
0000H  2870H  LIN    302
0000H  2878H  LIN    305
0000H  2881H  LIN    307
0000H  2887H  LIN    309
0000H  2891H  LIN    311
0000H  2896H  LIN    313
0000H  289FH  LIN    315
0000H  28ACH  LIN    317
0000H  28B6H  LIN    319
0000H  28BBH  LIN    321
0000H  28D5H  LIN    324
0000H  28E9H  LIN    327
0000H  28F8H  LIN    329
0000H  28FAH  LIN    332
0000H  28FFH  LIN    334
0000H  2908H  LIN    337
0000H  2920H  LIN    339
0000H  2950H  LIN    342
0000H  2967H  LIN    345
```


SER. #_____

```
0000H  2970H  LIN    346
0000H  297AH  LIN    348
0000H  297DH  LIN    350
0000H  2988H  LIN    352
0000H  298EH  LIN    354
0000H  29A0H  LIN    356
0000H  29ACH  LIN    358
0000H  29B6H  LIN    361
0000H  29C4H  LIN    364
0000H  29CCH  LIN    366
0000H  29CEH  LIN    368
0000H  29DCH  LIN    371
0000H  29E4H  LIN    373
0000H  29EDH  LIN    375
0000H  29F1H  LIN    377
0000H  29F6H  LIN    379
0000H  29FBH  LIN    382
0000H  2A09H  LIN    384
0000H  2A10H  LIN    387
0000H  2A21H  LIN    389
0000H  2A32H  LIN    391
0000H  2A36H  LIN    393
0000H  2A3CH  LIN    395
0000H  2A45H  LIN    397
0000H  2A4EH  LIN    400
0000H  2A55H  LIN    402
0000H  2A5FH  LIN    405
0000H  2A6CH  LIN    407
0000H  2A7AH  LIN    409
0000H  2A81H  LIN    411
0000H  2A8EH  LIN    413
0000H  2AA4H  LIN    415
0000H  2AA9H  LIN    417
0000H  2AB8H  LIN    419
0000H  2ACBH  LIN    426
0000H  2AD4H  LIN    429
0000H  2ADEH  LIN    431
0000H  2AE5H  LIN    434
0000H  2AF3H  LIN    437
0000H  2AFBH  LIN    439
0000H  2B03H  LIN    442
0000H  2B0DH  LIN    445
0000H  2B14H  LIN    448
0000H  2B1FH  LIN    451
0000H  2B26H  LIN    453
0000H  2B29H  LIN    456
0000H  2B34H  LIN    459
0000H  2B39H  LIN    461
0000H  2B3CH  LIN    464
0000H  2B53H  LIN    466
0000H  2B68H  LIN    470
0000H  2B71H  LIN    472
0000H  2B8DH  LIN    474
0000H  2B98H  LIN    476
0000H  2BA8H  LIN    478
0000H  2BB2H  LIN    482
0000H  2BC1H  LIN    485
0000H  2BCAH  LIN    487
0000H  2BDDH  LIN    489
0000H  2BE5H  LIN    491

0000H  2976H  LIN    347
0000H  297AH  LIN    349
0000H  2982H  LIN    351
0000H  298BH  LIN    353
0000H  299CH  LIN    355
0000H  29A7H  LIN    357
0000H  29B3H  LIN    360
0000H  29BDH  LIN    363
0000H  29C7H  LIN    365
0000H  29CEH  LIN    367
0000H  29D5H  LIN    370
0000H  29DFH  LIN    372
0000H  29E8H  LIN    374
0000H  29F1H  LIN    376
0000H  29F4H  LIN    378
0000H  29F8H  LIN    380
0000H  2A05H  LIN    383
0000H  2A0DH  LIN    385
0000H  2A15H  LIN    388
0000H  2A2EH  LIN    390
0000H  2A34H  LIN    392
0000H  2A39H  LIN    394
0000H  2A43H  LIN    396
0000H  2A48H  LIN    399
0000H  2A51H  LIN    401
0000H  2A58H  LIN    404
0000H  2A68H  LIN    406
0000H  2A6FH  LIN    408
0000H  2A7FH  LIN    410
0000H  2A84H  LIN    412
0000H  2A9AH  LIN    414
0000H  2AA6H  LIN    416
0000H  2AAEH  LIN    418
0000H  2AC6H  LIN    425
0000H  2AD4H  LIN    427
0000H  2AD9H  LIN    430
0000H  2AE3H  LIN    432
0000H  2AF0H  LIN    436
0000H  2AF8H  LIN    438
0000H  2B01H  LIN    440
0000H  2B08H  LIN    443
0000H  2B0FH  LIN    447
0000H  2B1FH  LIN    449
0000H  2B24H  LIN    452
0000H  2B26H  LIN    455
0000H  2B34H  LIN    457
0000H  2B37H  LIN    460
0000H  2B39H  LIN    463
0000H  2B51H  LIN    465
0000H  2B65H  LIN    468
0000H  2B6EH  LIN    471
0000H  2B74H  LIN    473
0000H  2B94H  LIN    475
0000H  2BA2H  LIN    477
0000H  2BAFH  LIN    479
0000H  2BB8H  LIN    483
0000H  2BC7H  LIN    486
0000H  2BD7H  LIN    488
0000H  2BE3H  LIN    490
0000H  2BE8H  LIN    492
```

CCP/M-86 2.0 V.4

SER. # CC-104

```
0000H  28ECH  LIN   493
0000H  28FCH  LIN   497
0000H  2C08H  LIN   499
0000H  2C11H  LIN   501
0000H  2C18H  LIN   503
0000H  2C1EH  LIN   505
0000H  2C28H  LIN   508
0000H  2C30H  LIN   511
0000H  2C3DH  LIN   513
0000H  2C46H  LIN   516
0000H  2C4CH  LIN   518
0000H  2C51H  LIN   520
0000H  2C62H  LIN   522
0000H  2C6AH  LIN   524
0000H  2C6FH  LIN   526
0000H  2C7DH  LIN   528
0000H  2C88H  LIN   530
0000H  2C92H  LIN   532
0000H  2CA4H  LIN   534
0000H  2CA9H  LIN   536
0000H  2CBBH  LIN   538
0000H  2CCAH  LIN   541
0000H  2CD4H  LIN   544
0000H  2CD9H  LIN   546
0000H  2CDFH  LIN   548
0000H  2CE5H  LIN   550
0000H  2CEAH  LIN   552
0000H  2CF0H  LIN   554
0000H  2CFAH  LIN   556
0000H  2CFFH  LIN   559
0000H  2D0FH  LIN   561
0000H  2D1DH  LIN   563
0000H  2D29H  LIN   565
0000H  2D2CH  LIN   567
0000H  2D35H  LIN   569
0000H  2D5FH  LIN   571
0000H  2D62H  LIN   573
0000H  2D6CH  LIN   575
0000H  2D74H  LIN   577
0000H  2D79H  LIN   579
0000H  2D83H  LIN   582
0000H  2D8CH  LIN   584
0000H  2D91H  LIN   586
0000H  2D96H  LIN   588
0000H  2DB2H  LIN   591
0000H  2DC3H  LIN   593
0000H  2DC8H  LIN   595
0000H  2DCDH  LIN   597
0000H  2DD3H  LIN   599
0000H  2DE8H  LIN   602
0000H  2E03H  LIN   604
0000H  2E15H  LIN   606
0000H  2E1AH  LIN   608
0000H  2E1FH  LIN   610
0000H  2E31H  LIN   612
0000H  2E36H  LIN   614
0000H  2E3CH  LIN   616
0000H  2E53H  LIN   618
0000H  2E58H  LIN   621
0000H  2E5EH  LIN   623
0000H  23EFH  LIN   495
0000H  2C02H  LIN   498
0000H  2C0FH  LIN   500
0000H  2C14H  LIN   502
0000H  2C1BH  LIN   504
0000H  2C25H  LIN   507
0000H  2C30H  LIN   509
0000H  2C37H  LIN   512
0000H  2C44H  LIN   515
0000H  2C49H  LIN   517
0000H  2C4FH  LIN   519
0000H  2C54H  LIN   521
0000H  2C67H  LIN   523
0000H  2C6CH  LIN   525
0000H  2C76H  LIN   527
0000H  2C86H  LIN   529
0000H  2C8BH  LIN   531
0000H  2C9BH  LIN   533
0000H  2CA6H  LIN   535
0000H  2CB8H  LIN   537
0000H  2CC2H  LIN   540
0000H  2CD1H  LIN   542
0000H  2CD6H  LIN   545
0000H  2CDCH  LIN   547
0000H  2CE2H  LIN   549
0000H  2CE7H  LIN   551
0000H  2CEDH  LIN   553
0000H  2CF7H  LIN   555
0000H  2CFCH  LIN   557
0000H  2D06H  LIN   560
0000H  2D19H  LIN   562
0000H  2D20H  LIN   564
0000H  2D29H  LIN   566
0000H  2D35H  LIN   568
0000H  2D38H  LIN   570
0000H  2D5FH  LIN   572
0000H  2D65H  LIN   574
0000H  2D71H  LIN   576
0000H  2D76H  LIN   578
0000H  2D7CH  LIN   580
0000H  2D86H  LIN   583
0000H  2D8EH  LIN   585
0000H  2D93H  LIN   587
0000H  2D99H  LIN   589
0000H  2DB5H  LIN   592
0000H  2DC5H  LIN   594
0000H  2DCAH  LIN   596
0000H  2DD0H  LIN   598
0000H  2DDAH  LIN   601
0000H  2DEFH  LIN   603
0000H  2E06H  LIN   605
0000H  2E17H  LIN   607
0000H  2E1CH  LIN   609
0000H  2E22H  LIN   611
0000H  2E33H  LIN   613
0000H  2E39H  LIN   615
0000H  2E4EH  LIN   617
0000H  2E56H  LIN   619
0000H  2E5BH  LIN   622
0000H  2E65H  LIN   624
```

```
0000H  2E7CH  LIN   625      0000H  2E93H  LIN   626
0000H  2E95H  LIN   627      0000H  2E98H  LIN   628
0000H  2E9FH  LIN   629      0000H  2EA7H  LIN   630
0000H  2EBEH  LIN   631      0000H  2ED5H  LIN   632
0000H  2EE9H  LIN   633      0000H  2EE8H  LIN   635
0000H  2EEEH  LIN   636      0000H  2EF1H  LIN   637
0000H  2EF4H  LIN   638      0000H  2EF6H  LIN   639
0000H  2EF9H  LIN   640      0000H  2EFCH  LIN   641
0000H  2F0AH  LIN   642      0000H  2F0FH  LIN   643
0000H  2F12H  LIN   644      0000H  2F14H  LIN   645
0000H  2F17H  LIN   647      0000H  2F1AH  LIN   648
0000H  2F1DH  LIN   649      0000H  2F24H  LIN   651
0000H  2F29H  LIN   652      0000H  2F2CH  LIN   653
0000H  2F2EH  LIN   654      0000H  2F35H  LIN   656
0000H  2F3AH  LIN   657      0000H  2F40H  LIN   659
0000H  2F50H  LIN   660      0000H  2F55H  LIN   661
0000H  2F58H  LIN   662      0000H  2F5BH  LIN   663
0000H  2F62H  LIN   664      0000H  2F65H  LIN   665
0000H  2F67H  LIN   666      0000H  2F6AH  LIN   667
0000H  2F6CH  LIN   668      0000H  2F73H  LIN   669
0000H  2F78H  LIN   670      0000H  2F7BH  LIN   671
0000H  2F7FH  LIN   672      0000H  2F82H  LIN   673
0000H  2F88H  LIN   674      0000H  2F8AH  LIN   675
0000H  2F8DH  LIN   676      0000H  2F93H  LIN   677
0000H  2F95H  LIN   679      0000H  2F98H  LIN   680
0000H  2F9BH  LIN   681      0000H  2FA2H  LIN   683
0000H  2FB2H  LIN   684      0000H  2FC2H  LIN   685
0000H  2FC4H  LIN   686      0000H  2FD7H  LIN   687
0000H  2FD9H  LIN   688      0000H  2FDCH  LIN   689
0000H  2FDFH  LIN   690      0000H  2FEEH  LIN   692
0000H  2FF5H  LIN   693      0000H  2FFDH  LIN   695
0000H  3003H  LIN   696      0000H  300AH  LIN   698
0000H  300CH  LIN   699      0000H  300FH  LIN   700
0000H  3011H  LIN   701      0000H  3014H  LIN   702
0000H  3017H  LIN   703      0000H  301AH  LIN   704
0000H  302CH  LIN   706      0000H  3038H  LIN   707
0000H  303EH  LIN   708      0000H  3045H  LIN   709
0000H  3047H  LIN   710      0000H  304AH  LIN   711
0000H  304CH  LIN   712      0000H  304FH  LIN   713
0000H  3055H  LIN   714      0000H  3058H  LIN   715
0000H  305FH  LIN   716      0000H  306AH  LIN   717
0000H  3071H  LIN   719      0000H  3074H  LIN   720
0000H  3078H  LIN   721      0000H  3070H  LIN   722
0000H  3080H  LIN   723      0000H  3082H  LIN   724
0000H  3085H  LIN   725      0000H  3088H  LIN   726
0000H  3091H  LIN   727      0000H  3094H  LIN   728
0000H  3097H  LIN   729      0000H  30A5H  LIN   730
0000H  30B4H  LIN   731      0000H  30C3H  LIN   733
0000H  30C6H  LIN   734      0000H  30CDH  LIN   735
0000H  30CFH  LIN   736      0000H  30D2H  LIN   737
0000H  30D4H  LIN   738      0000H  30D7H  LIN   740
0000H  30DAH  LIN   741      0000H  30E8H  LIN   742
0000H  30EFH  LIN   743      0000H  30FEH  LIN   744
0000H  3105H  LIN   745      0000H  3108H  LIN   746
0000H  310BH  LIN   747      0000H  3114H  LIN   748
0000H  3117H  LIN   749      0000H  312AH  LIN   750
0000H  3131H  LIN   751      0000H  3138H  LIN   752
0000H  313AH  LIN   754      0000H  3130H  LIN   755
0000H  3140H  LIN   756      0000H  3147H  LIN   758
0000H  3157H  LIN   759      0000H  315EH  LIN   760
```

```
0000H  3165H  LIN    761
0000H  317AH  LIN    763
0000H  317FH  LIN    765
0000H  3185H  LIN    767
0000H  3191H  LIN    770
0000H  3196H  LIN    772
0000H  31A2H  LIN    775
0000H  31A7H  LIN    777
0000H  31B4H  LIN    780
0000H  31B9H  LIN    782
0000H  31BEH  LIN    784
0000H  31C4H  LIN    786
0000H  31D6H  LIN    788
0000H  31E8H  LIN    791
0000H  31F1H  LIN    793
0000H  31F6H  LIN    795
0000H  31F8H  LIN    797
0000H  3201H  LIN    799
0000H  3219H  LIN    802
0000H  3224H  LIN    804
0000H  3236H  LIN    807
0000H  3241H  LIN    809
0000H  324AH  LIN    811
0000H  3264H  LIN    814
0000H  326DH  LIN    816
0000H  327FH  LIN    819
0000H  328AH  LIN    821
0000H  3293H  LIN    823
0000H  32A1H  LIN    827
0000H  32B5H  LIN    829
0000H  32BCH  LIN    832
0000H  32C1H  LIN    835
0000H  32CDH  LIN    838
0000H  32D2H  LIN    840
0000H  32D7H  LIN    842
0000H  32E4H  LIN    845
0000H  32EEH  LIN    847
0000H  32F8H  LIN    849
0000H  3304H  LIN    851
0000H  3314H  LIN    854
0000H  331EH  LIN    856
0000H  3330H  LIN    858
0000H  333AH  LIN    860
0000H  3344H  LIN    862
0000H  334EH  LIN    864
0000H  3358H  LIN    866
0000H  3362H  LIN    868
0000H  336CH  LIN    870
0000H  3376H  LIN    872
0000H  3380H  LIN    874
0000H  338AH  LIN    876
0000H  3394H  LIN    878
0000H  33C5H  LIN    880
0000H  33C8H  LIN    882
0000H  230EH  LIN    884

0000H  3167H  LIN    762
0000H  317CH  LIN    764
0000H  3182H  LIN    766
0000H  318CH  LIN    769
0000H  3194H  LIN    771
0000H  319DH  LIN    774
0000H  31A5H  LIN    776
0000H  31AEH  LIN    779
0000H  31B7H  LIN    781
0000H  31C0H  LIN    783
0000H  31C1H  LIN    785
0000H  31C7H  LIN    787
0000H  31E1H  LIN    789
0000H  31EEH  LIN    792
0000H  31F3H  LIN    794
0000H  31F8H  LIN    796
0000H  31FEH  LIN    798
0000H  3214H  LIN    801
0000H  321FH  LIN    803
0000H  3224H  LIN    805
0000H  323BH  LIN    808
0000H  3248H  LIN    910
0000H  325CH  LIN    813
0000H  3267H  LIN    815
0000H  326DH  LIN    817
0000H  3287H  LIN    820
0000H  3291H  LIN    822
0000H  329AH  LIN    825
0000H  32B2H  LIN    828
0000H  32B7H  LIN    831
0000H  32BFH  LIN    834
0000H  32C8H  LIN    837
0000H  32D0H  LIN    839
0000H  32D5H  LIN    841
0000H  32DAH  LIN    844
0000H  32E7H  LIN    846
0000H  32F4H  LIN    848
0000H  3301H  LIN    850
0000H  330AH  LIN    852
0000H  3318H  LIN    855
0000H  332BH  LIN    857
0000H  3335H  LIN    859
0000H  333FH  LIN    861
0000H  3349H  LIN    863
0000H  3353H  LIN    865
0000H  335DH  LIN    867
0000H  3367H  LIN    869
0000H  3371H  LIN    871
0000H  3378H  LIN    873
0000H  3385H  LIN    875
0000H  338FH  LIN    877
0000H  3399H  LIN    879
0000H  33C5H  LIN    881
0000H  33CFH  LIN    883
```


MEMORY MAP OF MODULE ASSLNK86
READ FROM FILE ASS86.LNK
WRITTEN TO FILE ASS86.ABS

SEGMENT MAP

START	STOP		LENGTH	ALIGN	NAME	CLASS
00000H	00000H	C	0001H	G	DATS	DATS
00000H	033CEH		33CFH	G	CODE	CODE
033D0H	035D3H		0204H	W	CONST	CONST
035D4H	03652H		007FH	W	DATA	DATA
03660H	03695H		0036H	G	??SEG	
03696H	036CBH		0036H	W	STACK	STACK
036CCH	036CCH		0000H	W	MEMORY	MEMORY

GROUP MAP

ADDRESS	GROUP OR SEGMENT NAME
00000H	CGROUP
	CODE
00000H	DGROUP
	DATS
	CONST
	DATA
	STACK
	MEMORY

```
                                title   'DDT86 1.1  10/2/81'
                        ;
                        ;       modified  5/14/81  R. Silberstein
                        ;       modified  6/15/81  R. Silberstein
                        ;       modified  8/12/81  R. Silberstein
                        ;       modified  9/6/81   R. Silberstein
                        ;       modified  9/16/81  R. Silberstein
                        ;       modified  10/1/81  R. Silberstein
                        ;
                        ;
                        ;       ****************************************
                        ;       *                                      *
                        ;       *       D D T  8 0 8 6 - 8 0 8 8       *
                        ;       *                                      *
                        ;       ****************************************
                        ;
0000                    debug           equ     00h     ;if set, use direct bios calls for console io
                        ;
0100                    ddt_org         equ     100h            ;origin of this module
1400                    lasmorg         equ     ddt_org+1300h   ;origin of disassembler
2300                    asmorg          equ     ddt_org+2200h   ;origin of assembler
                        ;
                                        cseg
                        ;
                                        org     005ch
005C                    fcb             rb      10h
006C                    fcb2            rb      14h
0080                    buff            rb      80h
                        ;
                                        org     lasmorg
1400 B80000             disem:          mov     ax,0
1403 C20400                             ret     4       ;remove parameters from stack
                        ;
                                        org     asmorg
2300 B80000             assem:          mov     ax,0
2303 C20400                             ret     4       ;remove parameters from stack
                        ;
                                        org     ddt_org
                        
0100 E94A01         024D                jmp     ddt86   ;ccp transfers control here
0103 E9EB02         03F1                jmp     conin
0106 E9BB02         03C4                jmp     plmconout
0109 E9C502         03D1                jmp     plmgetline
010C E9480A         0B5A                jmp     asment  ;get here on error in assem (pl/m)
010F E94508         0957                jmp     plmset  ;assembler link to ddt set/verify
                        ;
0112 CDE0               bdosint:        int     bdosi   ;this is only here for user to patch
                                                        ;actual bdos link gets set from here
0112                    bdosintloc      equ     offset bdosint
0113                    bdosintnum      equ     offset bdosint+1
                        ;
00E0                    bdosi           equ     224     ;bdos interrupt number
```

```
                        ;
                stsize  equ     96      ;stack size
                nbps    equ     2       ;number of breakpoints
                        ;
                ifmask16 equ    0200h   ;16-bit IF mask
                ifmask8 equ     02h     ;8-bit IF mask
                        ;
                lf      equ     0ah     ;line feed
                cr      equ     0dh     ;carriage return
                eol     equ     cr
                ctls    equ     13h     ;ascii ctl-s
                        ;
                        ;       ****************************************
                        ;       *                                      *
                        ;       *           m e s s a g e s            *
                        ;       *                                      *
                        ;       ****************************************
                        ;
0114 20434F50>952 copyright     db      ' COPYRIGHT (C) 1981, DIGITAL RESEARCH '
     494748542028
     432920313938
     312C20444947
     495441432052
     455345415243
     4820
                        ;
013A 444454383620 signon        db      'DDT86 1.','1' or 80h
     312E81
                        ;
0143 2031302F3032 DATE          DB      ' 10/02/81 '
     2F383120
014D 4108         regname       db      'A','X' or 80h
014F 4208                       db      'B','X' or 80h
0151 4308                       db      'C','X' or 80h
0153 4408                       db      'D','X' or 80h
0155 5350                       db      'S','P' or 80h
0157 4250                       db      'B','P' or 80h
0159 5349                       db      'S','I' or 80h
015B 4449                       db      'D','I' or 80h
015D 4353         segreg        db      'C','S' or 80h
015F 4453                       db      'D','S' or 80h
0161 5353                       db      'S','S' or 80h
0163 4553                       db      'E','S' or 80h
0165 4950                       db      'I','P' or 80h
                        ;
0167 4F444954535A flagname      db      'ODITSZAPC'
     415043
0170 0506070809A  flagbits      db      5,6,7,8,9,10,12,14,16
     0C0E10
                        ;
0179 4353         segnames      db      'C','S' or 80h
017B 4453                       db      'D','S' or 80h
017D 4553                       db      'E','S' or 80h
```

```
                                ;
0189 0D0A43414E4E       closem  db      cr,lf,'CANNOT CLOS','E' or 80h
     4F5420434C4F
     5345
0197 0D0A494E5355       loadm   db      cr,lf,'INSUFFICIENT MEMOR','Y' or 80h
     46464943494
     4E5420404540
     4F52D9
01AC 0D0A4E4F2053       makem   db      cr,lf,'NO SPAC','E' or 80h
     504143C5
01B6 0D0A4D454D4F       memm    db      cr,lf,'MEMORY REQUEST DENIE','D' or 80h
     525920524551
     554553542044
     454E4945C4
01CD 0D0A4E4F2046       openm   db      cr,lf,'NO FIL','E' or 80h
     494CC5
01D6 202053544152       readm   db      '  START      EN','D' or 80h
     542020202020
     20454EC4
01E6 0D0A56455249       verm    db      cr,lf,'VERIFY ERROR AT',' ' or 80h
     465920455252
     4F52204154A0
01F8 0D0A44495348       writem  db      cr,lf,'DISK WRITE ERRO','R' or 80h
     205752495445
     204552524FD2
                                ;
                                ;       ******************************************************
                                ;       *                                                    *
                                ;       *           i n i t i a l i z a t i o n              *
                                ;       *                                                    *
                                ;       ******************************************************
                                ;
                                setbdosint:                     ;copy vector at 0:bdosi*4 to (bdosi+1)*4
020A A01301                             mov     al,byte ptr .bdosintnum ;get bdos interrupt #
020D FEC0                               inc     al
020F A21E03                             mov     byte ptr .ddtbdosintnum,al      ;ddt uses the next interrupt internally
0212 2AE4                               sub     ah,ah
0214 D1E0                               shl     ax,1
0216 D1E0                               shl     ax,1            ;bdos int # * 4
0218 8BF8                               mov     di,ax           ;[di] points to new bdos interrupt vector
021A 8BF0                               mov     si,ax
021C 83EE04                             sub     si,4            ;[si] points to actual bdos interrupt vector
021F 1E                                 push    ds              ;save ds
0220 2BC0                               sub     ax,ax
0222 8EC0                               mov     es,ax           ;set es and ds to 0 to move interrupt vectors
0224 8ED8                               mov     ds,ax
0226 B90400                             mov     cx,4
```

```
                                ;
0229 F3A4               rep     movs    al,al           ;copy bdos interrupt vector to next int vector
022B 1F                         pop     ds              ;restore ds
022C C3                         ret
                                ;
                        checkcmdtail:                   ;if command tail not empty, assume E command
022D BE8000                     mov     si,offset buff
0230 B400                       mov     ah,0
0232 AC                         lods    al              ;get count from command tail
0233 0AC0                       or      al,al
0235 7415       024C            jz      cctret          ;nothing to do if tail empty
0237 3C40                       cmp     al,conbuffmax
0239 7602       023D            jbe     movcom
023B B040                       mov     al,conbuffmax   ;truncate, if needed
                        movcom:
023D 8BC8                       mov     cx,ax           ;count to [cx]
023F BFC311                     mov     di,offset conbuff
0242 1E                         push    ds
0243 07                         pop     es              ;point destination to ddt seg
0244 F3A4               rep     movs    al,al           ;copy command tail into ddt command buff
0246 B00D                       mov     al,eol
0248 AA                         stos    al              ;store terminator
0249 E84B0A     0C97            call    execute         ;command tail is assumed E command
                        cctret:
024C C3                         ret
                                ;
                        ddt86:
024D FC                         cld
024E 8C15BD11                   MOV     CCPSS,SS
0252 8926BF11                   MOV     CCPSP,SP        ;SAVE CCP STACK POINTER
0256 2E8C16B011                 MOV     USERSS,SS
025B 2E8926A411                 MOV     USERSP,SP       ;INITIALIZE USER'S MACHINE STATE TO CCP STACK
                                ;
0260 9C                         pushf
0261 58                         pop     ax              ;get flags
0262 250002                     and     ax,ifmask16     ;mask to IF bit
0265 88266B12                   mov     sysif,ah        ;save system IF state
0269 2EA3B611                   mov     userfl,ax       ;initialize user's flags to sysif
                                ;
026D 8CC8                       mov     ax,cs
026F FA                         cli                     ;entering critical region
0270 8ED0                       mov     ss,ax           ;set ss = cs
0272 BC1013                     mov     sp,offset stackp        ;set up stack pointer
0275 F6066B1202                 test    sysif,ifmask8   ;see if interrupts were enabled
027A 7401       027D            jz      d0              ;don't turn them on if they were off
027C FB                         sti
                        d0:                             ;exiting critical region
                                ;
027D E88AFF     020A            call    setbdosint      ;copy vector since ddt uses bdosi+1 internally
                                ;
0280 F6066F12FF                 test    savevecflag,0ffh        ;ddt interrupts saved on each g/t/u?
0285 7503       028A            jnz     d1                      ;if so, don't initialize here
0287 E8DA04     0764            call    bpvect                  ;if not, initialize them here
```

```
                                dl:
                                        if      debug
                                ;
                                        sub     ax,ax
                                        mov     es,ax
                                        mov     si,bdosi * 4 + 2
                                        mov     ax,es:[si]      ;get bdos segment
                                        mov     es,ax
                                        mov     biosentryseg,ax
                                        mov     di,biosentryoff
                                        mov     al,81h
                                        stos    al
                                        mov     al,0c3h
                                        stos    al
                                        mov     al,0h
                                        stos    al
                                        mov     al,25h
                                        stos    al
                                        mov     al,0ffh
                                        stos    al
                                        mov     al,0d3h
                                        stos    al
                                        mov     al,0cbh
                                        stos    al
                                ;
                                        endif
                                ;
028A BE3A01                             mov     si,offset signon        ;get sign on message
028D E8A001             043D            call    printm          ;and print it
                                ;
0290 E8A700             033A            CALL    VERSION
0293 3C30                               CMP     AL,30H          ;SEE IF WE ARE UNDER FILE SYSTEM III (MP/M)
0295 B700                               MOV     BH,0
0297 7207               02A0            JC      D2              ;IF EARLIER VERSION, SKIP
0299 B2FE                               MOV     DL,0FEH
029B E8EC00             038A            CALL    SETERRMODE      ;SO BDOS RETURNS TO DDT ON FILE ERRORS
029E B701                               MOV     BH,1
                                D2:
02A0 883E7112                           MOV     ERRMODE,BH
                                ;
02A4 E886FF             022D            call    checkcmdtail    ;if non-blank, do E command
                                ;
                                ;       ************************************************
                                ;       *                                              *
                                ;       *           w o r k i n g  l o o p             *
                                ;       *                                              *
                                ;       ************************************************
                                ;
02A7 FC                         start:  cld                     ;direction flag points up
02A8 BC1013                             mov     sp,offset stackp        ;make sure stack is right
02AB E86F01             041D            call    crlf            ;print crlf
02AE B02D                               mov     al,'-'          ;and prompt
```

```
                                ;
02B0 E87300             0326    call    conout
02B3 E82101             03D7    call    getline         ;get command line
02B6 E83B01             03F4    call    conin           ;read first char
02B9 3C0D                       cmp     al,eol
02BB 74EA               02A7    jz      start
02BD 3C3B                       CMP     AL,';'
02BF 74E6               02A7    JZ      START           ;IGNORE COMMENT LINES
02C1 2C41                       sub     al,'A'          ;check range for valid command
02C3 721A               02DF    jb      err
02C5 3C19                       cmp     al,'Z'-'A'
02C7 7716               02DF    ja      err
02C9 D0E0                       shl     al,1            ;* 2 (2 bytes per ctable entry)
02CB B400                       mov     ah,0
02CD 93                         xchg    ax,bx
02CE C6066C1201                 mov     numreq,1        ;most commands require an argument
02D3 C6056D1200                 mov     wmode,0         ;most commands are not word mode
02D8 2EFF97E902                 call    word ptr ctable[bx]     ;immed call command routine
02DD EBC8               02A7    jmps    start           ;start over
                                ;
                                err:
02DF E83B01             041D    call    crlf
02E2 B03F                       mov     al,'?'          ;error handler
02E4 E83F00             0326    call    conout          ;print error char
02E7 EBBE               02A7    jmps    start           ;stack maybe messed up, keep this jmp
                                ;
                                ;       ************************************************
                                ;       *                                              *
                                ;       *       c o m m a n d   j u m p   t a b l e    *
                                ;       *                                              *
                                ;       ************************************************
                                ;
02E9 2F08               ctable  dw      assm            ;assemble mnemonics
02EB 600B                       dw      BLOCKCOMPARE    ;COMPARE MEMORY BLOCKS
02ED DF02                       dw      err
02EF D60B                       dw      display         ;display memory
02F1 970C                       dw      execute         ;load user program for execution
02F3 150D                       dw      fill            ;fill memory with constant
02F5 5B0D                       dw      gouser          ;go to user program
02F7 BF0D                       dw      hexmath         ;compute hex sum and difference
02F9 DE0D                       dw      ifcb            ;input file control block
02FB DF02                       dw      err
02FD DF02                       dw      err
02FF 2E0E                       dw      lassm           ;disassemble memory
0301 A40E                       dw      move            ;move block
0303 DF02                       dw      err
0305 DF02                       dw      err
0307 DF02                       dw      err
0309 DF02                       dw      err
030B F20E                       dw      read            ;read file
030D 4C0F                       dw      setmem          ;set memory
030F A10F                       dw      trace           ;trace program execution
0311 D70F                       dw      untrace         ;untraced program execution
```

```
                                dw      verify          ;display file info
0313 DE0F                       dw      write           ;write memory block to disk
0315 3010                       dw      xcom            ;display/alter CPU state
0317 E410                       dw      err
0319 DF02                       dw      err
0318 DF02                       dw      err
                        ;
                        ;       ************************************************
                        ;       *                                              *
                        ;       *          b d o s   i n t e r f a c e         *
                        ;       *                                              *
                        ;       ************************************************
                        ;
                        bdos:                           ;this interrupt instruction is overwritten on initialization
                                                        ;the actual int # used is the one at bdosint: + 1
  031E                  ddtbdosintnum   equ     offset bdos + 1
                        ;
031D CDE0                               int     bdosi
031F C3                                 ret
                        ;
                                        if      debug
                        ;
                        bios:
                                        callf   dword ptr biosentryoff
                                        ret
                        ;
                                        endif
                        ;
                                        if      debug
                        ;
                        consin:
                                        mov     bx,9
                                        call    bios
                                        push    ax
                                        call    conout
                                        pop     ax
                                        ret
                        ;
                                        endif
                        ;
                                        if      not debug
                        ;
                        consin:
0320 B101                               mov     cl,1
0322 E8F8FF             0310            call    bdos
0325 C3                                 ret
                        ;
                                        endif
                        ;
                                        if      debug
                        ;
                        conout:
                                        mov     bx,0ch
```

```
372
373                                     mov     cl,al
374                                     jmp     bios
375                             ;
376                                     endif
377                             ;
378                                     if      not debug
379                             ;
380                             conout:
381   0326 B102                         mov     cl,2
382   0328 8AD0                         mov     dl,al
383   032A E8F0FF       031D            call    bdos
384   032D C3                           ret
385                             ;
386                                     endif
387                             ;
388                             rdconbuff:
389   032E B10A                         mov     cl,10
390   0330 E8EAFF       031D            call    bdos
391   0333 C3                           ret
392                             ;
393                                     if      debug
394                             ;
395                             constat:
396                                     mov     bx,6
397                                     jmp     bios
398                             ;
399                                     endif
400                             ;
401                                     if      not debug
402                             ;
403                             constat:
404   0334 B10B                         mov     cl,11
405   0336 E8E4FF       031D            call    bdos
406   0339 C3                           ret
407                             ;
408                                     endif
409                             ;
410                             VERSION:
411   033A B10C                         MOV     CL,12
412   033C E9DEFF       031D            JMP     BDOS
413                             ;
414                             open:
415   033F B10F                         mov     cl,15
416   0341 E8D9FF       031D            call    bdos
417   0344 FEC0                         inc     al              ;test for 0ffh returned
418   0346 7401         0349            jz      openerr
419   0348 C3                           ret
420                             openerr:
421   0349 BEC001                       mov     si,offset openm
422   034C EB70         03BE            jmps    errm
423                             ;
424                             close:
```

```
                                ;
                                delete:
0350 B113                       mov     cl,19
0352 E9B8FF             0310    jmp     bdos
                                ;
                                readsec:
0355 B114                       mov     cl,20
0357 E9B6FF             0310    jmp     bdos
                                ;
                                writesec:
035A B115                       mov     cl,21
035C E8B1FF             0310    call    bdos
035F 0AC0                       or      al,al
0361 7501               0364    jnz     writeerr
0363 C3                         ret
                                writeerr:
0364 BEF801                     mov     si,offset writem
0367 EB48               03B1    jmps    errm
                                ;
                                make:
0369 B116                       mov     cl,22
036B E8A2FF             0310    call    bdos
036E FEC0                       inc     al
0370 7401               0373    jz      makeerr
0372 C3                         ret
                                makeerr:
0373 BEAC01                     mov     si,offset makem
0376 EB39               03B1    jmps    errm
                                ;
                                setdma:
0378 B11A                       mov     cl,26
037A E993FF             0310    jmp     bdos
                                ;
                                SETERRMODE:
037D B12D                       MOV     CL,45
037F E98EFF             0310    JMP     BDOS
                                ;
                                setdmab:
0382 B133                       mov     cl,51
0384 E989FF             0310    jmp     bdos
                                ;
                                getmaxmem:
0387 B135                       mov     cl,53
0389 E884FF             0310    call    bdos
```

```
                                ;
                                allocabsmem:
0399 FEC0                               inc     al
0398 7408               03A8            jz      memerr
039D C3                                 ret
                                ;
                                allocabsmem:
039E B138                               mov     cl,56
03A0 E87AFF             031D            call    bdos
                                ;       inc     al
                                ;       jz      memerr
03A3 90                                 NOP
03A4 90                                 NOP
03A5 90                                 NOP
03A6 90                                 NOP             ;REPLACE INC AL, JZ MEMERR
03A7 C3                                 ret
                                memerr:
03A8 BEB601                             mov     si,offset memm
03AB EB11               03BE            jmps    errm
                                ;
                                freemem:
03AD B139                               mov     cl,57
03AF E96BFF             031D            jmp     bdos
                                ;
                                load:
03B2 B138                               mov     cl,59
03B4 E866FF             031D            call    bdos
03B7 40                                 inc     ax              ;test for 0ffffh returned
03B8 7401               03BB            jz      loaderr
03BA C3                                 ret
                                loaderr:
03BB BE9701                             mov     si,offset loadm
                                ;
                                errm:
03BE E87C00             043D            call    printm
03C1 E9E3FE             02A7            jmp     start
                                ;
                                plmconout:
03C4 55                                 push    bp
03C5 8BEC                               mov     bp,sp
03C7 8B4604                             mov     ax,4[bp]
03CA E859FF             0326            call    conout
03CD 5D                                 pop     bp
03CE C20200                             ret     2
                                ;
                                plmgetline:
03D1 55                                 push    bp
03D2 E80200             03D7            call    getline
03D5 5D                                 pop     bp              ;restore bp for pl/m
03D6 C3                                 ret
                                ;
                                ;       ************************************************
                                ;       *                                              *
                                ;       *     c o n s o l e   i / o   r o u t i n e s  *
```

```
                        ;       *                                               *
                        ;       ************************************************
                        ;
                                if      debug
                        ;
                        ctlx:
                                mov     al,'#'
                                call    conout
                                call    crlf
                        getline:
                                mov     conptr,0
                        get0:
                                call    consin
                                cmp     al,3
                                jz      ctlc
                                cmp     al,8
                                jz      backsp
                                cmp     al,24           ;ctl-x
                                jz      ctlx
                                cmp     al,cr
                                jz      getlinedone
                                cmp     conptr,conbuffmax
                                jnb     getlinedone
                                mov     di,offset conbuff       ;normal character store
                                add     di,conptr
                                mov     [di],al
                                inc     conptr
                                jmps    get0
                        getlinedone:
                                mov     di,offset conbuff
                                add     di,conptr
                                mov     byte ptr [di],eol
                                mov     conptr,0
                                ret
                        backsp:
                                cmp     conptr,0
                                jz      get0
                                dec     conptr
                                call    blank
                                mov     al,8
                                call    conout
                                jmps    get0
                        ctlc:
                                mov     cl,0
                                mov     dl,0
                                jmp     bdos
                        ;
                                endif
                        ;
                                if      not debug
                        ;
                        getline:
```

```
                                mov     dx,offset conbuffhdr
03D7 BAC111                     call    rdconbuff
03DA E851FF             032E    mov     bl,conbuffcnt
03DD 8A1EC211                   mov     bh,0
03E1 8700                       add     bx,offset conbuff
03E3 81C3C311                   mov     byte ptr [bx], eol
03E7 C60700D                    mov     conptr,0
03EA C70604120000               ret
03F0 C3                 ;
                                endif
                        ;
                        conin:
03F1 8EC311                     mov     si,offset conbuff
03F4 03360412                   add     si,conptr
03F8 AC                         lods    al
03F9 FF060412                   inc     conptr
                                                        ;fall thru to upper
03FD 3C61               upper:  cmp     al,'a'          ;less than 'a'
03FF 7206               0407    jb      upret           ;or
0401 3C7A                       cmp     al,'z'          ;greater than 'z'
0403 7702               0407    ja      upret           ;then no change
0405 245F                       and     al,5fh          ;else convert to uc
0407 C3                 upret:  ret
                        ;
                        ctlchek:                        ;check for ctl-s, ctl-q and ctl-c
0408 E829FF             0334    call    constat         ;keypress?
040B 0AC0                       or      al,al           ;zero?
040D 740D               041C    jz      ctlexit         ;no keypress so return
040F E80EFF             0320    call    consin          ;if keypress then get the data
0412 3C13                       cmp     al,ctls         ;check for ctl-s
0414 7403               0419    jz      kwait
0416 E98EFE             02A7    jmp     start           ;any other key will restart
0419 E804FF             0320 kwait:  call    consin     ;if ctl-s then wait for another keypress
                        ctlexit:
041C C3                         ret
                        ;
                        crlf:
041D B00D                       mov     al,cr           ;send cr and lf to console
041F E804FF             0326    call    conout
0422 B00A                       mov     al,lf
0424 E8FFFE             0326    call    conout
0427 C3                         ret
                        ;
                        CRLFCHK:                        ;DO CRLF AND CHECK FOR ABORT
0428 E8F2FF             0410    CALL    CRLF
042B E8D4FF             0408    CALL    CTLCHEK
042E C3                         RET
                        ;
                        blank:                          ;print a blank.
042F B020                       mov     al,' '
0431 E8F2FE             0326    call    conout
0434 C3                         ret
```

```
                                ;
                                tabs:                           ;print # blanks in cx
0435 51                                 push    cx
0436 E8F6FF             042F           call    blank
0439 59                                 pop     cx
043A E2F9               0435           loop    tabs
043C C3                                 ret
                                ;
                                printm:                         ;print the message at [si] on console
                                                                ;end of message indicated by parity set.
043D AC                                 lods    al              ;get a byte
043E A880                               test    al,80h          ;check for end of message
0440 7507               0449           jnz     pquit           ;quit if parity set
0442 56                                 push    si
0443 E8E0FE             0326           call    conout          ;otherwise display byte
0446 5E                                 pop     si
0447 EBF4               043D           jmps    printm          ;print more message
                                pquit:
0449 247F                               and     al,7fh          ;strip parity
044B E8D8FE             0326           call    conout          ;print last byte
044E C3                                 ret
                                ;
                                ascout:                         ;output [al] in ascii
044F 3C20                               cmp     al,' '
0451 7204               0457           jb      perout          ;less than blank?
0453 3C7E                               cmp     al,7eh
0455 7602               0459           jna     ascend
                                perout:
0457 B02E                               mov     al,'.'          ;output '.'
                                ascend:
0459 E8CAFE             0326           call    conout          ;else output ascii
045C C3                                 ret
                                ;
                                print8or16:                     ;print byte or word at es:[si] depending on wmode
045D 268B04                             mov     ax,es:[si]
0460 F6056D1201                         test    wmode,1
0465 7415               047C           jz      printbyte
0467 EB0C               0475           jmps    printword
                                ;
                                printdword:                     ;print double word as ssss:oooo
                                        ; called with:
                                        ;  es = segment
                                        ;  di = offset
0469 57                                 push    di
046A 8CC0                               mov     ax,es
046C E80600             0475           call    printword       ;print segment
046F B03A                               mov     al,':'
0471 E8B2FE             0326           call    conout
0474 58                                 pop     ax
                                ;
                                printword:                      ;print value in [ax] as 4 hex digits
0475 50                                 push    ax
```

```
                                ;
0476 8AC4                       mov     al,ah
0478 E80100         047C        call    printbyte
047B 58                         pop     ax
                                ;
                        printbyte:                      ;print value in [al] as 2 hex digits
047C 50                         push    ax
047D B104                       mov     cl,4
047F D2E8                       shr     al,cl           ;shift al right 4 places
0481 E80100         0485        call    printnibble     ;output upper nibble
0484 58                         pop     ax              ;restore al (now we do lower nibble)
                                ;
                        printnibble:                    ;print value in low 4 bits of [al] as a hex digit
0485 240F                       and     al,0fh          ;mask upper 4 bits
0487 0490                       add     al,90h
0489 27                         daa
048A 1440                       adc     al,40h
048C 27                         daa
048D E896FE         0326        call    conout
0490 C3                         ret
                                ;
                                ;       ****************************************
                                ;       *                                      *
                                ;       *     file name parsing routines       *
                                ;       *                                      *
                                ;       ****************************************
                                ;
                        parse:                          ;parse into fcb whose offset is in [di]
0491 0E                         push    cs
0492 07                         pop     es              ;set es=cs
0493 57                         push    di              ;save fcb address
0494 2AC0                       sub     al,al
0496 B92400                     mov     cx,36           ;fcblen
0499 F3AA                rep    stos    al              ;initialize fcb to 0
049B 5F                         pop     di              ;restore fcb address
                                ;
                        parse2:                         ;enter here to parse without clearing
                                                        ;assumes es = cs from parse:
049C 893EAE12                   mov     fcbadr,di       ;save fcb address
04A0 47                         inc     di              ;point to first byte of filename
04A1 E82200         04C6        call    setupdisk       ;check for d: and set drive byte
04A4 B90800                     mov     cx,8
04A7 E84800         04F5        call    fillfield       ;first item was disk, now get filename
04AA B90300                     mov     cx,3            ;length of file type
04AD 803EAD122E                 cmp     lastchar,'.'
04B2 7405           04B9        jz      filltype
04B4 E86000         0517        call    fillbl          ;fill type with blanks if no '.'
04B7 EB03           04BC        jmps    parseret
                        filltype:
04B9 E83900         04F5        call    fillfield       ;if '.', fill field from console buff
                        parseret:
04BC E85F00         051E        call    scanq           ;count '?'s in fcb
04BF A0AD12                     mov     al,lastchar
```

```
                                ret                     ;with last char scanned in [al]
                        parseerr:
04C3 E919FE     02DF            jmp     err
                        ;
                        setupdisk:                      ;set byte 0 of fcb according to char in fcb (1)
04C6 E828FF     03F1            call    conin
04C9 A2AD12                     mov     lastchar,al
04CC 3C20                       cmp     al,' '
04CE 74F6       04C6            jz      setupdisk       ;deblank input
04D0 3C0D                       cmp     al,eol
04D2 7413       04E7            jz      s1              ;can't be drive, decrement conptr to rescan
04D4 E81AFF     03F1            call    conin
04D7 3C3A                       cmp     al,':'
04D9 7508       04E3            jnz     s0              ;not a drive, subtract 2 from conptr to rescan
04DB A0AD12                     mov     al,lastchar     ;get drive char
04DE 2C40                       sub     al,'A'-1
04E0 4F                         dec     di              ;point to fcb (0)
04E1 AA                         stos    al              ;store drive byte
04E2 C3                         ret
                        s0:
04E3 FF0E0412                   dec     conptr
                        s1:
04E7 FF0E0412                   dec     conptr
04EB C3                         ret
                        ;
                        pdelim:                         ;check char in [al] for delimiter; return ZF if so.
04EC BFA212                     mov     di,offset delims
04EF B90800                     mov     cx,ndelims
04F2 F2AE               repnz   scas    al              ;look in table
04F4 C3                         ret
                        ;
                        fillfield:                      ;count in [cx], dest ptr in [di]
04F5 E8F9FE     03F1            call    conin
04F8 A2AD12                     mov     lastchar,al     ;save last char scanned
04FB 3C2A                       cmp     al,'*'
04FD 7505       0504            jnz     notast
04FF E81100     0513            call    fillq           ;fill with '?'
0502 EBF1       04F5            jmps    fillfield       ;continue till delimiter
                        notast:
0504 57                         push    di
0505 51                         push    cx
0506 E8E3FF     04EC            call    pdelim
0509 59                         pop     cx
050A 5F                         pop     di
050B 740A       0517            jz      fillbl          ;if delimiter, fill field with ' '
050D E3B4       04C3            jcxz    parseerr        ;error if count exceeded
050F AA                         stos    al              ;store char in fcb
0510 49                         dec     cx              ;decrement count
0511 EBE2       04F5            jmps    fillfield
                        fillq:
0513 B03F                       mov     al,'?'
0515 EB02       0519            jmps    fillx
```

```
                        fillb1:
0517 B020                       mov     al,' '
                        fillx:
0519 E302       0510            jcxz    filldone
051B F3AA               rep     stos    al              ;store '?' or ' '
                        filldone:
051D C3                         ret
                        ;
                        scanq:                          ;count '?'s in fcb, return ZF set if found
051E 8B3EAE12                   mov     di,fcbadr
0522 47                         inc     di              ;point to first char of filename
0523 B90B00                     mov     cx,11           ;11 chars to check
0526 B03F                       mov     al,'?'
0528 F2AE               repnz   scas    al
052A C3                         ret
                        ;
                        ;       ************************************
                        ;       *                                  *
                        ;       *     user CPU state routines      *
                        ;       *                                  *
                        ;       ************************************
                        ;
                        chkreg:                         ;if reg name in [ax] is valid, return with
                                                        ;register number in regnum
                                                        ;else go to error processor
052B B90E00                     mov     cx,totreg+1     ;number of names to check + 1
052E BF4D01                     mov     di,offset regname
0531 0E                         push    cs
0532 07                         pop     es
0533 F2AF               repnz   scas    ax
0535 E320       0557            jcxz    checkerr        ;not a valid reg name
0537 BA0D00                     mov     dx,totreg
053A 2BD1                       sub     dx,cx
053C 89165512                   mov     regnum,dx       ;save reg number
0540 C3                         ret
                        ;
                        checkflag:                      ;check for valid flag name
0541 B90A00                     mov     cx,nflag+1      ;number of names to check + 1
0544 BF6701                     mov     di,offset flagname
0547 0E                         push    cs
0548 07                         pop     es
0549 F2AE               repnz   scas    al
054B E30A       0557            jcxz    checkerr        ;not a valid flag name
054D BA0900                     mov     dx,nflag
0550 2BD1                       sub     dx,cx
0552 89165512                   mov     regnum,dx       ;save flag number
0556 C3                         ret
                        ;
                        checkerr:
0557 E985FD     02DF            jmp     err
                        ;
                        setreg:                         ;set reg whose number is in [cx] to value in [ax]
```

```
                                        mov     si,offset userrej
                                        add     si,cx
                                        add     si,cx
                                        mov     [si],ax
                                        ret
                                ;
                                printflags:                     ;print values of flags
                                        mov     cx,0
                                pf0:
                                        push    cx              ;save count
                        0586            call    printflag
                                        pop     cx
                                        inc     cx
                                        cmp     cx,9
                        0567            jb      pf0
                                        ret
                                ;
                                setflag:                        ;set flag whose # is in [cx] to value in [bx]
                                        mov     si,offset flagbits
                                        add     si,cx
                                        lods    al
                                        mov     cl,al
                                        mov     ax,0fffeh
                                        ror     ax,cl
                                        ror     bx,cl
                                        and     userfl,ax
                                        or      userfl,bx
                                        ret
                                ;
                                printflagname:                  ;print flag name whose # is in [cx]
                                        mov     si,offset flagname
                                        add     si,cx
                                        lods    al              ;get flag name
                        0326            call    conout
                                        ret
                                ;
                                getflag:                        ;check flag whose # is in [cx]
                                                                ;return with ZF set if flag is set
                                        mov     ax,1
                                        mov     si,offset flagbits
                                        add     si,cx
                                        mov     cl,[si]         ;get flagbits (flagnum)
                                        ror     ax,cl           ;get mask into position
                                        and     ax,userfl       ;see if bit set in CPU state
                                        ret
                                ;
                                printflagval:                   ;print value of flag (as 0 or 1) whose # is in [cx]
                        0597            call    getflag
                                        mov     al,'0'
                        0582            jz      pf2             ;if flag not set, print '0'
                                        mov     al,'1'          ;else print '1'
                                pf2:
```

```
902
903   05B2 E871FD      0326          call    conout
904   05B5 C3                        ret
905                            ;
906                            printflag:                      ;print flag (as flagname of '-') whose # is in [cx]
907   05B6 BE6701                    mov     si,offset flagname
908   05B9 03F1                      add     si,cx             ;point to flag name
909   05BB FF34                      push    word ptr [si]     ;save flag char
910   05BD E807FF      0597          call    getflag
911   05C0 58                        pop     ax                ;get flag char
912   05C1 7502        05C5          jnz     pname             ;if flag set, use flag char
913   05C3 B02D                      mov     al,'-'            ;else print hyphen
914                            pname:
915   05C5 E85EFD      0326          call    conout
916   05C8 C3                        ret
917                            ;
918                            PREG1:                          ;PRINT FIRST 6 REGISTER NAMES (FOR 40 COLUMNS)
919   05C9 B90000                    MOV     CX,0
920   05CC C606531206                MOV     NREG,6
921   05D1 EB0D        05E0          JMPS    PR0
922                            PREG2:                          ;PRINT NEXT 7 REGISTER NAMES (FOR 40 COLUMNS)
923   05D3 B90600                    MOV     CX,6
924   05D6 EB03        05DB          JMPS    PR00
925                            printregs:                      ;print register values
926   05D8 B90000                    mov     cx,0
927                            PR00:
928   05DB C606531200                MOV     NREG,13
929                            pr0:
930   05E0 E82900      060C          call    testregcl         ;see if reg should be printed
931   05E3 7308        05ED          jnb     pr2               ;don't print if carry not set
932   05E5 51                        push    cx
933   05E6 E80C00      05F5          call    printregval
934   05E9 E843FE      042F          call    blank
935   05EC 59                        pop     cx
936                            pr2:
937   05ED 41                        inc     cx
938   05EE 3A0E5312                  CMP     CL,NREG
939   05F2 72EC        05E0          jb      pr0
940   05F4 C3                        ret
941                            ;
942                            printregval:                    ;print value of reg whose # is in [cx]
943   05F5 BE9C11                    mov     si,offset userreg
944   05F8 03F1                      add     si,cx
945   05FA 03F1                      add     si,cx
946   05FC AD                        lods    ax
947   05FD E875FE      0475          call    printword
948   0600 C3                        ret
949                            ;
950                            printregname:                   ;print name of reg whose # is in [cx]
951   0601 BE4301                    mov     si,offset regname
952   0604 03F1                      add     si,cx
953   0606 03F1                      add     si,cx
954   0608 E832FE      043D          call    printm
```

```
 955
 956   060B C3                          ret
 957                                    ;
 958                                    testregcl:                      ;see if reg whose # is in [cl] should be printed
 959                                                                    ;return with carry set if so
 960   060C F6065412FF                  test    segflag,0ffh
 961   0611 7509          061C          jnz     printit                 ;print all reg's if segflag set
 962   0613 80F908                      cmp     cl,11                   ;otherwise, see if [cl] has seg reg #
 963   0616 7704          061C          ja      printit
 964   0618 80F908                      cmp     cl,8
 965   061B C3                          ret
 966                                    printit:
 967   061C F9                          stc
 968   061D C3                          ret
 969                                    ;
 970                                    SETUPHDR:
 971   061E E8FCFD        041D          call    crlf
 972   0621 B90800                      mov     cx,11
 973   0624 E80EFE        0435          call    tabs                    ;leave space for flags
 974   0627 B90000                      mov     cx,0
 975   062A C3                          RET
 976                                    ;
 977                                    PREGHDR1:                       ;PRINT HEADER FOR FIRST 6 REGS (FOR 40 COL)
 978   062B E8F0FF        061E          CALL    SETUPHDR
 979   062E C606531206                  MOV     NREG,6
 980   0633 EB10          0645          JMPS    PRH0
 981                                    PREGHDR2:                       ;PRINT HEADER FOR NEXT 7 REGISTERS
 982   0635 E8F7FD        042F          CALL    BLANK
 983   0638 B90600                      MOV     CX,6
 984   063B EB03          0640          JMPS    PRH00
 985                                    printregheader:                 ;print header for registers
 986   063D E8DEFF        061E          CALL    SETUPHDR
 987                                    PRH00:
 988   0640 C606531200                  MOV     NREG,13
 989                                    prh0:
 990   0645 E8C4FF        060C          call    testregcl               ;see if reg should be printed
 991   0648 7308          0655          jnb     prh1                    ;don't print if carry not set
 992   064A 51                          push    cx
 993   064B E8B3FF        0601          call    printregname
 994   064E B90300                      mov     cx,3
 995   0651 E8E1FD        0435          call    tabs
 996   0654 59                          pop     cx
 997                                    prh1:
 998   0655 41                          inc     cx
 999   0656 3A0E5312                    CMP     CL,NREG
1000   065A 72E9          0645          jb      prh0
1001   065C C3                          ret
1002                                    ;
1003                                    printinstr:                     ;disassemble instruction at [cs:ip]
1004   065D 2E8E06AC11                  mov     es,usercs
1005   0662 2E8B368411                  mov     si,userip
1006   0667 F6063912FF                  test    disempresent,0ffh
1007   066C 7410          067E          jz      pi1
```

```
1008
1009  066E 06                        push    es
1010  066F 56                        push    si
1011  0670 F6065412FF                test    segflag,0ffh
1012  0675 7403            067A      jz      pi0
1013  0677 E8A3FD          041D      call    crlf
1014                           pi0:
1015  067A E88300          1400      call    disem
1016  067D C3                        ret
1017                           pi1:
1018  067E 268A04                    mov     al,es:[si]
1019  0681 E8F8FD          047C      call    printbyte
1020  0684 C3                        ret
1021                           ;
1022                           printseginfo:                   ;print name, start and end address of segment whose
1023                                                           ;number is in [al] if length is non-zero.
1024  0685 8AC8                      mov     cl,al            ;save seg number
1025  0687 B306                      mov     bl,6
1026  0689 F6E3                      mul     bl               ;6 bytes per segment in base page
1027  068B 057212                    add     ax,offset basepagesave
1028  068E 8BF0                      mov     si,ax            ;si now points to entry in base page
1029  0690 AD                        lods    ax               ;get low 16 bits of length
1030  0691 8BD8                      mov     bx,ax            ;save in bx
1031  0693 AC                        lods    al               ;get high nibble of length
1032  0694 8AD0                      mov     dl,al            ;save it
1033  0696 B400                      mov     ah,0
1034  0698 0BC3                      or      ax,bx            ;test for zero length
1035  069A 7430            06CC      jz      psiret           ;if zero, no display
1036  069C AD                        lods    ax               ;get base
1037  069D 53                        push    bx               ;save low (length)
1038  069E 52                        push    dx               ;save high (length)
1039  069F 50                        push    ax               ;save base
1040  06A0 B500                      mov     ch,0             ;zero high byte of segment #
1041  06A2 D1E1                      shl     cx,1             ;* 2 (2 bytes per segment name)
1042  06A4 81C17901                  add     cx,offset segnames      ;cx now points to segment name
1043  06A8 51                        push    cx               ;save it
1044  06A9 E871FD          041D      call    crlf
1045  06AC 5E                        pop     si
1046  06AD E88DFD          043D      call    printm           ;print segment name
1047  06B0 E87CFD          042F      call    blank
1048  06B3 07                        pop     es               ;get base
1049  06B4 06                        push    es               ;save base
1050  06B5 BF0000                    mov     di,0
1051  06B8 E8AEFD          0469      call    printdword       ;print start address
1052  06BB E871FD          042F      call    blank
1053  06BE 58                        pop     bx               ;get base
1054  06BF 58                        pop     ax               ;get high (len)
1055  06C0 B10C                      mov     cl,12
1056  06C2 D3E0                      shl     ax,cl            ;move ls nibble of al to ms nibble of ah
1057  06C4 03C3                      add     ax,bx            ;add ms nibble of length to base
1058  06C6 8EC0                      mov     es,ax
1059  06C8 5F                        pop     di               ;get low (len)
1060  06C9 E89DFD          0469      call    printdword       ;print end address
```

```
1061
1062                                psiret:
1063    06CC C3                             ret
1064                                ;
1065                                setcpustate:                    ;set users regs after G command
1066    06CD A1B311                         MOV     AX,CCPSS
1067    06D0 2EA3B011                       MOV     USERSS,AX
1068    06D4 A1BF11                         MOV     AX,CCPSP
1069    06D7 2EA3A411                       MOV     USERSP,AX       ;RESET USER STACK POINTER TO CCP STACK
1070    06DB 8BC3                           mov     ax,bx           ;get ds base in [ax] (returned from bdos load)
1071    06DD 2EA3AE11                       mov     userds,ax       ;set user's ds
1072    06E1 2EA3AC11                       mov     usercs,ax
1073    06E5 2EA3B211                       mov     useres,ax
1074    06E9 A35A12                         mov     typelseg,ax
1075    06EC A35E12                         mov     type2seg,ax
1076    06EF 8EC0                           mov     es,ax
1077    06F1 26F606050001                   test    es:byte ptr .5, 1       ;test 8080 flag
1078    06F7 7405           06FE            jz      not8080
1079    06F9 B80001                         mov     ax,100h         ;default for userip, lasloc, disloc
1080    06FC EB2D           072B            jmps    setdone
1081                                not8080:
1082    06FE 26A10300                       mov     ax,es:.3        ;get cs base
1083    0702 2EA3AC11                       mov     usercs,ax
1084    0706 A35A12                         mov     typelseg,ax
1085    0709 26A10F00                       mov     ax,es:.15       ;get es base
1086    070D 08C0                           or      ax,ax
1087    070F 7404           0715            jz      sc1
1088    0711 2EA3B211                       mov     useres,ax       ;set it if there was one
1089                                sc1:
1090    0715 26A11500                       mov     ax,es:.21       ;get ss base
1091    0719 08C0                           or      ax,ax
1092    071B 740E           072B            jz      setdone
1093    071D 2EA3B011                       mov     userss,ax       ;set it if there was one
1094    0721 26A11200                       mov     ax,es:.18       ;get stack length
1095    0725 2EA3A411                       mov     usersp,ax       ;set user's sp
1096    0729 28C0                           sub     ax,ax           ;userip, lasloc, disloc = 0 for non-8080 model
1097                                setdone:
1098    072B 2EA3B411                       mov     userip,ax
1099    072F A33112                         mov     lasloc,ax
1100    0732 C3                             ret
1101                                ;
1102                                ;       ***********************************************
1103                                ;       *                                             *
1104                                ;       *     breakpoint/single step procedures       *
1105                                ;       *                                             *
1106                                ;       ***********************************************
1107                                ;
1108                                setbp:                          ;set breakpoint at address stored in dword at [di]
1109    0733 C435                           les     si,[di]         ;point [es]:[si] to breakpoint location
1110    0735 B0CC                           mov     al,0cch         ;int 3 instruction
1111    0737 268604                         xchg    al,es:[si]      ;set breakpoint and fetch instruction
1112    073A 884504                         mov     4[di],al        ;save user instruction in breakpoint table
1113    073D FE062612                       inc     bpcnt           ;increment breakpoint count
```

```
1114
1115  0741 C3                           ret
1116                                    ;
1117                                    bpclear:                        ;clear breakpoint table
1118  0742 28C9                         sub     cx,cx
1119  0744 8A0E2612                     mov     cl,bpcnt                ;get # of bp's to clear
1120  0748 BE2712                       mov     si,offset brklloc       ;point to bp table
1121                                    bpcloop:
1122  074B E30F               075C      jcxz    bpend                   ;0..quit
1123  074D AD                           lods    ax                      ;get bp offset
1124  074E 50                           push    ax                      ;save it
1125  074F AD                           lods    ax                      ;get bp segment
1126  0750 8EC0                         mov     es,ax
1127  0752 AC                           lods    al                      ;get inst byte
1128  0753 5F                           pop     di                      ;get bp offset
1129  0754 AA                           stos    al                      ;store user instruction back
1130  0755 E2F4               0748      loop    bpcloop
1131  0757 C606261200                   mov     bpcnt,0                 ;zero bp counter
1132  075C C3                 bpend:    ret
1133                                    ;
1134                                    BPV:
1135  075D F6066F12FF                   TEST    SAVEVECFLAG,0FFH
1136  0762 741C               0780      JZ      BPVECTRET
1137                                    bpvect:                         ;set up breakpoint/single step vectors
1138  0764 E81A00             0781      call    savevect
1139  0767 BA0000                       mov     dx,0
1140  076A 8EC2                         mov     es,dx                   ;make sure dest is absolute 0
1141  076C BF0400                       mov     di,4                    ;set up single step vector
1142  076F B8DF07                       mov     ax,offset ssentry       ;single step entry point
1143  0772 AB                           stos    ax                      ;save at ss vector
1144  0773 8CC8                         mov     ax,cs
1145  0775 AB                           stos    ax                      ;save cs
1146  0776 BF0C00                       mov     di,12                   ;set up breakpoint vector
1147  0779 B8D707                       mov     ax,offset breakentry    ;set up bp vector
1148  077C AB                           stos    ax                      ;save at bp vector
1149  077D 8CC8                         mov     ax,cs
1150  077F AB                           stos    ax                      ;save cs
1151                                    BPVECTRET:
1152  0780 C3                           ret
1153                                    ;
1154                                    savevect:                       ;save previous contents of 0:4 thru 0:f
1155  0781 BE0400                       mov     si,4
1156  0784 BF1A12                       mov     di,offset vectorsave
1157  0787 1E                           push    ds
1158  0788 07                           pop     es                      ;point to ddt segment
1159  0789 B90C00                       mov     cx,12
1160  078C 1E                           push    ds
1161  078D BA0000                       mov     dx,0
1162  0790 8EDA                         mov     ds,dx
1163  0792 F3A4               rep       movs    al,al
1164  0794 1F                           pop     ds
1165                                    svret:
1166  0795 C3                           ret
```

```
1167
1168                                    ;
1169                                    restorevect:                    ;restore previous contents of 0:4 thru 0:f
1170    0796 F6066F12FF                         test    savevecflag,0ffh
1171    079B 7410               07AD            jz      rvret
1172    079D BE1A12                             mov     si,offset vectorsave
1173    07A0 BF0400                             mov     di,4
1174    07A3 B90C00                             mov     cx,12
1175    07A6 BA0000                             mov     dx,0
1176    07A9 8EC2                               mov     es,dx
1177    07AB F3A4                       rep     movs    al,al
1178                                    rvret:
1179    07AD C3                                 ret
1180                                    ;
1181                                    SETDEFSEG:                      ;SET DEFAULT TYPE1SEG, LASLOC
1182    07AE 2E8B3EB411                         mov     di,userip
1183    07B3 893E3112                           mov     lasloc,di       ;set disassembler offset
1184    07B7 2E8E06AC11                         mov     es,usercs
1185    07BC 8C065A12                           mov     type1seg,es     ;set type1seg segment
1186    07C0 C3                                 RET
1187                                    ;
1188                                    breakaddr:                      ;print address where break occurred
1189    07C1 E859FC             0410            call    crlf
1190    07C4 B02A                               mov     al,'*'
1191    07C6 E85DF8             0326            call    conout
1192    07C9 2EA1AE11                           mov     ax,userds
1193    07CD A35E12                             mov     type2seg,ax     ;set type2 segment to userds
1194    07D0 E8DBFF             07AE            CALL    SETDEFSEG
1195    07D3 E893FC             0469            call    printdword      ;print break address
1196    07D6 C3                                 ret
1197                                    ;
1198                                    breakentry:                     ;breakpoint entry
1199    07D7 2EC606BC1101                       mov     breakfl,1
1200    07DD EB06               07E5            jmps    savecpu         ;common code for break and single step
1201                                    ssentry:                        ;single step entry
1202    07DF 2EC606BC1100                       mov     breakfl,0
1203                                    savecpu:
1204    07E5 2EA39C11                           mov     userax,ax
1205    07E9 2E891E9E11                         mov     userbx,bx
1206    07EE 2E890EA011                         mov     usercx,cx
1207    07F3 2E8916A211                         mov     userdx,dx
1208    07F8 2E8936A811                         mov     usersi,si
1209    07FD 2E893EAA11                         mov     userdi,di
1210    0802 2E892EA611                         mov     userbp,bp
1211    0807 2E8926A411                         mov     usersp,sp
1212    080C 2E8C06B211                         mov     useres,es
1213    0811 2E8C1EAE11                         mov     userds,ds
1214    0816 2E8C16B011                         mov     userss,ss
1215    081B 8BEC                               mov     bp,sp
1216    081D 8B4600                             mov     ax,[bp]
1217    0820 2EA3B411                           mov     userip,ax
1218    0824 8B4602                             mov     ax,2[bp]
1219    0827 2EA3AC11                           mov     usercs,ax
```

```
1220
1221   082B 894604                      mov     ax,4[bp]
1222   082E 2EA3B611                    mov     userfl,ax
1223                                    ;
1224   0832 8CC8                        mov     ax,cs
1225   0834 8ED8                        mov     ds,ax
1226   0836 8ED0                        mov     ss,ax
1227   0838 BC1013                      mov     sp,offset stackp
1228                                    ;
1229   083B F6066B1202                  test    sysif,ifmaskd   ;see whether interrupts should be enabled
1230   0840 7401          0843          jz      sav0
1231   0842 FB                          sti
1232                                sav0:
1233   0843 2E8306A41106                add     usersp,6        ;to make up for stacked cs, ip, and fl
1234   0849 FC                          cld
1235   084A E849FF        0796          call    restorevect
1236   084D 2EF606BC11FF                test    breakfl,0ffh
1237   0853 741A          086F          jz      sst0
1238                                    ;
1239                                break0:                         ;continuation of break processing
1240   0855 2EFF0EB411                  dec     userip          ;adjust user ip for breakpoint instr
1241   085A E8E5FE        0742          call    bpclear         ;clear all breakpoints
1242   085D F6064912FF                  test    skipbdos,0ffh   ;were we originally in trace mode?
1243   0862 7506          086A          jnz     sst00           ;if so, continue tracing
1244                                    ;
1245                                tracedone:
1246   0864 E85AFF        07C1          call    breakaddr
1247   0867 E93DFA        02A7          jmp     start
1248                                    ;
1249                                sst00:                          ;get here on breakpoint on return from bdos
1250   086A C606491200                  mov     skipbdos,0      ;no longer tracing thru bdos
1251                                sst0:                           ;continuation of single step processing
1252   086F 2E8126B611FF                and     userfl,0feffh   ;clear user trap flag
1253        FE
1254   0876 F6054A1201                  TEST    USERIFOFF,1
1255   087B 740C          0889          JZ      SST1
1256   087D C6054A1200                  MOV     USERIFOFF,0
1257   0882 2E810EB61100                OR      USERFL,200H     ;RESTORE USER IF
1258        02
1259                                SST1:
1260   0889 FF0E4612                    dec     tracecount
1261   088D 74D5          0864          jz      tracedone
1262   088F E876FB        0408          CALL    CTLCHEK         ;CHECK FOR ABORT
1263   0892 F6054812FF                  test    traceprint,0ffh
1264   0897 7403          089C          jz      tracerestore
1265   0899 E80608        10A2          call    xnohdr          ;display regs without header
1266                                    ;
1267                                tracerestore:                   ;enter here when in trace mode
1268   089C 2E8E06AC11                  mov     es,usercs
1269   08A1 2E8B36B411                  mov     si,userip
1270   08A6 A11201                      mov     ax,word ptr .bdosintloc ;get bdos interrupt instruction
1271   08A9 263B04                      cmp     ax,es:[si]      ;see if instruction to be traced is bdos int
1272   08AC 7518          08C6          jnz     tr00
```

```
1273
1274   08AE 8C0E2912                    mov     brklseg,ss
1275   08B2 83C602                      add     si,2            ;point to instruction after bdos int
1276   08B5 89362712                    mov     brklloc,si
1277   08B9 BF2712                      mov     di,offset brklloc
1278   08BC E874FE        0733          call    setbp           ;set breakpoint at return from bdos
1279   08BF C606491201                  mov     skipbdos,1      ;so we know we were in trace mode when we hit bp
1280   08C4 EB39          08FF          jmps    rstore          ;without setting single step flag
1281                            tr00:
1282   08C6 2EA1B611                    MOV     AX,USERFL
1283   08CA 0D0001                      OR      AX,100H         ;SET TRACE FLAG
1284   08CD A90002                      TEST    AX,200H         ;IS USER IF SET?
1285   08D0 7429          08FB          JZ      TR01
1286   08D2 25FFFD                      AND     AX,NOT 200h     ;CLEAR IT (SO WE DON'T END UP IN INT HANDLER)
1287   08D5 2E8E06AC11                  MOV     ES,USERCS
1288   08DA 2E8B36B411                  MOV     SI,USERIP
1289   08DF 268A1C                      MOV     BL,ES:[SI]      ;GET INSTRUCTION TO EXECUTE
1290   08E2 80FBFA                      CMP     BL,0FAH         ;IS IT CLI?
1291   08E5 7414          08FB          JZ      TR01
1292   08E7 80FBCF                      CMP     BL,0CFH         ;IRET?
1293   08EA 740F          08FB          JZ      TR01
1294   08EC 80FB9D                      CMP     BL,09DH         ;POPF?
1295   08EF 740A          08FB          JZ      TR01
1296   08F1 80FBCD                      CMP     BL,0CDH         ;INT?
1297   08F4 7405          08FB          JZ      TR01
1298   08F6 C6054A1201                  MOV     USERIFOFF,1     ;SET FLAG SO DDT86 WILL TURN IF BACK ON
1299                            TR01:
1300   08FB 2EA3B611                    MOV     USERFL,AX
1301                            rstore:                         ;enter here when in G mode
1302   08FF E85BFE        0750          CALL    BPV
1303   0902 FA                          cli
1304   0903 BC9C11                      mov     sp,offset userreg       ;point to reg save area
1305   0906 58                          pop     ax
1306   0907 5B                          pop     bx
1307   0908 59                          pop     cx
1308   0909 5A                          pop     dx
1309   090A 2E8F06BA11                  pop     savesp
1310   090F 5D                          pop     bp
1311   0910 5E                          pop     si
1312   0911 5F                          pop     di
1313   0912 1F                          pop     ds              ;throw away cs
1314   0913 1F                          pop     ds
1315   0914 2E8F06B811                  pop     savess
1316   0919 07                          pop     es
1317   091A 2E8E16B811                  mov     ss,savess       ;restore stack
1318   091F 2E8B26BA11                  mov     sp,savesp
1319   0924 2EFF36B611                  push    userfl          ;flags
1320   0929 2EFF36AC11                  push    usercs          ;cs
1321   092E 2EFF36B411                  push    userip          ;ip
1322   0933 CF                          iret                    ;transfer to user program
1323                            ;
1324                            ;       ************************************
1325                            ;       *                                  *
```

```
1326
1327                            ;       *     miscellaneous routines     *
1328                            ;       *                                *
1329                            ;       **********************************
1330                            ;
1331                            delim:
1332  0934 3C0D                         cmp     al,eol
1333  0936 740A         0942            jz      delret
1334  0938 3C2C                         cmp     al,',
1335  093A 7406         0942            jz      delret
1336  093C 3C20                         cmp     al,' '
1337  093E 7402         0942            jz      delret
1338  0940 3C3A                         cmp     al,':'
1339                            delret:
1340  0942 C3                           ret
1341                            ;
1342                            hexcon:
1343  0943 2C30                         sub     al,'0'
1344  0945 3C0A                         cmp     al,10
1345  0947 720A         0953            jb      hexret
1346  0949 04F9                         add     al,('0' - 'A' + 10) and 0ffh
1347  094B 3C10                         cmp     al,16
1348  094D 7305         0954            jnb     hexerr
1349  094F 3C0A                         cmp     al,10
1350  0951 7201         0954            jb      hexerr
1351                            hexret:
1352  0953 C3                           ret
1353                            hexerr:
1354  0954 E988F9       020F            jmp     err
1355                            ;
1356                            plmset:                         ;get here when assembler wants to set memory
1357  0957 8BEC                         mov     bp,sp           ;for parameter fetching
1358  0959 8B4602                       mov     ax,2[bp]        ;get value in [al]
1359  095C 8E065A12                     mov     es,typelseg     ;segment used in A command is in typel seg
1360  0960 8B7E04                       mov     di,4[bp]        ;get offset from stack
1361  0963 E81A00       0980            call    setbyte         ;set and verify
1362  0966 47                           inc     di              ;increment offset
1363  0967 7503         096C            jnz     psret           ;if incremented offset is non-zero, return
1364  0969 E93BF9       0247            jmp     start           ;otherwise exit A command, since wrap occurred
1365                            psret:
1366  096C C20400                       ret     4               ;remove 2 parameters
1367                            ;
1368                            set8or16:                       ;set byte or word at es:[di] depending on wmode
1369  096F F6065D1201                   test    wmode,1
1370  0974 740A         0980            jz      setbyte
1371                                                            ;fall through to setword
1372                            ;
1373                            setword:                        ;set word at es:[di] to [ax] and verify
1374                            ;
1375                            ;       NOTE:  THIS CODE COULD BE REPLACED BY THE FOLLOWING 4 INSTRUCTIONS
1376                            ;              FOR SYSTEMS IN WHICH MEMORY CAN ONLY BE ADDRESSED BY WORDS.
1377                            ;              HOWEVER, THIS WILL WRAP AROUND AND MODIFY LOCATIONS 0 IF
1378                            ;              [DI] CONTAINS OFFFEH.
```

```
1379
1380                            ;
1381                            ;       MOV     ES:[DI],AX
1382                            ;       CMP     ES:[DI],AX
1383                            ;       JNZ     BADVER
1384                            ;       RET
1385                            ;
1386   0976 50                          push    ax              ;save hi byte
1387   0977 E80600      0980            call    setbyte         ;set low byte
1388   097A 58                          pop     ax
1389   097B 8AC4                        mov     al,ah
1390   097D 47                          inc     di              ;point to next location
1391   097E 7408        0988            jz      sret            ;don't set byte if wraparound occurred
1392                                                            ;fall thru to setbyte
1393                            ;
1394                            setbyte:                        ;set byte at es:[di] to [al] and verify
1395   0980 268805                      mov     es:[di],al      ;store byte
1396   0983 263805                      cmp     es:[di],al      ;see if it got stored
1397   0986 7501        0989            jnz     badver
1398                            sret:
1399   0988 C3                          ret
1400                            ;
1401                            badver:
1402   0989 57                          push    di
1403   098A 06                          push    es
1404   098B BEE601                      mov     si,offset verm
1405   098E E8ACFA      043D            call    printm          ;print verify error message
1406   0991 07                          pop     es
1407   0992 5F                          pop     di
1408   0993 E8D3FA      0469            call    printdword
1409   0996 E90EF9      02A7            jmp     start
1410                            ;
1411                            inclor2:                        ;inc pointer at [si] by 1 or 2, depending on wmode
1412                                                            ;return with carry flag set if wrap occurred
1413   0999 A0D012                      mov     al,wmode
1414   099C 250100                      and     ax,1            ;mask to 0 or 1
1415   099F 40                          inc     ax              ;increment value is now 1 or 2
1416   09A0 0104                        add     [si],ax         ;increment pointer
1417   09A2 3904                        cmp     [si],ax         ;test for wraparound
1418   09A4 C3                          ret
1419                            ;
1420                            getnumber:                      ;get number from input buffer
1421                                                            ;returns:
1422                                                            ;bx = value of number from input
1423                                                            ;al = last character scanned (delimiter)
1424                                                            ;ah = blank flag (0 = blank, 1 = non-blank)
1425   09A5 2BDB                        sub     bx,bx           ;initialize value to 0
1426   09A7 8AE7                        mov     ah,bh           ;initialize blank flag to blank
1427                            getn0:
1428   09A9 E845FA      03F1            call    conin
1429   09AC E885FF      0934            call    delim           ;check for delimiter
1430   09AF 740D        09BE            jz      getnret         ;delimiter found, exit
1431   09B1 B104                        mov     cl,4
```

```
1432
1433  09B5 D3E3                      shl     bx,cl           ;make room for new nibble
1434  09B7 E883FF       0943         call    hexcon          ;convert ascii to binary
1435  09BA 02D8                      add     bl,al           ;add nibble
1436  09BC B401                      mov     ah,1            ;blank flag = non-blank
1437  09BE EBE8         09A9         jmps    getn0
1438                           getnret:
1439  09C0 C3                        ret
1440                           ;
1441                           getoffset:                      ;get offset from input line
1442                                                           ;offset is a non-blank number not followed by ":"
1443                                                           ;returns:
1444                                                           ;al = last char scanned (delimiter)
1445                                                           ;bx = value
1446  09C1 E8E3FF       09A5         call    getnumber       ;get value to bx
1447  09C4 0AE4                      or      ah,ah           ;check for blank entry
1448  09C6 7400         09D3         jz      geterr          ;don't allow blank entry
1449  09C8 3C3A                      cmp     al,':'          ;check delimiter for ":"
1450  09CA 7407         09D3         jz      geterr          ;don't allow ":" delimiter
1451  09CC C3                        ret
1452                           ;
1453                           getlastoffset:                  ;same as getoffset but delimiter must be a cr
1454  09CD E8F1FF       09BF         call    getoffset
1455  09D0 3C00                      cmp     al,eol
1456  09D2 7501         09D5         jnz     geterr
1457  09D4 C3                        ret
1458                           geterr:
1459  09D5 E909F9       02E1         jmp     err
1460                           ;
1461                           checkword:                      ;check for "W" following command letter
1462  09D8 E818FA       03F3         call    conin
1463  09DB 3C57                      cmp     al,'W'
1464  09DD 7506         09E5         jnz     chret
1465  09DF C6066D1201                mov     wmode,1
1466  09E4 C3                        ret
1467  09E5 FF0E0412            chret: dec     conptr          ;to rescan character
1468  09E9 C3                        ret
1469                           ;
1470                           checkreg:                       ;check for valid segment register prefix - <sr>:
1471                                                           ;called with:
1472                                                           ;bl = first char
1473                                                           ;bh = second char
1474                                                           ;returns:
1475                                                           ;si = offset to register, if found
1476                                                           ;zf = found flag (1 = found, 0 = not found)
1477  09EA 80CF80                    or      bh,80h          ;since they are defined like that
1478  09ED BD5D01                    mov     bp,offset segreg        ;point to seg reg names
1479  09F0 28F6                      sub     si,si           ;initialize index to 0
1480                           check0:
1481  09F2 3B1A                      cmp     bx,[bp+si]      ;is it a seg reg name
1482  09F4 740A         0A00         jz      checkret
1483  09F6 83C602                    add     si,2            ;point to next name
1484  09F9 83FE08                    cmp     si,8            ;check for done (4 seg reg names)
```

```
1485
1486   09FA 75F4          09F0          jnz     check0
1487   09FC 0BF6                        or      si,si           ;unset zero flag
1488                              checkret:
1489   09FE C3                          ret
1490                              ;
1491                              checksegreg:                    ;check for valid seg reg name
1492                                                              ;if found, return contents of seg reg
1493                                                              ;else reset input pointer for rescan
1494                                                              ;returns:
1495                                                              ;dx = seg reg value
1496                                                              ;zf = valid seg reg (1 = valid, 0 = not valid)
1497   09FF FF360412                    push    conptr          ;save input pointer for possible rescan
1498   0A03 E8E3F9        03F1          call    conin
1499   0A06 50                          push    ax
1500   0A07 E8E7F9        03F1          call    conin
1501   0A0A 50                          push    ax
1502   0A0B E8E3F9        03F1          call    conin
1503   0A0E 3C3A                        cmp     al,':'          ;valid seg reg must have colon
1504   0A10 5B                          pop     bx
1505   0A11 59                          pop     cx
1506   0A12 8AF9                        mov     bh,cl
1507   0A14 86DF                        xchg    bl,bh
1508   0A16 750C          0A24          jnz     notsr
1509   0A18 E8C0FF        09E8          call    checkreg        ;see if it's a valid name
1510   0A1B 7507          0A24          jnz     notsr
1511   0A1D 8DAC11                      mov     bp,offset usercs        ;point to saved user seg reg's
1512   0A20 8B12                        mov     dx,[bp+si]      ;get value of user seg reg
1513   0A22 59                          pop     cx              ;throw away saved input pointer
1514   0A23 C3                          ret
1515                              notsr:
1516   0A24 8F060412                    pop     conptr          ;reset for rescan
1517   0A28 C3                          ret
1518                              ;
1519                              getseganoff:                    ;get user location specification
1520                                                              ;may be one of the following forms:
1521                                                              ;<empty>
1522                                                              ;nnnn
1523                                                              ;sr:nnnn
1524                                                              ;mmmm:nnnn
1525                                                              ;if numreq set, <empty> is invalid
1526                                                              ;called with:
1527                                                              ;di = offset of 4 byte area containing <offset><segment>
1528                                                              ;numreq must have been initialized by calling routine
1529   0A29 E8D3FF        09FF          call    checksegreg     ;see if there is a segment prefix
1530   0A2C 7409          0A38          jz      gets0
1531   0A2E E874FF        09A5          call    getnumber       ;no segment prefix, check for number
1532   0A31 0AE4                        or      ah,ah           ;was there one?
1533   0A33 7412          0A47          jz      gets3
1534   0A35 8BD3                        mov     dx,bx           ;move number to dx
1535   0A37 3C3A                        cmp     al,':'          ;was delimiter a ':'
1536   0A39 7509          0A44          jnz     gets2
1537                              gets0:
```

```
1538
1539   0A3B E881FF        0A9F          call    getoffset       ;segment prefix present, must have number
1540   0A3E 8919                        mov     [di],bx         ;number goes to <offset>
1541   0A40 895502                      mov     2[di],dx        ;first number (or sr) goes to <segment>
1542   0A43 C3                          ret
1543                              gets2:
1544   0A44 8915                        mov     [di],dx         ;only one number, put it in <offset>
1545                              getsret:
1546   0A46 C3                          ret
1547                              gets3:
1548   0A47 F6066C12FF                  test    numreq,0ffh     ;blank field, see if ok
1549   0A4C 74F8          0A46          jz      getsret         ;ok, return with no change at [di]
1550   0A4E E98EF8        02DF          jmp     err             ;number was required
1551                              ;
1552                              ;       ******************************
1553                              ;       *                            *
1554                              ;       *     disk i/o routines      *
1555                              ;       *                            *
1556                              ;       ******************************
1557                              ;
1558                              readfile:                       ;read file in fcb into memory described in mcb
1559                                                              ;when done, mcb will have base and length of block to free
1560   0A51 A16612                      mov     ax,mcbbase
1561   0A54 A33C12                      mov     startreadseg,ax
1562   0A57 A34012                      mov     endreadseg,ax
1563   0A5A A34412                      mov     dmaseg,ax
1564   0A5D 28C0                        sub     ax,ax
1565   0A5F A33812                      mov     startreadoff,ax
1566   0A62 A33E12                      mov     endreadoff,ax
1567                              rf0:
1568   0A65 8B164412                    mov     dx,dmaseg
1569   0A69 E823F9        038F          call    setdmab         ;set dma base
1570   0A6C C70642120000                mov     dmaoff,0
1571                              rf1:
1572   0A72 8B164212                    mov     dx,dmaoff
1573   0A76 E80CF9        0385          call    setdma          ;set dma offset
1574   0A79 833E681208                  cmp     mcblen,8        ;8 paragraphs per sector
1575   0A7E 7239          0AB9          jb      readerr         ;if less than 8 pp's left, not enough memory
1576   0A80 BA5C00                      mov     dx,offset fcb
1577   0A83 E8DCF8        0362          call    readsec
1578   0A86 0AC0                        or      al,al           ;test value returned from bdos
1579   0A88 7528          0AB2          jnz     readdone
1580   0A8A 8306661208                  add     mcbbase,8       ;point mcb to next available paragraph
1581   0A8F 832E681208                  sub     mcblen,8        ;decrement # of available paragraphs
1582   0A94 A14212                      mov     ax,dmaoff
1583   0A97 047F                        add     al,7fh          ;add sector size - 1
1584   0A99 A33E12                      mov     endreadoff,ax
1585   0A9C A14412                      MOV     AX,DMASEG
1586   0A9F A34012                      MOV     ENDREADSEG,AX
1587   0AA2 810642128000                add     dmaoff,80h      ;increment dma offset
1588   0AA8 75C8          0A72          jnz     rf1             ;if no wrap occurred, simply continue
1589   0AAA 810644120010                add     dmaseg,1000h    ;else increment dma segment
1590   0AB0 EBB3          0A65          jmps    rf0
```

```
                    asmerr:
  1591  0B57 E9B5F7       02DF          jmp     err
  1592                           asment:                         ;arrive here on input error in pl/m
  1593  0B5A 8B2E0712                   mov     sp,asmsp_sav    ;reset stack to where it was
  1594  0B5E EBE0          0B40         jmps    asm0            ;go back for more input
  1595                           ;
  1596                           ;       ****************************
  1597                           ;       *                          *
  1598  4
  1599  0AC1 D3E8                       shr     ax,cl           ;divide offset by 16 - truncate ls nibble
  1600  0AC3 03064D12                   add     ax,startwriteseg        ;compute absolute paragraph number
  1601  0AC7 A33612                     mov     mcbbase,ax
  1602  0ACA A34412                     mov     dmaseg,ax
  1603  0ACD 8BD8                       mov     bx,ax           ;store start paragraph # in [bx]
  1604  0ACF A14F12                     mov     ax,endwriteoff
  1605  0AD2 D3E8                       shr     ax,cl
  1606  0AD4 03065112                   add     ax,endwriteseg  ;calculate absolute paragraph number of end
  1607  0AD8 2BC3                       sub     ax,bx           ;compute # of paragraphs to write
  1608  0ADA 7250          0B2C         jb      wferr           ;start can't be > end
  1609  0ADC A33812                     mov     mcblen,ax       ;store # paragraphs to write
  1610  0ADF BA5C00                     mov     dx,offset fcb
  1611  0AE2 E878F8        0350         call    delete
  1612  0AE5 F6067112FF                 TEST    ERRMODE,0FFH
  1613  0AEA 7408          0AF4         JZ      WF00
  1614  0AEC FEC0                       INC     AL              ;DID DELETE RETURN 0FFH?
  1615  0AEE 7504          0AF4         JNZ     WF00            ;IF NOT, OK
  1616  0AF0 0AE4                       OR      AH,AH           ;SEE IF EXTENDED OR PHYSICAL ERROR
  1617  0AF2 7538          0B2C         JNZ     WFERR           ;IF SO, DON'T CONTINUE
  1618                           WF00:
  1619  0AF4 BA5C00                     mov     dx,offset fcb
  1620  0AF7 E87CF8        0376         call    make
  1621                           wf0:
  1622  0AFA 8B164412                   mov     dx,dmaseg
  1623  0AFE E88EF8        038F         call    setdmab
  1624  0B01 C706421200000              mov     dmaoff,0        ;clear dma offset
  1625                           wf1:
  1626  0B07 8B164212                   mov     dx,dmaoff
  1627  0B0B E877F8        0385         call    setdma
  1628  0B0E BA5C00                     mov     dx,offset fcb
  1629  0B11 E853F8        0367         call    writesec
  1630  0B14 832E681208                 sub     mcblen,8        ;8 paragraphs per sector
  1631  0B19 7210          0B2B         jb      writedone
  1632  0B1B 810642128000               add     dmaoff,80h      ;increment dma pointer
  1633  0B21 75E4          0B07         jnz     wf1             ;loop if no wrap occurred
  1634  0B23 810644120010               add     dmaseg,1000h    ;if wrap occurred, increment dma segment
  1635  0B29 EBCF          0AFA         jmps    wf0
  1636                           writedone:
  1637  0B2B C3                         ret
  1638                           wferr:
  1639  0B2C E9B0F7        02DF         jmp     err
  1640                           ;
```

```
1641
1642                            eject
1643                            ;
1644                            ;       ************************************
1645                            ;       *                                  *
1646                            ;       *      a - assemble mnemonics      *
1647                            ;       *                                  *
1648                            ;       ************************************
1649                            ;
1650                            assm:
1651   0B2F F6060612FF                  test    assempresent,0ffh
1652   0B34 7421          0B57         jz      asmerr
1653   0B36 89260712                   mov     asmspsav,sp     ;save in case of error in pl/m
1654   0B3A BF5812                     mov     di,offset typelloc
1655   0B3D E8E9FF        0A29         call    getsegandoff    ;get start address
1656                            asm0:
1657   0B40 FF365A12                   push    typelseg        ;for pl/m call
1658   0B44 FF365812                   push    typelloc        ;for pl/m call
1659   0B48 E8B517        2300         call    assem           ;returns offset of next available byte
1660   0B4B 3B055812                   cmp     ax,typelloc     ;test for no input
1661   0B4F 7605          0B56         jna     asmret          ;done unless greater than original typelloc
1662   0B51 A35812                     mov     typelloc,ax     ;update typelloc
1663   0B54 EBEA          0B40         jmps    asm0
1664                           asmret:
1665   0B56 C3                         ret
1666                           asmerr:
1667   0357 E985F7        02DF         jmp     err
1668                           asment:                         ;arrive here on input error in pl/m
1669   0B5A 8B260712                   mov     sp,asmspsav     ;reset stack to where it was
1670   0B5E EBE0          0B40         jmps    asm0            ;go back for more input
1671                           ;
1672                           ;       ******************************
1673                           ;       *                            *
1674                           ;       *     B - BLOCK COMPARE      *
1675                           ;       *                            *
1676                           ;       ******************************
1677                           ;
1678                           BLOCKCOMPARE:
1679   0B60 BF5C12                     mov     di,offset type2loc
1680   0B63 E8C3FE        0A29         call    getsegandoff
1681   0B66 E856FE        09BF         call    getoffset       ;get end offset
1682   0B69 891E6012                   mov     usermax,bx
1683   0B6D 3C0D                       cmp     al,eol
1684   0B6F 7451          0BC2         jz      cmperr          ;need 3 arguments
1685   0B71 2B1E5C12                   sub     bx,type2loc
1686   0B75 724B          0BC2         jb      cmperr          ;error if start > end
1687   0B77 A15E12                     mov     ax,type2seg
1688   0B7A A36412                     mov     userseg2,ax     ;default to same seg as source
1689   0B7D BF5212                     mov     di,offset userloc2
1690   0B80 E8A6FE        0A29         call    getsegandoff    ;get destination address
1691   0B83 3C0D                       cmp     al,eol
1692   0B85 753B          0BC2         jnz     cmperr          ;error if more than 3 arguments
1693                           CMP0:
```

```
1694
1695  0B87 C4365C12                 LES     SI,DWORD PTR TYPE2LOC
1696  0B8B 268A04                   MOV     AL,ES:[SI]
1697  0B8E C4366212                 LES     SI,DWORD PTR USERLOC2
1698  0B92 263A04                   CMP     AL,ES:[SI]
1699  0B95 7415         0BAC        JZ      CMPCONT
1700  0B97 E88EF8       0428        CALL    CRLFCHK
1701  0B9A BF5C12                   MOV     DI,OFFSET TYPE2LOC
1702  0B9D E82500       0BC5        CALL    PRINTERROR
1703  0BA0 E88CF8       042F        CALL    BLANK
1704  0BA3 E889F8       042F        CALL    BLANK
1705  0BA6 BF6212                   MOV     DI,OFFSET USERLOC2
1706  0BA9 E81900       0BC5        CALL    PRINTERROR
1707                          CMPCONT:
1708  0BAC FF065C12                 INC     TYPE2LOC
1709  0BB0 A15C12                   MOV     AX,TYPE2LOC
1710  0BB3 39066012                 CMP     USERMAX,AX
1711  0BB7 7208         0BC1        JC      CMPDONE
1712  0BB9 FF066212                 INC     USERLOC2
1713  0BBD 7402         0BC1        JZ      CMPDONE         ;PREVENT WRAPAROUND
1714  0BBF EBC6         0B87        JMPS    CMP0
1715                          ;
1716                          CMPDONE:
1717  0BC1 C3                       RET
1718                          CMPERR:
1719  0BC2 E91AF7       02DF        JMP     ERR
1720                          ;
1721                          PRINTERROR:                     ;PRINT DWORD AT [DI], BYTE POINTED TO BY DWORD
1722  0BC5 C439                     LES     DI,[DI]
1723  0BC7 268A05                   MOV     AL,ES:[DI]
1724  0BCA 50                       PUSH    AX              ;SAVE BYTE AT ES:DI
1725  0BCB E898F8       0469        CALL    PRINTDWORD
1726  0BCE E85EF8       042F        CALL    BLANK
1727  0BD1 58                       POP     AX
1728  0BD2 E8A7F8       047C        CALL    PRINTBYTE
1729  0BD5 C3                       RET
1730                          ;
1731                          ;       *******************************
1732                          ;       *                             *
1733                          ;       *     d - display memory      *
1734                          ;       *                             *
1735                          ;       *******************************
1736                          ;
1737                          display:
1738  0BD6 C6066C1200               mov     numreq,0        ;ok to have no entries
1739  0BDB A15E12                   mov     ax,type2seg
1740  0BDE A30B12                   mov     disseg,ax       ;default to type2 seg
1741  0BE1 E8F2FD       09D6        call    checkword
1742  0BE4 F6067012FF               TEST    COL40,0FFH
1743  0BE9 7411         0BFC        JZ      DIS01
1744  0BEB F6056D1201               TEST    WMODE,1
1745  0BF0 7505         0BF7        JNZ     DIS00
1746  0BF2 A01312                   MOV     AL,ND40
```

```
1747
1748  0BF5 EB08          0BFF          JMPS    DTS02
1749                          DIS00:
1750  0BF7 A01412                      MOV     AL,NOW40        ;CHARS PER LINE FOR DW IN 40 COL MODE
1751  0BFA EB03          0BFF          JMPS    DTS02
1752                          DIS01:
1753  0BFC A01212                      MOV     AL,NO80         ;16 BYTES PER LINE IN NORMAL MODE
1754                          DTS02:
1755  0BFF A21112                      MOV     LINEMAX,AL
1756  0C02 BF0912                      mov     di,offset disloc
1757  0C05 E821FE        0A29          call    getsegandoff
1758  0C08 3C2C                        cmp     al,','
1759  0C0A A10812                      mov     ax,disseg
1760  0C0D A35E12                      mov     type2seg,ax     ;update default type2 seg
1761  0C10 7505          0C17          jnz     dis0            ;must be cr, no dismax entered
1762  0C12 E886FD        09CB          call    getlastoffset   ;get dismax
1763  0C15 EB1A          0C31          jmps    dis1
1764                          dis0:
1765  0C17 8B1E0912                    mov     bx,disloc       ;no dismax entered, calculate default
1766  0C1B A01112                      MOV     AL,LINEMAX
1767  0C1E 8A0E1512                    MOV     CL,NLINES
1768  0C22 F6E1                        MUL     CL
1769  0C24 FEC8                        DEC     AL
1770  0C26 03D8                        ADD     BX,AX
1771  0C28 3B1E0912                    cmp     bx,disloc       ;see if we went over ffff
1772  0C2C 7303          0C31          jnb     dis1
1773  0C2E BBFFFF                      mov     bx,0ffffh       ;set dismax if we wrapped around
1774                          dis1:
1775  0C31 891E0D12                    mov     dismax,bx
1776                          disp3:
1777  0C35 E8F0F7        0428          CALL    CRLFCHK
1778  0C38 C43E0912                    les     di,dword ptr disloc
1779  0C3C 893E0F12                    mov     tdisp,di
1780  0C40 E826F8        0469          call    printdword
1781                          disp4:
1782  0C43 E8E9F7        042F          call    blank
1783  0C46 C4360912                    les     si,dword ptr disloc
1784  0C4A E810F8        045D          call    print8or16
1785  0C4D BE0912                      mov     si,offset disloc
1786  0C50 E846FD        0999          call    inclor2
1787  0C53 7216          0C6B          jb      disp6           ;stop if wrap occurred
1788  0C55 A10912                      mov     ax,disloc
1789  0C58 2B060F12                    sub     ax,tdisp        ;calculate # bytes printed on line
1790  0C5C 3A061112                    CMP     AL,LINEMAX      ;SEE IF LINE FULL
1791  0C60 7409          0C6B          jz      disp6
1792  0C62 A10912                      mov     ax,disloc
1793  0C65 3B050D12                    cmp     ax,dismax       ;check for done
1794  0C69 76D8          0C43          jna     disp4
1795                          disp6:
1796  0C6B E8C1F7        042F          call    blank
1797                          disp7:
1798  0C6E 8E060B12                    mov     es,disseg
1799  0C72 8B360F12                    mov     si,tdisp
```

```
1800
1801   0C76 268A04                    mov     al,es:[si]
1802   0C79 E803F7        0A4F        call    ascout
1803   0C7C FF060F12                  inc     tdisp
1804   0C80 7411          0C93        jz      disp8           ;stop if wrap occurred
1805   0C82 A10F12                    mov     ax,tdisp
1806   0C85 3B060912                  cmp     ax,disloc
1807   0C89 75E3          0C6E        jnz     disp7
1808   0C8B 3B060D12                  cmp     ax,dismax
1809   0C8F 7702          0C93        ja      disp8
1810   0C91 EBA2          0C35        jmps    disp3
1811                      disp8:
1812   0C93 C3                        ret
1813                      ;
1814                      ;       ****************************************
1815                      ;       *                                      *
1816                      ;       *     e - load program for execution   *
1817                      ;       *                                      *
1818                      ;       ****************************************
1819                      ;
1820   0C94 434D44        cmd             db      'CMD'
1821                      ;
1822                      execute:
1823   0C97 E857F7        03F1            call    conin
1824   0C9A 3C00                          cmp     al,eol          ;check for no filename
1825   0C9C 7474          0D12            jz      eerr            ;don't allow no filename
1826   0C9E FF0E0412                      dec     conptr          ;to rescan character
1827   0CA2 BF5C00                        mov     di,offset fcb
1828   0CA5 E8E9F7        0491            call    parse
1829   0CA8 7468          0D12            jz      eerr            ;no '?' or '*' allowed
1830   0CAA 3C00                          cmp     al,eol
1831   0CAC 7564          0D12            jnz     eerr            ;eol must follow filename
1832   0CAE 0E                            push    cs
1833   0CAF 07                            pop     es              ;set es = cs
1834   0CB0 26803E650020                  cmp     es:fcb+9,' '    ;see if filetype blank
1835   0CB6 7508          0CC3            jnz     ex0
1836   0CB8 BE940C                        mov     si,offset cmd
1837   0CBB BF6500                        mov     di,offset fcb+9
1838   0CBE B90300                        mov     cx,3
1839   0CC1 F3A4              rep         movs    al,al           ;set filetype to 'CMD' if empty
1840                          ex0:
1841   0CC3 BA5C00                        mov     dx,offset fcb
1842   0CC6 E876F6        033F            call    open            ;see if file exists
1843   0CC9 C6065A12FF                    mov     mcbext,0ffh     ;free all allocations below DDT86
1844   0CCE BA6612                        mov     dx,offset mcb
1845   0CD1 E809F6        03AD            call    freemem         ;free all memory previously allocated under DDT
1846   0CD4 BA5C00                        mov     dx,offset fcb
1847   0CD7 E808F6        03B2            call    load            ;load user program
1848   0CDA 53                            push    bx              ;save ds base
1849   0CDB E8EFF9        06CD            call    setcpustate
1850   0CDE 5A                            pop     dx              ;get ds base back
1851   0CDF E8ADF6        038F            call    setdmab         ;set dma base
1852   0CE2 BA8000                        mov     dx,80h
```

```
1853
1854  0CE5 E893F6       0385          call    setdma          ;default to 80h in user's DS
1855  0CE8 BA5C00                     MOV     DX,OFFSET FCB
1856  0CEB E850F6       034E          CALL    CLOSE
1857  0CEE C6066E1245                 mov     mode,'E'
1858  0CF3 BE0000                     mov     si,0
1859  0CF6 BF7212                     mov     di,offset basepagesave
1860  0CF9 1E                         push    ds
1861  0CFA 07                         pop     es              ;set es to ddt's segment
1862  0CFB 2EA1AE11                   mov     ax,userds
1863  0CFF 1E                         push    ds              ;save it
1864  0D00 8ED8                       mov     ds,ax
1865  0D02 B93000                     mov     cx,48
1866  0D05 F3A4             rep       movs    al,al           ;copy user's base page into ddt save area
1867  0D07 1F                         pop     ds              ;restore ds
1868  0D08 E8D302       0FDE          call    verify          ;display load info
1869  0D0B FF0E0412                   DEC     CONPTR          ;TO RESCAN CR
1870  0D0F E9CC00       0DDE          JMP     IFCB            ;TO CLEAR FCB
1871                          eerr:
1872  0D12 E9CAF5       02DF          jmp     err
1873                          ;
1874                          ;       ****************************
1875                          ;       *                          *
1876                          ;       *     f - fill memory      *
1877                          ;       *                          *
1878                          ;       ****************************
1879                          ;
1880                          fill:
1881  0D15 E8BEFC       09D6          call    checkword       ;check for 'FW'
1882  0D18 BF5C12                     mov     di,offset type2loc
1883  0D1B E80BFD       0A29          call    getsegandoff
1884  0D1E E89EFC       098F          call    getoffset       ;get end address
1885  0D21 891E6012                   mov     usermax,bx      ;save end address
1886  0D25 E8A3FC       09CB          call    getlastoffset   ;get fill constant
1887  0D28 F6066D1201                 test    wmode,1
1888  0D2D 7504         0D33          jnz     fil0
1889  0D2F 0AFF                       or      bh,bh           ;if not wmode, high byte must be 0
1890  0D31 7525         0D58          jnz     filerr
1891                          fil0:
1892  0D33 8B0E6012                   mov     cx,usermax      ;get end address
1893  0D37 2B0E5C12                   sub     cx,type2loc     ;compare for valid range
1894  0D3B 721B         0D58          jb      filerr          ;error if start > end
1895                          fill:
1896  0D3D C43E5C12                   les     di,dword ptr type2loc
1897  0D41 8BC3                       mov     ax,bx           ;get fill constant
1898  0D43 E829FC       096F          call    set8or16
1899  0D46 BE5C12                     mov     si,offset type2loc
1900  0D49 E84DFC       0999          call    inclor2
1901  0D4C 7209         0D57          jb      filret          ;stop if wrap occurred
1902  0D4E A15C12                     mov     ax,type2loc
1903  0D51 3B066012                   cmp     ax,usermax
1904  0D55 76E6         0D3D          jbe     fill
1905                          filret:
```

```
1906
1907  0D57 C3                           ret
1908                              filerr:
1909  0D58 E9D4F5       020F            jmp     err
1910                              ;
1911                              ;       ************************************
1912                              ;       *                                  *
1913                              ;       *      g - go to user program      *
1914                              ;       *                                  *
1915                              ;       ************************************
1916                              ;
1917                              gouser:
1918  0D5B 2EA1AC11                       mov     ax,usercs
1919  0D5F A31812                         mov     goseg,ax        ;default goseg = usercs
1920  0D62 2EA1B411                       mov     ax,userip
1921  0D66 A31612                         mov     goloc,ax        ;default goloc = userip
1922  0D69 C6066C1200                     mov     numreq,0        ;number not required in G command
1923  0D6E BF1612                         mov     di,offset goloc
1924  0D71 E885FC       0A29              call    getsegandoff    ;get start address
1925  0D74 3C0D                           cmp     al,eol
1926  0D76 7428         0DA3              jz      gorestore       ;if eol, no breakpoints set
1927  0D78 A11812                         mov     ax,goseg
1928  0D7B A32912                         mov     brk1seg,ax
1929  0D7E A32E12                         mov     brk2seg,ax      ;defaults for breakpoint segments = goseg
1930  0D81 BF2712                         mov     di,offset brk1loc
1931  0D84 E8A2FC       0A29              call    getsegandoff    ;get first breakpoint
1932  0D87 50                             push    ax              ;save terminating char
1933  0D88 BF2712                         mov     di,offset brk1loc
1934  0D8B E8A5F9       0733              call    setbp           ;save breakpoint in table
1935  0D8E 58                             pop     ax              ;get char
1936  0D8F 3C0D                           cmp     al,eol
1937  0D91 7410         0DA3              jz      gorestore       ;only one breakpoint
1938  0D93 BF2C12                         mov     di,offset brk2loc
1939  0D96 E890FC       0A29              call    getsegandoff    ;get second breakpoint
1940  0D99 3C0D                           cmp     al,eol
1941  0D9B 751C         0DB9              jnz     goerr           ;only 2 breakpoints allowed
1942  0D9D BF2C12                         mov     di,offset brk2loc
1943  0DA0 E890F9       0733              call    setbp           ;set second breakpoint
1944                              gorestore:
1945  0DA3 C606491200                     mov     skipbdos,0      ;make sure it's 0 since we aren't in I/O mode
1946  0DA8 A11812                         mov     ax,goseg
1947  0DAB 2EA3AC11                       mov     usercs,ax       ;usercs = goseg
1948  0DAF A11612                         mov     ax,goloc
1949  0DB2 2EA3B411                       mov     userip,ax       ;userip = goloc
1950  0DB6 E946F8       08FF              jmp     rstore          ;restore user CPU state
1951                              goerr:
1952  0DB9 E886F9       0742              call    bpclear         ;in case any were set
1953  0DBC E920F5       02DF              jmp     err
1954                              ;
1955                              ;
1956                              ;       ************************
1957                              ;       *                      *
1958                              ;       *     h - hex math     *
```

```
1959
1960                            ;       *                       *
1961                            ;       *************************
1962                            ;
1963                            hexmath:
1964   0D8F E8FDF8      098F            call    getoffset
1965   0DC2 53                          push    bx              ;save first value
1966   0DC3 E805FC      09CB            call    getlastoffset
1967   0DC6 58                          pop     ax              ;get first value
1968   0DC7 50                          push    ax              ;save a copy
1969   0DC8 03C3                        add     ax,bx
1970   0DCA 53                          push    bx              ;save second value
1971   0DCB 50                          push    ax              ;save sum
1972   0DCC E84EF6      0410            call    crlf
1973   0DCF 58                          pop     ax              ;get sum
1974   0DD0 E8A2F6      0475            call    printword       ;print sum
1975   0DD3 E859F6      042F            call    blank
1976   0DD6 5B                          pop     bx              ;get second value
1977   0DD7 58                          pop     ax              ;get first value
1978   0DD8 2BC3                        sub     ax,bx
1979   0DDA E898F6      0475            call    printword       ;print difference
1980   0DDD C3                          ret
1981                            ;
1982                            ;       ****************************************
1983                            ;       *                                      *
1984                            ;       *     i - input file control block     *
1985                            ;       *                                      *
1986                            ;       ****************************************
1987                            ;
1988                            ifcb:
1989   0DDE FF360412                    push    conptr          ;save input pointer
1990   0DE2 BF5C00                      mov     di,offset fcb
1991   0DE5 E8A9F6      0491            call    parse
1992   0DE8 3C0D                        cmp     al,eol
1993   0DEA 7504        0DF0            jnz     i0              ;only one filename
1994   0DEC FF0E0412                    dec     conptr          ;to rescan eol and blank second filename in fcb
1995                            i0:
1996   0DF0 BF6C00                      mov     di,offset fcb2
1997   0DF3 E8A6F6      049C            call    parse2          ;parse second filename
1998   0DF6 1E                          push    ds
1999   0DF7 07                          pop     es              ;point to DDT's ds
2000   0DF8 8F060412                    pop     conptr          ;restore input pointer
2001   0DFC BF8100                      mov     di,81h          ;move command tail to [es]:[di]
2002   0DFF 2BC9                        sub     cx,cx           ;zero count
2003                            i1:
2004   0E01 E8EDF5      03F1            call    conin           ;get char from command tail
2005   0E04 3C0D                        cmp     al,eol          ;end of command tail?
2006   0E06 7404        0E0C            jz      i2
2007   0E08 AA                          stos    al              ;store in user's base page
2008   0E09 41                          inc     cx              ;increment count
2009   0E0A EBF5        0E01            jmps    i1              ;loop until eol
2010                            i2:
2011   0E0C B000                        mov     al,0
```

```
2012
2013   0E0E AA                          stos    al                  ;store 0 at end of string
2014   0E0F 26880E8000                  mov     es:.80h,cl          ;store count at start of buffer
2015   0E14 803E6E1245                  cmp     mode,'E'
2016   0E19 750F          0E2A          jnz     idone               ;if no file loaded with E command, we're done
2017   0E1B BE5C00                      mov     si,offset fcb
2018   0E1E 8BFE                        mov     di,si
2019   0E20 2E8E06AE11                  mov     es,userds
2020   0E25 83C126                      add     cx,38               ;total bytes to move = # in command + 36 (fcb)
2021                                                                ; +2 (0 at end of command and count byte)
2022   0E28 F3A4                rep     movs    al,al               ;move fcb from ddt86 basepage to user's basepage
2023                            idone:
2024   0E2A C3                          ret
2025                            ierr:
2026   0E2B E981F4        02DF          jmp     err
2027                            ;
2028                            ;       ************************************
2029                            ;       *                                  *
2030                            ;       *     l - list assembly code       *
2031                            ;       *                                  *
2032                            ;       ************************************
2033                            ;
2034                            lassm:
2035   0E2E F6063912FF                  test    disempresent,0ffh
2036   0E33 746C          0EA1          jz      laserr
2037   0E35 C606371200                  mov     lascntsw,0          ;don't use count if end addr specified
2038   0E3A C6066C1200                  mov     numreq,0            ;ok if no entries
2039   0E3F A15A12                      mov     ax,typelseg
2040   0E42 A33312                      mov     lasseg,ax           ;default to typel seg
2041   0E45 BF3112                      mov     di,offset lasloc
2042   0E48 E8DEFB        0A29          call    getsegandoff
2043   0E4B 3C0D                        cmp     al,eol
2044   0E4D A13312                      mov     ax,lasseg
2045   0E50 A35A12                      mov     typelseg,ax         ;update default typel seg
2046   0E53 7405          0E5A          jz      las0
2047   0E55 E873FB        09CB          call    getlastoffset       ;if ',', get end address
2048   0E58 EB0E          0E68          jmps    las1
2049                            las0:
2050   0E5A C606371201                  mov     lascntsw,1          ;disassemble fixed # of instructions
2051   0E5F A01512                      MOV     AL,NLINES
2052   0E62 A23812                      MOV     LASCNT,AL
2053   0E65 BBFFFF                      mov     bx,0ffffh           ;set lasmax to big number
2054                            las1:
2055   0E68 891E3512                    mov     lasmax,bx
2056                            las2:
2057   0E6C 8B3E3112                    mov     di,lasloc
2058   0E70 3B3E3512                    cmp     di,lasmax
2059   0E74 772A          0EA0          ja      lasret
2060   0E76 57                          push    di
2061   0E77 E8AEF5        0428          CALL    CRLFCHK
2062   0E7A 5F                          pop     di
2063   0E7B 8E063312                    mov     es,lasseg
2064   0E7F 06                          push    es
```

```
2065
2066   0E80 57                          push    di              ;for disem call (PL/M)
2067   0E81 E8E5F5       0469           call    printdword
2068   0E84 E8A8F5       042F           call    blank
2069   0E87 E87605       1400           call    disem
2070   0E8A 3B063112                    cmp     ax,lasloc
2071   0E8E 7210         0EA0           jb      lasret          ;stop if wrap occurred
2072   0E90 A33112                      mov     lasloc,ax
2073   0E93 F6063712FF                  test    lascntsw,0ffh
2074   0E98 74D2         0E6C           jz      las2
2075   0E9A FE0E3812                    dec     lascnt
2076   0E9E 75CC         0E6C           jnz     las2
2077                           lasret:
2078   0EA0 C3                          ret
2079                           laserr:
2080   0EA1 E933F4       02D7           jmp     err
2081                           ;
2082                           ;       ***************************
2083                           ;       *                         *
2084                           ;       *     m - move block      *
2085                           ;       *                         *
2086                           ;       ***************************
2087                           ;
2088                           move:
2089   0EA4 BF5C12                      mov     di,offset type2loc
2090   0EA7 E87FFB       0A29           call    getsegandoff
2091   0EAA E812FB       09BF           call    getoffset       ;get end offset
2092   0EAD 891E6012                    mov     usermax,bx
2093   0EB1 3C0D                        cmp     al,eol
2094   0EB3 743A         0EEF           jz      moverr          ;need 3 arguments
2095   0EB5 2B1E5C12                    sub     bx,type2loc
2096   0EB9 7234         0EEF           jb      moverr          ;error if start > end
2097   0EBB A15E12                      mov     ax,type2seg
2098   0EBE A35412                      mov     userseg2,ax     ;default to same seg as source
2099   0EC1 BF6212                      mov     di,offset userloc2
2100   0EC4 E862FB       0A29           call    getsegandoff    ;get destination address
2101   0EC7 3C0D                        cmp     al,eol
2102   0EC9 7524         0EEF           jnz     moverr          ;error if more than 3 arguments
2103                           mov0:
2104   0ECB C4365C12                    les     si,dword ptr type2loc
2105   0ECF 268A04                      mov     al,es:[si]      ;get source byte
2106   0ED2 C43E6212                    les     di,dword ptr userloc2
2107   0ED6 E8A7FA       0980           call    setbyte         ;put destination byte
2108   0ED9 FF065C12                    inc     type2loc
2109   0EDD 740F         0EEE           jz      movret          ;don't allow wraparound
2110   0EDF FF066212                    inc     userloc2
2111   0EE3 7409         0EEE           jz      movret          ;don't allow wraparound in destination segment
2112   0EE5 A15C12                      mov     ax,type2loc
2113   0EE8 3B065612                    cmp     ax,usermax      ;check for done
2114   0EEC 76D3         0EC3           jna     mov0
2115                           movret:
2116   0EEE C3                          ret
2117                           moverr:
```

```
                                ;
0EEF E9EDF3         02DF        jmp     err
                                ;
                                ;       ************************
                                ;       *                      *
                                ;       *     r - read file    *
                                ;       *                      *
                                ;       ************************
                                ;
                            read:
0EF2 E8FCF4         03F1        call    conin           ;get first command char
0EF5 3C00                       cmp     al,eol          ;check for no input
0EF7 7450           0F49        jz      rerr            ;filename must be included in command
0EF9 FF0E0412                   dec     conptr          ;to rescan first char
0EFD BF5C00                     mov     di,offset fcb
0F00 E88EF5         0491        call    parse
0F03 7444           0F49        jz      rerr            ;no '?' or '*' allowed
0F05 3C00                       cmp     al,eol
0F07 7540           0F49        jnz     rerr            ;no parameters after filename
0F09 BA5C00                     mov     dx,offset fcb
0F0C E830F4         033F        call    open
0F0F C7065612FFFF               mov     mcblen,0ffffh   ;largest memory request
0F15 BA6612                     mov     dx,offset mcb
0F18 E879F4         0394        call    getmaxmem       ;get size of largest chuck of memory
0F1B BA5612                     mov     dx,offset mcb
0F1E E87DF4         039E        call    allocabsmem     ;allocate block returned from getmaxmem
0F21 E82DFB         0A51        call    readfile        ;read file into memory block
0F24 C6066A1200                 mov     mcbext,0        ;only free memory at mbase
0F29 BA6612                     mov     dx,offset mcb
0F2C E87EF4         03AD        call    freemem         ;free memory not used in read (read updated mcb)
0F2F A13C12                     mov     ax,startreadseg
0F32 A35E12                     mov     type2seg,ax     ;set default type2 segment to file just read
0F35 A35A12                     mov     type1seg,ax     ;also type1 segment
0F38 28C0                       sub     ax,ax
0F3A A30912                     mov     disloc,ax       ;display pointer offset = 0
0F3D A33112                     mov     lasloc,ax       ;list pointer offset = 0
0F40 C6066E1252                 mov     mode,'R'        ;last disk input was read (not execute)
0F45 E89500         0FDE        call    verify
0F48 C3                         ret
                            rerr:
0F49 E993F3         02DF        jmp     err
                                ;
                                ;       ************************
                                ;       *                      *
                                ;       *     s - set memory   *
                                ;       *                      *
                                ;       ************************
                                ;
                            setmem:
0F4C E887FA         09D6        call    checkword       ;check for 'SW'
0F4F BF5C12                     mov     di,offset type2loc
0F52 E8D4FA         0A29        call    getsegandoff
                            set0:
```

```
2171
2172  0F55 E8C5F4        0410          call    crlf
2173  0F58 C43E5C12                    les     di,dword ptr type2loc
2174  0F5C E80AF5        0469          call    printdword
2175  0F5F E8CDF4        042F          call    blank
2176  0F62 C4365C12                    les     si,dword ptr type2loc
2177  0F66 E8F4F4        0450          call    print8or16
2178  0F69 E8C3F4        042F          call    blank
2179  0F6C E868F4        03D7          call    getline
2180  0F6F E87FF4        03F1          call    conin
2181  0F72 3C0D                        cmp     al,eol
2182  0F74 741F          0F95          jz      set2
2183  0F76 3C2E                        cmp     al,'.'
2184  0F78 7423          0F9D          jz      setret
2185  0F7A FF0E0412                    dec     conptr          ;to rescan first character
2186  0F7E E84AFA        09CB          call    getlastoffset
2187  0F81 8BC3                        mov     ax,bx           ;get new value to ax
2188  0F83 F6066D1201                  test    wmode,1
2189  0F88 7504          0F8E          jnz     set1
2190  0F8A 0AFF                        or      bh,bh
2191  0F8C 7510          0F9E          jnz     seterr          ;must be < 256 if not SW
2192                           set1:
2193  0F8E C43E5C12                    les     di,dword ptr type2loc
2194  0F92 E8DAF9        096F          call    set8or16
2195                           set2:
2196  0F95 BE5C12                      mov     si,offset type2loc
2197  0F98 E8FEF9        0999          call    inclor2
2198  0F9B 73B8          0F55          jnb     set0
2199                           setret:
2200  0F9D C3                          ret
2201                           seterr:
2202  0F9E E93EF3        02DF          jmp     err
2203                                   ;
2204                                   ;       ****************************************
2205                                   ;       *                                      *
2206                                   ;       *     t - trace program execution      *
2207                                   ;       *                                      *
2208                                   ;       ****************************************
2209                                   ;
2210                           trace:
2211  0FA1 C606481201                  mov     traceprint,1
2212                           trace0:                         ;untrace enters here with traceprint = 0
2213  0FA6 E848F4        03F1          call    conin
2214  0FA9 3C53                        cmp     al,'S'          ;check for TS
2215  0FAB B401                        mov     ah,1
2216  0FAD 7406          0FB5          jz      tr0             ;if TS, set segflag to 1
2217  0FAF FECC                        dec     ah              ;else set segflag to 0, and
2218  0FB1 FF0E0412                    dec     conptr          ;decrement pointer to rescan character
2219                           tr0:
2220  0FB5 88265412                    mov     segflag,ah      ;print segment registers or not
2221  0FB9 C706461201 00               mov     tracecount,1    ;default to 1 instruction trace
2222  0FBF E8E3F9        09A5          call    getnumber
2223  0FC2 3C0D                        cmp     al,eol
```

```
2224
2225  0FC4 750E          0FD4          jnz     traceerr        ;only 1 parameter allowed
2226  0FC6 0AE4                        or      ah,ah           ;see if a number was entered
2227  0FC8 7404          0FCE          jz      trace1          ;skip if no number typed
2228  0FCA 891E4612                    mov     tracecount,bx   ;store number of instructions to trace
2229                               trace1:
2230  0FCE E8C400        1095          call    xdisp           ;display CPU state
2231  0FD1 E9C8F8        089C          jmp     tracerestore    ;restore user's CPU state and return
2232                               traceerr:
2233  0FD4 E908F3        02DF          jmp     err
2234                               ;
2235                               ;       ****************************************
2236                               ;       *                                      *
2237                               ;       *     u - untraced program execution   *
2238                               ;       *                                      *
2239                               ;       ****************************************
2240                               ;
2241                               untrace:
2242  0FD7 C6054812 00                 mov     traceprint,0
2243  0FDC EBC8          0FA6          jmps    trace0          ;common code with trace command
2244                               ;
2245                               ;       ********************************
2246                               ;       *                              *
2247                               ;       *     v - display file info    *
2248                               ;       *                              *
2249                               ;       ********************************
2250                               ;
2251                               verify:
2252  0FDE A06E12                      mov     al,mode
2253  0FE1 3C52                        cmp     al,'R'
2254  0FE3 7407          0FEC          jz      verifyr
2255  0FE5 3C45                        cmp     al,'E'
2256  0FE7 7421          100A          jz      verifye
2257  0FE9 E9F3F2        02DF          jmp     err     ;neither R nor E command done
2258                               verifyr:
2259  0FEC E82EF4        041D          call    crlf
2260  0FEF BED601                      mov     si,offset readm
2261  0FF2 E843F4        043D          call    printm
2262  0FF5 E825F4        041D          call    crlf
2263  0FF8 C43E3A12                    les     di,dword ptr startreadoff
2264  0FFC E86AF4        0469          call    printdword
2265  0FFF E82DF4        042F          call    blank
2266  1002 C43E3E12                    les     di,dword ptr endreadoff
2267  1006 E860F4        0469          call    printdword
2268  1009 C3                          ret
2269                               ;
2270                               verifye:
2271  100A E810F4        041D          call    crlf
2272  100D B90300                      mov     cx,3
2273  1010 E822F4        0435          call    tabs
2274  1013 BED501                      mov     si,offset readm
2275  1016 E824F4        043D          call    printm          ;print header
2276  1019 B000                        mov     al,0            ;initialize count to 0
```

```
2277
2278                            v0:
2279   1018 50                          push    ax              ;save it
2280   1019 58                          pop     ax              ;get count
2281   101A 50                          push    ax              ;save it
2282   101B E864F6      0685            call    printseginfo    ;print name, start, end of segment if non-zero
2283   101E 58                          pop     ax              ;get count
2284   101F FEC0                        inc     al              ;increment it
2285   1021 803E771201                  cmp     byte ptr basepagesave+5,1       ;check for 8080 model
2286   1026 7407        102F            jz      verret          ;no more segments if 8080 model
2287   1028 3C08                        cmp     al,8            ;max of 8 segments described in base page
2288   102A 72EC        1018            jb      v0              ;done when count = 8
2289                            verret:
2290   102C C3                          ret
2291                            ;
2292                            ;       ****************************************
2293                            ;       *                                      *
2294                            ;       *     w - write memory block to disk   *
2295                            ;       *                                      *
2296                            ;       ****************************************
2297                            ;
2298                            write:
2299   102D A13C12                      mov     ax,startreadseg
2300   1030 A34D12                      mov     startwriteseg,ax
2301   1033 A13A12                      mov     ax,startreadoff
2302   1036 A34B12                      mov     startwriteoff,ax
2303   1039 A14012                      mov     ax,endreadseg
2304   103C A35112                      mov     endwriteseg,ax
2305   103F A13E12                      mov     ax,endreadoff
2306   1042 A34F12                      mov     endwriteoff,ax
2307   1045 E8A6F3      03F1            call    conin
2308   1048 3C0D                        cmp     al,eol          ;check for no parameters
2309   104A 7443        1092            jz      werr            ;must have a filename
2310   104C FF0E0412                    dec     conptr          ;to rescan first char
2311   1050 BF5C00                      mov     di,offset fcb
2312   1053 E838F4      0491            call    parse           ;get filename
2313   1056 7437        1092            jz      werr            ;don't allow '?' or '*'
2314   1058 3C0D                        cmp     al,eol
2315   105A 7509        1065            jnz     w0              ;not end of input - must be 2 parameters
2316   105C 803E6E1252                  cmp     mode,'R'        ;see if a file was read in
2317   1061 752C        1092            jnz     werr            ;no file read - must have start, end addresses
2318   1063 EB20        1085            jmps    w1              ;continue with write
2319                            w0:
2320   1065 A15A12                      mov     ax,typelseg
2321   1068 A34D12                      mov     startwriteseg,ax        ;set default to userds
2322   106B BF4B12                      mov     di,offset startwriteoff
2323   106E E8B5F9      0A29            call    getsegandoff    ;get start address
2324   1071 3C0D                        cmp     al,eol
2325   1073 741A        1092            jz      werr            ;need 2 parameters
2326   1075 A14D12                      mov     ax,startwriteseg
2327   1078 A35112                      mov     endwriteseg,ax  ;end defaults to start
2328   107B BF4F12                      mov     di,offset endwriteoff
2329   107E E8A5F9      0A29            call    getsegandoff    ;get end address
```

```
2330
2331   1084 3C00                        cmp     al,eol
2332   1086 750A          1092          jnz     werr            ;no more than 2 parameters
2333                      wl:
2334   1088 E831FA        0AC3          call    writefile
2335   108B BA5C00                      mov     dx,offset fcb
2336   108E E88DF2        034E          call    close
2337   1091 C3                          ret
2338                      werr:
2339   1092 E94AF2        02DF          jmp     err
2340                      ;
2341                      ;       ****************************************
2342                      ;       *                                      *
2343                      ;       *     x - display/alter CPU state      *
2344                      ;       *                                      *
2345                      ;       ****************************************
2346                      ;
2347                      xdisp:                          ;display CPU state
2348   1095 E816F7        07AE          CALL    SETDEFSEG       ;SET TYPE1SEG, LASLOC TO CS:IP
2349   1098 F6067012FF                  TEST    COL40,0FFH
2350   109D 751A          10B9          JNZ     XD40
2351   109F E898F5        0630          call    printregheader
2352                      xnohdr:                         ;entry point to display CPU state without header
2353   10A2 F6067012FF                  TEST    COL40,0FFH
2354   10A7 7517          10C0          JNZ     XNH40
2355   10A9 E871F3        041D          call    crlf
2356   10AC E8B5F4        0564          call    printflags
2357   10AF E87DF3        042F          call    blank
2358   10B2 E823F5        05D8          call    printregs
2359   10B5 E8A5F5        065D          call    printinstr      ;disassemble instruction at [cs:ip]
2360   10B8 C3                          ret
2361                      ;
2362                      XD40:
2363   10B9 E86FF5        062B          CALL    PREGHDR1
2364   10BC B001                        MOV     AL,1
2365   10BE EB02          10C2          JMPS    XD0
2366                      XNH40:
2367   10C0 B000                        MOV     AL,0
2368                      XD0:
2369   10C2 50                          PUSH    AX              ;SAVE HEADER/NO HEADER FLAG
2370   10C3 E857F3        041D          CALL    CRLF
2371   10C6 E898F4        0564          CALL    PRINTFLAGS
2372   10C9 E863F3        042F          CALL    BLANK
2373   10CC E8FAF4        05C9          CALL    PREG1
2374   10CF E848F3        041D          CALL    CRLF
2375   10D2 58                          POP     AX
2376   10D3 FEC8                        DEC     AL
2377   10D5 7506          10DD          JNZ     XD1
2378   10D7 E858F5        0635          CALL    PREGHDR2
2379   10DA E840F3        041D          CALL    CRLF
2380                      XD1:
2381   10DD E8F3F4        05D3          CALL    PREG2
2382   10E0 E87AF5        065D          CALL    PRINTINSTR
```

```
2383
2384  10E3 C3                          RET
2385                          xcom:
2386  10E4 C606541201                  mov     segflag,1       ;display seg reg's in x command
2387  10E9 E805F3       03F1           call    conin
2388  10EC 3C0D                        cmp     al,eol          ;check for command by itself
2389  10EE 74A5         1095           jz      xdisp           ;if so, simply display CPU state
2390  10F0 A25712                      mov     xtemp,al        ;else save char
2391  10F3 E8F8F2       03F1           call    conin
2392  10F6 3C0D                        cmp     al,eol          ;check for single character after X
2393  10F8 746A         1164           jz      xflag           ;if so, must be a flag name
2394  10FA 8A265712                    mov     ah,xtemp        ;else it's a reg name
2395  10FE 0C80                        or      al,80h          ;since names are declared that way
2396  1100 86C4                        xchg    al,ah           ;since that's how it is in memory
2397  1102 E826F4       052B           call    chkreg          ;check for valid reg name + store number in regnum
2398  1105 E8E9F2       03F1           call    conin
2399  1108 3C0D                        cmp     al,eol
2400  110A 7555         1161           jnz     xerr            ;eol must follow reg name
2401                          x0:
2402  110C E80EF3       041D           call    crlf
2403  110F 8B0E5512                    mov     cx,regnum
2404  1113 E8EBF4       0601           call    printregname
2405  1116 E816F3       042F           call    blank
2406  1119 8B0E5512                    mov     cx,regnum
2407  111D E8D5F4       05F5           call    printregval
2408  1120 E80CF3       042F           call    blank
2409  1123 E8B1F2       03D7           call    getline
2410  1126 E8C8F2       03F1           call    conin
2411  1129 3C2E                        cmp     al,'.'
2412  112B 746E         119B           jz      xret            ;done when '.' entered
2413  112D FF0E0412                    dec     conptr          ;else rescan character
2414  1131 E871F8       09A5           call    getnumber
2415  1134 3C0D                        cmp     al,eol
2416  1136 7529         1161           jnz     xerr            ;eol must follow number
2417  1138 0AE4                        or      ah,ah           ;see if non-blank entry
2418  113A 7419         1155           jz      xnext           ;if blank, go to next reg
2419  113C 8B0E5512                    mov     cx,regnum
2420  1140 8BC3                        mov     ax,bx           ;get new value
2421  1142 80F908                      cmp     cl,8            ;are we updating cs?
2422  1145 7503         114A           jnz     x1
2423  1147 A35A12                      mov     typelseg,ax     ;if so, update default type1 segment
2424                          x1:
2425  114A 80F909                      cmp     cl,9            ;are we updating ds?
2426  114D 7503         1152           jnz     x2
2427  114F A35E12                      mov     type2seg,ax     ;if so, update default type2 segment
2428                          x2:
2429  1152 E805F4       055A           call    setreg
2430                          xnext:
2431  1155 FF065512                    inc     regnum
2432  1159 833E551200                  cmp     regnum,totreg
2433  115E 72AC         110C           jb      x0
2434  1160 C3                          ret
2435                          xerr:
```

```
2436
2437  1161 E978F1        02DF          jmp     err
2438                          ;
2439                          xflag:
2440  1164 A05712                     mov     al,xtemp        ;get flag name
2441  1167 E8D7F3        0541         call    checkflag       ;check for valid flag name
2442  116A E830F2        0419         call    crlf
2443  116D 8B0E5512                   mov     cx,regnum       ;restore flag number
2444  1171 E819F4        058D         call    printflagname
2445  1174 E888F2        042F         call    blank
2446  1177 8B0E5512                   mov     cx,regnum
2447  117B E823F4        05A9         call    printflagval
2448  117E E8AEF2        042F         call    blank
2449  1181 E853F2        03D7         call    getline
2450  1184 E81EF8        09A5         call    getnumber
2451  1187 3C0D                       cmp     al,eol
2452  1189 75D6          1161         jnz     xerr            ;eol must follow number
2453  118B 0AE4                       or      ah,ah           ;see if non-blank entry
2454  118D 740C          119B         jz      xret            ;if blank, done
2455  118F 83FB01                     cmp     bx,1
2456  1192 77CD          1161         ja      xerr            ;flag value must be 0 or 1
2457  1194 8B0E5512                   mov     cx,regnum
2458  1198 E8D8F3        0573         call    setflag
2459                          xret:
2460  119B C3                         ret
2461                          ;
```

```
2462
2463                                    eject
2464                            ;       ************************************
2465                            ;       *                                  *
2466                            ;       *       d a t a   a r e a          *
2467                            ;       *                                  *
2468                            ;       ************************************
2469                            ;
2470                            ;       user regs must be in cseg, others may be in dseg
2471                            ;
2472   119C 0000                userax  dw      0
2473   119E 0000                userdx  dw      0
2474   11A0 0000                usercx  dw      0
2475   11A2 0000                userdx  dw      0
2476   11A4 0000                usersp  dw      0
2477   11A6 0000                userbp  dw      0
2478   11A8 0000                usersi  dw      0
2479   11AA 0000                userdi  dw      0
2480   11AC 0000                usercs  dw      0
2481   11AE 0000                userds  dw      0
2482   11B0 0000                userss  dw      0
2483   11B2 0000                useres  dw      0
2484   11B4 0000                userip  dw      0
2485   11B6 0000                userfl  dw      0
2486     119C                   userreg equ     userax
2487                            ;
2488   11B8 0000                savess  dw      0               ;temp holder for sp
2489   11BA 0000                savesp  dw      0               ;temp holder for sp
2490                            ;
2491   11BC                     breakfl rs      1               ;break/single step flag (must be in CS)
2492                            ;
2493     11BD                   endcs   equ     $
2494                            ;
2495                                    dseg
2496                                    org     offset endcs
2497                            ;
2498                            ;       CCP STACK LOCATION
2499                            ;
2500   11BD 0000                CCPSS           DW      0
2501   11BF 0000                CCPSP           DW      0
2502                            ;
2503                            ;       console buffer declarations
2504                            ;
2505     0040                   conbuffmax      equ     64
2506   11C1 40                  conbuffhdr      db      conbuffmax
2507   11C2 00                  conbuffcnt      db      0
2508   11C3                     conbuff         rs      conbuffmax+1    ;leave room for eol
2509   1204 0000                conptr          dw      0
2510                            ;
2511                            ;       a command declarations
2512                            ;
2513   1206 01                  assempresent    db      1       ;assembler in memory flag
2514   1207                     asmspsav        rw      1       ;temporary save for stack pointer
```

```
2515
2516                            ;
2517                            ;       d command declarations
2518                            ;
2519   1209 0000                disloc          dw      0       ;offset for display
2520   120b 0000                disseg          dw      0       ;segment for display
2521   120D                     dismax          rw      1       ;end offset of display
2522   120F                     tdisp           rw      1       ;temporary storage for disloc
2523   1211 00                  LINEMAX         DB      0       ;# OF BYTES PER LINE
2524   1212 10                  ND80            DB      16      ;16 BYTES PER LINE IN 80 COL MODE
2525   1213 06                  ND40            DB      6       ;6 BYTES PER LINE IN 40 COL MODE
2526   1214 08                  NDW40           DB      8       ;8 BYTES (4 WORDS) FOR DW IN 40 COL MODE
2527   1215 0C                  NLINES          DB      12      ;DEFAULT NUMBER OF LINES FOR L, D COMMANDS
2528                            ;
2529                            ;       g command declarations
2530                            ;
2531   1216 0000                goloc           dw      0
2532   1218 0000                goseg           dw      0
2533   121A                     vectorsave      rs      12      ;save area for bytes at 0004h to 000fh
2534   1226 00                  bpcnt           db      0       ;breakpoint count
2535   1227 0000                brk1loc         dw      0
2536   1229 0000                brk1seg         dw      0
2537   122B 00                  brk1byt         db      0
2538   122C 0000                brk2loc         dw      0
2539   122E 0000                brk2seg         dw      0
2540   1230 00                  brk2byt         db      0
2541                            ;
2542                            ;       l command declarations
2543                            ;
2544   1231 0000                lasloc          dw      0
2545   1233 0000                lasseg          dw      0
2546   1235 0000                lasmax          dw      0
2547   1237                     lascntsw        rb      1       ;# instructions specified or not
2548   1238                     lascnt          rb      1       ;number of instructions to disassemble
2549   1239 01                  disempresent    db      1       ;disassembler in memory flag
2550                            ;
2551                            ;       r command declarations
2552                            ;
2553   123A                     startreadoff    rw      1       ;offset where file read starts
2554   123C                     startreadseg    rw      1       ;segment where file read starts
2555   123E                     endreadoff      rw      1       ;offset where file read ends
2556   1240                     endreadseg      rw      1       ;segment where file read ends
2557   1242                     dmaoff          rw      1       ;offset of 20-bit dma address
2558   1244                     dmaseg          rw      1       ;segment of 20-bit dma address
2559                            ;
2560                            ;       t/u command declarations
2561                            ;
2562   1246                     tracecount      rw      1       ;number of instructions to trace
2563   1248                     traceprint      rb      1       ;display CPU state on each step flag
2564   1249 00                  skipbdos        db      0       ;set when trace suspended during BDOS call
2565   124A 00                  USERIFOFF       DB      0       ;SET WHEN DDT86 MUST REENABLE USER IF
2566                            ;
2567                            ;       w command declarations
```

```
2568
2569                            ;
2570   124B                     startwriteoff   rw      1       ;offset where file write starts
2571   124D                     startwriteseg   rw      1       ;segment where file write starts
2572   124F                     endwriteoff     rw      1       ;offset where file write ends
2573   1251                     endwriteseg     rw      1       ;segment where file write ends
2574                            ;
2575                            ;       x command declarations
2576                            ;
2577   1253 00                  NREG            DB      0       ;CURRENT NUMBER OF REGISTER NAMES TO DISPLAY
2578                                                            ;(MAY DIFFER FOR 40 COLUMN MODE)
2579     000D                   TOTREG          EQU     13      ;TOTAL NUMBER OF REGISTER NAMES
2580     0009                   nflag           equ     9       ;number of flag names
2581   1254 01                  segflag         db      1       ;print segment register flag
2582   1255                     regnum          rw      1       ;temp for reg/flag number
2583   1257                     xtemp           rb      1       ;temp for first char of reg name
2584                            ;
2585   1258 0000                type1loc        dw      0       ;offset for type 1 commands
2586   125A 0000                type1seg        dw      0       ;segment for type 1 commands
2587   125C 0000                type2loc        dw      0       ;offset for type 2 commands
2588   125E 0000                type2seg        dw      0       ;segment for type 2 commands
2589   1260                     usermax         rw      1
2590   1262                     userloc2        rw      1
2591   1264                     userseg2        rw      1
2592                            ;
2593                            ;       memory control block declarations
2594                            ;
2595   1266                     mcb             rs      0       ;used in reading/writing file
2596   1266 0000                mcbbase         dw      0       ;segment of memory block
2597   1268 0000                mcblen          dw      0       ;length of memory block
2598   126A 00                  mcbext          db      0       ;returned value from bdos memory functions
2599                            ;
2600   126B                     sysif           rb      1       ;system interrupt flag
2601   126C                     numreq          rb      1       ;number required or optional
2602   126D                     wmode           rb      1       ;set for DW, FW and SW commands
2603   126E 49                  mode            db      'I'     ;last disk access with 'R' or 'E' command
2604   126F 00                  savevecflag     db      0       ;save/restore bp and ss vectors, or not
2605   1270 00                  COL40           DB      0       ;PATCHED TO 1 FOR 40 COLUMN CONSOLE
2606   1271 00                  ERRMODE         DB      0       ;SET IF BDOS RETURN ERR MODE SET (V3.0)
2607                            ;
2608   1272                     basepagesave    rb      48      ;copy of user's base page
2609                            ;
2610                            ;       parsing declarations
2611                            ;
2612   12A2 203D2F2C3A3B        delims          db      ' ', '=', '.', ',', ':', ';', '[', ']', '<', '>', eol
2613        5B5D3C3E0D
2614     000B                   ndelims         equ     offset $ - offset delims
2615   12AD 00                  lastchar        db      0
2616   12AE 0000                fcbadr          dw      0       ;temp storage for fcb address
2617                            ;
2618   12B0                                     rs      stsize  ;stack size
2619   1310                     stackp          rs      0       ;top of stack
2620                            ;
```

```
2621
2622                                    if      debug
2623                            ;
2624                            biosentryoff    dw      0a00h   ;empty part of ccp (hopefully)
2625                            biosentryseg    dw      0       ;same as bdos segment
2626                            ;
2627                                    endif
2628                            ;
2629                                    end
2630
2631
2632  END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR: 32%
```

```
GETLASTOFFSET   09CB L  1453# 1762  1886  1966  2047  2186
GETLINE         03D7 L   268   524   541#  583&  2179  2409  2443
GETMAXMEM       0394 L   475# 2141
GETNO           09A9 L  1427# 1437
GETNRET         09BE L  1430  1438#
GETNUMBER       09A5 L  1420# 1446  1531  2222  2414  2450
GETOFFSET       09BF L  1441# 1454  1539  1681  1884  1964  2091
GETS0           0A3B L  1530  1537#
GETS2           0A44 L  1536  1543#
GETS3           0A47 L  1533  1547#
GETSEGANDOFF    0A29 L  1519# 1655  1680  1690  1757  1885  1924  1931  1939  2042
                        2090  2100  2169  2323  2329
GETSRET         0A46 L  1545# 1549
GOERR           0DB9 L  1941  1951#
GOLOC           1216 V  1921  1923  1948  2531#
GORESTORE       0DA3 L  1926  1937  1944#
GOSEG           1218 V  1919  1927  1946  2532#
GOUSER          0D5B L   304  1917#
HEXCON          0943 L  1342# 1434
HEXERR          0954 L  1348  1350  1353#
HEXMATH         0DBF L   305  1963#
HEXRET          0953 L  1345  1351#
I0              0DF0 L  1993  1995#
I1              0E01 L  2003# 2009
I2              0E0C L  2006  2010#
IDONE           0E2A L  2016  2023#
IERR            0E2B L  2025#
IFCB            0DDE L   306  1870  1988#
IFMASK16        0200 N    58#  195
IFMASK8         0002 N    59#  203  1229
INC10R2         0999 L  1411# 1786  1900  2197
KWAIT           0419 L   615   617#
LAS0            0E5A L  2046  2049#
LAS1            0E68 L  2048  2054#
LAS2            0E6C L  2056# 2074  2076
LASCNT          1238 V  2052  2075  2548#
LASCNTSW        1237 V  2037  2050  2073  2547#
LASERR          0EA1 L  2036  2079#
LASLOC          1231 V  1099  1183  2041  2057  2070  2072  2153  2544#
LASMAX          1235 V  2055  2058  2546#
LASMORG         1400 N    21#   31
LASRET          0EA0 L  2059  2071  2077#
LASSEG          1233 V  2040  2044  2063  2545#
LASSM           0E2E L   309  2034#
LASTCHAR        12AD V   734   742   750   758   777  2615#
LF              000A N    61#   114   117   121   123   127   132   135   624
LINEMAX         1211 V  1755  1766  1790  2523#
LOAD            03B2 L   501# 1847
LOADERR         03BB L   505   507#
LOADM           0197 V   117#  508
MAKE            0376 L   453# 1620
MAKEERR         0380 L   457   459#
MAKEM           01AC V   121#  460
```

```
MCB             1266 V   1844  2140  2142  2146  2595#
MCBBASE         1266 V   1560  1580  1601  2596#
MCBEXT          126A V   1843  2145  2598#
MCBLEN          1268 V   1574  1581  1609  1630  2139  2597#
MEMERR          03A8 L    480   493#
MEMM            0186 V    123#  494
MODE            126E V   1857  2015  2154  2252  2316  2603#
MOVD            0EC8 L   2103#  2114
MOVCOM          023D L    172   174#
MOVE            0EA4 L    310  2088#
MOVERR          0EEF L   2094  2096  2102  2117#
MOVRET          0EEE L   2109  2111  2115#
NBPS            0002 N     56#
ND40            1213 V   1746  2525#
ND80            1212 V   1753  2524#
NDELIMS         0006 N    771  2614#
NDW40           1214 V   1750  2526#
NFLAG           0009 N    834   840  2580#
NLINES          1215 V   1767  2051  2527#
NOT8080         06FE L   1078  1081#
NOTAST          0504 L    779   782#
NOTSR           0A24 L   1508  1510  1515#
NREG            1253 V    920   928   938   979   988   999  2577#
NUMREQ          126C V    281  1548  1738  1922  2038  2601#
OPEN            033F L    414#  1842  2138
OPENERR         0349 L    418   420#
OPENM           01CD V    127#  421
PARSE           0491 L    717#  1828  1991  2133  2312
PARSE2          049C L    726#  1997
PARSEERR        04C3 L    745#  789
PARSERET        04BC L    737   740#
PDELIM          04FC L    769#  785
PERROUT         0457 L    662   665#
PF0             0567 L    858#  864
PF2             05D2 L    899   901#
PI0             067A L   1012  1014#
PI1             067E L   1007  1017#
PLMCONOUT       03C4 L     43   514#
PLMGETLINE      03D1 L     44   522#
PLMSET          0957 L     46  1356#
PNAME           05C5 L    912   914#
PQUIT           0449 L    650   655#
PR0             05E0 L    921   929#  939
PR00            05DB L    924   927#
PR2             05ED L    931   936#
PREG1           05C9 L    918#  2373
PREG2           05D3 L    922#  2381
PREGHDR1        062B L    977#  2363
PREGHDR2        0635 L    981#  2378
PRH0            0645 L    980   989#  1000
PRH00           0640 L    984   987#
PRH1            0655 L    991   997#
PRINT8UR16      045D L    671#  1784  2177
```

```
PRINTBYTE          047C L   674   692   695# 1019  1728
PRINTDWORD         0469 L   677# 1051  1060  1195  1408  1725  1780  2067  2174  2264
                            2267
PRINTERROR         0BC5 L  1702  1706  1721#
PRINTFLAG          0586 L   860   906#
PRINTFLAGNAME      058D L   879# 2444
PRINTFLAGS         0564 L   856# 2356  2371
PRINTFLAGVAL       05A9 L   896# 2447
PRINTINSTR         065D L  1003# 2359  2382
PRINTIT            061C L   961   963   965#
PRINTM             043D L   242   511   646#  654   954  1046  1405  2261  2275
PRINTNIBBLE        0485 L   699   702#
PRINTREGHEADER     063D L   985# 2351
PRINTREGNAME       0601 L   950#  993  2404
PRINTREGS          05D8 L   925# 2358
PRINTREGVAL        05F5 L   933   942# 2407
PRINTSEGINFO       0685 L  1022# 2282
PRINTWORD          0475 L   675   683   688#  947  1974  1979
PSIRET             06CC L  1035  1062#
PSRET              096C L  1363  1365#
RDCONBUFF          032E L   388#  586
READ               0EF2 L   315  2127#
READDONE           0AB2 L  1579
READERR            0AB9 L  1575
READFILE           0A51 L  1558# 2144
READM              01D6 V   129# 2260  2274
READSEC            0362 L   439# 1577
REGNAME            014D V    85#  823   951
REGNUM             1255 V   830   842  2403  2406  2419  2431  2432  2443  2446  2457
                           2582#
RERR               0F49 L  2130  2134  2136  2157#
RESTOREVECT        0796 L  1169# 1235
RF0                0A65 L  1567# 1590
RF1                0A72 L  1571# 1588
RSTORE             08FF L  1280  1301# 1950
RVRET              07AD L  1171  1178#
S0                 04E3 L   757   763#
S1                 04E7 L   754   765#
SAV0               0843 L  1230  1232#
SAVECPU            07E5 L  1200  1203#
SAVESP             118A V  1309  1318  2489#
SAVESS             1188 V  1315  1317  2488#
SAVEVECFLAG        126F V   210  1135  1170  2604#
SAVEVECT           0781 L  1138  1154#
SC1                0715 L  1087  1089#
SCANQ              051E L   741   805#
SEGFLAG            1254 V   960  1011  2220  2386  2581#
SEGNAMES           0179 V   104# 1042
SEGREG             015D V    93# 1478
SET0               0F55 L  2170# 2198
SET1               0F8E L  2189  2192#
SET2               0F95 L  2182  2195#
SETBOR16           096F L  1368# 1898  2194
```

```
SETBDOSINT      020A L   145#  206
SETBP           0733 L  1108#  1278  1934  1943
SETBYTE         0980 L  1361   1370  1387  1394#  2107
SETCPUSTATE     06CD L  1065#  1849
SETDEFSEG       07AE L  1181#  1194  2348
SETDMA          0385 L   463#  1573  1627  1854
SETDMAB         038F L   471#  1569  1623  1851
SETDONE         0728 L  1080   1092  1097#
SETERR          0F9E L  2191   2201#
SETERRMODE      038A L   249    467#
SETFLAG         0573 L   867#  2458
SETMEM          0F4C L   316   2166#
SETREG          055A L   848#  2429
SETRET          0F9D L  2184   2199#
SETUPDISK       04C6 L   730    748#  752
SETUPHDR        061E L   970#   978   985
SETWORD         0976 L  1373#
SIGNON          013A V    80#   241
SKIPBDOS        1249 V  1242   1250  1279  1945  2564#
SRET            0988 L  1391   1398#
SSENTRY         07DF L  1142   1201#
SSIO            086F L  1237   1251#
SSIOO           086A L  1243   1249#
SSI1            0889 L  1255   1259#
STACKP          1310 V   202    263  1227  2619#
START           02A7 L   262#   271   273   284   290   512   616  1247  1364  1409
STARTREADOFF    123A V  1565   2263  2301  2553#
STARTREADSEG    123C V  1561   2148  2299  2554#
STARTWRITEOFF   124B V  2302   2322  2570#
STARTWRITESEG   124D V  1600   2300  2321  2326  2571#
STSIZE          0060 N    55#  2618
SVRET           0795 L  1165#
SYSIF           126B V   196    203  1229  2600#
TABS            0435 L   639#   643   973   995  2273
TDISP           120F V  1779   1789  1799  1803  1805  2522#
TESTREGCL       060C L   930    958#  990
TOTREG          000D N   822    828  2432  2579#
TRO             0FB5 L  2216   2219#
TR00            08C6 L  1272   1281#
TR01            08FB L  1285   1291  1293  1295  1297  1299#
TRACE           0FA1 L   317   2210#
TRACE0          0FA6 L  2212#  2243
TRACE1          0FCE L  2227   2229#
TRACECOUNT      1246 V  1260   2221  2228  2562#
TRACEDONE       0864 L  1245#  1261
TRACEERR        0FD4 L  2225   2232#
TRACEPRINT      1248 V  1263   2211  2242  2563#
TRACERESTORE    089C L  1264   1267#  2231
TYPE1LOC        1258 V  1654   1658  1660  1662  2585#
TYPE1SEG        125A V  1074   1084  1185  1359  1657  2039  2045  2150  2320  2423
                        2586#
TYPE2LOC        125C V  1679   1685  1695  1701  1708  1709  1882  1893  1896  1899
                        1902   2089  2095  2104  2108  2112  2168  2173  2176  2193
```

```
                    2196  2587#
TYPE2SEG    125E V  1075  1193  1687  1739  1760  2097  2149  2427  2589#
UNTRACE     0FD7 L   318  2241#
UPPER       03FD L   602#
UPRET       0407 L   603   605   607#
USERAX      119C V  1204  2472# 2486
USERBP      11A6 V  1210  2477#
USERBX      119E V  1205  2473#
USERCS      11AC V  1004  1072  1083  1184  1219  1268  1287  1320  1511  1918
                    1947  2480#
USERCX      11A0 V  1206  2474#
USERDI      11AA V  1209  2479#
USERDS      11AE V  1071  1192  1213  1862  2019  2481#
USERDX      11A2 V  1207  2475#
USERES      11B2 V  1073  1088  1212  2483#
USERFL      11B6 V   197   875   875   893  1222  1252  1257  1282  1300  1319
                    2485#
USERIPOFF   124A V  1254  1256  1298  2565#
USERIP      11B4 V  1005  1098  1182  1217  1240  1269  1288  1321  1920  1949
                    2484#
USERLOC2    1262 V  1689  1697  1705  1712  2099  2106  2110  2590#
USERMAX     1260 V  1682  1710  1885  1892  1903  2092  2113  2589#
USERREG     119C V   850   943  1304  2486#
USERSEG2    1264 V  1688  2094  2591#
USERSI      11A8 V  1208  2478#
USERSP      11A4 V   191  1069  1095  1211  1233  2476#
USERSS      11B0 V   190  1067  1093  1214  2482#
VO          101B L  2278# 2288
VECTORSAVE  121A V  1156  1172  2533#
VERIFY      0FDE L   320  1868  2155  2251#
VERIFYE     100A L  2256  2270#
VERIFYR     0FEC L  2254  2258#
VERM        01E6 V   132# 1404
VERRET      102F L  2286  2289#
VERSION     033A L   244   410#
WO          1068 L  2315  2319#
W1          1088 L  2318  2333#
WERR        1092 L  2309  2313  2317  2325  2332  2338#
WF0         0AFA L  1621# 1635
WF00        0AF4 L  1613  1615  1618#
WF1         0B07 L  1625# 1633
WFERR       0B2C L  1608  1617  1638#
WMODE       126D V   282   673  1369  1413  1465  1744  1887  2188  2602#
WRITE       1030 L   321  2298#
WRITEDONE   0B2B L  1631  1636#
WRITEERR    0371 L   447   449#
WRITEFILE   0ABC L  2334
WRITEM      01F8 V   135#  450
WRITESEC    0367 L   443# 1629
XO          110C L  2401# 2433
X1          114A L  2422  2424#
X2          1152 L  2426  2428#
XCOM        10E4 L   322  2385#
```

```
XD0             10C2 L  2365  2368#
XD1             10DD L  2377  2380#
XD40            1089 L  2350  2362#
XD1SP           1095 L  2230  2347# 2389
XERR            1161 L  2400  2416  2435# 2452  2456
XFLAG           1164 L  2393  2439#
XNEXT           1155 L  2418  2430#
XNH40           10C0 L  2354  2366#
XNUMBR          10A2 L  1265  2352#
XRET            119B L  2412  2454  2459#
XTEMP           1257 V  2390  2394  2440  2585#
```