MAC-8

HEXADECIMAL CODING CHART

PA-800517
Issue 1
December 1978

Prepared for the Microprocessor System Department
and the
Continuing Education Department
by
Technical Documentation Department,
Bell Laboratories, 1978

PA-800517
Issue 1
December 1978

ORGANIZATION

This hexadecimal (hex.) coding chart allows you to translate your program assembly language into a machine code that the MAC-8 microprocessor can store, manipulate, and process. The chart comprises the following parts:

Part 1 – Dyadic Instructions

A dyadic instruction is defined as having two operands, one designated as the source and one designated as the destination. The result of the operation is stored in the destination. The dyadic instructions consist of the following operations.

- Move
- Arithmetical add and subtract
- Logical AND, OR, exclusive OR, compare, and test

All dyadic operations use both 8- and 16-bit operands, except test which uses 8-bit operands only.

Part 2 – Monadic Instructions

A monadic instruction is defined as having one operand that serves as the source and the destination, but is designated as the destination. The monadic instructions consist of the following operations.

- Zero
- Negate
- Increment
- Decrement
- Complement
- Arithmetical shift
- Logical shift
- Rotate 8
- Rotate 9

All monadic operations use 8-bit operands; the zero, increment, and decrement operations use both 8- and 16-bit operands.

Part 3 – Miscellaneous Dyadic and Monadic Instructions

These instructions consist of the following operations.

- Find, clear, count ones, and load address into 16-bit register
- Register pointer manipulation (load, bump, and debump)
- Stack pointer manipulation (load, logical add, arithmetical add)
- Condition register manipulation (set and clear)
- Memory stack save and restore (push and pop)

Part 4 – Transfer Instructions : Unconditional and Condition Register Bit Conditional

Transfer instructions alter the order of instruction execution. These instructions come in two forms: one form is unconditional and uses abbreviated addressing modes; the other form is conditional and is dependent upon a particular condition bit being set or cleared. The condition bits are neg, zero, ovfl, carry, ones, odd, enable, flag, lt, lteq, llteq, hom, shovfl, and always. The last six bits are logical combinations of the first eight bits. The unconditional and bit conditional instructions consist of the following operations.

- Branch (unconditional jump)
- Conditional jump (local or global)
- Unconditional call
- Conditional call
- Unconditional return
- Conditional return

Part 5 – Transfer Instructions : Register Bit Conditional and Miscellaneous

These instructions consist of the following operations.

- Conditional jump (local only)
- Halt
- No operation (nop)

Part 6 – Summary of Machine Codes

All MAC-8 hex. machine codes are listed in numerical order.

GLOSSARY

To understand the hex. coding chart, you have to become familiar with the designations and symbols and their meanings that are listed below

Memory Register Identification

Designation	Meaning
ad	**a** register used as the destination
as	**a** register used as the source
bd	**b** register used as the destination
bs	**b** register used as the source
dd	**b** register used as a pointer to 16-bit data
ds	**b** register used as a pointer to 16-bit data

MAC-8 Microprocessor Register Identification

Designation	Meaning
cr	condition register
pc	program counter
rp	register pointer
sp	stack pointer

Number Identification

Designation	Meaning
M	8-bit immediate data or address offset
N	8-bit immediate data or address offset
V	16-bit immediate data or address
W	16-bit immediate data or address

Register and Number Operation Symbols

In the following, x and y are used to denote the contents of registers, contents of memory addresses, or immediate data.

Symbols	Meaning
x = y	replace x with y
x Λ y	bit-by-bit exclusive OR of x and y
x | y	bit-by-bit inclusive OR of x and y
x & y	bit-by-bit AND of x and y
x -- y	x minus y
x + y	x plus y
*x	the contents of the memory address(es) pointed to by x; if x represents a 16-bit number, *x represents the contents of that 16-bit number used as an address
*x++	after operating on the contents of the memory address(es) pointed by x, increment x; if two successive memory addresses are referenced by the instruction, x is incremented by 2
*(x + N)	the contents of the memory address that is N addresses above the address pointed to by x
–x	the value of x negated (2s complement)
++x	the value of x incremented by 1
––x	the value of x decremented by 1
∿x	the value of x complemented (1s complement)
x*2	x arithmetically shifted left one bit (multiplied by 2)
x/2	x arithmetically shifted right one bit (divided by 2)
x<<1	x logically shifted left one bit
x>>1	x logically shifted right one bit
x<<<1	x rotated left one bit
x>>>1	x rotated right one bit
x$<<1	x rotated left through carry one bit
x>>$1	x rotated right through carry one bit
&x	the address of x
!x	nontrue condition (where x represents a register bit)

HOW TO USE THE HEX. CODING CHART

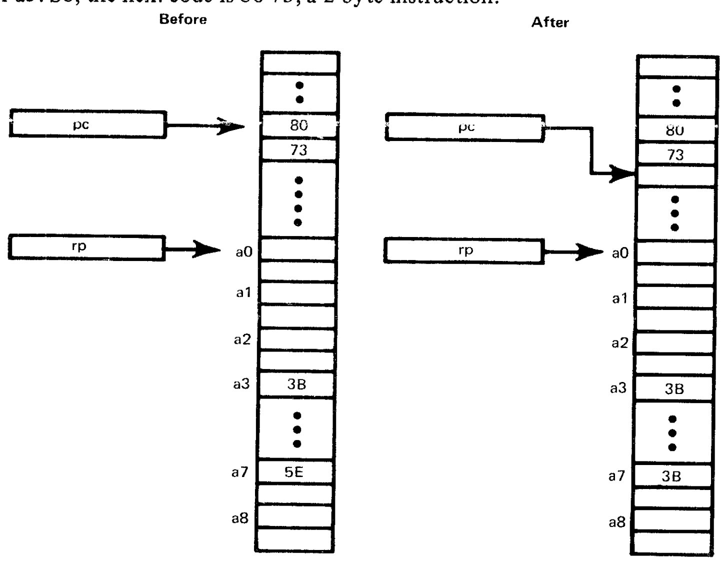
To translate your program into hex. code, you have to examine each assembly language instruction, determine its type (dyadic, monadic, etc.) and, within the type, what kind of operation is performed (move, add, etc.). For example, if you want to move the contents of register a3 to register a7, the assembly language instruction is

a7=a3

This is a dyadic instruction for a move between registers operation. If you look in Part 1 of the chart, under the Addressing Mode (the Register and Register entry) and Move columns, you will find the corresponding instruction

ad=as
80
ds

The operation code (opcode) is 80. It is followed by the **a** register that is used as the destination, which is register a7, and the **a** register that is used as the source, which is register a3. So, the hex. code is 80 73, a 2-byte instruction.



Memory Before and After Execution of 80 73

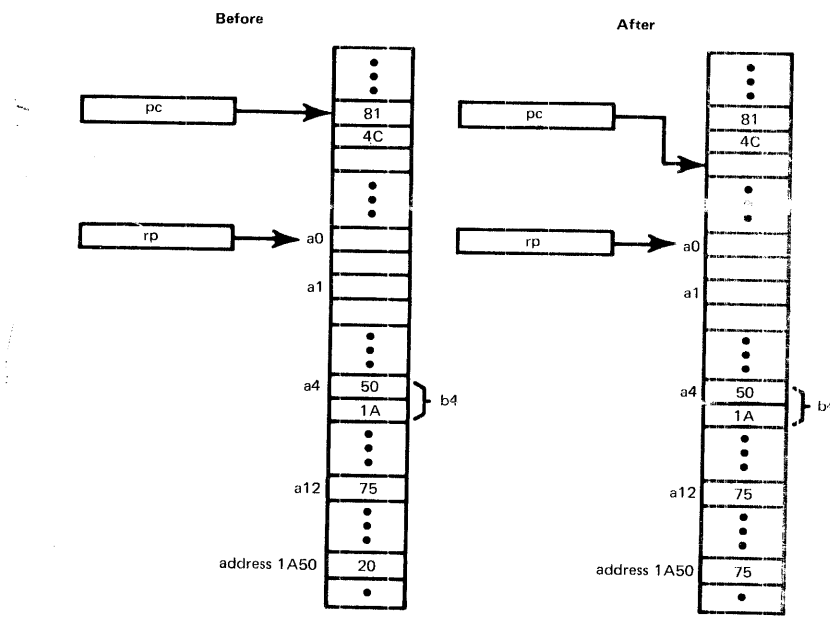
To move the contents of register a12 to the address pointed to by register b4, the assembly language instruction is

*b4=a12

This is a dyadic instruction for a move operation between an indirect address and a register, and in Part 1 of the chart, under the Addressing Mode (the Indirect and Register entry) and Move columns, you will find the corresponding instruction

*bd=as
81
ds

The opcode is 81, and it is followed by the **b** register that is used as a pointer to the destination, which is register b4, and the **a** register that is used as the source, which is register a12. So, the hex. code is 81 4C, a 2-byte instruction.



Note: Register b4 is assumed to contain address 1A50.

Memory Before and After Execution of 81 4C

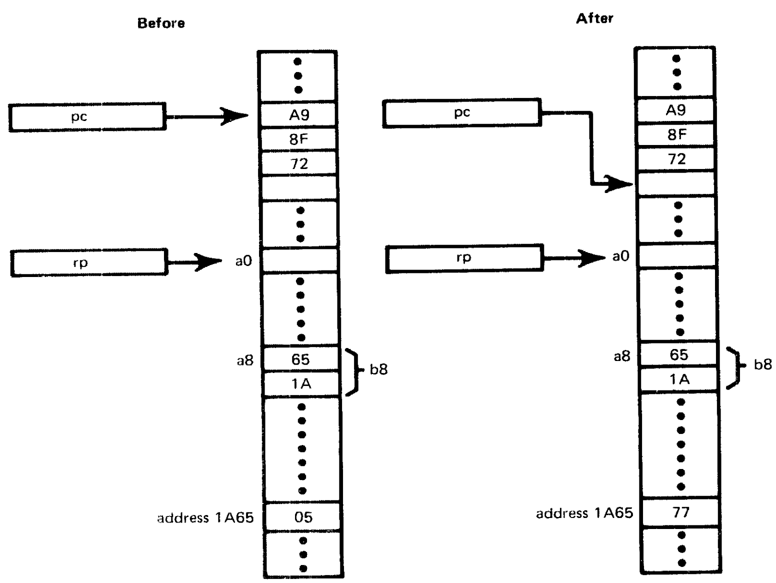
To add 72 to the contents of the address pointed to by register b8 and store the result in the same address, the assembly language instruction is

*b8=*b8+72

This is a dyadic instruction for an add operation between an indirect address and immediate data, and in Part 1, under the Addressing Mode (the Indirect and Immediate entry) and Add columns, you will find the corresponding instruction

*bd=*bd+N
A9
dF
N

The opcode is A9, and it is followed by the **b** register that is used as the pointer to the destination, which is register b8, hex. digit F, and the number that is to be added, which is 72. So, the hex. code is A9 8F 72, a 3-byte instruction.



Note: Register b8 is assumed to contain address 1A65.

Memory Before and After Execution of A9 8F 72

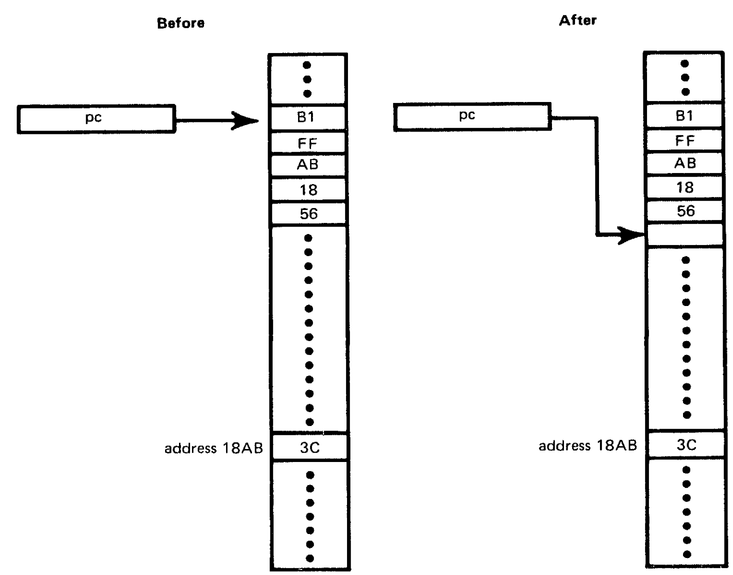
To compare the contents of the memory address 18AB with the number 56, the assembly language instruction is

*18AB–56

This is a dyadic instruction for a compare operation between a direct address and immediate data, and in Part 1, under the Addressing Mode (the Direct and Immediate entry) and Compare columns, you will find the corresponding instruction

*W–N
B1
FF
W(LO)
W(HI)
N

The opcode is B1 FF. It is followed by the low contents (eight least significant bits) and the high contents (eight most significant bits) of the address, which are AB and 18, respectively, and the number with which the contents of the address are to be compared, which is 56. So, the hex. code is B1 FF AB 18 56, a 5-byte instruction.



Note: No changes occur in memory contents as a result of the comparison; however, since the result of the comparison is negative, the neg bit in the cr is set to 1.

Memory Before and After Execution of B1 FF AB 18 56

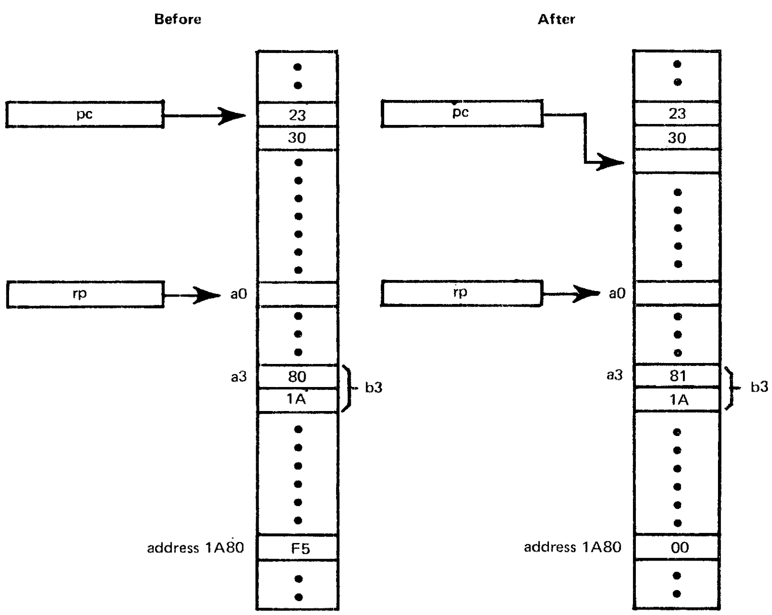
To place all zeros in the memory address pointed to by register b3 and then increment that address, the assembly language instruction is

*b3++=0

This is a monadic instruction for a zero operation with automatic incrementing of the indirect address, and in Part 2, under the Addressing Mode (the Automatic Increment entry) and Zero columns, you will find the corresponding instruction

*bd++=0
23
d0

The opcode is 23, and it is followed by the register that is used as the destination, which is register b3, and 0. So, the hex. code is 23 30, a 2-byte instruction.



Note: Register b3 is assumed to initially contain address 1A80.

Memory Before and After Execution of 23 30

To jump to address 1850, the assembly language instruction is

goto *0x1850

(The prefix 0x tells the assembler that the following number is a hex. number.)

This is a transfer instruction for an unconditional jump operation, and in Part 4, under the Jump Instructions and Always columns, you will find the corresponding instruction

goto *W
59
W(LO)
W(HI)

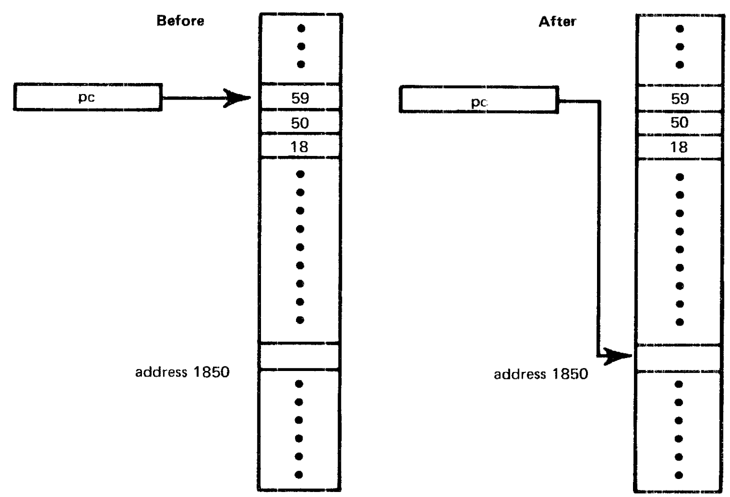
The opcode is 59. It is followed by the low contents (eight least significant bits) and the high contents (eight most significant bits) of the address to which the jump is to be made, which are 50 and 18, respectively. So, the hex. code is 59 50 18, a 3-byte instruction.

You can also use the instruction

if (condition) goto *W (where the condition is always)

49
FF
W(LO)
W(HI)

However, the hex. code is 49 FF 50 18, a 4-byte instruction.



Memory Before and After Execution of 59 50 18

To call the subroutine at the address pointed to by register b10 if the zero condition is not met, the assembly language instruction is

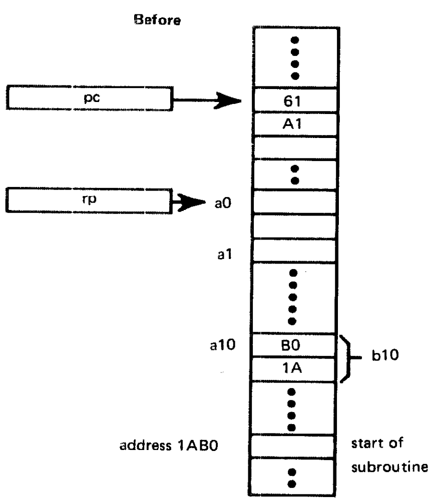
if(!zero)*b10( )

This is a conditional transfer instruction, and in Part 4, under the Call Instructions and Zero columns, you will find the corresponding instruction

if(!condition)*bd( ) (where the condition is zero)

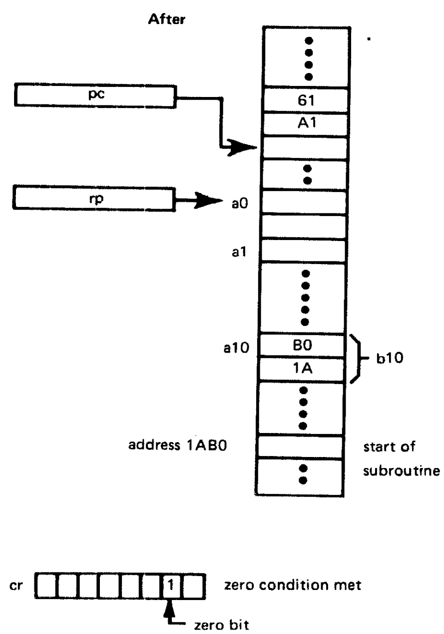
61
d1

The opcode is 61, and it is followed by the register that contains the address of the subroutine to be called, which is register b10, and 1. So, the hex. code is 61 A1, a 2-byte instruction.

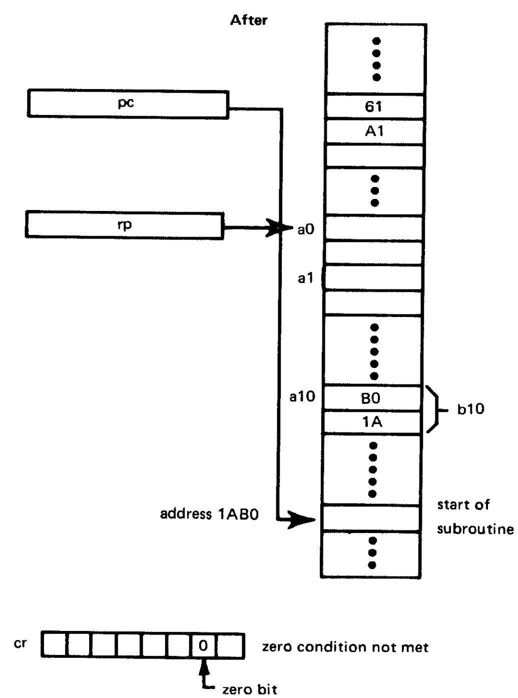


Note: Register b10 is assumed to contain address 1AB0.

Memory Before and After Execution of 61 A1



Zero Condition Met



Zero Condition Not Met

To jump seven locations ahead of the first instruction byte in your program if bit 3 of register a14 is a 1, the assembly language instruction is

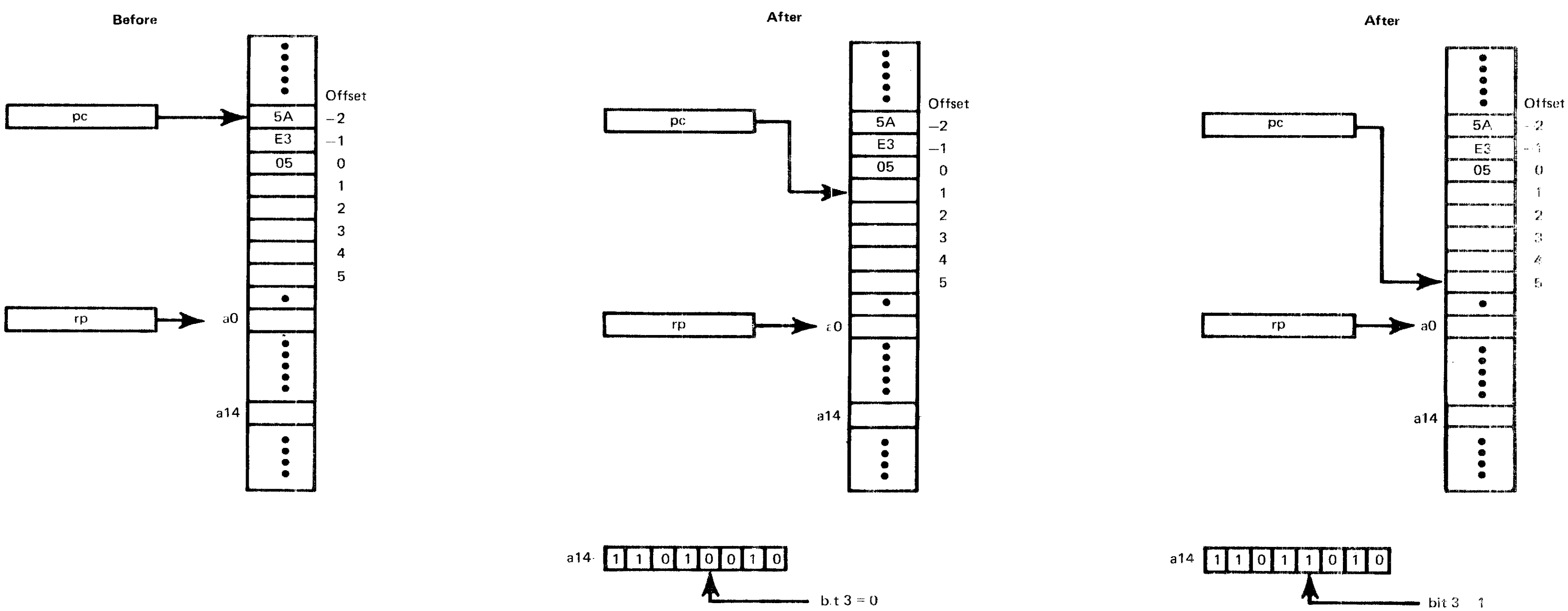
if(bit(3,a14))goto *(pc+5)

(The offset is 5 rather than 7 because the pc will contain an address two above that of the first instruction byte.)

This is a register bit conditional transfer instruction, and in Part 5, under the Jump Instructions and Bit No. 3 columns, you will find the corresponding instruction

if(bit(n,as))goto *(pc+N)
5A
s3
N

The opcode is 5A, and it is followed by the register number, the bit number, and the number of locations to be jumped. So, the hex. code is 5A E3 05, a 3-byte instruction.



Memory Before and After Execution of 5A E3 05

Condition Not Met

Condition Met

PART 1 — DYADIC INSTRUCTIONS

<table>
<tr><th rowspan="2">ADDRESSING MODE</th><th colspan="8">OPERATION</th><th colspan="2">DESTINATION–SOURCE RANGE</th></tr>
<tr><th>MOVE</th><th>XOR</th><th>OR</th><th>AND</th><th>SUBTRACT</th><th>ADD</th><th>COMPARE</th><th>TEST</th><th>d</th><th>s</th></tr>
<tr><td>Register and Register</td><td>ad=as
80
ds</td><td>ad=adΛas
88
ds</td><td>ad=ad | as
90
ds</td><td>ad=ad&as
98
ds</td><td>ad=ad–as
A0
ds</td><td>ad=ad+as
A8
ds</td><td>ad–as
B0
ds</td><td>test(ad,as)
B8
ds</td><td>0–15</td><td>0–14</td></tr>
<tr><td>Register and Immediate</td><td>ad=N
80
dF
N</td><td>ad=adΛN
88
dF
N</td><td>ad=ad | N
90
dF
N</td><td>ad=ad&N
98
dF
N</td><td>ad=ad–N
A0
dF
N</td><td>ad=ad+N
A8
dF
N</td><td>ad–N
B0
dF
N</td><td>test(ad,N)
B8
dF
N</td><td>0–15</td><td></td></tr>
<tr><td>Register and Indirect</td><td>ad=*bs
85
ds</td><td>ad=adΛ*bs
8D
ds</td><td>ad=ad | *bs
95
ds</td><td>ad=ad&*bs
9D
ds</td><td>ad=ad–*bs
A5
ds</td><td>ad=ad+*bs
AD
ds</td><td>ad–*bs
B5
ds</td><td>test(ad,*bs)
BD
ds</td><td>0–15</td><td>0–14</td></tr>
<tr><td>Register and Direct</td><td>ad=*W
85
dF
W(LO)
W(HI)</td><td>ad=adΛ*W
8D
dF
W(LO)
W(HI)</td><td>ad=ad | *W
95
dF
W(LO)
W(HI)</td><td>ad=ad&*W
9D
dF
W(LO)
W(HI)</td><td>ad=ad–*W
A5
dF
W(LO)
W(HI)</td><td>ad=ad+*W
AD
dF
W(LO)
W(HI)</td><td>ad–*W
B5
dF
W(LO)
W(HI)</td><td>test(ad,*W)
BD
dF
W(LO)
W(HI)</td><td>0–15</td><td></td></tr>
<tr><td>Register and Automatic Increment</td><td>ad=*bs++
87
ds</td><td>ad=adΛ*bs++
8F
ds</td><td>ad=ad | *bs++
97
ds</td><td>ad=ad&*bs++
9F
ds</td><td>ad=ad–*bs++
A7
ds</td><td>ad=ad+*bs++
AF
ds</td><td>ad–*bs++
B7
ds</td><td>test(ad,*bs++)
BF
ds</td><td>0–15</td><td>0–15</td></tr>
<tr><td>Register and Offset Memory</td><td>ad=*(bs+N)
86
ds
N</td><td>ad=adΛ*(bs+N)
8E
ds
N</td><td>ad=ad | *(bs+N)
96
ds
N</td><td>ad=ad&*(bs+N)
9E
ds
N</td><td>ad=ad–*(bs+N)
A6
ds
N</td><td>ad=ad+*(bs+N)
AE
ds
N</td><td>ad–*(bs+N)
B6
ds
N</td><td>test(ad,*(bs+N))
BE
ds
N</td><td>0–15</td><td>0–14</td></tr>
<tr><td>Register and Offset Stack</td><td>ad=*(sp+N)
86
dF
N</td><td>ad=adΛ*(sp+N)
8E
dF
N</td><td>ad=ad | *(sp+N)
96
dF
N</td><td>ad=ad&*(sp+N)
9E
dF
N</td><td>ad=ad–*(sp+N)
A6
dF
N</td><td>ad=ad+*(sp+N)
AE
dF
N</td><td>ad–*(sp+N)
B6
dF
N</td><td>test(ad,*(sp+N))
BE
dF
N</td><td>0–15</td><td></td></tr>
<tr><td>Indirect and Register</td><td>*bd=as
81
ds</td><td>*bd=*bdΛas
89
ds</td><td>*bd=*bd | as
91
ds</td><td>*bd=*bd&as
99
ds</td><td>*bd=*bd–as
A1
ds</td><td>*bd=*bd+as
A9
ds</td><td>*bd–as
B1
ds</td><td>test(*bd,as)
B9
ds</td><td>0–14</td><td>0–14</td></tr>
<tr><td>Indirect and Immediate</td><td>*bd=N
81
dF
N</td><td>*bd=*bdΛN
89
dF
N</td><td>*bd=*bd | N
91
dF
N</td><td>*bd=*bd&N
99
dF
N</td><td>*bd=*bd–N
A1
dF
N</td><td>*bd=*bd+N
A9
dF
N</td><td>*bd–N
B1
dF
N</td><td>test(*bd,N)
B9
dF
N</td><td>0–14</td><td></td></tr>
<tr><td>Indirect and 16-Bit Register See Note</td><td>*dd=bs
C1
ds</td><td>*dd=*ddΛbs
C9
ds</td><td>*dd=*dd | bs
D1
ds</td><td>*dd=*dd&bs
D9
ds</td><td>*dd=*dd–bs
E1
ds</td><td>*dd=*dd+bs
E9
ds</td><td>*dd–bs
F1
ds</td><td></td><td>0–14</td><td>0–14</td></tr>
</table>

Note: Instructions operating on 16-bit data identify the LO byte of the result with the specified memory address and the HI byte with the succeeding memory address.

PART 1 – DYADIC INSTRUCTIONS (Continued)

<table>
<tr><th rowspan="2">ADDRESSING MODE</th><th colspan="8">OPERATION</th><th colspan="2">DESTINATION–SOURCE RANGE</th></tr>
<tr><th>MOVE</th><th>XOR</th><th>OR</th><th>AND</th><th>SUBTRACT</th><th>ADD</th><th>COMPARE</th><th>TEST</th><th>d</th><th>s</th></tr>
<tr><td>Indirect and
16-Bit
Immediate
See Note</td><td>*dd=W
C1
dF
W(LO)
W(HI)</td><td>*dd=*dd∧W
C9
dF
W(LO)
W(HI)</td><td>*dd=*dd | W
D1
dF
W(LO)
W(HI)</td><td>*dd=*dd&W
D9
dF
W(LO)
W(HI)</td><td>*dd=*dd–W
E1
dF
W(LO)
W(HI)</td><td>*dd=*dd+W
E9
dF
W(LO)
W(HI)</td><td>*dd–W
F1
dF
W(LO)
W(HI)</td><td></td><td>0–14</td><td></td></tr>
<tr><td>Automatic
Increment
and Register</td><td>*bd++=as
83
ds</td><td>*bd++=*bd++∧as
8B
ds</td><td>*bd++=*bd++ | as
93
ds</td><td>*bd++=*bd++&as
9B
ds</td><td>*bd++=*bd++–as
A3
ds</td><td>*bd++=*bd++ + as
AB
ds</td><td>*bd++–as
B3
ds</td><td>test(*bd++,as)
BB
ds</td><td>0–15</td><td>0–14</td></tr>
<tr><td>Automatic
Increment
and Immediate</td><td>*bd++=N
83
dF
N</td><td>*bd++=*bd++∧N
8B
dF
N</td><td>*bd++=*bd++ | N
93
dF
N</td><td>*bd++=*bd++&N
9B
dF
N</td><td>*bd++=*bd++–N
A3
dF
N</td><td>*bd++=*bd++ + N
AB
dF
N</td><td>*bd++–N
B3
dF
N</td><td>test(*bd++,N)
BB
dF
N</td><td>0–15</td><td></td></tr>
<tr><td>Automatic
Increment and
16-Bit Register
See Note</td><td>*dd++=bs
C3
ds</td><td>*dd++=*dd++∧bs
CB
ds</td><td>*dd++=*dd++ | bs
D3
ds</td><td>*dd++=*dd++&bs
DB
ds</td><td>*dd++=*dd++–bs
E3
ds</td><td>*dd++=*dd++ + bs
EB
ds</td><td>*dd++–bs
F3
ds</td><td></td><td>0–15</td><td>0–14</td></tr>
<tr><td>Automatic
Increment and
16-Bit Immediate
See Note</td><td>*dd++=W
C3
dF
W(LO)
W(HI)</td><td>*dd++=*dd++∧W
CB
dF
W(LO)
W(HI)</td><td>*dd++=*dd++ | W
D3
dF
W(LO)
W(HI)</td><td>*dd++=*dd++&W
DB
dF
W(LO)
W(HI)</td><td>*dd++=*dd++–W
E3
dF
W(LO)
W(HI)</td><td>*dd++=*dd++ + W
EB
dF
W(LO)
W(HI)</td><td>*dd++–W
F3
dF
W(LO)
W(HI)</td><td></td><td>0–15</td><td></td></tr>
<tr><td>Offset
Memory and
Register</td><td>*(bd+N)=as
82
ds
N</td><td>*(bd+N)=*(bd+N)∧ as
8A
ds
N</td><td>*(bd+N)=*(bd+N) | as
92
ds
N</td><td>*(bd+N)=*(bd+N)&as
9A
ds
N</td><td>*(bd+N)=*(bd+N)–as
A2
ds
N</td><td>*(bd+N)=*(bd+N)+as
AA
ds
N</td><td>*(bd+N)–as
B2
ds
N</td><td>test (*(bd+N),as)
BA
ds
N</td><td>0–14</td><td>0–14</td></tr>
<tr><td>Offset
Memory and
Immediate</td><td>*(bd+N)=M
82
dF
N
M</td><td>*(bd+N)=*(bd+N)∧M
8A
dF
N
M</td><td>*(bd+N)=*(bd+N) | M
92
dF
N
M</td><td>*(bd+N)=*(bd+N)&M
9A
dF
N
M</td><td>*(bd+N)=*(bd+N)–M
A2
dF
N
M</td><td>*(bd+N)=*(bd+N)+M
AA
dF
N
M</td><td>*(bd+N)–M
B2
dF
N
M</td><td>test(*(bd+N),M)
BA
dF
N
M</td><td>0–14</td><td></td></tr>
<tr><td>Offset
Memory and
Offset Memory</td><td>*(bd+N)=
*(bs+M)
C4
ds
M
N</td><td>*(bd+N)=*(bd+N)∧
*(bs+M)
CC
ds
M
N</td><td>*(bd+N)=*(bd+N) |
*(bs+M)
D4
ds
M
N</td><td>*(bd+N)=*(bd+N)&
*(bs+M)
DC
ds
M
N</td><td>*(bd+N)=*(bd+N)–
*(bs+M)
E4
ds
M
N</td><td>*(bd+N)=*(bd+N)+
*(bs+M)
EC
ds
M
N</td><td>*(bd+N)–
*(bs+M)
F4
ds
M
N</td><td>test(*(bd+N),
*(bs+M))
FC
ds
M
N</td><td>0–14</td><td>0–14</td></tr>
</table>

Note: Instructions operating on 16-bit data identify the LO byte of the result with the specified memory address and the HI byte with the succeeding memory address.

PART 1 – DYADIC INSTRUCTIONS (Continued)

	OPERATION								DESTINATION–SOURCE RANGE	
ADDRESSING MODE	**MOVE**	**XOR**	**OR**	**AND**	**SUBTRACT**	**ADD**	**COMPARE**	**TEST**	**d**	**s**
Offset Memory and Offset Stack	*(bd+N)= *(sp+M) C4 dF M N	*(bd+N)=*(bd+N)∧ *(sp+M) CC dF M N	*(bd+N)=*(bd+N)| *(sp+M) D4 dF M N	*(bd+N)=*(bd+N)& *(sp+M) DC dF M N	*(bd+N)=*(bd+N)– *(sp+M) E4 dF M N	*(bd+N)=*(bd+N)+ *(sp+M) EC dF M N	*(bd+N)–*(sp+M) F4 dF M N	test(*(bd+N), *(sp+M)) FC dF M N	0–14	
Offset Memory and 16-Bit Register See Note	*(dd+N)=bs C2 ds N	*(dd+N)=*(dd+N)∧bs CA ds N	*(dd+N)=*(dd+N)|bs D2 ds N	*(dd+N)=*(dd+N)&bs DA ds N	*(dd+N)=*(dd+N)–bs E2 ds N	*(dd+N)=*(dd+N)+bs EA ds N	*(dd+N)–bs F2 ds N		0–14	0–14
Offset Memory and 16-Bit Immediate See Note	*(dd+N)=W C2 dF N W(LO) W(HI)	*(dd+N)=*(dd+N) ∧ W CA dF N W(LO) W(HI)	*(dd+N)=*(dd+N)| W D2 dF N W(LO) W(HI)	*(dd+N)=*(dd+N)&W DA dF N W(LO) W(HI)	*(dd+N)=*(dd+N)–W E2 dF N W(LO) W(HI)	*(dd+N)=*(dd+N)+W EA dF N W(LO) W(HI)	*(dd+N)–W F2 dF N W(LO) W(HI)		0–14	
Direct and Register	*W=as 81 Fs W(LO) W(HI)	*W=*W∧as 89 Fs W(LO) W(HI)	*W=*W|as 91 Fs W(LO) W(HI)	*W=*W&as 99 Fs W(LO) W(HI)	*W=*W–as A1 Fs W(LO) W(HI)	*W=*W+as A9 Fs W(LO) W(HI)	*W–as B1 Fs W(LO) W(HI)	test(*W,as) B9 Fs W(LO) W(HI)		0–14
Direct and Immediate	*W=N 81 FF W(LO) W(HI) N	*W=*W∧N 89 FF W(LO) W(HI) N	*W=*W|N 91 FF W(LO) W(HI) N	*W=*W&N 99 FF W(LO) W(HI) N	*W=*W–N A1 FF W(LO) W(HI) N	*W=*W+N A9 FF W(LO) W(HI) N	*W–N B1 FF W(LO) W(HI) N	test(*W,N) B9 FF W(LO) W(HI) N		
Direct and 16-Bit Register See Note	*W=bs C1 Fs W(LO) W(HI)	*W=*W∧bs C9 Fs W(LO) W(HI)	*W=*W|bs D1 Fs W(LO) W(HI)	*W=*W&bs D9 Fs W(LO) W(HI)	*W=*W–bs E1 Fs W(LO) W(HI)	*W=*W+bs E9 Fs W(LO) W(HI)	*W–bs F1 Fs W(LO) W(HI)			0–14
Direct and 16-Bit Immediate See Note	*W=V C1 FF W(LO) W(HI) V(LO) V(HI)	*W=*W∧V C9 FF W(LO) W(HI) V(LO) V(HI)	*W=*W|V D1 FF W(LO) W(HI) V(LO) V(HI)	*W=*W&V D9 FF W(LO) W(HI) V(LO) V(HI)	*W=*W–V E1 FF W(LO) W(HI) V(LO) V(HI)	*W=*W+V E9 FF W(LO) W(HI) V(LO) V(HI)	*W–V F1 FF W(LO) W(HI) V(LO) V(HI)			

Note: Instructions operating on 16-bit data identify the LO byte of the result with the specified memory address and the HI byte with the succeeding memory address.

PART 1 – DYADIC INSTRUCTIONS (Continued)

ADDRESSING MODE	OPERATION								DESTINATION–SOURCE RANGE	
	MOVE	XOR	OR	AND	SUBTRACT	ADD	COMPARE	TEST	d	s
Offset Stack and Register	*(sp+N)=as 82 Fs N	*(sp+N)=*(sp+N)Λ as 8A Fs N	*(sp+N)=*(sp+N) | as 92 Fs N	*(sp+N)=*(sp+N)&as 9A Fs N	*(sp+N)=*(sp+N)–as A2 Fs N	*(sp+N)=*(sp+N)+as AA Fs N	*(sp+N)–as B2 Fs N	test(*(sp+N),as) BA Fs N		0–14
Offset Stack and Immediate	*(sp+N)=M 82 FF N M	*(sp+N)=*(sp+N)Λ M 8A FF N M	*(sp+N)=*(sp+N) | M 92 FF N M	*(sp+N)=*(sp+N)&M 9A FF N M	*(sp+N)=*(sp+N)–M A2 FF N M	*(sp+N)=*(sp+N)+M AA FF N M	*(sp+N)–M B2 FF N M	test(*(sp+N),M) BA FF N M		
Offset Stack and Offset Memory	*(sp+N)= *(bs+M) C4 Fs M N	*(sp+N)=*(sp+N)Λ *(bs+M) CC Fs M N	*(sp+N)=*(sp+N) | *(bs+M) D4 Fs M N	*(sp+N)=*(sp+N)& *(bs+M) DC Fs M N	*(sp+N)=*(sp+N)– *(bs+M) E4 Fs M N	*(sp+N)=*(sp+N)+ *(bs+M) EC Fs M N	*(sp+N)=*(bs+M) F4 Fs M N	test(*(sp+N), *(bs+M)) FC Fs M N		0–14
Offset Stack and Offset Stack	*(sp+N)= *(sp+M) C4 FF M N	*(sp+N)=*(sp+N)Λ *(sp+M) CC FF M N	*(sp+N)=*(sp+N) | *(sp+M) D4 FF M N	*(sp+N)=*(sp+N)& *(sp+M) DC FF M N	*(sp+N)=*(sp+N)– *(sp+M) E4 FF M N	*(sp+N)=*(sp+N)+ *(sp+M) EC FF M N	*(sp+N)=*(sp+M) F4 FF M N	test(*(sp+N), *(sp+M)) FC FF M N		
Offset Stack and 16-Bit Register See Note	*(dsp+N)=bs C2 Fs N	*(dsp+N)= *(dsp+N)Λbs CA Fs N	*(dsp+N)= *(dsp+N) | bs D2 Fs N	*(dsp+N)= *(dsp+N)&bs DA Fs N	*(dsp+N)= *(dsp+N)–bs E2 Fs N	*(dsp+N)= *(dsp+N)+bs EA Fs N	*(dsp+N)–bs F2 Fs N			0–14
Offset Stack and 16-Bit Immediate See Note	*(dsp+N)=W C2 FF N W(LO) W(HI)	*(dsp+N)= *(dsp+N)ΛW CA FF N W(LO) W(HI)	*(dsp+N)= *(dsp+N) | W D2 FF N W(LO) W(HI)	*(dsp+N)= *(dsp+N)&W DA FF N W(LO) W(HI)	*(dsp+N)= *(dsp+N)–W E2 FF N W(LO) W(HI)	*(dsp+N)= *(dsp+N)+W EA FF N W(LO) W(HI)	*(dsp+N)–W F2 FF N W(LO) W(HI)			

Note: Instructions operating on 16-bit data identify the LO byte of the result with the specified memory address and the HI byte with the succeeding memory address.

PART 1 – DYADIC INSTRUCTIONS (Continued)

ADDRESSING MODE	OPERATION MOVE	XOR	OR	AND	SUBTRACT	ADD	COMPARE	TEST	DESTINATION–SOURCE RANGE d	s
16-Bit Register and 16-Bit Register	bd=bs C0 ds	bd=bd∧bs C8 ds	bd=bd|bs D0 ds	bd=bd&bs D8 ds	bd=bd–bs E0 ds	bd=bd+bs E8 ds	bd–bs F0 ds		0–15	0–14
16-Bit Register and 16-Bit Immediate	bd=W C0 dF W(LO) W(HI)	bd=bd∧W C8 dF W(LO) W(HI)	bd=bd|W D0 dF W(LO) W(HI)	bd=bd&W D8 dF W(LO) W(HI)	bd=bd–W E0 dF W(LO) W(HI)	bd=bd+W E8 dF W(LO) W(HI)	bd–W F0 dF W(LO) W(HI)		0–15	
16-Bit Register and Indirect See Note	bd=*ds C5 ds	bd=bd∧*ds CD ds	bd=bd|*ds D5 ds	bd=bd&*ds DD ds	bd=bd–*ds E5 ds	bd=bd+*ds ED ds	bd–*ds F5 ds		0–15	0–14
16-Bit Register and Direct See Note	bd=*W C5 dF W(LO) W(HI)	bd=bd∧*W CD dF W(LO) W(HI)	bd=bd|*W D5 dF W(LO) W(HI)	bd=bd&*W DD dF W(LO) W(HI)	bd=bd–*W E5 dF W(LO) W(HI)	bd=bd+*W ED dF W(LO) W(HI)	bd=*W F5 dF W(LO) W(HI)		0–15	
16-Bit Register and Automatic Increment See Note	bd=*ds++ C7 ds	bd=bd∧*ds++ CF ds	bd=bd|*ds++ D7 ds	bd=bd&*ds++ DF ds	bd=bd–*ds++ E7 ds	bd=bd+*ds++ EF ds	bd–*ds++ F7 ds		0–15	0–15
16-Bit Register and Offset Memory See Note	bd=*(ds+N) C6 ds N	bd=bd∧*(ds+N) CE ds N	bd=bd|*(ds+N) D6 ds N	bd=bd&*(ds+N) DE ds N	bd=bd–*(ds+N) E6 ds N	bd=bd+*(ds+N) EE ds N	bd–*(ds+N) F6 ds N		0–15	0–14
16-Bit Register and Offset Stack See Note	bd=*(dsp+N) C6 dF N	bd=bd∧*(dsp+N) CE dF N	bd=bd|*(dsp+N) D6 dF N	bd=bd&*(dsp+N) DE dF N	bd=bd–*(dsp+N) E6 dF N	bd=bd+*(dsp+N) EE dF N	bd–*(dsp+N) F6 dF N		0–15	

Note: Instructions operating on 16-bit data identify the LO byte of the result with the specified memory address and the HI byte with the succeeding memory address.

PART 2 – MONADIC INSTRUCTIONS

OPERATION

ADDRESSING MODE	ZERO	NEGATE	INCREMENT	DECREMENT	COMPLEMENT	SHIFT ARITHMETICAL LEFT	SHIFT ARITHMETICAL RIGHT
Register	ad=0 20 d0	ad=–ad 24 d0	++ad 28 d0	––ad 28 d8	ad=~ad 2C d0	ad=ad*2 30 d1	ad=ad/2 30 dF
Indirect	*bd=0 21 d0	*bd=–*bd 25 d0	++*bd 29 d0	––*bd 29 d8	*bd=~*bd 2D d0	*bd=*bd*2 31 d1	*bd=*bd/2 31 dF
Automatic Increment	*bd++=0 23 d0	*bd++= –*bd++ 27 d0	++*bd++ 2B d0	––*bd++ 2B d8	*bd++= ~*bd++ 2F d0	*bd++= *bd++*2 33 d1	*bd++= *bd++/2 33 dF
Offset Memory	*(bd+N)=0 22 d0 N	*(bd+N)= –*(bd+N) 26 d0 N	++*(bd+N) 2A d0 N	––*(bd+N) 2A d8 N	*(bd+N)= ~*(bd+N) 2E d0 N	*(bd+N)= *(bd+N)*2 32 d1 N	*(bd+N)= *(bd+N)/2 32 dF N
Direct	*W=0 21 F0 W(LO) W(HI)	*W=–*W 25 F0 W(LO) W(HI)	++*W 29 F0 W(LO) W(HI)	––*W 29 F8 W(LO) W(HI)	*W=~*W 2D F0 W(LO) W(HI)	*W=*W*2 31 F1 W(LO) W(HI)	*W=*W/2 31 FF W(LO) W(HI)
Offset Stack	*(sp+N)=0 22 F0 N	*(sp+N)= –*(sp+N) 26 F0 N	++*(sp+N) 2A F0 N	––*(sp+N) 2A F8 N	*(sp+N)= ~*(sp+N) 2E F0 N	*(sp+N)= *(sp+N)*2 32 F1 N	*(sp+N)= *(sp+N)/2 32 FF N
16-Bit Register	bd=0 60 d0		++bd 68 d0	––bd 68 d8			

ADDRESSING MODE	SHIFT LOGICAL LEFT	SHIFT LOGICAL RIGHT	ROTATE 8 LEFT	ROTATE 8 RIGHT	ROTATE 9 LEFT	ROTATE 9 RIGHT	DESTINATION RANGE d
Register	ad=ad<<1 38 d1	ad=ad>>1 38 dF	ad=ad<<<1 34 d1	ad=ad>>>1 34 dF	ad=ad$<<1 3C d1	ad=ad >>$1 3C dF	0–15
Indirect	*bd=*bd<<1 39 d1	*bd=*bd>>1 39 dF	*bd=*bd<<<1 35 d1	*bd=*bd>>>1 35 dF	*bd=*bd$ <<1 3D d1	*bd=*bd >>$1 3D dF	0–14
Automatic Increment	*bd++= *bd++<<1 3B d1	*bd++= *bd++>>1 3B dF	*bd++= *bd++<<<1 37 d1	*bd++= *bd++>>>1 37 dF	*bd++= *bd++$<<1 3F d1	*bd++= *bd++>>$1 3F dF	0–15
Offset Memory	*(bd+N)= *(bd+N)<<1 3A d1 N	*(bd+N)= *(bd+N)>>1 3A dF N	*(bd+N)= *(bd+N)<<<1 36 d1 N	*(bd+N)= *(bd+N)>>>1 36 dF N	*(bd+N)= *(bd+N)$<<1 3E d1 N	*(bd+N)= *(bd+N)>>$1 3E dF N	0–14
Direct	*W=*W<<1 39 F1 W(LO) W(HI)	*W=*W>>1 39 FF W(LO) W(HI)	*W=*W<<<1 35 F1 W(LO) W(HI)	*W=*W>>>1 35 FF W(LO) W(HI)	*W=*W$<<1 3D F1 W(LO) W(HI)	*W=*W >>$1 3D FF W(LO) W(HI)	
Offset Stack	*(sp+N)= *(sp+N)<<1 3A F1 N	*(sp+N)= *(sp+N)>>1 3A FF N	*(sp+N)= *(sp+N)<<<1 36 F1 N	*(sp+N)= *(sp+N)>>>1 36 FF N	*(sp+N)= *(sp+N)$<<1 3E F1 N	*(sp+N)= *(sp+N)>>$1 3E FF N	
16-Bit Register							0–15

PART 3 – MISCELLANEOUS DYADIC AND MONADIC INSTRUCTIONS

SPECIAL ADDRESSING MODE	OPERATION						DESTINATION–SOURCE RANGE d	 s
Register and Register	**FIND LEFT ONES** ad=flo(as) 0C ds	**FIND, CLEAR LEFT ONES** ad=floc(as) 4C ds	**COUNT ONES** ad=bitsum(as) 0E ds				0–15	0–15
16-Bit Register	**LOAD REGISTER ADDRESS** bd=&bs 6D ds	**LOAD INSTRUCTION ADDRESS** bd=&*pc 6D dF	**LOAD MEMORY ADDRESS** bd=&*(bs+N) 6F ds N	**LOAD STACK ADDRESS** bd=&*(sp+N) 6F dF N	**BYTE SWAP** swap(bd) 6A d0	**SIGN EXTEND** extend(bd) 62 d0	0–15	0–14
	LOGICAL ADD REGISTER bd=bd+ logical (as) 75 ds	**LOGICAL ADD IMMEDIATE** bd=bd+ logical (N) 75 dF N	**ARITHMETICAL ADD REGISTER** bd=bd+as 7D ds	**ARITHMETICAL ADD IMMEDIATE** bd=bd+N 7D dF N			0–14	0–14
rp **See Note**	**LOAD IMMEDIATE** rp=W 4D 0F W(LO) W(HI)	**LOAD REGISTER** rp=bs 4D 0s	**BUMP 4** rp=rp+8 42	**BUMP 8** rp=rp+16 43	**DEBUMP 4** rp=rp–8 4A	**DEBUMP 8** rp=rp–16 4B		0–14
sp	**LOAD IMMEDIATE** sp=W 0D 0F W(LO) W(HI)	**LOAD REGISTER** sp=bs 0D 0s	**LOGICAL ADD REGISTER** sp=sp+ logical (as) 75 Fs	**LOGICAL ADD IMMEDIATE** sp=sp+ logical (N) 75 FF N	**ARITHMETICAL ADD REGISTER** sp=sp+as 7D Fs	**ARITHMETICAL ADD IMMEDIATE** sp=sp+N 7D FF N		0–14

Note: The last three bits of the rp are always 0s.

PART 3 – MISCELLANEOUS DYADIC AND MONADIC INSTRUCTIONS (Continued)

SPECIAL ADDRESSING MODE	OPERATION				DESTINATION–SOURCE RANGE d	s
cr See Note	**SET CERTAIN BITS** set (N) 01 N	**CLEAR CERTAIN BITS** clear (N) 03 N				
Stack	**SAVE 8-BIT REGISTER** push (as) 06 0s	**SAVE 16-BIT REGISTER** push (bs) 46 0s	**SAVE rp** push (rp) 47	**SAVE cr** push (cr) 07		0–15
Stack	**RESTORE 8-BIT REGISTER** ad=pop( ) 04 d0	**RESTORE 16-BIT REGISTER** bd=pop( ) 44 d0	**RESTORE rp** rp=pop( ) 45	**RESTORE cr** cr=pop( ) 05	0–15	

Note: The bits to be set or cleared are determined by the location of 1s in N. The order of the bits of the cr is as follows: 7, flag; 6, enable; 5, odd; 4, ones; 3, carry; 2, ovfl; 1, zero; 0, neg.

PART 4 – TRANSFER INSTRUCTIONS: UNCONDITIONAL AND CONDITION REGISTER BIT CONDITIONAL

JUMP INSTRUCTIONS	UNCONDITIONAL (ALWAYS)	CONDITION NEG	ZERO	OVFL	CARRY	ONES	ODD	ENABLE	FLAG	LT	LTEQ	LLTEQ	HOMOG	SHOVFL	DESTINATION RANGE d
goto *W	59 W(LO) W(HI)														
goto *(pc+N)	58 N														
if(condition) goto *W	49 FF W(LO) W(HI)	49 F0 W(LO) W(HI)	49 F1 W(LO) W(HI)	49 F2 W(LO) W(HI)	49 F3 W(LO) W(HI)	49 F4 W(LO) W(HI)	49 F5 W(LO) W(HI)	49 F6 W(LO) W(HI)	49 F7 W(LO) W(HI)	49 F8 W(LO) W(HI)	49 F9 W(LO) W(HI)	49 FA W(LO) W(HI)	49 FB W(LO) W(HI)	49 FC W(LO) W(HI)	
if(!condition) goto *W	41 FF W(LO) W(HI)	41 F0 W(LO) W(HI)	41 F1 W(LO) W(HI)	41 F2 W(LO) W(HI)	41 F3 W(LO) W(HI)	41 F4 W(LO) W(HI)	41 F5 W(LO) W(HI)	41 F6 W(LO) W(HI)	41 F7 W(LO) W(HI)	41 F8 W(LO) W(HI)	41 F9 W(LO) W(HI)	41 FA W(LO) W(HI)	41 FB W(LO) W(HI)	41 FC W(LO) W(HI)	
if(condition) goto *bd	49 dF	49 d0	49 d1	49 d2	49 d3	49 d4	49 d5	49 d6	49 d7	49 d8	49 d9	49 dA	49 dB	49 dC	0–14
if(!condition) goto *bd	41 dF	41 d0	41 d1	41 d2	41 d3	41 d4	41 d5	41 d6	41 d7	41 d8	41 d9	41 dA	41 dB	41 dC	0–14
if(condition) goto *(bd+N)	48 dF N	48 d0 N	48 d1 N	48 d2 N	48 d3 N	48 d4 N	48 d5 N	48 d6 N	48 d7 N	48 d8 N	48 d9 N	48 dA N	48 dB N	48 dC N	0–14
if(!condition) goto *(bd+N)	40 dF N	40 d0 N	40 d1 N	40 d2 N	40 d3 N	40 d4 N	40 d5 N	40 d6 N	40 d7 N	40 d8 N	40 d9 N	40 dA N	40 dB N	40 dC N	0–14
if(condition) goto *(pc+N)	48 FF N	48 F0 N	48 F1 N	48 F2 N	48 F3 N	48 F4 N	48 F5 N	48 F6 N	48 F7 N	48 F8 N	48 F9 N	48 FA N	48 FB N	48 FC N	
if(!condition) goto *(pc+N)	40 FF N	40 F0 N	40 F1 N	40 F2 N	40 F3 N	40 F4 N	40 F5 N	40 F6 N	40 F7 N	40 F8 N	40 F9 N	40 FA N	40 FB N	40 FC N	

PART 4 — TRANSFER INSTRUCTIONS: UNCONDITIONAL AND CONDITION REGISTER BIT CONDITIONAL (Continued)

CALL INSTRUCTIONS	UNCONDITIONAL (ALWAYS)	CONDITION NEG	ZERO	OVFL	CARRY	ONES	ODD	ENABLE	FLAG	LT	LTEQ	LLTEQ	HOMOG	SHOVFL	DESTINATION RANGE d
*W( )	79 W(LO) W(HI)														
if(condition) *W( )	69 FF W(LO) W(HI)	69 F0 W(LO) W(HI)	69 F1 W(LO) W(HI)	69 F2 W(LO) W(HI)	69 F3 W(LO) W(HI)	69 F4 W(LO) W(HI)	69 F5 W(LO) W(HI)	69 F6 W(LO) W(HI)	69 F7 W(LO) W(HI)	69 F8 W(LO) W(HI)	69 F9 W(LO) W(HI)	69 FA W(LO) W(HI)	69 FB W(LO) W(HI)	69 FC W(LO) W(HI)	
if(!condition) *W( )	61 FF W(LO) W(HI)	61 F0 W(LO) W(HI)	61 F1 W(LO) W(HI)	61 F2 W(LO) W(HI)	61 F3 W(LO) W(HI)	61 F4 W(LO) W(HI)	61 F5 W(LO) W(HI)	61 F6 W(LO) W(HI)	61 F7 W(LO) W(HI)	61 F8 W(LO) W(HI)	61 F9 W(LO) W(HI)	61 FA W(LO) W(HI)	61 FB W(LO) W(HI)	61 FC W(LO) W(HI)	
if(condition) *bd( )	69 dF	69 d0	69 d1	69 d2	69 d3	69 d4	69 d5	69 d6	69 d7	69 d8	69 d9	69 dA	69 dB	69 dC	0–14
if(!condition) *bd( )	61 dF	61 d0	61 d1	61 d2	61 d3	61 d4	61 d5	61 d6	61 d7	61 d8	61 d9	61 dA	61 dB	61 dC	0–14
RETURN INSTRUCTIONS															
return	66														
i return( )	67														
if(condition) return	64 0F	64 00	64 01	64 02	64 03	64 04	64 05	64 06	64 07	64 08	64 09	64 0A	64 0B	64 0C	
if(!condition) return	65 0F	65 00	65 01	65 02	65 03	65 04	65 05	65 06	65 07	65 08	65 09	65 0A	65 0B	65 0C	

PART 5 — TRANSFER INSTRUCTIONS: REGISTER BIT CONDITIONAL AND MISCELLANEOUS

	REGISTER BIT NUMBER (n)								SOURCE RANGE
JUMP INSTRUCTIONS	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**s**
if(bit(n,as)) goto *(pc+N)	5A s0 N	5A s1 N	5A s2 N	5A s3 N	5A s4 N	5A s5 N	5A s6 N	5A s7 N	0–15
if(!bit(n,as)) goto *(pc+N)	52 s0 N	52 s1 N	52 s2 N	52 s3 N	52 s4 N	52 s5 N	52 s6 N	52 s7 N	0–15
if(bit(n,*bs)) goto *(pc+N)	5B s0 N	5B s1 N	5B s2 N	5B s3 N	5B s4 N	5B s5 N	5B s6 N	5B s7 N	0–15
if(!bit(n,*bs)) goto *(pc+N)	53 s0 N	53 s1 N	53 s2 N	53 s3 N	53 s4 N	53 s5 N	53 s6 N	53 s7 N	0–15
if(bit(n,*bs++)) goto *(pc+N)	7B s0 N	7B s1 N	7B s2 N	7B s3 N	7B s4 N	7B s5 N	7B s6 N	7B s7 N	0–15
if(!bit(n,*bs++)) goto *(pc+N)	73 s0 N	73 s1 N	73 s2 N	73 s3 N	73 s4 N	73 s5 N	73 s6 N	73 s7 N	0–15

MISCELLANEOUS

halt( )	78
nop( )	7F

Note 1: Register bit number 7 is the most significant bit.

Note 2: N is a positive or negative number (signed 2s complement).

PART 6 – SUMMARY OF MACHINE CODES

MACHINE CODE BYTE 1	2	3	4	5	6	ASSEMBLY LANGUAGE INSTRUCTION	ADDRESSING MODE	OPERATION	CONDITION (See Note)
01	N					set(N)	cr	Set Certain Bits	M
03	N					clear(N)	cr	Clear Certain Bits	M
04	d0					ad=pop( )	Stack	Restore 8-Bit Register	N
05						cr=pop( )	Stack	Restore cr	S
06	0s					push(as)	Stack	Save 8-Bit Register	N
07						push(cr)	Stack	Save cr	N
0C	ds					ad=flo(as)	Register and Register	Find Left Ones	F
0D	0s					sp=bs	sp	Load Register	N
0D	0F	W(LO)	W(HI)			sp=W	sp	Load Immediate	N
0E	ds					ad=bitsum(as)	Register and Register	Count Ones	F
20	d0					ad=0	Register	Zero	F
21	d0					*bd=0	Indirect	Zero	F
21	F0	W(LO)	W(HI)			*W=0	Direct	Zero	F
22	d0	N				*(bd+N)=0	Offset Memory	Zero	F
22	F0	N				*(sp+N)=0	Offset Stack	Zero	F
23	d0					*bd++=0	Automatic Increment	Zero	F
24	d0					ad=−ad	Register	Negate	A
25	d0					*bd=−*bd	Indirect	Negate	A
25	F0	W(LO)	W(HI)			*W=−*W	Direct	Negate	A
26	d0	N				*(bd+N)=−*(bd+N)	Offset Memory	Negate	A
26	F0	N				*(sp+N)=−*(sp+N)	Offset Stack	Negate	A
27	d0					*bd++=−*bd++	Automatic Increment	Negate	A
28	d0					++ad	Register	Increment	A
28	d8					−−ad	Register	Decrement	A
29	d0					++*bd	Indirect	Increment	A
29	d8					−−*bd	Indirect	Decrement	A
29	F0	W(LO)	W(HI)			++*W	Direct	Increment	A
29	F8	W(LO)	W(HI)			−−*W	Direct	Decrement	A
2A	d0	N				++*(bd+N)	Offset Memory	Increment	A
2A	d8	N				−−*(bd+N)	Offset Memory	Decrement	A
2A	F0	N				++*(sp+N)	Offset Stack	Increment	A
2A	F8	N				−−*(sp+N)	Offset Stack	Decrement	A
2B	d0					++*bd++	Automatic Increment	Increment	A
2B	d8					−−*bd++	Automatic Increment	Decrement	A
2C	d0					ad=~ad	Register	Complement	F
2D	d0					*bd= ~ *bd	Indirect	Complement	F
2D	F0	W(LO)	W(HI)			*W= ~ *W	Direct	Complement	F

Note: The Condition column identifies which bits of the cr are affected by the operation. The characters are defined as follows: A, affects the neg, odd, zero, ones, ovfl, and carry bits; F, affects neg, odd, zero, and ones bits; H, affects enable bit; M, mask determines bits affected; N, no effect; and S, byte popped determines bits affected. Illegal opcodes have no effect on the bits of the cr. The odd and ones bits are affected by all 16-bit dyadic instructions (move, exclusive OR, OR, AND, subtract, add, compare) and by 16-bit increment and decrement instructions in an undefined manner. The ovfl bit is affected by 8- and 16-bit increment and decrement instructions and by all rotate and shift instructions in an undefined manner. The conditions lt, lteq, llteq, homog, and shovfl are not part of the cr but are logical combinations of the cr bits. They are derived as follows: lt = neg Λ ovfl; lteq = zero | (neg Λ ovfl); llteq = carry | zero; homog = zero | ones; and shovfl = neg Λ carry.

MACHINE CODE BYTE 1	2	3	4	5	6	ASSEMBLY LANGUAGE INSTRUCTION	ADDRESSING MODE	OPERATION	CONDITION
2E	d0	N				*(bd+N)= ~ *(bd+N)	Offset Memory	Complement	F
2E	F0	N				*(sp+N)= ~ *(sp+N)	Offset Stack	Complement	F
2F	d0					*bd++=~*bd++	Automatic Increment	Complement	F
30	d1					ad=ad*2	Register	Left Arithmetical Shift	A
30	dF					ad=ad/2	Register	Right Arithmetical Shift	A
31	d1					*bd=*bd*2	Indirect	Left Arithmetical Shift	A
31	dF					*bd=*bd/2	Indirect	Right Arithmetical Shift	A
31	F1	W(LO)	W(HI)			*W=*W*2	Direct	Left Arithmetical Shift	A
31	FF	W(LO)	W(HI)			*W=*W/2	Direct	Right Arithmetical Shift	A
32	d1	N				*(bd+N)=*(bd+N)*2	Offset Memory	Left Arithmetical Shift	A
32	dF	N				*(bd+N)=*(bd+N)/2	Offset Memory	Right Arithmetical Shift	A
32	F1	N				*(sp+N)=*(sp+N)*2	Offset Stack	Left Arithmetical Shift	A
32	FF	N				*(sp+N)=*(sp+N)/2	Offset Stack	Right Arithmetical Shift	A
33	d1					*bd++=*bd++*2	Automatic Increment	Left Arithmetical Shift	A
33	dF					*bd++=*bd++/2	Automatic Increment	Right Arithmetical Shift	A
34	d1					ad=ad <<< 1	Register	Left Rotate 8	A
34	dF					ad=ad >>> 1	Register	Right Rotate 8	A
35	d1					*bd=*bd <<<1	Indirect	Left Rotate 8	A
35	dF					*bd=*bd >>>1	Indirect	Right Rotate 8	A
35	F1	W(LO)	W(HI)			*W=*W <<<1	Direct	Left Rotate 8	A
35	FF	W(LO)	W(HI)			*W=*W >>>1	Direct	Right Rotate 8	A
36	d1	N				*(bd+N)=*(bd+N) <<<1	Offset Memory	Left Rotate 8	A
36	dF	N				*(bd+N)=*(bd+N) >>>1	Offset Memory	Right Rotate 8	A
36	F1	N				*(sp+N)=*(sp+N) <<< 1	Offset Stack	Left Rotate 8	A
36	FF	N				*(sp+N)=*(sp+N) >>> 1	Offset Stack	Right Rotate 8	A
37	d1					*bd++=*bd++ <<<1	Automatic Increment	Left Rotate 8	A
37	dF					*bd++=*bd++ >>>1	Automatic Increment	Right Rotate 8	A
38	d1					ad=ad <<1	Register	Left Logical Shift	A
38	dF					ad=ad >>1	Register	Right Logical Shift	A
39	d1					*bd=*bd<<1	Indirect	Left Logical Shift	A
39	dF					*bd=*bd>>1	Indirect	Right Logical Shift	A
39	F1	W(LO)	W(HI)			*W=*W <<1	Direct	Left Logical Shift	A
39	FF	W(LO)	W(HI)			*W=*W >>1	Direct	Right Logical Shift	A
3A	d1	N				*(bd+N)=*(bd+N) <<1	Offset Memory	Left Logical Shift	A
3A	dF	N				*(bd+N)=*(bd+N) >>1	Offset Memory	Right Logical Shift	A
3A	F1	N				*(sp+N)=*(sp+N) <<1	Offset Stack	Left Logical Shift	A
3A	FF	N				*(sp+N)=*(sp+N) >>1	Offset Stack	Right Logical Shift	A
3B	d1					*bd++=*bd++ <<1	Automatic Increment	Left Logical Shift	A
3B	dF					*bd++=*bd++ >>1	Automatic Increment	Right Logical Shift	A
3C	d1					ad=ad$ <<1	Register	Left Rotate 9	A
3C	dF					ad=ad >>$1	Register	Right Rotate 9	A
3D	d1					*bd=*bd$<<1	Indirect	Left Rotate 9	A
3D	dF					*bd=*bd>>$1	Indirect	Right Rotate 9	A
3D	F1	W(LO)	W(HI)			*W=*W$ <<1	Direct	Left Rotate 9	A
3D	FF	W(LO)	W(HI)			*W=*W >>$1	Direct	Right Rotate 9	A
3E	d1	N				*(bd+N)=*(bd+N)$ <<1	Offset Memory	Left Rotate 9	A
3E	dF	N				*(bd+N)=*(bd+N) >>$1	Offset Memory	Right Rotate 9	A
3F	F1	N				*(sp+N)=*(sp+N)$ <<1	Offset Stack	Left Rotate 9	A
3F	FF	N				*(sp+N)=*(sp+N) >>$1	Offset Stack	Right Rotate 9	A
3F	d1					*bd++=*bd++$ <<1	Automatic Increment	Left Rotate 9	A
3F	dF					*bd++=*bd++ >>$1	Automatic Increment	Right Rotate 9	A

MACHINE CODE BYTE 1	2	3	4	5	6	ASSEMBLY LANGUAGE INSTRUCTION	ADDRESSING MODE	OPERATION	CONDITION
40	dc	N	See Note 1			if(!condition)goto*(bd+N)		Jump	N
40	Fc	N	See Note 1			if(!condition)goto*(pc+N)		Jump	N
41	dc		See Note 1			if(!condition)goto*bd		Jump	N
41	Fc	W(LO)	W(HI)	See Note 1		if(!condition)goto*W		Jump	N
42						rp=rp+8	rp	Bump 4	N
43						rp=rp+16	rp	Bump 8	N
44	d0					bd=pop( )	Stack	Restore 16-Bit Register	N
45						rp=pop( )	Stack	Restore rp	N
46	0s					push(bs)	Stack	Save 16-Bit Register	N
47						push(rp)	Stack	Save rp	N
48	dc	N	See Note 1			if(condition)goto*(bd+N)		Jump	N
48	Fc	N	See Note 1			if(condition)goto*(pc+N)		Jump	N
49	dc		See Note 1			if(condition)goto*bd		Jump	N
49	Fc	W(LO)	W(HI)	See Note 1		if(condition)goto*W		Jump	N
4A						rp=rp−8	rp	Debump 4	N
4B						rp=rp−16	rp	Debump 8	N
4C	ds					ad=floc(as)	Register and Register	Find, Clear Left Ones	F
4D	0s					rp=bs	rp	Load Register	N
4D	0F	W(LO)	W(HI)			rp=W	rp	Load Immediate	N
52	sn	N	See Note 2			if(!bit(n,as))goto*(pc+N)		Jump	N
53	sn	N	See Note 2			if(!bit(n,*bs))goto*(pc+N)		Jump	N
58	N					goto*(pc+N)		Unconditional Jump	N
59	W(LO)	W(HI)				goto*W		Unconditional Jump	N
5A	sn	N	See Note 2			if(bit(n,as))goto*(pc+N)		Jump	N
5B	sn	N	See Note 2			if(bit(n,*bs))goto*(pc+N)		Jump	N
60	d0					bd=0	16-Bit Register	Zero	A
61	dc		See Note 1			if(!condition)*bd( )		Call	N
61	Fc	W(LO)	W(HI)	See Note 1		if(!condition)*W( )		Call	N
62	d0					extend (bd)	16-Bit Register	Sign Extend	N
64	0c			See Note 1		if(condition)return		Return	N
65	0c			See Note 1		if(!condition)return		Return	N
66						return		Return	N
67						i return( )		Interrupt Return	S
68	d0					++bd	16-Bit Register	Increment	A
68	d8					−−bd	16-Bit Register	Decrement	A
69	dc			See Note 1		if(condition)*bd( )		Call	N
69	Fc	W(LO)	W(HI)	See Note 1		if(condition)*W( )		Call	N
6A	d0					swap(bd)	16-Bit Register	Byte Swap	N
6D	ds					bd=&bs	16-Bit Register	Load Register Address	N
6D	dF					bd=&*pc	16-Bit Register	Load Instruction Address	N
6F	ds	N				bd=&*(bs+N)	16-Bit Register	Load Memory Address	N
6F	dF	N				bd=&*(sp+N)	16-Bit Register	Load Stack Address	N
73	sn	N		See Note 2		if(!bit(n,*bs++))goto*(pc+N)		Jump	N

Note 1: c is the condition. Its value is as follows: 0, neg; 1, zero; 2, ovfl; 3, carry; 4, ones; 5, odd; 6, enable; 7, flag; 8, lt; 9, lteq; A, llteq; B, homog; C, shovfl; F, always.

Note 2: n is the register bit number.

PART 6 – SUMMARY OF MACHINE CODES (Continued)

MACHINE CODE BYTE 1	2	3	4	5	6	ASSEMBLY LANGUAGE INSTRUCTION	ADDRESSING MODE	OPERATION	CONDITION
75	ds					bd=bd+logical(as)	16-Bit Register	Logical Add Register	N
75	dF	N				bd=bd+logical(N)	16-Bit Register	Logical Add Immediate	N
75	Fs					sp=sp+logical(as)	sp	Logical Add Register	N
75	FF	N				sp=sp+logical(N)	sp	Logical Add Immediate	N
78						halt( )		Halt	H
79	W(LO)	W(HI)				*W( )		Unconditional Call	N
7B	sn	N	See Note			if(bit(n,*bs++))goto*(pc+N)		Jump	N
7D	ds					bd=bd+as	16-Bit Register	Arithmetical Add Register	N
7D	dF	N				bd=bd+N	16-Bit Register	Arithmetical Add Immediate	N
7D	Fs					sp=sp+as	sp	Arithmetical Add Register	N
7D	FF	N				sp=sp+N	sp	Arithmetical Add Immediate	N
7F						nop( )		No Operation	N
80	ds					ad=as	Register and Register	Move	F
80	dF	N				ad=N	Register and Immediate	Move	F
81	ds					*bd=as	Indirect and Register	Move	F
81	dF	N				*bd=N	Indirect and Immediate	Move	F
81	Fs	W(LO)	W(HI)			*W=as	Direct and Register	Move	F
81	FF	W(LO)	W(HI)	N		*W=N	Direct and Immediate	Move	F
82	ds	N				*(bd+N)=as	Offset Memory and Register	Move	F
82	dF	N	M			*(bd+N)=M	Offset Memory and Immediate	Move	F
82	Fs	N				*(sp+N)=as	Offset Stack and Register	Move	F
82	FF	N	M			*(sp+N)=M	Offset Stack and Immediate	Move	F
83	ds					*bd++=as	Automatic Increment and Register	Move	F
83	dF	N				*bd++=N	Automatic Increment and Immediate	Move	F
85	ds					ad=*bs	Register and Indirect	Move	F
85	dF	W(LO)	W(HI)			ad=*W	Register and Direct	Move	F
86	ds	N				ad=*(bs+N)	Register and Offset Memory	Move	F
86	dF	N				ad=*(sp+N)	Register and Offset Stack	Move	F
87	ds					ad=*bs++	Register and Automatic Increment	Move	F
88	ds					ad=ad Λ as	Register and Register	Exclusive OR	F
88	dF	N				ad=ad Λ N	Register and Immediate	Exclusive OR	F
89	ds					*bd=*bd Λ as	Indirect and Register	Exclusive OR	F
89	dF	N				*bd=*bd Λ N	Indirect and Immediate	Exclusive OR	F
89	Fs	W(LO)	W(HI)			*W=*W Λ as	Direct and Register	Exclusive OR	F
89	FF	W(LO)	W(HI)	N		*W=*W Λ N	Direct and Immediate	Exclusive OR	F
8A	ds	N				*(bd+N)=*(bd+N) Λ as	Offset Memory and Register	Exclusive OR	F
8A	dF	N	M			*(bd+N)=*(bd+N) Λ M	Offset Memory and Immediate	Exclusive OR	F
8A	Fs	N				*(sp+N)=*(sp+N) Λ as	Offset Stack and Register	Exclusive OR	F
8A	FF	N	M			*(sp+N)=*(sp+N) Λ M	Offset Stack and Immediate	Exclusive OR	F
8B	ds					*bd++=*bd++ Λ as	Automatic Increment and Register	Exclusive OR	F
8B	dF	N				*bd++=*bd++ Λ N	Automatic Increment and Immediate	Exclusive OR	F
8D	ds					ad=ad Λ *bs	Register and Indirect	Exclusive OR	F
8D	dF	W(LO)	W(HI)			ad=ad Λ *W	Register and Direct	Exclusive OR	F
8E	ds	N				ad=ad Λ *(bs+N)	Register and Offset Memory	Exclusive OR	F
8E	dF	N				ad=ad Λ *(sp+N)	Register and Offset Stack	Exclusive OR	F
8F	ds					ad=ad Λ *bs++	Register and Automatic Increment	Exclusive OR	F
90	ds					ad=ad | as	Register and Register	OR	F
90	dF	N				ad=ad | N	Register and Immediate	OR	F

Note: n is the register bit number.

MACHINE CODE BYTE 1	2	3	4	5	6	ASSEMBLY LANGUAGE INSTRUCTION	ADDRESSING MODE	OPERATION	CONDITION
91	ds					*bd=*bd|as	Indirect and Register	OR	F
91	dF	N				*bd=*bd|N	Indirect and Immediate	OR	F
91	Fs	W(LO)	W(HI)			*W=*W|as	Direct and Register	OR	F
91	FF	W(LO)	W(HI)	N		*W=*W|N	Direct and Immediate	OR	F
92	ds	N				*(bd+N)=*(bd+N)|as	Offset Memory and Register	OR	F
92	dF	N	M			*(bd+N)=*(bd+N)|M	Offset Memory and Immediate	OR	F
92	Fs	N				*(sp+N)=*(sp+N)|as	Offset Stack and Register	OR	F
92	FF	N	M			*(sp+N)=*(sp+N)|M	Offset Stack and Immediate	OR	F
93	ds					*bd++=*bd++|as	Automatic Increment and Register	OR	F
93	dF	N				*bd++=*bd++|N	Automatic Increment and Immediate	OR	F
95	ds					ad=ad|*bs	Register and Indirect	OR	F
95	dF	W(LO)	W(HI)			ad=ad|*W	Register and Direct	OR	F
96	ds	N				ad=ad|*(bs+N)	Register and Offset Memory	OR	F
96	dF	N				ad=ad|*(sp+N)	Register and Offset Stack	OR	F
97	ds					ad=ad|*bs++	Register and Automatic Increment	OR	F
98	ds					ad=ad&as	Register and Register	AND	F
98	dF	N				ad=ad&N	Register and Immediate	AND	F
99	ds					*bd=*bd&as	Indirect and Register	AND	F
99	dF	N				*bd=*bd&N	Indirect and Immediate	AND	F
99	Fs	W(LO)	W(HI)			*W=*W&as	Direct and Register	AND	F
99	FF	W(LO)	W(HI)	N		*W=*W&N	Direct and Immediate	AND	F
9A	ds	N				*(bd+N)=*(bd+N)&as	Offset Memory and Register	AND	F
9A	dF	N	M			*(bd+N)=*(bd+N)&M	Offset Memory and Immediate	AND	F
9A	Fs	N				*(sp+N)=*(sp+N)&as	Offset Stack and Register	AND	F
9A	FF	N	M			*(sp+N)=*(sp+N)&M	Offset Stack and Immediate	AND	F
9B	ds					*bd++=*bd++&as	Automatic Increment and Register	AND	F
9B	dF	N				*bd++=*bd++&N	Automatic Increment and Immediate	AND	F
9D	ds					ad=ad&*bs	Register and Indirect	AND	F
9D	dF	W(LO)	W(HI)			ad=ad&*W	Register and Direct	AND	F
9E	ds	N				ad=ad&*(bs+N)	Register and Offset Memory	AND	F
9E	dF	N				ad=ad&*(sp+N)	Register and Offset Stack	AND	F
9F	ds					ad=ad&*bs++	Register and Automatic Increment	AND	F
A0	ds					ad=ad–as	Register and Register	Subtract	A
A0	dF	N				ad=ad–N	Register and Immediate	Subtract	A
A1	ds					*bd=*bd–as	Indirect and Register	Subtract	A
A1	dF	N				*bd=*bd–N	Indirect and Immediate	Subtract	A
A1	Fs					*W=*W–as	Direct and Register	Subtract	A
A1	FF	W(LO)	W(HI)	N		*W=*W–N	Direct and Immediate	Subtract	A
A2	ds	N				*(bd+N)=*(bd+N)–as	Offset Memory and Register	Subtract	A
A2	dF	N	M			*(bd+N)=*(bd+N)–M	Offset Memory and Immediate	Subtract	A
A2	Fs	N				*(sp+N)=*(sp+N)–as	Offset Stack and Register	Subtract	A
A2	FF	N	M			*(sp+N)=*(sp+N)–M	Offset Stack and Immediate	Subtract	A
A3	ds					*bd++=*bd++–as	Automatic Increment and Register	Subtract	A
A3	dF	N				*bd++=*bd++–N	Automatic Increment and Immediate	Subtract	A
A5	ds					ad=ad–*bs	Register and Indirect	Subtract	A
A5	dF	W(LO)	W(HI)			ad=ad–*W	Register and Direct	Subtract	A
A6	ds	N				ad=ad–*(bs+N)	Register and Offset Memory	Subtract	A
A6	dF	N				ad=ad–*(sp+N)	Register and Offset Stack	Subtract	A
A7	ds					ad=ad–*bs++	Register and Automatic Increment	Subtract	A

PART 6 – SUMMARY OF MACHINE CODES (Continued)

MACHINE CODE BYTE 1	2	3	4	5	6	ASSEMBLY LANGUAGE INSTRUCTION	ADDRESSING MODE	OPERATION	CONDITION
A8	ds					ad=ad+as	Register and Register	Add	A
A8	dF	N				ad=ad+N	Register and Immediate	Add	A
A9	ds					*bd=*bd+as	Indirect and Register	Add	A
A9	dF	N				*bd=*bd+N	Indirect and Immediate	Add	A
A9	Fs	W(LO)	W(HI)			*W=*W+as	Direct and Register	Add	A
A9	FF	W(LO)	W(HI)	N		*W=*W+N	Direct and Immediate	Add	A
AA	ds	N				*(bd+N)=*(bd+N)+as	Offset Memory and Register	Add	A
AA	dF	N	M			*(bd+N)=*(bd+N)+M	Offset Memory and Immediate	Add	A
AA	Fs	N				*(sp+N)=*(sp+N)+as	Offset Stack and Register	Add	A
AA	FF	N	M			*(sp+N)=*(sp+N)+M	Offset Stack and Immediate	Add	A
AB	ds					*bd++=*bd++ +as	Automatic Increment and Register	Add	A
AB	dF	N				*bd++=*bd++ +N	Automatic Increment and Immediate	Add	A
AD	ds					ad=ad+*bs	Register and Indirect	Add	A
AD	dF	W(LO)	W(HI)			ad=ad+*W	Register and Direct	Add	A
AE	ds	N				ad=ad+*(bs+N)	Register and Offset Memory	Add	A
AE	dF	N				ad=ad+*(sp+N)	Register and Offset Stack	Add	A
AF	ds					ad=ad+*bs++	Register and Automatic Increment	Add	A
B0	ds					ad–as	Register and Register	Compare	A
B0	dF	N				ad–N	Register and Immediate	Compare	A
B1	ds					*bd–as	Indirect and Register	Compare	A
B1	dF	N				*bd–N	Indirect and Immediate	Compare	A
B1	Fs	W(LO)	W(HI)			*W–as	Direct and Register	Compare	A
B1	FF	W(LO)	W(HI)	N		*W–N	Direct and Immediate	Compare	A
B2	ds	N				*(bd+N)–as	Offset Memory and Register	Compare	A
B2	dF	N	M			*(bd+N)–M	Offset Memory and Immediate	Compare	A
B2	Fs	N				*(sp+N)–as	Offset Stack and Register	Compare	A
B2	FF	N	M			*(sp+N)–M	Offset Stack and Immediate	Compare	A
B3	ds					*bd++–as	Automatic Increment and Register	Compare	A
B3	dF	N				*bd++–N	Automatic Increment and Immediate	Compare	A
B5	ds					ad–*bs	Register and Direct	Compare	A
B5	dF	W(LO)	W(HI)			ad–*W	Register and Direct	Compare	A
B6	ds	N				ad–*(bs+N)	Register and Offset Memory	Compare	A
B6	dF	N				ad–*(sp+N)	Register and Offset Stack	Compare	A
B7	ds					ad–*bs++	Register and Automatic Increment	Compare	A
B8	ds					test(ad,as)	Register and Register	Test	F
B8	dF	N				test(ad,N)	Register and Immediate	Test	F
B9	ds					test(*bd,as)	Indirect and Register	Test	F
B9	dF	N				test(*bd,N)	Indirect and Immediate	Test	F
B9	Fs	W(LO)	W(HI)			test(*W,as)	Direct and Register	Test	F
B9	FF	W(LO)	W(HI)	N		test(*W,N)	Direct and Immediate	Test	F
BA	ds	N				test(*(bd+N),as)	Offset Memory and Register	Test	F
BA	dF	N	M			test(*(bd,N),M)	Offset Memory and Immediate	Test	F
BA	Fs	N				test(*(sp+N),as)	Offset Stack and Register	Test	F
BA	FF	N	M			test(*(sp+N),M)	Offset Stack and Immediate	Test	F
BB	ds					test(*bd++,as)	Automatic Increment and Register	Test	F
BB	dF	N				test(*bd++,N)	Automatic Increment and Immediate	Test	F
BD	ds					test(ad,*bs)	Register and Indirect	Test	F
BD	dF	W(LO)	W(HI)			test(ad,*W)	Register and Direct	Test	F
BE	ds	N				test(ad,*(bs+N))	Register and Offset Memory	Test	F
BE	dF	N				test(ad,*(sp+N))	Register and Offset Stack	Test	F

MACHINE CODE BYTE 1	2	3	4	5	6	ASSEMBLY LANGUAGE INSTRUCTIONS	ADDRESSING MODE	OPERATION	CONDITION
BF	ds					test(ad,*bs++)	Register and Automatic Increment	Test	F
C0	ds					bd=bs	16-Bit Register and 16-Bit Register	Move	F
C0	dF	W(LO)	W(HI)			bd=W	16-Bit Register and 16-Bit Immediate	Move	F
C1	ds					*dd=bs	Indirect and 16-Bit Register	Move	F
C1	dF	W(LO)	W(HI)			*dd=W	Indirect and 16-Bit Immediate	Move	F
C1	Fs	W(LO)	W(HI)			*W=bs	Direct and 16-Bit Register	Move	F
C1	FF	W(LO)	W(HI)	V(LO)	V(HI)	*W=V	Direct and 16-Bit Immediate	Move	F
C2	ds	N				*(dd+N)=bs	Offset Memory and 16-Bit Register	Move	F
C2	dF	N	W(LO)	W(HI)		*(dd+N)=W	Offset Memory and 16-Bit Immediate	Move	F
C2	Fs	N				*(dsp+N)=bs	Offset Stack and 16-Bit Register	Move	F
C2	FF	N	W(LO)	W(HI)		*(dsp+N)=W	Offset Stack and 16-Bit Immediate	Move	F
C3	ds					*dd++=bs	Automatic Increment and 16-Bit Register	Move	F
C3	dF	W(LO)	W(HI)			*dd++=W	Automatic Increment and 16-Bit Immediate	Move	F
C4	ds	M	N			*(bd+N)=*(bs+M)	Offset Memory and Offset Memory	Move	F
C4	dF	M	N			*(bd+N)=*(sp+M)	Offset Memory and Offset Stack	Move	F
C4	Fs	M	N			*(sp+N)=*(bs+M)	Offset Stack and Offset Memory	Move	F
C4	FF	M	N			*(sp+N)=*(sp+M)	Offset Stack and Offset Stack	Move	F
C5	ds					bd=*ds	16-Bit Register and Indirect	Move	F
C5	dF	W(LO)	W(HI)			bd=*W	16-Bit Register and Direct	Move	F
C6	ds	N				bd=*(ds+N)	16-Bit Register and Offset Memory	Move	F
C6	dF	N				bd=*(dsp+N)	16-Bit Register and Offset Stack	Move	F
C7	ds					bd=*ds++	16-Bit Register and Automatic Increment	Move	F
C8	ds					bd=bd Λ bs	16-Bit Register and 16-Bit Register	Exclusive OR	F
C8	dF	W(LO)	W(HI)			bd=bd Λ W	16-Bit Register and 16-Bit Immediate	Exclusive OR	F
C9	ds					*dd=*dd Λ bs	Indirect and 16-Bit Register	Exclusive OR	F
C9	dF	W(LO)	W(HI)			*dd=*dd Λ W	Indirect and 16-Bit Immediate	Exclusive OR	F
C9	Fs	W(LO)	W(HI)			*W=*W Λ bs	Direct and 16-Bit Register	Exclusive OR	F
C9	FF	W(LO)	W(HI)	V(LO)	V(HI)	*W=*W Λ V	Direct and 16-Bit Immediate	Exclusive OR	F
CA	ds	N				*(dd+N)=*(dd+N) Λ bs	Offset Memory and 16-Bit Register	Exclusive OR	F
CA	dF	N	W(LO)	W(HI)		*(dd+N)=*(dd+N) Λ W	Offset Memory and 16-Bit Immediate	Exclusive OR	F
CA	Fs	N				*(dsp+N)=*(dsp+N) Λ bs	Offset Stack and 16-Bit Register	Exclusive OR	F
CA	FF	N	W(LO)	W(HI)		*(dsp+N)=*(dsp+N) Λ W	Offset Stack and 16-Bit Immediate	Exclusive OR	F
CB	ds					*dd++=*dd++ Λ bs	Automatic Increment and 16-Bit Register	Exclusive OR	F
CB	dF	W(LO)	W(HI)			*dd++=*dd++ Λ W	Automatic Increment and 16-Bit Immediate	Exclusive OR	F
CC	ds	M	N			*(bd+N)=*(bd+N) Λ *(bs+M)	Offset Memory and Offset Memory	Exclusive OR	F
CC	dF	M	N			*(bd+N)=*(bd+N) Λ *(sp+M)	Offset Memory and Offset Stack	Exclusive OR	F
CC	Fs	M	N			*(sp+N)=*(sp+N) Λ *(bs+M)	Offset Stack and Offset Memory	Exclusive OR	F
CC	FF	M	N			*(sp+N)=*(sp+N) Λ *(sp+M)	Offset Stack and Offset Stack	Exclusive OR	F
CD	ds					bd=bd Λ *ds	16-Bit Register and Indirect	Exclusive OR	F
CD	dF	W(LO)	W(HI)			bd–bd Λ *W	16-Bit Register and Direct	Exclusive OR	F
CE	ds	N				bd=bd Λ *(ds+N)	16-Bit Register and Offset Memory	Exclusive OR	F
CE	dF	N				bd=bd Λ *(dsp+N)	16-Bit Register and Offset Stack	Exclusive OR	F
CF	ds					bd=bd Λ *ds++	16-Bit Register and Automatic Increment	Exclusive OR	F
D0	ds					bd=bd | bs	16-Bit Register and 16-Bit Register	OR	F
D0	dF	W(LO)	W(HI)			bd=bd | W	16-Bit Register and 16-Bit Immediate	OR	F
D1	ds					*dd=*dd | bs	Indirect and 16-Bit Register	OR	F
D1	dF	W(LO)	W(HI)			*dd=*dd | W	Indirect and 16-Bit Immediate	OR	F
D1	Fs	W(LO)	W(HI)			*W=*W | bs	Direct and 16-Bit Register	OR	F
D1	FF	W(LO)	W(HI)	V(LO)	V(HI)	*W=*W | V	Direct and 16-Bit Immediate	OR	F

PART 6 – SUMMARY OF MACHINE CODES (Continued)

<table>
<tr><th colspan="6">MACHINE CODE BYTE</th><th rowspan="2">ASSEMBLY LANGUAGE INSTRUCTIONS</th><th rowspan="2">ADDRESSING MODE</th><th rowspan="2">OPERATION</th><th rowspan="2">CONDITION</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th></tr>
<tr><td>D2</td><td>ds</td><td>N</td><td></td><td></td><td></td><td>*(dd+N)=*(dd+N) | bs</td><td>Offset Memory and 16-Bit Register</td><td>OR</td><td>F</td></tr>
<tr><td>D2</td><td>dF</td><td>N</td><td>W(LO)</td><td>W(HI)</td><td></td><td>*(dd+N)=*(dd+N |W</td><td>Offset Memory and 16-Bit Immediate</td><td>OR</td><td>F</td></tr>
<tr><td>D2</td><td>Fs</td><td>N</td><td></td><td></td><td></td><td>*(dsp+N)=*(dsp+N) |bs</td><td>Offset Stack and 16-Bit Register</td><td>OR</td><td>F</td></tr>
<tr><td>D2</td><td>FF</td><td>N</td><td>W(LO)</td><td>W(HI)</td><td></td><td>*(dsp+N)=*(dsp+N) | W</td><td>Offset Stack and 16-Bit Immediate</td><td>OR</td><td>F</td></tr>
<tr><td>D3</td><td>ds</td><td></td><td></td><td></td><td></td><td>*dd++=*dd++ | bs</td><td>Automatic Increment and 16-Bit Register</td><td>OR</td><td>F</td></tr>
<tr><td>D3</td><td>dF</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>*dd++=*dd++ | W</td><td>Automatic Increment and 16-Bit Immediate</td><td>OR</td><td>F</td></tr>
<tr><td>D4</td><td>ds</td><td>M</td><td>N</td><td></td><td></td><td>*(bd+N)=*(bd+N) | *(bs+M)</td><td>Offset Memory and Offset Memory</td><td>OR</td><td>F</td></tr>
<tr><td>D4</td><td>dF</td><td>M</td><td>N</td><td></td><td></td><td>*(bd+N)=*(bd+N) | *(sp+M)</td><td>Offset Memory and Offset Stack</td><td>OR</td><td>F</td></tr>
<tr><td>D4</td><td>Fs</td><td>M</td><td>N</td><td></td><td></td><td>*(sp+N)=*(sp+N) | *(bs+M)</td><td>Offset Stack and Offset Memory</td><td>OR</td><td>F</td></tr>
<tr><td>D4</td><td>FF</td><td>M</td><td>N</td><td></td><td></td><td>*(sp+N)=*(sp+N) | *(sp+M)</td><td>Offset Stack and Offset Stack</td><td>OR</td><td>F</td></tr>
<tr><td>D5</td><td>ds</td><td></td><td></td><td></td><td></td><td>bd=bd | *ds</td><td>16-Bit Register and Indirect</td><td>OR</td><td>F</td></tr>
<tr><td>D5</td><td>dF</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>bd=bd | *W</td><td>16-Bit Register and Direct</td><td>OR</td><td>F</td></tr>
<tr><td>D6</td><td>ds</td><td>N</td><td></td><td></td><td></td><td>bd=bd | *(ds+N)</td><td>16-Bit Register and Offset Memory</td><td>OR</td><td>F</td></tr>
<tr><td>D6</td><td>dF</td><td>N</td><td></td><td></td><td></td><td>bd=bd | *(dsp+N)</td><td>16-Bit Register and Offset Stack</td><td>OR</td><td>F</td></tr>
<tr><td>D7</td><td>ds</td><td></td><td></td><td></td><td></td><td>bd=bd | *ds++</td><td>16-Bit Register and Automatic Increment</td><td>OR</td><td>F</td></tr>
<tr><td>D8</td><td>ds</td><td></td><td></td><td></td><td></td><td>bd=bd&bs</td><td>16-Bit Register and 16-Bit Register</td><td>AND</td><td>F</td></tr>
<tr><td>D8</td><td>dF</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>bd=bd&W</td><td>16-Bit Register and 16-Bit Immediate</td><td>AND</td><td>F</td></tr>
<tr><td>D9</td><td>ds</td><td></td><td></td><td></td><td></td><td>*dd=*dd&bs</td><td>Indirect and 16-Bit Register</td><td>AND</td><td>F</td></tr>
<tr><td>D9</td><td>dF</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>*dd=*dd&W</td><td>Indirect and 16-Bit Immediate</td><td>AND</td><td>F</td></tr>
<tr><td>D9</td><td>Fs</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>*W=*W&bs</td><td>Direct and 16-Bit Register</td><td>AND</td><td>F</td></tr>
<tr><td>D9</td><td>FF</td><td>W(LO)</td><td>W(HI)</td><td>V(LO)</td><td>V(HI)</td><td>*W=*W&V</td><td>Direct and 16-Bit Immediate</td><td>AND</td><td>F</td></tr>
<tr><td>DA</td><td>ds</td><td>N</td><td></td><td></td><td></td><td>*(dd+N)=*(dd+N)&bs</td><td>Offset Memory and 16-Bit Register</td><td>AND</td><td>F</td></tr>
<tr><td>DA</td><td>dF</td><td>N</td><td>W(LO)</td><td>W(HI)</td><td></td><td>*(dd+N)=*(dd+N)&W</td><td>Offset Memory and 16-Bit Immediate</td><td>AND</td><td>F</td></tr>
<tr><td>DA</td><td>Fs</td><td>N</td><td></td><td></td><td></td><td>*(dsp+N)=*(dsp+N)&bs</td><td>Offset Stack and 16-Bit Register</td><td>AND</td><td>F</td></tr>
<tr><td>DA</td><td>FF</td><td>N</td><td>W(LO)</td><td>W(HI)</td><td></td><td>*(dsp+N)=*(dsp+N)&W</td><td>Offset Stack and 16-Bit Immediate</td><td>AND</td><td>F</td></tr>
<tr><td>DB</td><td>ds</td><td></td><td></td><td></td><td></td><td>*dd++=*dd++&bs</td><td>Automatic Increment and 16-Bit Register</td><td>AND</td><td>F</td></tr>
<tr><td>DB</td><td>dF</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>*dd++=*dd++&W</td><td>Automatic Increment and 16-Bit Immediate</td><td>AND</td><td>F</td></tr>
<tr><td>DC</td><td>ds</td><td>M</td><td>N</td><td></td><td></td><td>*(bd+N)=*(bd+N)&*(bs+M)</td><td>Offset Memory and Offset Memory</td><td>AND</td><td>F</td></tr>
<tr><td>DC</td><td>dF</td><td>M</td><td>N</td><td></td><td></td><td>*(bd+N)=*(bd+N)&*(sp+M)</td><td>Offset Memory and Offset Stack</td><td>AND</td><td>F</td></tr>
<tr><td>DC</td><td>Fs</td><td>M</td><td>N</td><td></td><td></td><td>*(sp+N)=*(sp+N)&*(bs+M)</td><td>Offset Stack and Offset Memory</td><td>AND</td><td>F</td></tr>
<tr><td>DC</td><td>FF</td><td>M</td><td>N</td><td></td><td></td><td>*(sp+N)=*(sp+N)&*(sp+M)</td><td>Offset Stack and Offset Stack</td><td>AND</td><td>F</td></tr>
<tr><td>DD</td><td>ds</td><td></td><td></td><td></td><td></td><td>bd=bd&*ds</td><td>16-Bit Register and Indirect</td><td>AND</td><td>F</td></tr>
<tr><td>DD</td><td>dF</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>bd=bd&*W</td><td>16-Bit Register and Direct</td><td>AND</td><td>F</td></tr>
<tr><td>DE</td><td>ds</td><td>N</td><td></td><td></td><td></td><td>bd=bd&*(ds+N)</td><td>16-Bit Register and Offset Memory</td><td>AND</td><td>F</td></tr>
<tr><td>DE</td><td>dF</td><td>N</td><td></td><td></td><td></td><td>bd=bd&*(dsp+N)</td><td>16-Bit Register and Offset Stack</td><td>AND</td><td>F</td></tr>
<tr><td>DF</td><td>ds</td><td></td><td></td><td></td><td></td><td>bd=bd&*ds++</td><td>16-Bit Register and Automatic Increment</td><td>AND</td><td>F</td></tr>
<tr><td>E0</td><td>ds</td><td></td><td></td><td></td><td></td><td>bd=bd–bs</td><td>16-Bit Register and 16-Bit Register</td><td>Subtract</td><td>A</td></tr>
<tr><td>E0</td><td>dF</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>bd=bd–W</td><td>16-Bit Register and 16-Bit Immediate</td><td>Subtract</td><td>A</td></tr>
<tr><td>E1</td><td>ds</td><td></td><td></td><td></td><td></td><td>*dd=*dd–bs</td><td>Indirect and 16-Bit Register</td><td>Subtract</td><td>A</td></tr>
<tr><td>E1</td><td>dF</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>*dd=*dd–W</td><td>Indirect and 16-Bit Immediate</td><td>Subtract</td><td>A</td></tr>
<tr><td>E1</td><td>Fs</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>*W=*W–bs</td><td>Direct and 16-Bit Register</td><td>Subtract</td><td>A</td></tr>
<tr><td>E1</td><td>FF</td><td>W(LO)</td><td>W(HI)</td><td>V(LO)</td><td>V(HI)</td><td>*W=*W–V</td><td>Direct and 16-Bit Immediate</td><td>Subtract</td><td>A</td></tr>
<tr><td>E2</td><td>ds</td><td>N</td><td></td><td></td><td></td><td>*(dd+N)=*(dd+N)–bs</td><td>Offset Memory and 16-Bit Register</td><td>Subtract</td><td>A</td></tr>
<tr><td>E2</td><td>dF</td><td>N</td><td>W(LO)</td><td>W(HI)</td><td></td><td>*(dd+N)=*(dd+N)–W</td><td>Offset Memory and 16-Bit Immediate</td><td>Subtract</td><td>A</td></tr>
<tr><td>E2</td><td>Fs</td><td>N</td><td></td><td></td><td></td><td>*(dsp+N)=*(dsp+N)–bs</td><td>Offset Stack and 16-Bit Register</td><td>Subtract</td><td>A</td></tr>
<tr><td>E2</td><td>FF</td><td>N</td><td>W(LO)</td><td>W(HI)</td><td></td><td>*(dsp+N)=*(dsp+N)–W</td><td>Offset Stack and 16-Bit Immediate</td><td>Subtract</td><td>A</td></tr>
<tr><td>E3</td><td>ds</td><td></td><td></td><td></td><td></td><td>*dd++=*dd++–bs</td><td>Automatic Increment and 16-Bit Register</td><td>Subtract</td><td>A</td></tr>
<tr><td>E3</td><td>dF</td><td>W(LO)</td><td>W(HI)</td><td></td><td></td><td>*dd++=*dd++–W</td><td>Automatic Increment and 16-Bit Immediate</td><td>Subtract</td><td>A</td></tr>
</table>

MACHINE CODE BYTE 1	2	3	4	5	6	ASSEMBLY LANGUAGE INSTRUCTIONS	ADDRESSING MODE	OPERATION	CONDITION
E4	ds	M	N			*(bd+N)=*(bd+N)–*(bs+M)	Offset Memory and Offset Memory	Subtract	A
E4	dF	M	N			*(bd+N)=*(bd+N)–*(sp+M)	Offset Memory and Offset Stack	Subtract	A
E4	Fs	M	N			*(sp+N)=*(sp+N)–*(bs+M)	Offset Stack and Offset Memory	Subtract	A
E4	FF	M	N			*(sp+N)=*(sp+N)–*(sp+M)	Offset Stack and Offset Stack	Subtract	A
E5	ds					bd=bd–*ds	16-Bit Register and Indirect	Subtract	A
E5	dF	W(LO)	W(HI)			bd=bd–*W	16-Bit Register and Direct	Subtract	A
E6	ds	N				bd=bd–*(ds+N)	16-Bit Register and Offset Memory	Subtract	A
E6	dF	N				bd=bd–*(dsp+N)	16-Bit Register and Offset Stack	Subtract	A
E7	ds					bd=bd–*ds++	16-Bit Register and Automatic Increment	Subtract	A
E8	ds					bd=bd+bs	16-Bit Register and 16-Bit Register	Add	A
E8	dF	W(LO)	W(HI)			bd=bd+W	16-Bit Register and 16-Bit Immediate	Add	A
E9	ds					*dd=*dd+bs	Indirect and 16-Bit Register	Add	A
E9	dF	W(LO)	W(HI)			*dd=*dd+W	Indirect and 16-Bit Immediate	Add	A
E9	Fs	W(LO)	W(HI)			*W=*W+bs	Direct and 16-Bit Register	Add	A
E9	FF	W(LO)	W(HI)	V(LO)	V(HI)	*W=*W+V	Direct and 16-Bit Immediate	Add	A
EA	ds	N				*(dd+N)=*(dd+N)+bs	Offset Memory and 16-Bit Register	Add	A
EA	dF	N	W(LO)	W(HI)		*(dd+N)=*(dd+N)+W	Offset Memory and 16-Bit Immediate	Add	A
EA	Fs	N				*(dsp+N)=*(dsp+N)+bs	Offset Stack and 16-Bit Register	Add	A
EA	FF	N	W(LO)	W(HI)		*(dsp+N)=*(dsp+N)+W	Offset Stack and 16-Bit Immediate	Add	A
EB	ds					*dd++=*dd++ +bs	Automatic Increment and 16-Bit Register	Add	A
EB	dF	W(LO)	W(HI)			*dd++=*dd++ +W	Automatic Increment and 16-Bit Immediate	Add	A
EC	ds	M	N			*(bd+N)=*(bd+N)+*(bs+M)	Offset Memory and Offset Memory	Add	A
EC	dF	M	N			*(bd+N)=*(bd+N)+*(sp+M)	Offset Memory and Offset Stack	Add	A
EC	Fs	M	N			*(sp+N)=*(sp+N)+*(bs+M)	Offset Stack and Offset Memory	Add	A
EC	FF	M	N			*(sp+N)=*(sp+N)+*(sp+M)	Offset Stack and Offset Stack	Add	A
ED	ds					bd=bd+*ds	16-Bit Register and Indirect	Add	A
ED	dF	W(LO)	W(HI)			bd=bd+*W	16-Bit Register and Direct	Add	A
EE	ds	N				bd=bd+*(ds+N)	16-Bit Register and Offset Memory	Add	A
EE	dF	N				bd=bd+*(dsp+N)	16-Bit Register and Offset Stack	Add	A
EF	ds					bd=bd+*ds++	6-Bit Register and Automatic Increment	Add	A
F0	ds					bd–bs	16-Bit Register and 16-Bit Register	Compare	A
F0	dF	W(LO)	W(HI)			bd–W	16-Bit Register and 16-Bit Immediate	Compare	A
F1	ds					*dd–bs	Indirect and 16-Bit Register	Compare	A
F1	dF	W(LO)	W(HI)			*dd–W	Indirect and 16-Bit Immediate	Compare	A
F1	Fs	W(LO)	W(HI)			*W–bs	Direct and 16-Bit Register	Compare	A
F1	FF	W(LO)	W(HI)	V(LO)	V(HI)	*W–V	Direct and 16-Bit Immediate	Compare	A
F2	ds	N				*(dd+N)–bs	Offset Memory and 16-Bit Register	Compare	A
F2	dF	N	W(LO)	W(HI)		*(dd+N)–W	Offset Memory and 16-Bit Immediate	Compare	A
F2	Fs	N				*(dsp+N)–bs	Offset Stack and 16-Bit Register	Compare	A
F2	FF	N	W(LO)	W(HI)		*(dsp+N)–W	Offset Stack and 16-Bit Immediate	Compare	A
F3	ds					*dd++–bs	Automatic Increment and 16-Bit Register	Compare	A
F3	dF	W(LO)	W(HI)			*dd++–W	Automatic Increment and 16-Bit Immediate	Compare	A
F4	ds	M	N			*(bd+N)–*(bs+M)	Offset Memory and Offset Memory	Compare	A
F4	dF	M	N			*(bd+N)–*(sp+M)	Offset Memory and Offset Stack	Compare	A
F4	Fs	M	N			*(sp+N)=*(bs+M)	Offset Stack and Offset Memory	Compare	A
F4	FF	M	N			*(sp+N)=*(sp+M)	Offset Stack and Offset Stack	Compare	A
F5	ds					bd–*ds	16-Bit Register and Indirect	Compare	A
F5	dF	W(LO)	W(HI)			bd=*W	16-Bit Register and Direct	Compare	A

PART 6 — SUMMARY OF MACHINE CODES (Continued)

MACHINE CODE BYTE 1	2	3	4	5	6	ASSEMBLY LANGUAGE INSTRUCTIONS	ADDRESSING MODE	OPERATION	CONDITION
F6	ds	N				bd–*(ds+N)	16-Bit Register and Offset Memory	Compare	A
F6	dF	N				bd–*(dsp+N)	16-Bit Register and Offset Stack	Compare	A
F7	ds					bd–*ds++	16-Bit Register and Automatic Increment	Compare	A
FC	ds	M	N			test(*(bd+N),*(bs+M))	Offset Memory and Offset Memory	Test	F
FC	dF	M	N			test(*(bd+N),*(sp+M))	Offset Memory and Offset Stack	Test	F
FC	Fs	M	N			test(*(sp+N),*(bs+M))	Offset Stack and Offset Memory	Test	F
FC	FF	M	N			test(*(sp+N),*(sp+M))	Offset Stack and Offset Stack	Test	F

PA-800517
Issue 1
December 1978