HONEYWELL

DPS 6 & LEVEL 6 PROGRAMMER'S POCKET GUIDE

HARDWARE

DPS 6 & LEVEL 6 PROGRAMMER'S POCKET GUIDE

SUBJECT

DPS 6 and Level 6 Programmer's Reference Information for Models DPS 6/30, 6/32, 6/34, 6/38, 6/48, 6/54, 6/74, 6/76, 6/92, 6/94, and 6/96

ORDER NUMBER

CU75-00

October 1981

Honeywell

Preface

The information presented in this Pocket Guide covers the complete set of DPS 6 instructions. Details specific to a given DPS 6 model may be found in the *DPS 6 Central Processor Operation Manual,* (Order No. CT43); details specific to a given Level 6 model, in the *Level 6 Systems Handbook* (Order No. CC71).

For all software aspects of this Pocket Guide, refer to the *Level 6 GCOS 6 Assembly Language Reference Manual* (Order No. CB07). For further information, please contact your Honeywell Marketing Representative or Honeywell Field Service Department.

 File No.: 1R03, 1S03

CONTENTS

TABLES

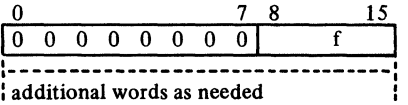
SECTION 1

INSTRUCTION TYPES

BASIC INSTRUCTIONS

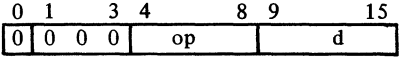
Type/Source Format **Memory Format**

Generic (GE)
[label] op



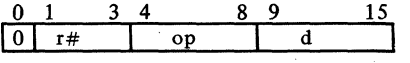
f – Function.

Branch on Indicators (BI)
[label] op branchloc



d – Displacement, calculated by the assembler as follows:
If "<branchloc" is coded, d = 0 and the next word(s) contain a pointer to branchloc.
If "branchloc" is coded, d = 1 and the next word contains the displacement (DSP) to branchloc. If ">branchloc" is coded, d = -64 through +63, which is the displacement to branchloc; ">branchloc" must not tag this instruction or the next word.

Branch on Registers (BR)
[label] op r#, branchloc

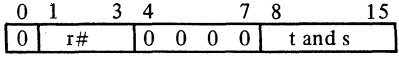


d – See "Branch on Indicators," above.

Note:

Refer to *Assembly Language Reference Manual*, CB07.

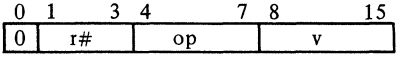
Shift Short (SHS) and Shift Long (SHL) [label] op r#,s



t – Type and direction of shift. Requires bits 8 – 11 for SHS and bits 8 – 10 for SHL.

s – Shift distance. Requires bits 12 – 15 for SHS and bits 11 – 15 for SHL. (If s = 0, the shift distance is obtained from R1.)

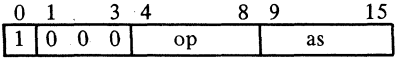
Short Value Immediate (SI) [label] op r#,v



v – Value between -128 and +127.

Input/Output (I/O)

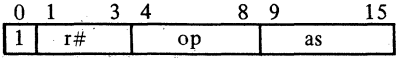
Single Operand (SO) [label] op as[,maskword]



as – See "Address Syllable."

maskword – Used only in SAVE, RSTR, and (optionally) bit instructions. If maskword is all zeros, the mask is obtained from R1.

Double Operand (DO) [label] op r#,as [,maskword]



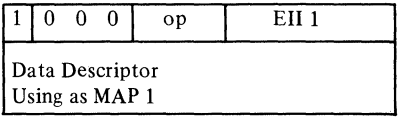
as – See "Address Syllable."

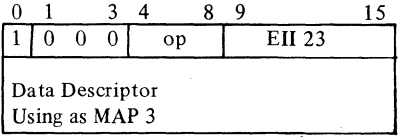
maskword – Used only in SRM instruction. If maskword is all zeros, the mask is obtained from R1.

DSP	16-bit displacement (-32,768 through +32,767)
op	Operation code
r#	Register number
[]	Optional

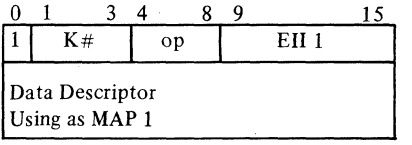
EXTENDED INTEGER INSTRUCTIONS[1]

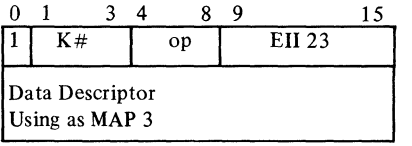
Single Operand (SO)





Double Operand (DO)



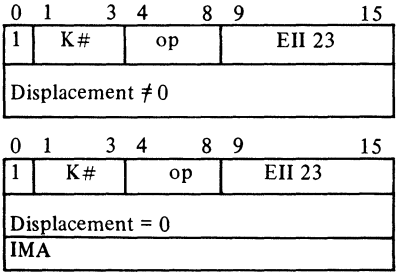


Note:

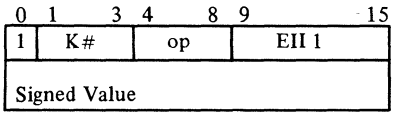
Refer to *Assembly Language Reference Manual*, CB07.

[1] Available only on DPS 6 Models 92 and 96.

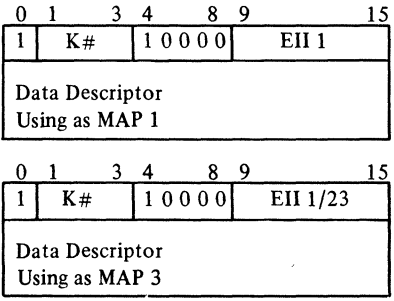
Branch-on-K-Register (BK)



Short Value Immediate (SI)



Load Index and Address (LXA)



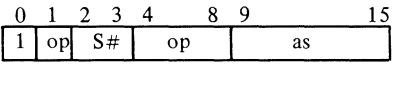
IMA – Immediate Address
op – Operation code
R# – Register number

Note:

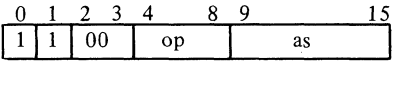
Refer to *Assembly Language Reference Manual*, CB07.

SCIENTIFIC PROCESSOR INSTRUCTIONS[1]

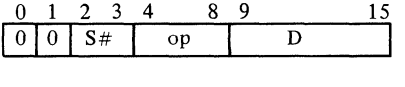
Scientific Double Operand (DO)



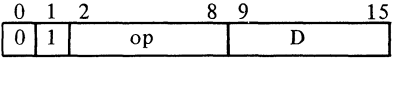
Scientific Single Operand (SO)



Branch on Scientific Register (BR)

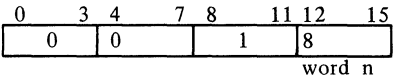


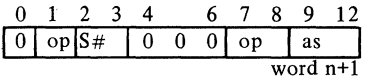
Branch on Scientific Indication (BI)



as – Address Syllable
D – Displacement
op – Operation code
S# – Scientific accumulator number

Scientific Intrinsic





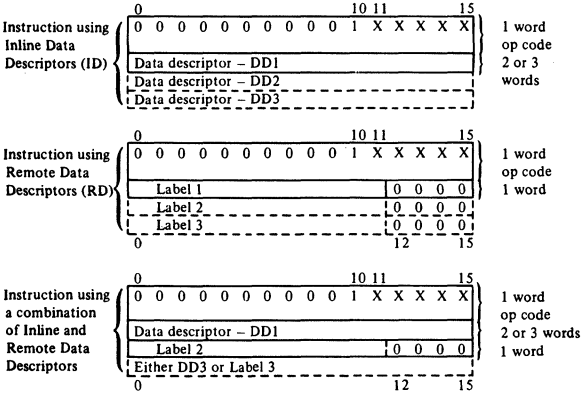
as – Address Syllable
op – Operation code
S# – Scientific accumulator number

Note:

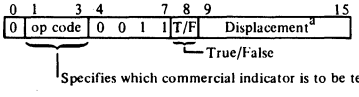
Refer to *Assembly Language Reference Manual,* CB07.

[1] Available only on DPS 6 Models 92 and 96.

COMMERCIAL PROCESSOR INSTRUCTIONS[1]



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; if it is 1, the next sequential word specifies the displacement (in words) from the address of this displacement word to the destination; otherwise, the displacement value specified is the displacement, in twos complement form, from the current instruction to the destination.

Note:

Refer to *Assembly Language Reference Manual*, CB07.

[1] Available only in Extended mode on DPS 6 Models 92 and 96.

SECTION 2

BASIC INSTRUCTIONS

Mnemonic	Type	Description	Operation
ADD	DO	Add to R Reg	R# ← R# + [EA] l(ov), l(c) affected
ADV	SI	Add Value to R Reg	R# ← R# + V l(ov), l(c) affected
ACQ	GE	Acquire Stack Space	See Note
AID	SO	Add Double Word Integer	R6, R7, ← R6, R7 + [EA] l(ov), l(c) affected
AND	DO	AND With R Reg	R# ← R# ∧ [EA]
ANH	DO	AND Halfword With R Reg	R# ← R# ∧ [EA] sign extended
ASD	GE	Activate Segment Descriptor	See Note
B	BI	Branch	P ← EA
BAG	BI	Br Alg Greater Than	If l(g) ⊕ l(u) = 1, then P ← EA
BAGE	BI	Br Alg Greater Than or Equal	If l(l) ⊕ l(u) = 0, then P ← EA
BAL	BI	Br Alg Less Than	If l(l) ⊕ l(u) = 1, then P ← EA
BALE	BI	Br Alg Less Than or Equal	If l(g) ⊕ l(u) = 0, then P ← EA
BBF	BI	Br on Bit Test Indicator False	If l(b) = 0, then P ← EA
BBT	BI	Br on Bit Test Indicator True	If l(b) = 1, then P ← EA
BCF	BI	Br on Carry False	If l(c) = 0, then P ← EA
BCT	BI	Br on Carry True	If l(c) = 1, then P ← EA
BDEC	BR	Br After Decrementing R Reg	R# ← R# + FFFF; if R# ≠ FFFF, then P ← EA
BE	BI	Br Equal	If l(l) ∨ l(g) = 0, then P ← EA
BEVN	BR	Br if R Reg Even	If R#(15) = 0, then P ← EA
BEZ	BR	Br if R Reg Equal to Zero	If R# = 0, then P ← EA
BG	BI	Br Greater Than	If l(g) = 1, then P ← EA
BGE	BI	Br Greater Than or Equal	If l(l) = 0, then P ← EA
BGEZ	BR	Br if R Reg Greater Than or Equal to Zero	If R#(0) = 0, then P ← EA
BGZ	BR	Br if R Reg Greater Than Zero	If R#(1:15) ≠ 0 and R#(0) = 0, then P ← EA
BINC	BR	Br After Incrementing R Reg	R# ← R# + 0001; If R# ≠ 0, then P ← EA
BIOF	BI	Br on I/O Indicator False	If l(i) = 0, then P ← EA
BIOT	BI	Br on I/O Indicator True	If l(i) = 1, then P ← EA
BL	BI	Br Less Than	If l(l) = 1, then P ← EA
BLE	BI	Br Less Than or Equal	If l(g) = 0, then P ← EA
BLEZ	BR	Br if R Reg Less Than or Equal to Zero	If R#(0) = 1 or R# = 0, then P ← EA
BLZ	BR	BR if R Reg Less Than Zero	If R#(0) = 1, then P ← EA
BNE	BI	Br Not Equal	If l(l) ∨ l(g) = 1, then P ← EA
BNEZ	BR	Br if R Reg Not Equal to Zero	If R# ≠ 0, then P ← EA

Note:

Refer to *Assembly Language Reference Manual*, CB07.

Mnemonic	Type	Description	Operation
BNOV	BI	Br on Not Overflow	If I(ov) = 0, then P ← EA
BODD	BR	Br if R Reg Odd	If R#(15) = 1, then P ← EA
BOV	BI	Br on Overflow	If I(ov) = 1, then P ← EA
BRK	GE	Breakpoint	Generate trap via Trap Vector #2.
BSE	BI	Br Signs Equal	If I(u) = 0, then P ← EA
BSS	GE	Bit String Search	See Note
BSU	BI	Br Signs Unlike	If I(u) = 1, then P ← EA
CAD	SO	Carry Add	[EA] ← [EA] + I(c) I(ov), I(c) affected
CL	SO	Clear Memory	[EA] ← 0000
CLH	SO	Clear Memory Halfword	[EA] ← 00
CMB	DO	Compare B Reg	B# :: [EA] I(g); I(l) affected I(u) affected but undefined
CMH	DO	Compare R Reg With Halfword	R# :: [EA] sign extended I(g), I(l), I(u) affected
CMN	SO	Compare With Null	[EA] :: Null Address I(g), I(l) affected I(u) affected but undefined
CMR	DO	Compare R Reg	R# :: [EA] I(g), I(l), (u) affected
CMV	SI	Compare R Reg With Value	R# :: V sign extended I(g), I(l), I(u) affected
CMZ	SO	Compare With Zero	[EA] :: 0000 I(g), I(l), I(u) affected
CNFG	GE	Configure	See Note
CPL	SO	Complement	[EA] ← [EA] ⊕ FFFF
CVP	GE	Convert Virtual Address to Physical Address	See Note
DAL	SHL	Dbl Shift Arith Left	See Note
DAR	SHL	Dbl Shift Arith Right	See Note
DCL	SHS	Dbl Shift Closed Left	See Note
DCR	SHS	Dbl Shift Closed Right	See Note
DEC	SO	Decrement	[EA] ← [EA] + FFFF; I(b) ← [EA] (0) I(ov), I(c), I(b) affected
DIV	DO	Divide R Reg	R# ← R# / [EA] except if r# = R7, then R7 ← R6, R7 / [EA] and R6 ← remainder I(ov), I(c) affected
DOL	SHL	Dbl Shift Open Left	See "Shift Instructions"
DOR	SHL	Dbl Shift Open Right	See "Shift Instructions"
DQA	GE	Dequeue by Address	See Note
DQH	GE	Dequeue from Head	See Note
ENT	SO	Enter	P ← EA; S(p) ← 0
HLT	GE	Halt	See Note
INC	SO	Increment	I(b) ← [EA] (0); [EA] ← [EA] + 0001 I(ov), I(c), I(b) affected
IO	I/O	Input/Output (Word)	See Note
IOH	I/O	Input/Output Halfword	See Note
IOLD	I/O	Input/Output Load	See Note
JMP	SO	Jump	P ← EA
LAB	DO	Load EA into B Reg	B# ← EA
LB	SO	Load Bit	I(b) ← [EA] i (if indexed) I(b) ← V[EA] m (if masked) I(b) affected

Note:

Refer to *Assembly Language Reference Manual*, CB07.

Mnemonic	Type	Description	Operation
LBC	SO	Load Bit and Complement	l(b) ← [EA] i; [EA] i ← $\overline{[EA]\ i}$ (if indexed) l(b) ← ∨ [EA] m; [EA] m ← $\overline{[EA]\ m}$ (if masked) l(b) affected
LBF	SO	Load Bit and Set False	l(b) ← [EA] i; [EA] i ← 0 (if indexed) l(b) ← ∨ [EA] m; [EA] m ← 0 (if masked) l(b) affected
LBS	SO	Load Bit and Swap	l(b) ↔ [EA] i (if indexed) Temp ← l(b); l(b) ← ∨ [EA] m; [EA] m ← Temp (if masked) i(b) affected
LBT	SO	Load Bit and Set True	l(b) ← [EA] i; [EA] i ← 1 (if indexed) l(b) ← ∨ [EA] m; [EA] m ← 1 (if masked) l(b) affected
LDB	DO	Load B Reg	B# ← [EA]
LDH	DO	Load Halfword Into R Reg	R# ← [EA] sign extended
LDI	SO	Load Doubleword Integer	R6, R7 ← [EA]
LDR	DO	Load R Reg	R# ← [EA]
LDT	GE	Load T Register	T ← B#
LDV	SI	Load Value Into R Reg	R# ← ∨ sign extended
LEV	SO	Level Change	See Note
LLH	DO	Load Logical Halfword	R#(8:15) ← [EA]; R#(0:7) ← 00
LNJ	DO	Load B Reg and Jump	B# ← NSIA;P ← EA
LRDB	GE	Load Remote Descriptor Base Register	RDBR ← B3
MCL	GE	Monitor Call Via Trap	Generate trap via Trap Vector #1
MFL	GE	Modify Frame Length	See Note
MLV	SI	Multiply R Reg by Value	R# ← R# * ∨ except if r# = 7, then R6, R7 ← R7 * ∨ l(ov) affected
MMM	GE	Memory to Memory Move	See Note
MTM	DO	Modify Test M Reg	See Note
MUL	DO	Multiply R Reg	R# ← R# * [EA] except if r# = R7, then R6, R7 ← R7 * [EA] l(ov) affected
NEG	SO	Negate	[EA] ← 0000 [EA] l(ov), l(c) affected
NOP	BI	No Operation	None
OR	DO	OR With R Reg	R# ← R# ∨ [EA]
ORH	DO	OR Halfword With R Reg	R# ← R# ∨ [EA] sign extended
QOH	GE	Queue on Head	See Note
QOT	GE	Queue on Tail	See Note
RLQ	GE	Relinquish Stack Space	See Note
RSTR	SO	Restore Context	See Note
RTCF	GE	Real Time Clock Off	See Note
RTCN	GE	Real Time Clock On	See Note
RTT	GE	Return From Trap	See Note
SAL	SHS	Sgl Shift Arith Left	See "Shift Instructions"

Note:

Refer to *Assembly Language Reference Manual,* CB07.

Mnemonic	Type	Description	Operation
SAR	SHS	Sgl Shift Arith Right	See "Shift Instructions"
SAVE	SO	Save Context	See Note
SCL	SHS	Sgl Shift Closed Left	See "Shift Instructions"
SCR	SHS	Sgl Shift Closed Right	See "Shift Instructions"
SDI	SO	Store Doubleword Integer	[EA] ← R6, R7
SID	SO	Subtract Doubleword Integer	R6, R7 ← R6, R7 + [EA] +1; l(ov), l(c) affected
SOL	SHS	Sgl Shift Open Left	See "Shift Instructions"
SOR	SHS	Sgl Shift Open Right	See "Shift Instructions"
SQA	GE	Search Queue by Address	See Note
SQP	GE	Search Queue by Priority	See Note
SRDB	GE	Store Remote Descriptor Base Register	B3 ← RDBR
SRM	DO	Store Reg Masked	[EA] ← (R# ∧ m) ([EA] ∧ $\overline{m}$)
STB	DO	Store B Reg	[EA] ← B#
STH	DO	Store Halfword From R Reg	[EA] ← R# (8:15)
STM	DO	Store M Reg	[EA] (8:15) ← M1 [EA] (0:7) ← FF if r# ≠ M1, generate trap via Trap Vector #5 (Model 33 only)
STR	DO	Store R Reg	[EA] ← R#
STS	SO	Store S Reg	[EA] ← S
STT	GE	Store T Reg	B ← T
SUB	DO	Subtract From R Reg	R# ← R# − [EA] l(ov), l(c) affected
SWB	DO	Swap B Reg	B# ↔ [EA]
SWR	DO	Swap R Reg	R# ↔ [EA]
VLD	GE	Validate	See Note
WDTF	GE	Watchdog Timer Off	See Note
WDTN	GE	Watchdog Timer On	See Note
XOH	DO	Exclusive OR Halfword with R Reg	R# ← R# ⊕ [EA] sign extended
XOR	DO	Exclusive OR With R Reg	R# ← R# ⊕ [EA]

B#	Value store specified B register
EA	Effective address
[EA]	Value of operand (bit, halfword, word, doubleword) pointed to by the effective address
[EA] (0)	Value of bit 0 of operand pointed to by the effective address
[EA] (0:7)	Value of bits 0 through 7 of operand pointed to by the effective address
[EA] (8:15)	Value of bits 8 through 15 of operand pointed to by the effective address
[EA] i	Value of single bit obtained through an indexed address
$\overline{[EA]\ i}$	Complement value of single bit obtained through an indexed address
[EA] m	Value of each masked bit in the word pointed to by the effective address
$\overline{[EA]\ m}$	Complement value of each masked bit in the word pointed to by the effective address
∨[EA] m	Single bit value obtained by an inclusive OR operation on the logical product of (1) the designated mask and (2) the word pointed to by the effective address

Note:

Refer to *Assembly Language Reference Manual*, CB07.

I(b)	"Bit Test" indicator bit of I register
I(c)	"Carry" indicator bit of I register
I(g)	"Greater than" indicator bit of I register
I(i)	"Input/output" indicator bit of I register
I(l)	"Less than" indicator bit of I register
I(ov)	"Overflow" indicator bit of I register
I(u)	"Unlike signs" indicator bit of I register
m	Value of mark operand
$\overline{m}$	Complement of mask operand value
M#	Specified M register
NSIA	Next sequential instruction address
P	Program counter (P register)
r#	Register number
R#	Specified R register
R#(0)	Bit 0 of specified R register
R#(15)	Bit 15 of specified R register
R#(0:7)	Bits 0 through 7 of specified R register
R#(1:15)	Bits 1 through 15 of specified R register
R#(8:15)	Bits 8 through 15 of specified register
S	S register
S(p)	"Privilege state" bits of S register
Temp	Temporary storage location for a single bit; the value stored in this temporary storage location
T	Stack address register
V	Value in bits 8 through 15 of this instruction
$\oplus$	Exclusive OR
$\vee$	Inclusive OR
$\wedge$	Logical AND
/	Division operator
*	Multiplication operator
;	Separator for nonsimultaneous operations
::	Is compared with
$\leftarrow$	Is replaced by
$\leftrightarrow$	Is exchanged with

SECTION 3

EXTENDED INTEGER INSTRUCTIONS[1] (ALPHABETICAL)

Mnemonic	Type	Description	Operation
ADDK	DO	Add to K Reg	[K#] ← [K#] + [EA] I(c), I(ov) affected
ADVK	SI	Add Value to K Reg	[K#] ← [K#] + V I(c), I(ov) affected
ANDK	DO	AND with K Reg	[K#] ← [K#] ∧ [EA]
CMK	DO	Compare K Reg	[K#] :: [EA] I(g), I(l), I(u) affected
CMVK	SI	Compare K Reg with Value	[K#] :: V I(g), I(l), I(u) affected
CPLK	SO	Complement	[EA] ← [EA]
DALK	SHL	Dbl Shift Arith Left	See "Shift Instructions"
DARK	SHL	Dbl Shift Arith Right	See "Shift Instructions"
DCLK	SHL	Dbl Shift Closed Left	See "Shift Instructions"
DCRK	SHL	Dbl Shift Closed Right	See "Shift Instructions"
DECK	SO	Decrement	[EA] ← [EA] -1 I(ov), I(c) affected
DIVK	DO	Divide K Reg	[K#] ← [K#]/[EA] I(ov), I(c) affected
DOLK	SHL	Dbl Shift Open Left	See "Shift Instructions"
DORK	SHL	Dbl Shift Open Right	See "Shift Instructions"
INCK	SO	Increment	[EA] ← [EA] +1 I(ov), I(c) affected
KBEVN	BK	Branch if K Reg Even	If K# (31) = 0, then P ← EA
KBEZ	BK	Branch if K Reg Equal to Zero	If K# = 0, then P ← EA
KBGEZ	BK	Branch if K Reg Greater Than or Equal to Zero	If K# (0) = 0, then P ← EA
KBGZ	BK	Branch if K Reg Greater Than Zero	If K# (1:31) = 0 and K# (0) = 0, then P ← EA
KBLEZ	BK	Branch if K Reg Equal to or Less Than Zero	If K# (0) = 1 or K# = 0, then P ← EA
KBLZ	BK	Branch if K Reg Less Than Zero	If K# (0) = 1, then P ← EA
KBNEZ	BK	Branch if K Reg Not Equal to Zero	If K# = 0, then P ← EA
KBODD	BK	Branch if K Reg Odd	If K# (31) = 1, then P ← EA
LDK	DO	Load K Reg	[K#] ← [EA]

Note:

Refer to *Assembly Language Reference Manual*, CB07.

[1] Available only on DPS 6 Models 92 and 96.

Mnemonic	Type	Description	Operation
LDVK	SI	Load Value in K Reg	[K#] ← V
LXA	TO	Load Index and Address	See Note
MLVK	SI	Multiply K Reg by Value	[K#] ← [K#] * V I(ov) affected
MULK	DO	Multiply K Reg	[K#] ← [K#] * [EA] I(ov) affected
NEGK	SO	Negate	[EA] ← 0 − [EA] I(ov), I(c) affected
ORK	DO	OR with K Reg	[K#] ← [K#] V [EA]
STK	DO	Store K Reg	[EA] ← [K#] I(ov) affected
SUBK	SO	Subtract from K Reg	[K#] ← [K#] − [EA] I(c), I(ov) affected
SWK	DO	Swap K Reg	[K#] ↔ [EA] I(ov) affected
XORK	DO	Exclusive OR with K Reg	[K#] ← [K#] ⊕ [EA]

Note:

Refer to *Assembly Language Reference Manual*, CB07.

EA	Effective address
[EA]	Value of operand (bit, halfword, word, doubleword) pointed to by the effective address
EA	Complement value of each bit in the word pointed to by the effective address
I(c)	"Carry" indicator bit of 1 register
I(g)	"Greater than" indicator bit of 1 register
I(l)	"Less than" indicator bit of 1 register
I(ov)	"Overflow" indicator bit of 1 register
I(u)	"Unlike signs" indicator bit of 1 register
P	Program counter (P register)
K#	Specified K register
K#(0)	Bit 0 of specified K register
K#(31)	Bit 31 of specified K register
K#(1:31)	Bits 1 through 31 of specified K register
⊕	Exclusive OR
v	Inclusive OR Logical AND
/	Division operator
*	Multiplication operator
::	Is compared with
←	Is replaced with
↔	Is exchanged with
V	Value in bits 0 through 15 of the second word of the instruction

SECTION 4

SCIENTIFIC INSTRUCTIONS (ALPHABETICAL)

Mnemonic	Type	Description	Operation
SAD	DO	Scientific Add	[SA#] ← [SA#] + [EA]
SART[1]	DO	Scientific Arc Tangent	[SA#] ← TAN^{-1} [EA]
SBE	BI	Branch if Equal	If SI (L) V SI (G) = 0, then [P] ← EA
SBEU	BI	Branch on Exponent Underflow	If [SI (EUF)] = 1, then [P] ← EA
SBEZ	BR	Branch on SA = 0	If [SA# (f)] = 0, then [P] ← EA
SBG	BI	Branch on Greater Than	If [SI (G)] = 1, then [P] ← EA
SBGE	BI	Branch on Greater Than or Equal to	If [SI (L)] = 0, then [P] ← EA
SBGEZ	BR	Branch on SA > 0	If [SA# (s)] = 0, or if [SA# (f)] = 0, then [P] ← EA
SBGZ	BR	Branch on SA > 0	If [SA# (f)] ≠ 0 and [SA# (s)] = 0, then [P] ← EA
SBL	BI	Branch on Less Than	If [SI (L)] = 1, then [P] ← EA
SBLE	BI	Branch on Less Than or Equal to	If [SI (G)] : 0, then [P] ← EA
SBLEZ	BR	Branch on SA < 0	If [SA# (f)] = 0 or if [SA# (s)] = 1, then [P] ← EA
SBLZ	BR	Branch on SA < 0	If [SA# (s)] = 1 and [SA# (f)] ≠ 0, then [P] ← EA
SBNE	BI	Branch on Not Equal	If [SI (L) V SI (G)] = 1, then [P] ← EA
SBNEU	BI	Branch on No Exponent Underflow	If [SI (EUF)] = 0, then [P] ← EA
SBNEZ	BR	Branch on SA ≠ 0	If [SA# (f)] ≠ 0, then [P] ← EA
SBNPE	BI	Branch on No Precision Error	If [SI (PE)] = 0, then [P] ← EA
SBNSE	BI	Branch on No Significance Error	If [SI (SE)] = 0, then [P] ← EA
SBPE	BI	Branch on Precision Error	If [SI (PE)] = 1, then [P] ← EA
SBSE	BI	Branch on Significance Error	If [SI (SE)] = 1, then [P] ← EA
SCM	DO	Scientific Compare	SI (G), SI (L) ← [SA#] :: [EA]
SCOS[1,2]	DO	Scientific Trigonometric Cosine	[SA#] ← COSINE [EA]
SCZD	SO	Scientific Compare to Zero – 2 words	SI (G), SI (L) ← [EA] :: Zero

[1] Available only on DPS 6 Models 92 and 96.
[2] Units are in radian measure.

Mnemonic	Type	Description	Operation
SCZQ	SO	Scientific Compare to Zero – 4 words	SI (G), SI (L) ← [EA] :: Zero
SDTH[1,2]	DO	Convert Hex to Decimal	[SA] ← [EA]
SDV	DO	Scientific Divide	[SA#] ← [SA#] / [EA]
SEXP[1]	DO	Scientific Exponentiation	[SA#] ← e[EA]
SHTD[1,2]	DO	Scientific Convert Hexadecimal to Decimal	[EA] ← [SA]
SLD	DO	Scientific Load	[SA#] ← [EA]
SLOG[1]	DO	Scientific Natural Logarithm	[SA#] ← LOG_e [EA]
SML	DO	Scientific Multiply	[SA#] ← [SA#] * [EA]
SNGD	SO	Scientific Negate – 2 words	[EA (s)] ← $\overline{[EA\ (s)]}$
SQRT[1]	DO	Scientific Square Root	[SA#] ← [EA][1,2]
SNGQ	SO	Scientific Negate	[EA (s)] ← [EA (s)]
SSB	SO	Scientific Subtract	[SA#] ← [SA#] – [EA]
SSIN[1,3]	DO	Scientific Trigonometric Sine	[SA#] ← SINE [EA]
SST	DO	Scientific Store	[EA] ← [SA#]
SSW	DO	Scientific Swap	[SA#] ↔ [EA]
where	f = Fraction (mantissa) excluding sign SA = Scientific Accumulator		

Note:
These instructions are software simulated except on Models 48, 54, 74, 76, 92, 94, and 96.

[1] Available only on DPS 6 Models 92 and 96.
[2] Refer to *Assembly Language Reference Manual*, CB07.
[3] Units are in radian measure.

SECTION 5

COMMERCIAL INSTRUCTIONS (ALPHABETICAL)

Mnemonic[1]	Type	Description	Operation
ACM	A	Alphanumeric Compare	[DD1] :: [DD2] → CI(G,L)
ALR	A	Alphanumeric Move	[DD1] → [DD2]
AME	E	Alphanumeric Move and Edit	[DD1] edited → [DD2]; [DD3] specifies Micro-ops
CBD	N	Convert Binary to Decimal	[DD1] converted → [DD2]
CBE	B	Branch if Equal	If CI (G and L) = 0, then [P] ← EA
CBG	B	Branch if Greater	If CI (G) = 1, then [P] ← EA
CBGE	B	Branch if Greater Than or Equal	If CI (L) = 0, then [P] ← EA
CBL	B	Branch if Less	If CI (L) = 1, then [P] ← EA
CBLE	B	Branch if Less Than or Equal	If CI (G) = 0, then [P] ← EA
CBNE	B	Branch if Not Equal	If CI (G or L) = 1, then [P] ← EA
CBNOV	B	Branch if No Overflow	If CI (OV) = 0, then [P] ← EA
CBNSF	B	Branch if No Sign Fault	If CI (SF) = 0, then [P] ← EA
CBNTR	B	Branch on No Truncation	If CI (TR) = 0, then [P] ← EA
CBOV	B	Branch on Overflow	If CI (OV) = 1, then [P] ← EA
CBSF	B	Branch on Sign Fault	If CI (SF) = 1, then [P] ← EA
CBTR	B	Branch on Truncation	If CI (TR) = 1, then [P] ← EA
CDB	N	Convert Decimal to Binary	[DD1] converted → [DD2]
CSNCB	B	Synchronize and Branch	Prevents CP from going to next instruction unit previous commercial instruction is completed; then performs unconditional branch
CSYNC	B	Synchronize	Prevents CP from going to next instruction until previous commercial instruction is completed
DAD	N	Decimal Add	[DD2] + [DD1] → [DD2]
DCM	N	Decimal Compare	[DD1] :: [DD2] → IND
DDV	N	Decimal Divide	[DD2]/[DD1] → [DD3]; R → [DD2]
DLS	E	Decimal Left Shift	Shift [DD1] left "d" positions
DMC	N	Decimal Move and Convert	[DD1] converted → [DD2]
DME	E	Decimal Move and Edit	[DD1] Edited → [DD2]; [DD3] specifies Micro-ops
DML	N	Decimal Multiply	[DD2] * [DD1] → [DD2]
DRS	E	Decimal Right Shift	Shift [DD1] right "d" position

[1] All DPS includes CIP functionality.

[2] DLS and DRS are software instructions that generate the DSH code and the appropriate value of the shift control word. Refer to footnote 1 on page 6-3 and to the *GCOS 6 Assembly Language Reference Manual*, Order No. CB07, for a complete description.

Mnemonic[1]	Type	Description	Operation
DSB	N	Decimal Subtract	[DD2] – [DD1] → [DD2]
DSH	N	Decimal Shift	Shift [DD1] left "d" positions Shift [DD1] right "d" positions
MAT	A	Alphanumeric Move and Translate	[DD1] translate → [DD2]; [DD3] specifies 256-byte Translate Table
SRCH	A	Alphanumeric Search	[DD3] is searched using [DD1] and [DD2] for some purposes
VRFY	A	Alphanumeric Verify	[DD3] is verified using [DD1] and [DD2] for some purposes

where:

DD1, 2 and 3 = Data Description 1, 2, or 3
N = Numeric Type
A = Alphanumeric Type
E = Edit Type
B = Branch Type

[1] All DPS includes CIP functionality.

[2] DLS and DRS are software instructions that generate the DSH code and the appropriate value of the shift control word. Refer to footnote 1 on page 6-3 and to the *GCOS 6 Assembly Language Reference Manual*, Order No. CB07, for a complete description.

SECTION 6

INSTRUCTIONS (NUMERICAL)

H1	H2	H3	H4	Mnemonic
Generic (GE)				
0	0	0	0	HLT
0	0	0	1	MCL
0	0	0	2	BRK
0	0	0	3	RTT
0	0	0	4	RTCN
0	0	0	5	RTCF
0	0	0	6	WDTN
0	0	0	7	WDTF
0	0	0	8	MMM
0	0	0	A	ASD
0	0	0	B	VLD
0	0	0	C	LRDB
0	0	0	D	SRDB
0	0	1	0	LDT*
0	0	1	0	STT*
0	0	1	0	ACQ*
0	0	1	0	RLQ*
0	0	1	0	MFL
0	0	1	1	CNFG
0	0	1	2	BSS
0	0	1	8	Scientific Intrinsics
0	0	1	B	CVP
0	0	2	–	Commercial
0	0	6	0	DQA
0	0	6	1	QOT
0	0	6	2	DQH
0	0	6	3	QOH
0	0	6	4	SQA
0	0	6	6	SQP

*Function determined by second word of instruction. See *Assembly Language Reference Manual*, CB07.

Commercial Branches				
1	3	0+x	–	CBOV
1	3	8+x	–	CBNOV
2	3	0+x	–	CBTR
2	3	8+x	–	CBNTR
3	3	0+x	–	CBSF
3	3	8+x	–	CBNSF
4	3	0+x	–	CSYNC
4	4	8+x	–	CSNCB
5	3	0+x	–	CBNE
5	3	8+x	–	CBE
6	3	0+x	–	CBG
6	3	8+x	–	CBLE
7	3	0+x	–	CBL
7	3	8+x	–	CBGE

H1	H2	H3	H4	Mnemonic
Branch on Indicators (BI)				
0	2	0+x	–	BL
0	2	8+x	–	BGE
0	3	0+x	–	BG
0	3	8+x	–	BLE
0	4	0+x	–	BOV
0	4	8+x	–	BNOV
0	5	0+x	–	BBT
0	5	8+x	–	BBF
0	6	0+x	–	BCT
0	6	8+x	–	BCF
0	7	0+x	–	BIOT
0	7	8+x	–	BIOF
0	8	0+x	–	BAL
0	8	8+x	–	BAGE
0	9	0+x	–	BE
0	9	8+x	–	BNE
0	A	0+x	–	BAG
0	A	8+x	–	BALE
0	B	0+x	–	BSU
0	B	8+x	–	BSE
0	F	0+x	–	NOP
0	F	8+x	–	B
Scientific Branches				
0+r	4	0+x	–	SBLZ
4	4	0+x	–	SBL
5	4	0+x	–	SBPE
6	4	0+x	–	SBSE
7	4	0+x	–	SBEU
0+r	4	8+x	–	SBGEZ
4	4	8+x	–	SBGE
5	4	8+x	–	SBNPE
6	4	8+x	–	SBNSE
7	4	8+x	–	SBNEU
0+r	5	0+x	–	SBEZ
4	5	0+x	–	SBE
0+r	5	8+x	–	SBNEZ
4	5	8+x	–	SBNE
0+r	6	0+x	–	SBGZ
4	6	0+x	–	SBG
0+r	6	8+x	–	SBLEZ
4	6	8+x	–	SBLE
Branch on Registers (BR)				
0+r	7	0+x	–	BDEC
0+r	7	8+x	–	BINC
0+r	8	0+x	–	BLZ
0+r	8	8+x	–	BGEZ
0+r	9	0+x	–	BEZ
0+r	9	8+x	–	BNEZ
0+r	A	0+x	–	BGZ
0+r	A	8+x	–	BLEZ
0+r	B	0+x	–	BEVN
0+r	B	8+x	–	BODD
Short Value Immediate (SI)				
0+r	C	–	–	LDV
0+r	D	–	–	CMV
0+r	E	–	–	ADV
0+r	F	–	–	MLV

H1	H2	H3	H4	Mnemonic
Input/Output (I/O)				
8	0	0+x	–	IO
8	1	0+x	–	IOH
8	1	8+x	–	IOLD
Single Operand (SO)				
8	2	0+x	–	NEG
8	2	8+x	–	LB
8	3	8+x	–	JMP
8	4	0+x	–	AID
8	4	8+x	–	SID
8	6	0+x	–	CPL
8	7	0+x	–	CL
8	7	8+x	–	CLH
8	8	0+x	–	LBF
8	8	8+x	–	DEC
8	9	0+x	–	LBT
8	9	8+x	–	CMZ
8	A	0+x	–	LBS
8	A	8+x	–	INC
8	B	0+x	–	LBC
8	B	8+x	–	ENT
8	C	0+x	–	STS
8	C	8+x	–	LDI
8	D	0+x	–	SDI
8	D	8+x	–	CMN
8	E	0+x	–	LEV
8	E	8+x	–	CAD
8	F	0+x	–	SAVE
8	F	8+x	–	RSTR
Double Operand (DO)				
8+r	0	0+x	–	MTM
8+r	0	8+x	–	LDH
8+r	1	8+x	–	CMH
8+r	2	0+x	–	SUB
8+r	2	8+x	–	LLH
8+r	3	0+x	–	DIV
8+r	3	8+x	–	LNJ
8+r	4	0+x	–	OR
8+r	4	8+x	–	ORH
8+r	5	0+x	–	AND
8+r	5	8+x	–	ANH
8+r	6	0+x	–	XOR
8+r	6	8+x	–	XOH
8+r	7	0+x	–	STM
8+r	7	8+x	–	STH
8+r	8	0+x	–	LDR
8+r	9	0+x	–	CMR
8+r	A	0+x	–	ADD
8+r	A	8+x	–	SRM
8+r	B	0+x	–	MUL
8+r	B	8+x	–	LAB
8+r	C	8+x	–	LDB
8+r	D	8+x	–	CMB
8+r	E	0+x	–	SWR
8+r	E	8+x	–	SWB
8+r	F	0+x	–	STR
8+r	F	8+x	–	STB
Scientific				
C	8	8+x	–	SCZD
C+r	8	8+x	–	SCM
8+r	8	8+x	–	SLD
C	9	8+x	–	SNGD
C+r	9	8+x	–	SSB
8+r	9	8+x	–	SAD
C	C	0+x	–	SCZQ
C+r	C	0+x	–	SDV
8+r	C	0+x	–	SML
C	D	0+x	–	SNGQ
C+r	D	0+x	–	SSW
8+r	D	0+x	–	SST
Scientific Intrinsics				
0+r	0	8+x	–	SART
4+r	0	0+x	–	SCOS
0+r	1	8+x	–	SDTH[1]
0+r	1	0+x	–	SEXP[1]
4+r	1	8+x	–	SHTD[1]
4+r	1	0+x	–	SLOG
4+r	0	8+x	–	SQRT
0+r	0	0+x	–	SSIN[1]

[1] Refer to *Assembly Language Reference Manual*, CB07.

H1	H2	H3	H4	Mnemonic
Commercial				
0	0	2	0	– VRFY
0	0	2	1	– ALR
0	0	2	2	– ACM
0	0	2	3	– MAT
0	0	2	4	– AME
0	0	2	5	– DMC
0	0	2	6	– DME
0	0	2	7	– CBD
0	0	2	8	– SRCH
0	0	2	9	– DML
0	0	2	A	– CDB
0	0	2	B	– DDV
0	0	2	C	– DAD
0	0	2	D	– DSB
0	0	2	E	– DSH[2]
0	0	2	F	– DCM

where:

r = Register number contained in bits 1 through 3 or 2 through 3
x = Value of bits 9 through 11

[2] For the DSH instruction, shift control direction is specified by the internal shift control word 2 (SCW2) bit 0. If SCW2 bit 0 = 0, shift is left; if = 1, shift is right. For the instruction format in which direction is specified, refer to the *GCOS 6 Assembly Language Reference Manual*, Order No. CB07. Two additional software instructions are available, DLS (Decimal Left Shift) and DRS (Decimal Right Shift), and are also discussed in the same manual. When these instructions are used, the internal code of the DSH instruction (002E) is generated along with the appropriate value in the SCW2.

SECTION 7

EXTENDED INTEGER INSTRUCTIONS[1] (NUMERICAL)

H1	H2	H3	H4	Mnemonic
Branch on Registers (BK)				
8+k	0	7	C	KBLZ
8+k	0	F	C	KBGEZ
8+k	1	7	C	KBEZ
8+k	1	F	C	KBNEZ
8+k	2	7	C	KBGZ
8+k	2	F	C	KBLEZ
8+k	3	7	C	KBEVN
8+k	3	F	C	KBODD
Short Value Immediate (SI)				
8+k	0	E	C	LDVK
8+k	1	E	C	CMVK
8+k	2	E	C	ADVK
8+k	3	E	C	MLVK

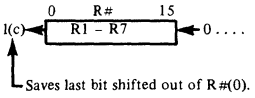
H1	H2	H3	H4	Mnemonic
Single Operand (SO)				
8	C	0+y	–	INCK
8	D	0+y	–	DECK
8	E	0+y	–	NEGK
8	F	0+y	–	CPLK
Double Operand (DO)				
8+r	8	0+y	–	LXA
8+k	8	8+y	–	STK
8+k	9	0+y	–	ANDK
8+k	9	8+y	–	SWK
8+k	A	0+y	–	XORK
8+k	A	8+y	–	SUBK
8+k	B	0+y	–	ORK
8+k	B	8+y	–	DIVK
8+k	C	8+y	–	LDK
8+k	D	8+y	–	CMK
8+k	E	8+y	–	ADDK
8+k	F	8+y	–	MULK

[1] Available only on DPS 6 Models 92 and 96.

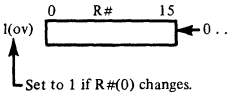
SECTION 8

SHIFT INSTRUCTIONS

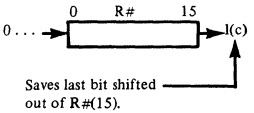
SOL
r00#



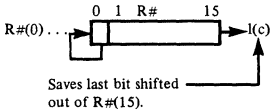
SAL
r02#



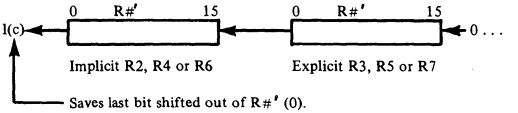
SOR
r04#



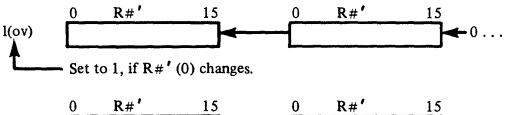
SAR
r06#



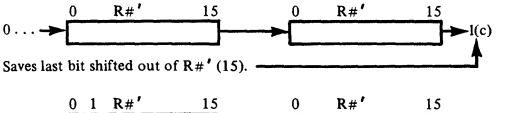
DOL
r08#



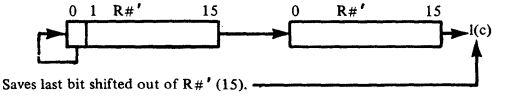
DAL
r0A#



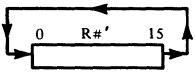
DOR
r0C#



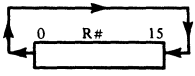
DAR
r0E#



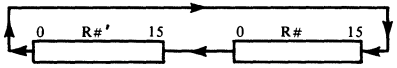
SCR
r01#



SCL
r05#

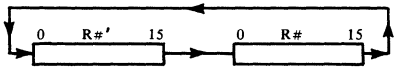


DCL
r03#



Implicit R2, R4, or R6 Explicit R3, R5 or R7

DCR
r07#

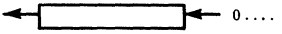


SHIFT INSTRUCTIONS

l(c)	c bit of 1 reg. contains most recent bit discarded
l(ov)	ov bit of 1 reg. set if sign changed, else cleared
r	R1 through R7, or, R3, R5 or R7
#	shift distance 1-15 or 1-31; 0 is special case

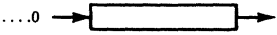
Commercial Shift Instructions

(DSH)
DLS

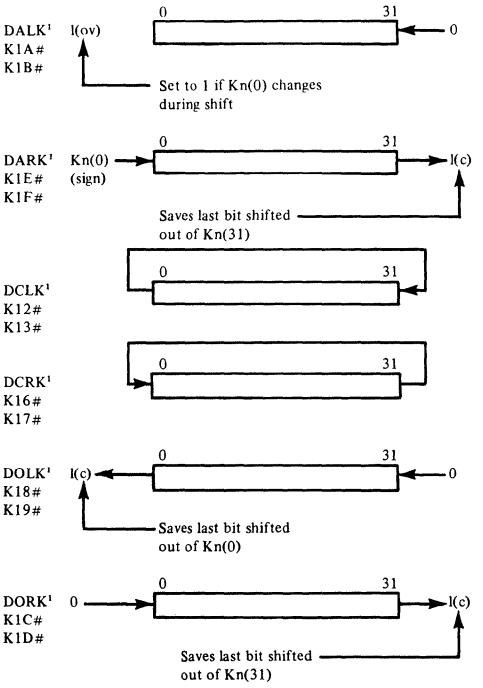


Set ov if a non-zero character is shifted out.

(DSH)
DRS



Round least significant digit if rounding is specified



\# – Distance shifted 1-15; if D = 0 distance in R1 bits is 11 to 15.

[1] Available only on DPS 6 Models 92 and 96.

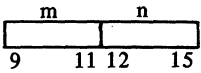
SECTION 9

ADDRESS SYLLABLE

ADDRESS SYLLABLE DEFINITIONS

D	16-bit signed displacement in words (-32,768) through +32,767) that follows the word containing the address syllable – for P+D and C(P+D), D is added to the address of the word containing D
Δ	32-bit signed displacement in words
@	Indirect operator
+r	Specifies indexing
+k	Specifies indexing
FB	FT + L
FT	Address of the top element of the current active frame in the stack
IMA	Immediate address
IMO	Immediate operand
IV	Interrupt vector
IA	Intermediate address
B	Base Register
Bk	B register number 1, 2, or 3
K	Double word operand register
Bn	B register number n
R	Word operand register
P	Program Counter which points to the word containing D
( )	Logical binding
[ ]	Contents of
+	Addition operation
-	Subtraction operation
0	Specifies an offset in bits
↑	Increment symbol; increment occurs after the effective address is obtained but before execution of the instruction
↓	Decrement symbol; decrement occurs before the effective address is obtained
EII1	Identifies an extended integer instruction and specifies that AS map 1 is to be used
EII23	Identifies an extended integer instruction, and specifies that AS map 3 is to be used
Rn	R register number n

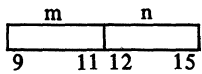
COMMERCIAL ADDRESS SYLLABLE



m – Address modifier
n – Register number (when 1 through 7)

n / m	0	1 – 7	8	9 – F
0	Remote Descriptor	Bn+D $Bn . value	P+D label . value	@ [B(n-8) +D] *$Bn .value
1		Bn+D+R1 $Bn . value . $R1	P+D+R1 label . value . $R1	@ [B(n-8) +D]+R1 *$Bn . value . $R1
2		Bn+D+R2 $Bn . value . $R2	P+D+R2 label . value . $R2	@ [B(n-8) +D]+R2 *Bn . value . $R2
3		Bn+D+R3 $Bn . value . $R3	P+D+R3 label . value . $R3	@ [B(n-8)+D] +R3 *Bn . value. $R3
4		Bn+D+R4. $Bn . value. $R4	@ [P+D] *label . value	@ [B(n-8)+D] +R4 *Bn . value . $R4
5		Bn+D+R5 $Bn . value . $R5	RFU	@ [B(n-8)+D] +R5 *Bn . value . $R5
6		Bn+D+R6 $Bn . value . $R6	P + Δ	@ [B(n-8)+D] +R6 *Bn . value . $R6
7		Bn+D+R7 $Bn . value . $R7	IMO	@ [B(n-8)+D] +R7 *Bn . value . $R7

NON-COMMERCIAL INSTRUCTIONS



m – Address modifier
n – Register number (when 1 through 7)

In the table below, the underlined items indicate the nature of the address obtained by the corresponding values of the address syllable. Indented below each item is the format of the related source language.

n / m	0	1 – 7	8	9 – B	C	D – F
0	IMA <label	Bn $Bn	@IMA @ <label	@B(n-8) @$Bn		
1	IMA+R1 <label . $R1	Bn+R1 $Bn . $R1	@IMA+R1 @ <label . $R1	@B(n-8) + R1 @ $Bn . $R1		
2	IMA+R2 <label . $R2	Bn+R2 $Bn . $R2	@IMA+R2 @ <label . $R2	@B(n-8) + R2 @ $Bn . $R2		
3	IMA+R3 <label . $R3	Bn+R3 $Bn . $R3	@IMA+R3 @ <label . $R3	@B(n-8)+R3 @$Bn . $R3		
4	P+D label	Bn+D $Bn . value	@ [P+D] @ label	@ [B(n-8) +D] $Bn . value		
5	RFU (TV05)	REG = $Bn or = $Rn	RFU (TV05)	B(n-8) + R1 $BK – $R1	RFU	B(n-C) + R1 $BK. + $R1
6	(↓) FT (↑) +$FT -	(↓) Bn -$Bn	FT+D $FT . value	B(n-8) + R2 $BK. – $R2	EII1	B(n-C) + R2 $BK . + $R2
7	IMO = value or = label	Bn(↑) +$Bn	IV+D $IV . value	B(n-8) + R3 $BK. – $R3	EII23	B(n-C) + R3 $BK . + $R3

n / m	0	1 – 3 4 – 7	8	9 – F
0	Remote Descriptor	FT+D+Rn $FT . value . $Rn →	@FT (↑) @FT	RFU (TV05)
1		Bn+D+K1 $Bn . value . $K1 →	RFU (TV05)	
2		Bn+D+K2 $Bn . value . $K2 →		
3		Bn+D+K3 $Bn . value . $K3 →		
4		@ [FT+D] +Rn $FT . value . $Rn	@ [FT+D] @ $FT. value →	
5		= Kn = $Kn	RFU (TV05)	
6		RFU (TV05)		
7				

Note:

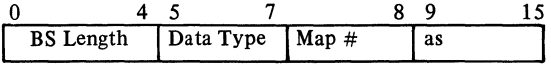
Available only on DPS 6 Models 92 and 96. The optional data-type expression (see Section 10), as an adjunct to the address expression, is for EII usage.

SECTION 10

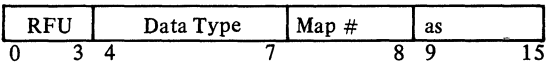
DATA DESCRIPTORS

EXTENDED INTEGER DESCRIPTORS[1]

For all EIIs except the LXA instruction, the following applies:



or



MAP # – specifies the Address Syllable (as) map as a function of the as in Word 1 (i.e., EII1 or EII23).

as in WORD 1	MAP #	as MAP #
EII1	0	RFU (TV05)
EII1	1	1
EII23	0	RFU
EII23	1	3

BS LENGTH – specifies the length of the bit string when data type is Bit String. If data is Bit String and BS LENGTH equals zero, the contents of $R1 (bits 11-15) are used.

DATA TYPE – specifies the atom size and value of the operand as defined in Table 10-1.

Note:

Refer to *Assembly Language Reference Manual,* CB07.

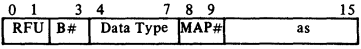
[1] Available only on DPS 6 Models 92 and 96.

TABLE 10-1. DATA TYPE FORMATS

Data Type Bits	Memory Operand	IMO	=Kn	Reg
X 0 0 0	U Bit String	Invalid (TV16)	Invalid (TV16)	Invalid (TV16)
X 0 0 1	S Bit String	"	"	"
0 0 1 0	Invalid (TV16)	"	"	"
0 0 1 1	"	"	"	"
0 1 0 0	U Half Word	"	"	"
0 1 0 1	S Half Word	"	"	"
0 1 1 0	U Word	U Word	"	U Word
0 1 1 1	S Word	S Word	"	S Word
1 0 1 0	Invalid (TV16)	Invalid	"	"
1 0 1 1	S Double Word	S Double Word	S Double Word	S Double Word
1 1 0 0	Invalid (TV16)	Invalid (TV16)	Invalid (TV16)	Invalid (TV16)
1 1 0 1	"	"	"	"
1 1 1 0	Address	Address Reg	Address Reg	Address Reg
1 1 1 1	Invalid (TV16)	Invalid (TV16)	Invalid (TV16)	Invalid (TV16)

- o X = least significant bit of BS length field
- o U = unsigned
- o S = signed

For the LXA instruction the following applies:



MAP – specifies the as map as a function of the as in the instruction word. (See the definition of this bit in the previous Data description.)

B# – specifies in which B-register the word portion of the address is to be stored.

Bits 1 through 3 of the op code word define in which R-register the index portion of the address is to be stored, right-justified.

DATA TYPE – specifies the size of the data being pointed to.

Data Type Bits	Size
4 5 6 7	
X 0 0 X	Bit
0 0 1 0	Digit (4 bits)
0 1 0 X	Byte
0 1 1 X	Word
1 0 1 1	Double-Word
1 1 0 1	Quad-Word
1 1 1 0	Address

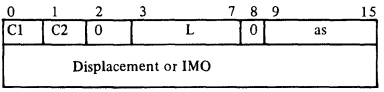
Note:

Refer to *Assembly Language Reference Manual*, CB07.

For bit, digit, and byte sizes, the atom index is stored in $Rn. For all the other atom sizes, $Rn is cleared.

COMMERCIAL DESCRIPTORS[1]

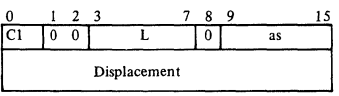
Alphanumeric Format



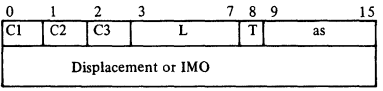
Word 1

Word 2

Binary Format



Decimal Format



Word 1:

- C1, C2, C3 – Control bits which specify byte or half-byte offset and sign information
- L – Specifies the length of the operand
- T – Specifies the data type:
  - o T = 0, data is string decimal;
  - o T = 1, data is packed decimal.
- as – Specifies the address syllable

Word 2:

The contents of Word 2 is either a displacement or an immediate operand as defined by the as.

Note:

Refer to *Assembly Language Reference Manual*, CB07.

[1] Available only on DPS 6 Models 92 and 96.

SECTION 11

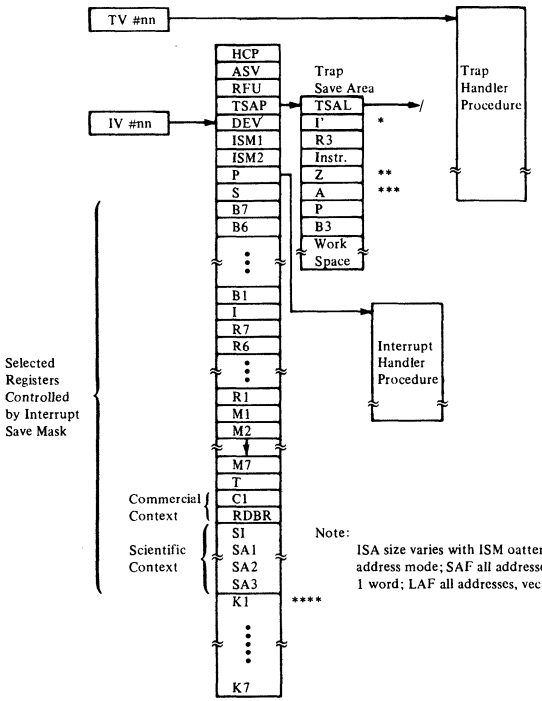
HARDWARE-DEDICATED MEMORY

SAF Memory Location	Hardware Nomenclature	LAF Memory Location**		Contents
0000	RHU/RSU	0000	–	Entry to Power Failure Restart Routine
000A	NATSAP3			Next Available
000C	NATSAP2			TSA pointers
000E	NATSAP1			
0010	NATSAP0	0010		
	RHU			
0014	RTCI	0014	–	RTC Initial Value
0015	RTCC	0015	–	RTC Current Value
0016	RTCL	0016	–	RTC Level
0017	WDTC	0017	–	WDT Current Value
001A	RHU	001A	–	Display Pointer***
001F	MERC	001F	–	Memory Error Count
0020	0 15	0020		
0021	16 31	0021		
0022	32 IAF 47	0022	–	Interrupt Level Activity Flags
0023	48 63	0023		
	RHU			Associated Event
0052	TV #46	0024	–	RFU
	TV #33	003E	–	SIP ILL Argument ***
	TV #32	0040	–	SIP/CIP Protection Violation***
0061	TV #31	0042	–	SIP QLT FAULT
0062	TV #30	0044	–	CIP QLT FAULT
0063	TV #29	0046	–	Commercial Overflow*
0064	TV #28	0048	–	Commercial Truncation*
0065	TV #27	004A	–	Commercial Illegal Character
0066	TV #26	004C	–	Commercial Illegal Specification
0067	TV #25	004E	–	Commercial Divide by Zero
0068	TV #24	0050	–	Memory or Megabus Error seen by SIP or CIP
0069	TV #23	0052	–	Unavailable Resource referenced by SIP or CIP
006A	TV #22	0054	–	Scientific Precision Error*
006B	TV #21	0056	–	Scientific Significance Error*
006C	TV #20	0058	–	Scientific Program Error
006D	TV #19	005A	–	Scientific Exponent Underflow*
006E	TV #18	005C	–	RFU
006F	TV #17	005E	–	Uncorrectable Memory or Megabus Error
0070	TV #16	0060	–	Program Error
0071	TV #15	0062	–	Unavailable Resource
0072	TV #14	0064	–	Protection Violation
0073	TV #13	0066	–	Privilege Violation
0074	TV #12	0068	–	Remote, remote data descriptor
0075	TV #11	006A	–	RFU
0076	TV #10	006C	–	Stack Overflow
0077	TV #9	006E	–	Stack Underflow
0078	TV #8	0070	–	Scientific Exponent Overflow
0079	TV #7	0072	–	Scientific Divide by Zero
007A	TV #6	0074	–	Integer Register Overflow*
007B	TV #5	0076	–	Unimplemented Instruction
007C	TV #4	0078	–	RSU
007D	TV #3	007A	–	Uninstalled Scientific Instruction
007E	TV #2	007C	–	Trace*/Breakpoint Trap
007F	TV #1	007E	–	Monitor Call Trap
0080	IV #0	0080	–	Interrupt Vector, Level 0
0081	IV #1	0082	–	Interrupt Vector, Level 1
	⋮			
00BE	IV #62	00FC	–	Interrupt Vector, Level 62
00BF	IV #63	00FE	–	Interrupt Vector, Level 63
00C0-00FF	RHU	NONE		

*Maskable Trap Conditions
**All LAF addresses and vectors are contained in two memory words.
***Available only on DPS 6 Models 92 and 96.

SECTION 12

TRAP VECTOR AND INTERRUPT VECTOR LINKAGE



Note:
ISA size varies with ISM oatterns and with
address mode; SAF all addresses, vectors a
1 word; LAF all addresses, vectors are 2 w

*I' Format

TRAP#	1

Trap# = 40_{16} – TV#
I = Copy of I Register

**Z Format

0	1 3	4 7	8 9	10 11	12 15
REG	000	BI	R	NTO	IS

REG – 'A' word validity
1 = A word is invalid
0 = A word is valid

BI – bit/byte index field
For bit instruction, BI = 4 low-order bits of index register
For byte instruction, BI = X000 where X is the low-order bit of the index register

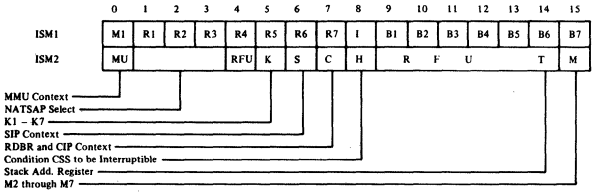
R – Ring Number

IS – Instruction Size

***A Format – address associated with Trap, see *Assembly Language Reference Manual*, CB07.

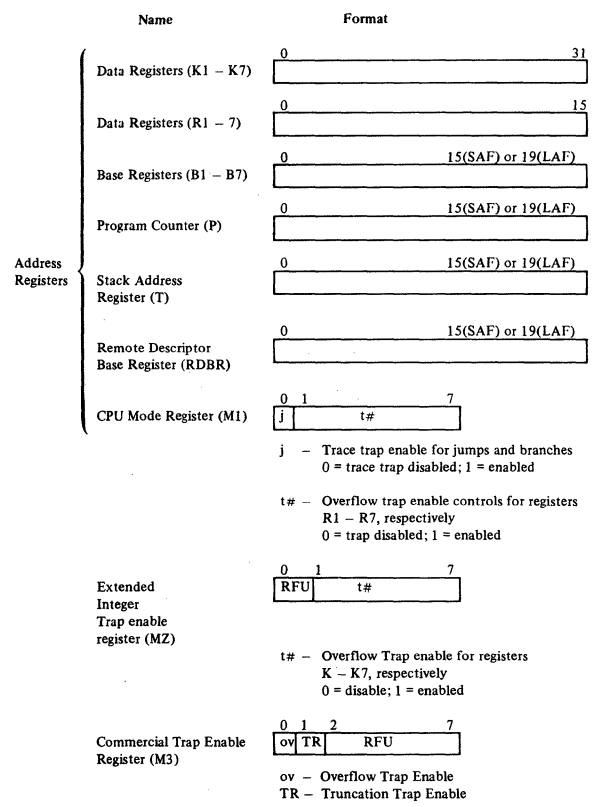
****Available only on DPS 6 Models 92 and 96.

The format of ISM is as follows:

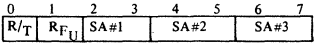


SECTION 13

REGISTERS



SIP Operating Mode Register (M4)

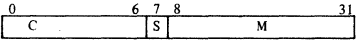


R/T – Round/Truncate Mode
0 = Truncate; 1 = Round

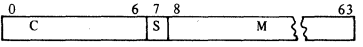
SA# – Scientific accumulator number

	Memory data length (words)	Accumulator data length (words)
00	2	2
01	2	4
10	4	2
11	4	4

Floating-Point Data Short precision (32-bit)



Long precision (64-bit)

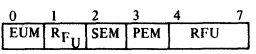


C – Represents the characteristic (excess 64 power-of-16 exponent) of the number. The characteristic represents exponents with a range from -64 to +63. Since the characteristics have no sign bit, the number 64 (decimal) is effectively added to each exponent, thus allowing a characteristic range of 0 to 127 to represent exponents with a range of -64 to +63.

S – Sign bit (0 = +; 1 = -) of the mantissa.

M – Mantissa – a normalized unsigned hexadecimal fraction (i.e., $1 > M \geq 1/16$).

SIP Trap Enable Register (M5)

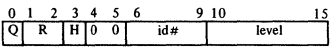


EUM – Exponent Underflow Trap Enable;
0 = disabled, 1 = enabled

SEM – Significance Error Trap Enable;
0 = disabled, 1 = enabled

PEM – Precision Error Trap Enable;
0 = disabled, 1 = enabled

System Status/Security Register (S)



R – Ring Number (privilege state)
IX = privilege rings; OX = user rings

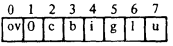
id# – Processor identity (channel number)

level – Interrupt priority level – 0 (high) through 63 (low)

Q – QLT Fault

H – Super halt

CPU Indicator Register (I)



ov – "Overflow" indicator

c – "Carry" indicator

b – "Bit test" indicator

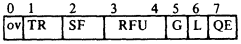
i – "Input/output" indicator

g – "Greater than" indicator

l – "Less than" indicator

u – "Unlike signs" indicator

Commercial Indicator Register (CI)



ov – Overflow occurred during decimal instruction

TR – Alphanumeric result is truncated

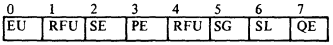
SF – Sign fault (negative operand is stored in unsigned field)

G – Greater than

L – Less than

QE – QLT error

SIP Indicator Register (SI)



EU – Exponent underflow

SE – Significance error

PE – Precision error

SG – Greater than

SL – Less than

QE – QLT error

SECTION 14

ASCII – HEXADECIMAL HEXADECIMAL – DECIMAL CONVERSION TABLES

ASCII – HEXADECIMAL

ASCII character = 2 hexadecimal digits (8 bits): H1 H2

H2 \ H1	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

HEXADECIMAL – DECIMAL

Word							
Byte				Byte			
H1	Decimal	H2	Decimal	H3	Decimal	H4	Decimal
0	0	0	0	0	0	0	0
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61440	F	3840	F	240	F	15

Some useful PUP verbs:

MEMORY	P+CTLY	14C
DISK-STAT	P	02A8
LINE-STAT	P	12A8
PRIORITY	P	8099
!MDUMP	P	17C

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 155 Gordon Baker Road, Willowdale, Ontario M2H 3N7
In the U.K.: Great West Road, Brentford, Middlesex TW8 9DH
In Australia: 124 Walker Street, North Sydney, N.S.W. 2060
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

34779, 1.5682, Printed in U.S.A. CU75-00