PDP-1 COMPUTER
ELECTRICAL ENGINEERING DEPARTMENT
CAMBRIDGE, MASSACHUSETTS 02139

SUPPLEMENT TO PDP-35
INDEX REGISTER and MICRO-PROGRAM INSTRUCTION CLASS

June 12, 1968

An index register is an active register which is particularly useful for modifying operand addresses. Its name derives from its use in indexing through arrays since its contents (the index) can be added to the address field of an instruction (array base address) to determine the effective address for the desired operand. In dealing with linked data structures it is often convenient to load a "pointer" into the index register and to have a "displacement" quantity in the address field of the instruction. Thus the effective address becomes the pointer value offset by the displacement.

On the PDP-1 the index register (XR) is an 18-bit register. Effective address calculation is performed by one's complement addition of the low order 12-bits of the XR and the address field of the instruction. The result is a 15-bit extended address which is relative to the current core if the machine is not in extend mode. If the machine is in extend mode, the 15-bit index sum is absolute. The indexing operation does not require any time in addition to the instruction time.

Since there are no spare bits available in the op code to designate the indexing operation the "1" bit (bit 5) is used to designate either indirection or indexing, depending upon the state of the machine as set by mode instructions. The machine remains in any given mode until another mode setting instruction is executed. Mode setting instructions are:

```
nam     /set normal mode, subsequent "1" bits will
        /mean indirection

iam     /set index mode, subsequent "1" bits will
        /cause an effective address to be calculated
        /using the XR.
```

```
aam         /if in normal mode the next (only) instruction
            /indexed, independent of the "i" bit.
            /if in index mode, the next (only) instruction
            /will be indirected.
            /if the "i" bit of the next instruction is
            /set both indexing and indirection will
            /occur, in that order.
            /(i.e., indexing first.)
```

Instruction to load the index register from a memory register:

```
lxr         /load index register (similar to lac and lio)
```

Instructions in the PDP-1 micro-program instruction class are quite useful for incrementing (stepping) the XR contents, and exchanging and/or operating upon the contents of the XR in conjunction with other active registers.

The instructions jmp, jdp, jda, and jsp always behave as if in normal mode. aam still applies, however.

Examples:

In the index mode the instruction

```
lac i array2          load the AC with the content
                      of memory
```

location array2+C(XR).

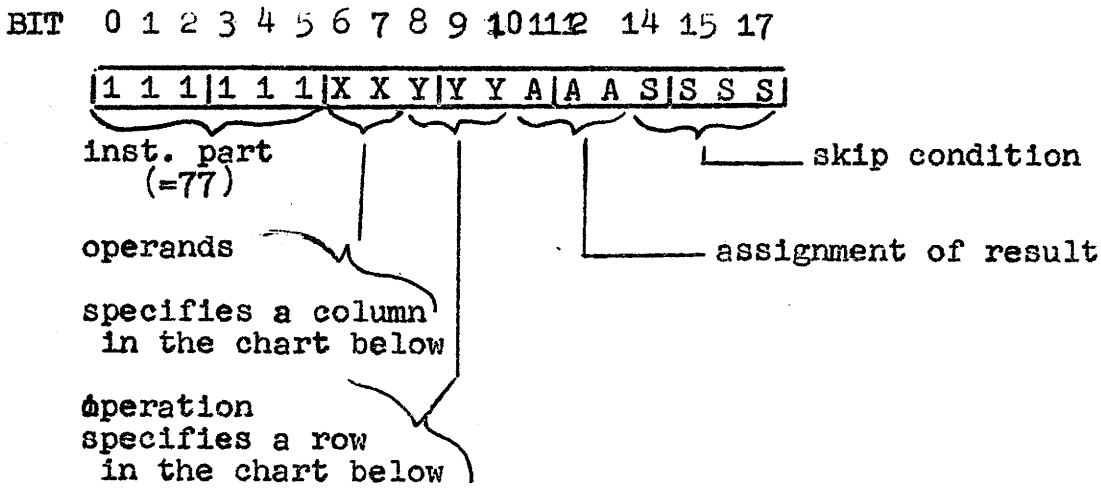
```
/program which stores zeros in all of array2

n=100
dimension array2(n)

        iam                     /enter indexing mode
        lxr (-n                 /initial value into XR
        dzm i array2+n          /store a zero into the array
        SXXP                    /step xr and skip if it is +0
        jmp .-2                 /continue loop
          .
          .                     /that does it.
```

MICRO-PROGRAM INSTRUCTION CLASS

Sample Micro-program instructions

symbolic	octal	action
A+I	773600	computes sum of AC and IO and does nothing at all with it.
A+IA	773700	the sum of the AC and IO is put into the AC.
A+IAIX	773760	the sum of the AC and IO is put into the AC, IO, and XR.
A+I<P	773606	skips if the sum of the AC and IO is less than -0 or equal to +0.
A+I<P|	773611	skip if the sum of the AC and IO is not less than -0 or equal to +0.
A+IX≥	773633	the sum of the AC and IO is put into the XR. If the sum was -0,+0, or greater than +0, the instruction, will skip.
X→IXA|	776420	exchanges the IO and XR. The "old" IO is also put into the AC. This instruction skips always.
ZAIX	771160	the AC, IO, and XR are cleared.
SA>	771210	skip if the AC plus one is greater than +0.
SAAP	771302	add one (step) to the AC and skip if it is +0.
TXM	776001	skip if the XR is -0 and clear the XR.
CXP	776202	skip if the XR is -0.
TAXI	774060	transfer the contents of the AC into the XR and IO, the AC is cleared.
TAAXI	774160	same as TAXI, but the AC is not cleared.

The micro-program instruction class has the following instruction format.



symbolic Specification for Micro-program operations

A = AC, I = IO, X = XR

YYY \ XX	00	01	10	11
000		TI	TA	TX
001		CI	CA	CX
010		I (A→I)	A (X→A)	I (X→I)
011		AMI	XMA	XMI
100	Z (ZERO)	AVI	XVA	XVI
101	SA (AC)+1	A∧I	X∧A	X∧I
110	SI (IO)+1	A~I	X~A	X~I
111	SX (XR)+1	A+I	X+A	X+I

S (Step, ie., add one)

OPERATION (result)

T (True, test, or Transfer)

C (Complement (negative))

exchange (see explanation below)

M (arithmetic Minus)

V (inclusive OR)

∧ (bitwise AND)

~ (exclusive OR)

+ (arithmetic Plus)

The functions of the result assignment and skip condition fields are as follows:

AAA	Bit 11=1 will put the result in the AC	(A)
	Bit 12=1 will put the result in the IO	(I)
	Bit 13=1 will put the result in the XR	(X)
SSSS:	Bit 14=1 will cause a skip if the result is >+0	(>)
	Bit 15=1 will cause a skip if the result is <-0	(<)
	Bit 16=1 will cause a skip if the result is = +0	(P)
	Bit 17=1 will cause a skip if the result is = -0	(M)

The execution of micro-program instruction computes the result specified by the XX and YYY bits (i.e from Table on previous page), puts this result in the specified register (s) and skips if any of the specified conditions are true. Note that skip condition is evaluated on the result of the micro-program, not the final contents of any given register, and the result need not be assigned to any register. Thus, it is possible to test the sum of the AC and IO to see if it is equal to -0 or greater than +0 without destroying the contents of these registers. The instruction to do this is 773611 or, symbolicly, A+IM>. All opr 1 instructions take 5 μseconds.

The "T" operation clears the transmitted register after the transfer. This may be prevented by assigning the result back to the transmitted register. See the examples.

The exchange operation is a special case. The result I (A→I) means that the result of the function is the old IO register, but in addition the accumulator is placed in the IO. This secondary assignment is done before the assignment given by the AAA bits in the instruction. Thus, the instruction A→I (772400) will move the AC to the IO, A→IA (772500) will swap the AC and IO, and A→II (772440) is a nop (because the primary assignment is to the IO and the result is the old IO).

USAGE

The assembler considers any symbol consisting of capital letters as a micro-program instruction. The entire instruction must be in upper case, and may appear in any expression, e.g., storage word, constant, etc. When typing into ID the instruction must be preceded by a single quote (').

Micro-programs are specified by concatenating three "fields" - the result field, the assignment field, and the skip field. The characters in all of these must be in upper case and there should be no separator between the fields.

<result field><assignment field><skip field>

<result field> must be one of the twenty-eight results given in the Table on a previous page.

<assignment field> may be null (no characters) or any combination of A, I, and X to specify in which registers the result will be placed.

<skip field> may be null or contain any combination of >, <, P, M, |, _, and =.

> means skip if result is greater than +0

P means skip if result is equal to +0

M means skip if result is equal to -0

< means skip if result is less than -0

| means complement the specified skip conditions

= and _ means skip if result is either +0 or -0

Since the octal representation of a microprogram instruction is computed by exclusive - OR'ing all of the specifications within each field, redundant specifications may lead to unexpected results. For example TAIII is the same as TAI; and TAI>> is the same as TAI.

An error in syntax results in a uer error message from the assembler. Note that the add and minus instructions don't always set minus zero to plus zero and do not set the overflow flip flop. The step instruction does set a minus zero to plus zero, i.e. $-1 \rightarrow +0$.