29

CODING TABLE: 9000 A INSTRUCTION SET

6-18-7ϕ

* IMPLIES A OR B

D/I A/B Z/C H/S H/C ALL CODED AS 0/1

*± 7_8

höst [illegible]

GROUP	OCTAL	INSTR	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMORY REFERENCE GROUP 1 (10)	-0-----	AD*	D/I	0	0	0	A/B	Z/C	← MEMORY ADDRESS → (1 7 7 7)									
	-1-----	CP*	D/I	0	0	1	A/B	Z/C										
	-2-----	LD*	D/I	0	1	0	A/B	Z/C										
	-3-----	ST*	D/I	0	1	1	A/B	Z/C										
	-4-----	IOR	D/I	1	0	0	0	Z/C										
	-4-----	ISZ	D/I	1	0	0	1	Z/C										
	-5-----	AND	D/I	1	0	1	0	Z/C										
	-5-----	DSZ	D/I	1	0	1	1	Z/C										
	-6----	JSM	D/I	1	1	0	0	Z/C										
×1	-6----	JMP	D/I	1	1	0	1	Z/C										
SHIFT-ROTATE GROUP 4 (8) ×2	~ vacated bits = bit 15 07---0	A*R	0	1	1	1	A/B	—	—	← SHIFT CODE n-1 → x	x	x	x	—	0	0	0	0
	~ vacated bits = 0 07---2	S*R	0	1	1	1	A/B	—	—					—	0	0	1	0
	07---4	S*L	0	1	1	1	A/B	—	—					—	0	1	0	0
	07---6	R*R	0	1	1	1	A/B	—	—					—	0	1	1	0
			M												X			
ALTER-SKIP GROUP 3 (9) ×2	07---0	SZ*	0	1	1	1	A/B	0	← SKIP CODE → x	x	x	x	x	0	1	0	0	0
	07---0	RZ*	0	1	1	1	A/B	1						0	1	0	0	0
	07---0	SI*	0	1	1	1	A/B	0						1	1	0	0	0
	07---0	RI*	0	1	1	1	A/B	1						1	1	0	0	0
	07---1	SL*	0	1	1	1	A/B	H/S						H/C	1	0	0	1
	07---2	S*M	0	1	1	1	A/B	H/S						H/C	1	0	1	0
	07---3	S*P	0	1	1	1	A/B	H/S	1	0	1	0	0	H/C	1	0	1	1
	07---4	SES	0	1	1	1	A/B	H/S						H/C	1	1	0	0
	07---5	SEC	0	1	1	1	A/B	H/S						H/C	1	1	0	1
			M												X			

GROUP	OCTAL	INSTR	M	(2)	1	0	15	14	13	12	X
REGISTER REFERENCE GROUP 2 (14) ×1	07--17	ADA	0 1 1 1	A/B	—	—	D/I	0	0	0	0 1 1 1 1
	07--37	ADB	0 1 1 1	A/B	—	—	D/I	0	0	0	1 1 1 1 1
	07--57	CPA	0 1 1 1	A/B	—	—	D/I	0	0	1	0 1 1 1 1
	07--77	CPB	0 1 1 1	A/B	—	—	D/I	0	0	1	1 1 1 1 1
	07--17	LDA	0 1 1 1	A/B	—	—	D/I	0	1	0	0 1 1 1 1
	07--37	LDB	0 1 1 1	A/B	—	—	D/I	0	1	0	1 1 1 1 1
	07-557	STA,I	0 1 1 1	A/B	—	—	1	0	1	1	0 1 1 1 1
	07-577	STB,I	0 1 1 1	A/B	—	—	1	0	1	1	1 1 1 1 1
	07--17	IOR	0 1 1 1	A/B	—	—	D/I	1	0	0	0 1 1 1 1
	07-637	ISZ I	0 1 1 1	A/B	—	—	1	1	0	0	1 1 1 1 1
	07--57	AND	0 1 1 1	A/B	—	—	D/I	1	0	1	0 1 1 1 1
	07-677	DSZ,I	0 1 1 1	A/B	—	—	1	1	0	1	1 1 1 1 1
	07-717	JSM,I	0 1 1 1	A.	—	—	1	1	1	0	0 1 1 1 1
	07--37	JMP	0 1 1 1	A/B	—	—	D/I	1	1	0	1 1 1 1 1

only use in indirect mode

SKIP - Sprung

geändert Sprung Gruppe

only digits (7

- complement for shift

* * * * * *

Instruction Description - 9000A

General Comments

These 71 instructions are all implemented serially by the micro-processor in a time period which varies according to specific instruction, to whether or not it is indirect, and to whether or not the skip condition has been met.

Upon completion of the execution of each instruction, the program counter (P register) has been incremented by one except for instructions JMP, JSM, and the skip instructions in which the skip condition has been met. The M-register is left with contents identical to the P-register. The contents of the addressed memory location and the A and B registers are left unchanged unless specified otherwise.

Memory Reference Group

The 14 memory reference instructions refer to a specific address in memory determined by the address field <m>, by the ZERO/CURRENT page bit, and by the DIRECT/INDIRECT bit. This description assumes familiarity with page addressing and indirect addressing, which are both described in detail in the -hp-2116 computer reference manuals.

The address field <m> is a 10 bit field consisting of bits 0 through 9. The ZERO/CURRENT page bit is bit 10 and the DIRECT/INDIRECT bit is bit 15, except for reference to the A or B register in which case bit 8 becomes the DIRECT/INDIRECT bit. An indirect reference is denoted by a <,I> following the address <m>.

REGISTER REFERENCE OF A OR B REGISTER: If the location <A> or <B> is used in place of <m> for any memory reference instruction, the instruction will treat the contents of A or B exactly as it would the contents of location <m>. See footnote on special restriction for direct register reference of A or B.

ADA m,I Add to A. The contents of the addressed memory location m are added (binary add) to contents of the A register, and the sum remains in the A register. If carry occurs from bit 15, the E register is loaded with 0001, otherwise E is left unchanged.

ADB m,I Add to B. Otherwise identical to ADA.

Memory Reference Group (continued)

CPA m,I Compare to A and skip if unequal. The contents of the addressed memory location are compared with the contents of the A register. If the two 16-bit words are different, the next instruction is skipped; that is, the P and M registers are advanced by two instead of one. Otherwise, the next instruction will be executed in normal sequence.

CPB m,I Compare to B and skip is unequal. Otherwise identical to CPA.

LDA m,I Load into A. The A register is loaded with the contents of the addressed memory location.

LDB m,I Load into B. The B register is loaded with the contents of the addressed memory location.

STA m,I Store A. The contents of the A register are stored into the addressed memory location. The previous contents of the addressed memory location are lost.

STB m,I Store B. Otherwise identical to STA.

IOR m,I "Inclusive OR" to A. The contents of the addressed location are combined with the contents of the A register as an "INCLUSIVE OR" logic operation. only with A register

ISZ m,I Increment and Skip if Zero. The ISZ instruction adds ONE to the contents of the addressed memory location. If the result of this operation is ZERO, the next instruction is skipped; that is, the P and M registers are advanced by TWO instead of ONE. The incremental value is written back into the addressed memory location. Use of ISZ with the A or B register is limited to indirect reference; see footnote on restrictions.

AND m,I Logical "AND" to A. The contents of the addressed location are combined with the contents of the A register as an "AND" logic operation. only with A register

DSZ m,I Decrement and Skip if Zero. The DSZ instruction subtracts ONE from the contents of the addressed memory location. If the result of this operation is zero, the next instruction is skipped. The decremented value is written back into the addressed memory location. Use of DSZ with the A or B register is limited to indirect reference; see footnote on restrictions.

Memory Reference Group (continued)

JSM m,I Jump to Subroutine. The JSM instruction permits jumping to a subroutine in either ROM or R/W memory. The contents of the P register is stored at the address contained in location 1777 (stack pointer). The contents of the stack pointer is incremented by one, and both M and P are loaded with the referenced memory location.

JMP m,I Jump. This instruction transfers control to the contents of the addressed location. That is, the referenced memory location is loaded into both M and P registers, effecting a jump to that location.

Shift-Rotate Group

The eight shift-rotate instructions all contain a 4 bit variable shift field <n> which permits a shift of one through 16 bits; that is, $1 \leq n \leq 16$. If <n> is omitted, the shift will be treated as a one bit shift. The shift code appearing in bits 8,7,6,5 is the binary code for n-1, except for SAL and SBL, in which cases the complementary code for n-1 is used.

AAR n Arithmetic right shift of A. The A register is shifted right n places with the sign bit (bit 15) filling all vacated bit positions. That is, the n+1 most significant bits become equal to the sign bit.

ABR n Arithmetic right shift of B. Otherwise identical to AAR.

SAR n Shift A right. The A register is shifted right n places with all vacated bit positions cleared. That is, the n most significant bits become equal to zero.

SBR n Shift B right. Otherwise identical to SAR.

SAL n Shift A left. The A register is shifted left n places with the n least significant bits equal to zero.

SBL n Shift B left. Otherwise identical to SAL.

RAR n Rotate A right. The A register is rotated right n places, with bit 0 rotated around to bit 15.

RBR n Rotate B right. Otherwise identical to RAR.

Alter-Skip Group

The sixteen alter-skip instructions all contain a 5-bit variable skip field <n> which, upon meeting the skip condition, permits a relative branch to any one of 32 locations. Bits 9,8,7,6,5 are coded for positive or negative relative branching in which the number <n> is the number to be added to the current address, (skip in forward direction), and the number <-n> is the number to be subtracted from the current address, (skip in negative direction). If <n> is omitted, it will be interpreted as a ONE.

<n>=0	CODE=00000	REPEAT SAME INSTRUCTION
<n>=1	CODE=00001	DO NEXT INSTRUCTION
<n>=2	CODE=00010	SKIP ONE INSTRUCTION
<n>=15	CODE=01111	ADD 15 TO ADDRESS
<n>=-1	CODE=11111	DO PREVIOUS INSTRUCTION
<n>=-16	CODE=10000	SUBTRACT 16 FROM ADDRESS
<n>=nothing	CODE=00001	DO NEXT INSTRUCTION

The alter bits consist of bits 10 and bits 4. The letter <s> following the instruction places a ONE in bit 10 which causes the tested bit to be set after the test. Similarly the letter <c> will place a ONE in bit 4 to clear the test bit. If both a set and clear bit are given, the set will take precedence. Alter bits do not apply to SZA, SZB, SIA, and SIB.

SZA	n	Skip if A zero. If all 16 bits of the A register are zero, skip to location defined by n.
SZB	n	Skip if B zero. Otherwise identical to SZA.
RZA	n	Skip if A not zero. This is a "Reverse Sense" skip of S
RZB	n	Skip if B not zero. Otherwise identical to RZA.
SIA	n	Skip if A zero; then increment A. The A register is tested for zero, then incremented by one. If all 16 bits of A were zero before incrementing, skip to location defined by n.
SIB	n	Skip if B zero; then increment B. Otherwise identical to SIA.
RIA	n	Skip if A not zero; then increment A. This is a "Reverse Sense" skip of SIA.
RIB	n	Skip if B not zero; then increment B. Otherwise identical to RIA.
SLA	n,S/C	Skip if Least Significant bit of A is zero. If the least significant bit (bit 0) of the A register is zero, skip to location defined by n. If either S or C is present, the test bit is altered accordingly after test.

T 29

Alter-Skip Group (continued)

SLB	n, S/C Skip if Least Significant bit of B is zero. Otherwise identical to SLA.
SAM	n, S/C Skip if A is Minus. If the sign bit (bit 15) of the A register is a ONE, skip to location defined by n. If either S or C is present, bit 15 is altered after the test.
SBM	n, S/C Skip if B is Minus. Otherwise identical to SAM.
SAP	n, S/C Skip if A is Positive. If the sign bit (bit 15) of the A register is a ZERO, skip to location defined by n. If either S or C is present, bit 15 is altered after the test.
SBP	n, S/C Skip if B is Positive. Otherwise identical to SAP.
SES	n, S/C Skip if Least Significant bit of E is Set. If bit 0 of the E register is a ONE, skip to location defined by n. If either S or C is present, the entire E register is set or cleared respectively.
SEC	n, S/C Skip if Least Significant bit of E is Clear. If bit 0 of the E register is a ZERO, skip to location defined by n. If either S or C is present, the entire E register is set or cleared respectively.

Complement-Execute-DMA Group.

These seven instructions include complement operations and several special-purpose instructions chosen to speed up printing and extended memory operations.

CMA	Complement A. The A register is replaced by its One's complement.
CMB	Complement B. The B register is replaced by its One's complement.
TCA	Two's Complement A. The A register is replaced by its One's Complement and incremented by one.
TCB	Two's complement B. The B register is replaced by its One's Complement and incremented by one.
EXA	Execute A. The contents of the A register are treated as the current instruction, and executed in the normal manner. The A register is left unchanged unless the instruction code causes A to be altered.
EXB	Execute B. Otherwise identical to EXA.
DMA	Direct Memory Access. The DMA control in Extended Memory is enabled by setting the indirect bit in M and giving a WTM instruction. The next ROM clock transfers A→M and the following two cycles transfer B→M. ROM clock then remains inhibited until released by DMA control.

MAC Instruction Group.

A total of 16 MAC instructions are available for operation

(a) with the whole floating-point data (like transfer, shifts, etc), or

(b) with two floating-point data words to speed up digit and word loops in arithmetic routines.

NOTE: $<A_{0-3}>$ means: contents of A-register bit 0 to 3

AR 1	is a mnemonix for arithmetic pseudo-register located in R/W memory on addresses 1744 to 1747 (octal)
AR 2	is a mnemonix for arithmetic pseudo-register located in R/W memory on addresses 1754 to 1757 (octal)
D_i	means: mantissas i-th decimal digit; most significant digit is D1 least significant digit is D12 decimal point is located between D1 and D2

Every operation with mantissa means BCD-coded decimal operation.

RET Return
16-bit-number stored at highest occupied address in stack is transferred to P- and M-registers. Stack pointer (=next free address in stack) is decremented by one.
<A>, <B>, <E> unchanged.

MOV Move overflow
The contents of E-register is transferred to A_{0-3}. Rest of A-register and E-register are filled by zeros.
<B> unchanged.

CLR Clear a floating-point data register in R/W memory on location <A>
ZERO→<A>, <A>+1, <A>+2, <A>+3
<A>, >B>, <E> unchanged

XFR Floating-point data transfer in R/W memory from location <A> to location <B>.
Routine starts with exponent word transfer.
Data on location <A> is unchanged.
<E> unchanged.

MRX AR1 mantissa is shifted to right n-times. Exponent word remains unchanged.

$<B_{0-3}> = n$ (binary coded)

1st shift: $<A_{0-3}> \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; D_{12} is lost

jth shift: $\emptyset \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; D_{12} is lost

nth shift: $\emptyset \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; $D_{12} \rightarrow A_{0-3}$

$\emptyset \rightarrow E$, A_{4-15}

each shift: $<B_{0-3}> - 1 \rightarrow B_{0-3}$

$<B_{4-15}>$ unchanged

MRY AR2 mantissa is shifted to right n-times. Otherwise identical to MRX

MLS AR2 mantissa is shifted to left once. Exponent word remains unchanged.

$\emptyset \rightarrow D_{12}$; $D_i \rightarrow D_{i-1}$; $D_1 \rightarrow A_{0-3}$ $\emptyset \rightarrow E$, A_{4-15}

$<B>$ unchanged

DRS AR1 mantissa is shifted to right once Exponent word remains unchanged

$\emptyset \rightarrow D_1$; $D_i \rightarrow D_{i+1}$; $D_{12} \rightarrow A_{0-3}$

ZERO $\rightarrow$ E and A_{4-15}

$<B>$ unchanged

DLS AR1 mantissa is shifted to left once. Exponent word remains unchanged.

$<A_{0-3}> \rightarrow D_{12}$; $D_i \rightarrow D_{i-1}$; $D_1 \rightarrow A_{0-3}$

$\emptyset \rightarrow E$, A_{4-15}

$<B>$ unchanged

FXA Fixed-point addition
Mantissas in pseudo-registers AR2 and AR1 are added together and result is placed into AR2. Both exponent words remain unchanged. When overflow occurs "0001" is set into E-reg., in opposite case $<E>$ will be zero.

$<AR2> + <AR1> + DC \rightarrow AR2$

DC = $\emptyset$ if $<E>$ was 0000 before routine execution

DC = 1 if $<E>$ was 1111 before routine execution

$<B>$, $<AR1>$ unchanged

FMP Fast multiply
Mantissas in pseudo-registers AR2 and AR1 are added together $\langle B_{0-3}\rangle$-times and result is placed into AR2. Total decimal overflow is placed to A_{0-3}. Both exponent words remain unchanged.

$$\langle AR2\rangle + \langle AR1\rangle * \langle B_{0-3}\rangle + DC \rightarrow AR2$$

DC = 0 if <E> was 0000 before routine execution
DC = 1 if <E> was 1111 before routine execution
ZERO → E, A_{4-15}

<AR1> unchanged

FDV Fast divide
Mantissas in pseudo-registers AR2 and AR1 are added together so many times until first decimal overflow occurs. Result is placed into AR2. Both exponent words remain unchanged. Each addition without overflow causes +1 increment of <B>.

1st addition: <AR2> + <AR1> + DC → AR2
DC = 0 if <E> was 0000 before routine execution
DC = 1 if <E> was 1111 before routine execution
next additions: <AR2> + <AR1> → AR2
ZERO → E
<AR1> unchanged

CMX 10's complement of AR1 mantissa is placed back to AR1, and ZERO is set into E-register. Exponent word remains unchanged
<B> unchanged

CMY 10's complement of AR2 mantissa.
Otherwise identical to CMY

MDI Mantissa decimal increment.
Mantissa on location <A> is incremented by decimal ONE on D_{12} level, result is placed back into the same location, and zero is set into E-reg.
Exponent word is unchanged.
When overflow occurs, result mantissa will be
1,000 0000 0000 (dec)
and 0001 (bin) will be set into E-reg.
<B> unchanged.

NRM Normalization
Mantissa in pseudo-register AR2 is rotated to the left to get $D_1 \neq 0$. Number of these 4-bit left shifts is stored in B_{0-3} in binary form ($\langle B_{4-15}\rangle = 0$)

when $\langle B_{0-3}\rangle = 0, 1, 2, \ldots, 11$ (dec) $\Rightarrow \langle E\rangle = 0000$
When $\langle B_{0-3}\rangle = 12$ (dec) $\Rightarrow$ mantissa is zero, and $\langle E\rangle = 0001$

Exponent word remains unchanged
<A> unchanged.