Section 4

Instructions and Addressing

INSTRUCTION SET SUMMARY

The instruction set consists of over 100 instructions, grouped as follows:

- o Single Operand
- o Double Operand
- o Short Value Immediate
- o Branch on Register
- o Branch on Indicator
- o Shift
- o Input/Output
- o Generic
- o Scientific
- o Commercial

These are listed below in Table 4-1 and summarized following the table. The instruction formats for each type and the addressing modes are detailed later in this section.

TABLE 4-1. INSTRUCTION SET SUMMARY

SINGLE OPERAND INSTRUCTIONS

Modify	Description
INC	Increment
DEC	Decrement
NEG	Negate
CPL	Complement
CL	Clear
CLH	Clear halfword
CMZ	Compare to zero
CMN	Compare to null
CAD	Add carry bit

Control	Description
STS	Store S-register
JMP	Jump
ENT	Enter
LEV	Change level
SAVE	Save context
RSTR	Restore context

Bit	Description
LB	Load bit
LBF	Load bit and set false
LBT	Load bit and set true
LBC	Load bit and complement
LBS	Load bit and swap

Double Word	Description
AID	Add double integer[a]
LDI	Load double integer
SDI	Store double integer
SID	Subtract double integer[a]

DOUBLE OPERAND INSTRUCTIONS

Word	Description
LDR	Load R-register
STR	Store R-register

TABLE 4-1 (CONT). INSTRUCTION SET SUMMARY

Word	Description
SRM	Store R-register through mask
SWR	Swap R-register
CMR	Compare to R-register
ADD	Add to R-register
SUB	Subtract from R-register
MUL	Multiply R-register
DIV	Divide R-register
OR	Inclusive OR with R-register
XOR	Exclusive OR with R-register
AND	AND with R-register

Byte	Description
LDH	Load halfword into R-register
STH	Store R-register halfword
CMH	Compare halfword to R-register
ORH	Halfword inclusive OR with R-register
XOH	Halfword exclusive OR with R-register
ANH	AND halfword with R-register
LLH	Load logical halfword into R-register

Mode and Base Register	Description
MTM	Modify/test M-register
STM	Store M-register
LDB	Load B-register
STB	Store B-register
CMB	Compare to B-register
SWB	Swap B-register
LAB	Load effective address into B-register
LNJ	Load B-register and jump

SHORT VALUE IMMEDIATE INSTRUCTIONS

Instruction	Description
LDV	Load value into R-register
CMV	Compare value to R-register
ADV	Add value to R-register
MLV	Multiply R-register by value

BRANCH INSTRUCTIONS

Branch on Register	Description
BLZ	Branch if R-register less than zero
BGEZ	Branch if R-register greater than or equal to zero
BEZ	Branch if R-register equal to zero
BNEZ	Branch if R-register not equal to zero
BGZ	Branch if R-register greater than zero
BLEZ	Branch if R-register less than or equal to zero
BODD	Branch if R-register odd
BEVN	Branch if R-register even
BINC	Branch and increment
BDEC	Branch and decrement

Branch on Indicator	Description
B	Branch
NOP	No operation
BE	Branch if equal
BNE	Branch if not equal
BAL	Branch if algebraically less than
BAGE	Branch if algebraically greater than or equal to
BAG	Branch if algebraically greater than
BALE	Branch if algebraically less than or equal to
BL	Branch if less than
BGE	Branch if greater than or equal to
BG	Branch if greater than
BLE	Branch if less than or equal to
BSU	Branch if signs unlike
BSE	Branch if signs equal
BCT	Branch if carry true
BCF	Branch if carry false
BBT	Branch if bit test indicator true
BBF	Branch if bit test indicator false

TABLE 4-1 (CONT). INSTRUCTION SET SUMMARY

Branch on Indicator	Description
BIOT	Branch if I/O indicator true
BIOF	Branch if I/O indicator false
BOV	Branch if R-register overflow
BNOV	Branch if no R-register overflow

SHIFT INSTRUCTIONS

Shift Short	Description
SOL	Single shift open left
SCL	Single shift closed left
SAL	Single shift arithmetic left
SOR	Single shift open right
SCR	Single shift closed right
SAR	Single shift arithmetic right

Shift Long	Description
DOL	Double shift open left
DCL	Double shift closed left
DAL	Double shift arithmetic left
DOR	Double shift open right
DCR	Double shift closed right
DAR	Double shift arithmetic right

INPUT/OUTPUT INSTRUCTIONS

Instruction	Description
IO	Input/output word
IOH	Input/output halfword
IOLD	Input/output load

GENERIC INSTRUCTIONS

Instruction	Description
CNFG	Configure
HLT	Halt
MCL	Monitor call
RTT	Return from trap
RTCN	Real-time clock on
RTCF	Real-time clock off
WDTN	Watchdog timer on[b]
WDTF	Watchdog timer off[b]
BRK	Breakpoint trap
MMM	Memory to memory move[a]
ASD	Activate segment descriptor[c]
VLD	Validate address, range and access rights[c]
QOH	Queue on head[a]
QOT	Queue on tail[a]
DQH	Dequeue from head[a]
DQA	Dequeue on address[a]
RLQ	Relinquish stack space[a]
LDT	Load T-register[a]
ACQ	Acquire stack space[a]
STT	Store T-register[a]
LRDB	Load remote descriptor base[c]
SRDB	Store remote descriptor base[c]

SCIENTIFIC INSTRUCTIONS[d]

Single Operand	Description
SCZD	Scientific compare to zero two words
SCZQ	Scientific compare to zero four words
SNGD	Scientific negate two words
SNGQ	Scientific negate four words

Double Operand	Description
SLD	Scientific load
SST	Scientific store
SCM	Scientific compare
SAD	Scientific add
SSB	Scientific subtract
SML	Scientific multiply
SDV	Scientific divide
SSW	Scientific swap

Scientific Accumulator Branch	Description
SBLZ	Branch if SA less than zero
SBGEZ	Branch if SA greater than or equal to zero
SBEZ	Branch if SA equal to zero
SBNEZ	Branch if SA not equal to zero
SBGZ	Branch if SA greater than zero
SBLEZ	Branch if SA less than or equal to zero

TABLE 4-1 (CONT). INSTRUCTION SET SUMMARY

Scientific Indicator Branch	Description
SBL	Branch if less than
SBGE	Branch if greater than or equal
SBE	Branch if equal
SBNE	Branch if not equal
SBG	Branch if greater than
SBLE	Branch if less than or equal
SBPE	Branch if precision error
SBNPE	Branch if no precision error
SBSE	Branch if significance error
SBNSE	Branch if no significance error
SBEU	Branch if exponent underflow
SBNEU	Branch if no exponent underflow

COMMERCIAL INSTRUCTIONS[e]

Numeric	Description
DAD	Decimal add
DSB	Decimal subtract
DML	Decimal multiply
DDV	Decimal divide
DCM	Decimal compare
DMC	Decimal move and convert
DSH	Decimal shift
CBD	Convert binary to decimal
CDB	Convert decimal to binary

Alphanumeric	Description
ALR	Alphanumeric move
ACM	Alphanumeric compare
MAT	Alphanumeric move and translate
SRCH	Alphanumeric search
VRF	Alphanumeric verify

Edit	Description
DME	Decimal move and edit
AME	Alphanumeric move and edit

Commercial Branch	Description
CBOV	Branch on overflow
CBNOV	Branch on no overflow
CBTR	Branch on truncation
CBNTR	Branch on no truncation
CBSF	Branch on sign fault
CBNSF	Branch on no sign fault
CSYNC	Synchronize
CSNCB	Synchronize and branch
CBE	Branch if equal
CBNE	Branch if not equal
CBG	Branch if greater
CBGE	Branch if greater than or equal
CBLE	Branch if less than or equal
CBL	Branch if less

[a]Traps on Models 23 and 33.

[b]Traps on Model 33 without Watchdog Timer option.

[c]Traps on models without MMU option.

[d]Traps on models without SIP option.

[e]Traps on Models 23, 33, 43, and 53.

1. *Single Operand* instructions can address memory (or a register) in the same way as double operand instructions, but they do not need a register address. A typical single operand instruction is the Clear (CL) instruction, which clears the addressed memory location to zero. In assembly notation, this instruction could be written:

   CL LOC

2. *Double Operand* instructions are memory reference instructions in which the first operand is a register address and the second operand is usually a memory address, although for register-to-register instructions the second address also specifies a register. A typical double operand instruction is an (ADD) instruction, which adds the contents

of the addressed memory location (or register) to the general (R) register specified by the first operand. Thus, the instruction ADD $R1, LOC adds the contents of memory location LOC to register R1.[1]

3. *Branch on Register* instructions are similar to double operand instructions in that they must specify a general register, R1 through R7, and also a memory address to which control will be transferred if the tested condition is true. A typical branch on register instruction is Branch if Register Odd (BODD), which might be written:

   BODD $R6, LOC

   This would test register 6 to see whether it were even or odd, and if it were odd the program would branch to location LOC.

4. *Branch on Indicator* instructions are similar to branch on register instructions, but the op code specifies an indicator and no register address is required. A typical instruction is Branch if Greater than (BG), which will branch if the G (greater than) indicator is set. This will be written:

   BG LOC

5. *Short Value Immediate* instructions do not reference memory, but specify a register and an 8-bit immediate operand which is contained in the instruction itself. For example, if it were desired to add the quantity 2 to register R3, the Add Value (ADV) instruction could be used. This would be written:

   ADV $R3, =2

6. *Shift* instructions are used to shift either single general registers or pairs of general registers. The first operand specifies the register itself or, in the case of a double word shift, the right-hand (odd) register of a pair. The second operand usually specifies the number of positions to be shifted. A typical shift instruction is Shift Closed Left (SCL), which rotates the contents of a register "n" positions to the left. For example, to rotate R6 four places to the left, the following instruction would be used:

   SCL $R6, 4

   To rotate both R6 and R7 together, a Double Closed Left would be utilized:

   DCL $R7, 4

7. *Generic* instructions have no variable addresses and need only an op code. Typically, these are control instructions. A typical instruction in this group is Monitor Call (MCL), which generates an automatic trap via vector #1.

8. *Input/Output* instructions enable the processor to communicate directly with input/output channels by sending the channel either an output command or an input command request (see Section 2). A typical I/O command is the I/O Load (IOLD) instruction, which sends *both* an address and a range to the addressed channel. Thus, this instruction has three operands and could be written:

   IOLD ADDR, CHAN, RANGE

   This instruction in machine language, depending upon the address form used, could occupy from 3 to 9 words of memory.

9. *Scientific* instructions are all executed by the SIP when it is configured (optional on Models 43, 47, 53, and 57; not available on Models 23 and 33). If the SIP is not configured (or offered), then the scientific instructions are trapped and emulated by software (assuming that the SIP software simulator is configured).

10. *Commercial* instructions are all executed by the commercial processor (standard on Models 47 and 57; not available on Models 23, 33, 43, and 53). On the latter models, the commercial instructions are trapped and emulated by software (assuming that the commercial processor software simulator is configured).

[1] Instruction examples will be given in Assembly Notation. For details, see the *Level 6 Assembly Language Manual,* Order No. AS31.

SAF and LAF Mode Impact on Instructions

The operation mode of the CP impacts instructions in two ways:

- o Instruction size – this is a factor whenever an instruction specifies an IMA form of addressing. In SAF mode the instruction consists of two words while in LAF mode the instruction consists of three words. Consequently using the wrong instruction size not only results in the erroneous execution of the instruction but also results in mispositioning of the program counter.
- o Instruction Execution – this relates to those instructions which operate on base registers or address information since addresses are 16 bits in SAF mode and 20 bits in LAF mode. Consequently when either loading or storing a B-register or address information, the correct size storage is required. The following instructions operate on address information: LDB, STB, CMB, SWB, SAVE, and RSTR.

SAF/LAF Independent Code (SLIC)

Two techniques are available to achieve SAF/LAF independence:

- o *SAF/LAF Independence by Reassembly.* A program must be reassembled for the addressing mode in which it will execute. Rules required to achieve this are provided in the *GCOS 6 Program Preparation Manual,* Order Number CB01. Refer specifically to Appendix A.
- o *SAF/LAF Independence at Loading.* A program is modified at the time it is loaded for the addressing mode in which it will execute. Detailed rules for writing software in this fashion are described in Appendix A of CB01.

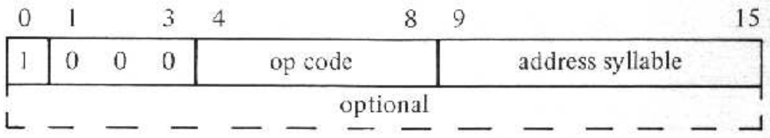
Pre-fetch Capability and Self-Modifying Code

Model 43 and larger models have a pre-fetch or "look ahead" capability in which two words following the current instruction are pre-fetched to achieve greater processing speed. Therefore, programmers should avoid using an instruction modifying another that follows it without an intervening branch, since the modification might take place in the memory location from which the instruction has already been pre-fetched.

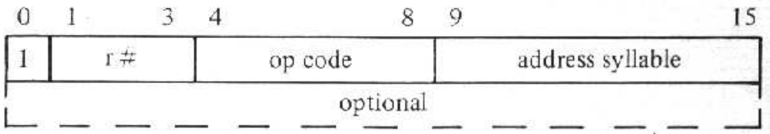
INSTRUCTION FORMATS AND ADDRESSING MODES

Single and Double Operand Instructions

The format for single and double operand instructions is as follows:



SINGLE OPERAND



DOUBLE OPERAND

The significance of the bits is as follows: bit 0 is always a 1; bits 1, 2 and 3 are 0 for single operand instructions and define a register number (1–7) in double operand instructions (the op code defines whether this is one of the 7 general (R) registers or one of the 7 address (B) registers); bits 4 to 8 define the operation code; bits 9–15 are the Address Syllable and are used to define either:

- o a *location in memory* that contains an operand
- o a *register* that contains an operand
- o an *immediate operand* where the operand is contained in the subsequent word(s) of the instruction

Single and double operand instructions can be one to four words in length depending on the addressing mode utilized. The major breakdowns are register addressing, memory addressing, and immediate addressing. An assembly language example for an add instruction is shown next to each mode. For further information on instruction addressing in assembly language, see the referenced assembly language manual for Level 6 systems. Table 4-2 summarizes the addressing modes.

CZ38-00, p. A-

TABLE 4-2. SUMMARY OF ADDRESSING MODES FOR SINGLE AND DOUBLE OPERAND INSTRUCTIONS

Operand Location	Types of Addressing	Instruction Length (Words)[a]	Assembly Example Using ADD Command
Register	Register Addressing	1	ADD $R6, = $R5
Instruction	Immediate Operand	2[b]	ADD $R6, = 1000
Memory	Absolute (Immediate Address)	2 (3 LAF mode)	
	• Direct		ADD $R6, <LOC
	• Indirect		ADD $R6, *<LOC
	• Indexed		ADD $R6, <LOC. $R3
	• Indirect Indexed		ADD $R6, *<LOC. $R3
Memory	Base Addressing	1	
	• Direct		ADD $R6, $B7
	• Indirect		ADD $R6, *$B7
	• Indexed		ADD $R6, $B7. $R3
	• Indirect Indexed		ADD $R6, *$R7. $R3
	• Pre-Decrement		ADD $R6, – $B7
	• Post-Increment		ADD $R6, + $B7
	• Auto-Indexed, Pre-Decrement		ADD $R6, $B3. – $R3
	• Auto-Indexed, Post-Increment		ADD $R6, $B3. + $R3
Memory	Relative Addressing	2	
	• P-Relative Direct		ADD $R6, LOC
	• P-Relative Indirect		ADD $R6, *LOC
	• Base Relative, Direct		ADD $R6, $B7. – 5
	• Base Relative, Indirect		ADD $R6, *$B7.7
	• Interrupt Vector Relative		ADD $R6, $IV.7

[a]Add additional word for mask when required.

[b]Three for LDI, SDI, AID, SID, and Scientific, or LAF mode for LDB, STB, CMB, SWB, and CMN.

The addressing mode is defined by the address syllable; these are as follows:

Absolute addressing – (also called immediate address mode). In SAF mode, a two-word instruction is used, with the second word containing a 16-bit word absolute address that describes a location from 0 to 64K. In LAF mode, a three-word instruction is used, with the last two words containing a 20-bit word absolute address that describes a location from 0 to 1M. This address can be:

- o a direct address
- o an indirect address
- o a direct address indexed by the contents of R1, R2, or R3
- o an indirect address that is post-indexed by the contents of R1, R2, or R3

Base addressing – one-word instructions that define one of the seven base registers (B1-B7) as containing the address of the operand. The address in the register can be:

- o a direct address
- o an indexed address
- o an indirect address
- o an indirect address post-indexed

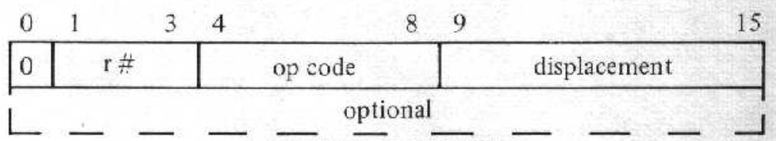
Some extremely powerful additional forms of base addressing are provided. These are still all one-word instructions:

- o Base, pre-decrement (also called push addressing). In this mode one is subtracted from the contents of the base register prior to its being used as an address – unless it is a multiword operation, in which case two or more are subtracted.
- o Base, post-increment (also called pop addressing). Here one (or more, as above) is added to the contents of the base register after it has been used as the base.
- o Base, auto-indexed. Here the contents of an index register R1, R2, or R3 are either pre-decremented (push indexed) or post-incremented (pop indexed) before/after being added to the contents of an address register B1, B2, or B3.

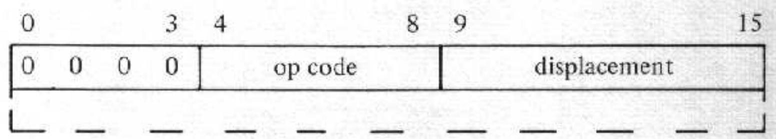
Relative addressing – two-word instructions where the second word contains an algebraic displacement (±32K) relative to either the program counter (P relative), a base register (base relative), or the interrupt vector for the current central processor level (IV relative). The resultant address can be utilized as either a direct or an indirect address (except for IV relative, which is direct only). This does not change in LAF mode (i.e., the 16-bit displacement is still used).

Branch Instructions

There are two types of branch instructions: branch on register and branch on indicator. The formats are as follows:



BRANCH ON REGISTER



BRANCH ON INDICATOR

Table 4-3 shows the three types of addressing modes that can be utilized with branch instructions together with the assembler mnemonics for each.

TABLE 4-3. BRANCH INSTRUCTION ADDRESSING FORMS (BG INSTRUCTION SHOWN)

Short Displacement	1 Word	BG >LOC
Long Displacement	2 Words	BG LOC
Absolute (Immediate Address)	2 Words (3 LAF mode)	BG <LOC

These instructions again can be either single- or multiword instructions. Three addressing modes are possible with branch instructions: short displacement, long displacement, and absolute.

Short Displacement Addressing

In this mode a displacement is contained within a one-word instruction. The displacement is a 7-bit algebraic quantity that is applied to the contents of the program counter. Utilizing this mode of addressing, the program can branch to 64 locations prior to the instruction or 63 locations after it. Displacements of zero and one are not allowed.

Long Displacement Addressing

This mode of addressing is identical to the P-relative addressing mode in single and double operand instructions. The second word of the instruction contains a signed, 16-bit value (±32K) displacement from that word.

Absolute Addressing (Immediate Address)

This is also identical to single and double operand instructions. In SAF mode, a two-word instruction is used, with the second word containing an absolute 16-bit word address that describes a location from 0 to 64K. In LAF mode, a three-word instruction is used, with the last two words containing an absolute 20-bit word address that describes a location from 0 to 1M.

Short Value Immediate Instructions

The format for these instructions is as follows:

0	1 3	4 7	8 15
0	r#	op code	value

Bits 1–3 must specify a general (R) register number. Bits 8–15 contain an arithmetic value between -128 and +127. This value (with its sign extended) is used as an operand by the instructions that utilize this short value immediate addressing form.

Short Value Immediate	1 Word	ADV $R6, =6

Shift Instructions

Shift instructions have the following format:

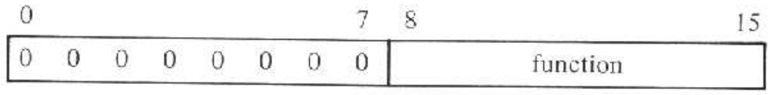
0	1 3	4			7	8 15
0	r#	0	0	0	0	type, direction, & distance

Bits 8–15 are used to specify the type, direction and number of places to be shifted. If the distance field is zero, register R1 will contain the shift distance. Short shifts can specify a distance of up to 15 places; long shifts, up to 31 places. Bits 1–3 specify a general (R) register number. If a double shift is to be executed, this field must address the right-hand (odd) registers as shown below.

Short shift	1 Word	SAL SR5, 6
Long shift	1 Word	DAL $R5, 26

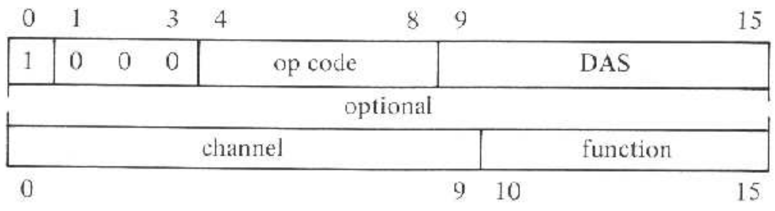
Generic Instructions

Generic instructions have the following format:
Bits 0–7 must be zeros, while bits 8–15 specify the function.

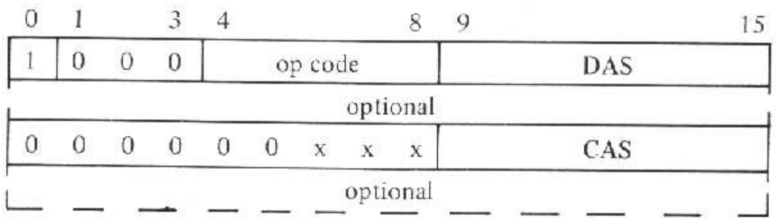


Input/Output Instructions

There are two types of input/output instructions. The first type is used by the input/output word (IO) or by the input/output half-word (IOH) instructions. This is the instruction that is used to place an I/O command on the Level 6 bus (see Section 2). An I/O command consists of a channel number and a function code on the address bus, and a 16-bit data word on the data bus. The instruction format to do this is as follows:

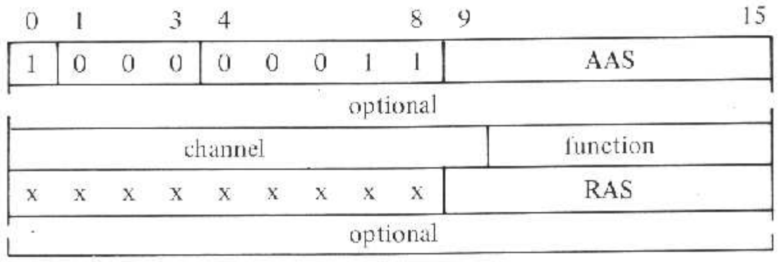


The address of the data word is defined by the Data Address Syllable (DAS) in the least significant 7 bits of the instruction and by a second word or third word (LAF mode), if needed. The addressing forms are the same as for single operand instruction addressing and the second word will be needed for absolute addressing or relative addressing forms. The last word of the instruction contains the channel number and the function code. If it is desired not to embed the channel number and the function code in the procedure, then the instruction can take the following format:

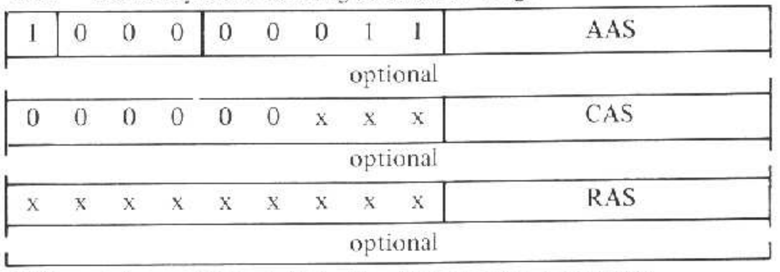


In this case the Channel Address Syllable (CAS) bits point to the location of a word containing the channel and function. Again, a second or third (LAF mode) word may be required to define this address.

The second type of I/O instruction is the IOLD instruction. This is similar except that instead of placing one word of data on the I/O bus it places the address and range that are required to set up a DMA transfer. The format is the same as for the I/O instructions, except that a third address must be specified. Again, this can be one or more words, depending upon the addressing mode utilized. The two cases are thus as follows, with the first embedding the control in the procedure and the second having the control word nonprocedural:



AAS – address syllable defining buffer address
RAS – address syllable defining location of range



CAS – address syllable defining location of word containing channel and function

Scientific Instructions

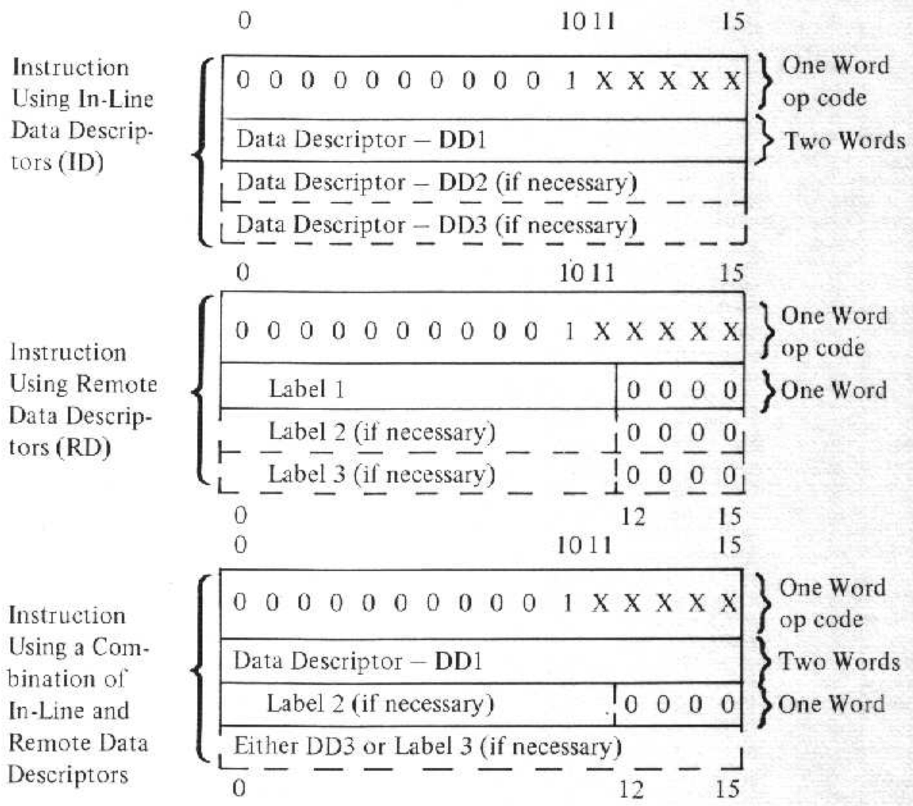
Scientific instructions take the formats of double operand instructions. They are not executed by hardware on Models 23 and 33 but rather cause traps to unique software routines which execute the instructions. On the Model 43 and larger, an optional Scientific Instruction Processor (SIP) is offered. Two trap handlers, the Floating-Point Simulator, entered via trap vector #3, and the Scientific Branch Simulator, entered via trap vector #5, are available and are described in the *GCOS/BES1/2 Executive Modules I/O* manual, Order No. AU45; the *GCOS 6 MOD 400 System Building* manual, Order No. CB23 and the *GCOS 6 System Service and Macro Calls* manual, Order No. CB08.

Commercial Instructions

Commercial instructions take the formats of double operand instructions. They are not executed by hardware on Models 23, 33, 43, and 53, but rather cause traps to unique software routines that execute the instructions. On the Models 47 and 57, a commercial processor is standard. The Commercial Instruction Simulator is the trap handler that is entered via trap vector #5 and is described in the *GCOS 6 MOD 400 System Building* manual, Order No. CB23 and the *GCOS 6 MOD 400 System Service and Macro Calls* manual, Order No. CB08. See also the *GCOS 6 Assembly Language Reference* manual, Order No. CB07.

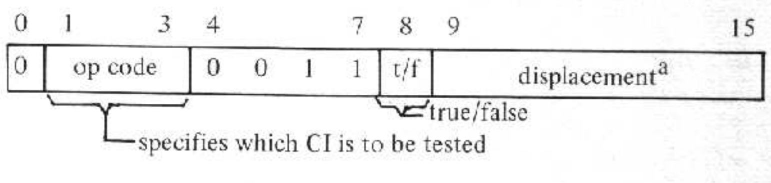
The basic format of CIP instructions is as follows:

Format of Alphanumeric, Numeric, and Edit Instructions


0
10 11
15
Instruction Using In-Line Data Descriptors (ID)
0 0 0 0 0 0 0 0 0 0 1 X X X X X
One Word op code
Data Descriptor – DD1
Two Words
Data Descriptor – DD2 (if necessary)
Data Descriptor – DD3 (if necessary)
Instruction Using Remote Data Descriptors (RD)
Label 1
0 0 0 0
One Word
Label 2 (if necessary)
Label 3 (if necessary)
12
Instruction Using a Combination of In-Line and Remote Data Descriptors
Either DD3 or Label 3 (if necessary)

One, two or three operands are required depending on the CIP instruction. All CIP instructions, except branch CIP instructions, require at least one data descriptor. A data descriptor specifies the type of data on which the instruction is to operate and the location of the data. In a CIP instruction, a label occupies the 12 high-order bits of a word and is capable of addressing any of up to 4K remote data descriptors. The label designates an offset from the remote descriptor base address contained in the CP remote descriptor base register (RDBR). This register can be accessed by use of the CP instructions LRDB and SRDB.

Format of Branch Instructions



[a]If the displacement value specified is 0, the location to be branched to is specified in the next sequential word (two words if in LAF mode); if it is 1, the next sequential word specifies the displacement (in words) from the address of this displacement word; otherwise, the displacement value specified is the displacement, in two's complement form, from the current instruction to the destination.