Sauiers 26

PDP-II

ENGINEERING SPECIFICATIONS

NUMBER: ES:0001
DATE: 28 March 1968

BASIC PDP-11/30 DESCRIPTION
and
INSTRUCTION SET

Originated by: H. McFarland

Approved by:

H L McFarland
Architecture

Hardware

George W Thissell
Systems Software

Rich Merrill
Applications Software

Roger C Cady
Project Manager

This document is COMPANY CONFIDENTIAL and is for internal company use only. Copying is expressly prohibited.

ABSTRACT

Brief description of organization and concept of PDP-11 system and detailed discussion of the basic instruction set for the PDP-11/30 processor; includes basic Assembler Syntax.

Sample programs

Multiple precision (48 bit) integer arith (add & subtract)

Boot?

These specifications are for the instruction set implemented in the PDP-11/30 system. This initial system will contain a 4K word memory and have an average cycle time of 2.2us.

THESE SPECIFICATIONS ARE FOR INTERNAL DIGITAL USE ONLY AND ARE NOT TO BE RELEASED TO ANYONE OUTSIDE OF THE PDP-11 PROJECT WITHOUT APPROVAL OF PDP-11 ENGINEERING.

Future PDP-11 Engineering Specifications will deal with definition of expansion instructions for extended arithmetic, double precision, floating point, exec/user operations, memory relocation and protect, and other larger system features.

PDP-11 Engineering
March 26, 1969

PDP-11 DESCRIPTION AND INSTRUCTION SET

The PDP-11 is a two's-complement arithmetic, stored-program computer with a basic 16 bit instruction word. It is a two-address machine with a very powerful multi-accumulator/general register configuration, and is capable of doing memory-to-memory arithmetic operations. Most instructions can specify either byte or word data, enabling use as an 8- or 16-bit processor. When the byte mode is used, the accumulators operate as true 8-bit registers on two's-complement numbers. The memory is byte addressable with instructions that occupy 2 bytes.

The basic instruction organization for binary two address operations is

B	OP	SOURCE	DESTINATION
15	14 12	11 6	5 0

where: B indicates byte or word operation (except for multiply/ divide instructions).

OP designates the operation to be performed.

SOURCE defines the location of source data.

DESTINATION defines the location of the destination data

SOURCE and DESTINATION are configured identically.

M	D	REG	
11 10	9	8 6	SOURCE
5 4	3	2 0	DESTINATION

M is the mode	00	The register (bits 8-6 or 2-0) itself
	01	Autoincrement using the register specified
	10	Autodecrement using the register specified
	11	Register specified is indexed by the following word.

Does register Ø index?

D is the defer bit

REG is the register

000	RØ	general hardware register/accumulators
001	R1	
010	R2	
011	R3	
100	R4	
101	R5	
110	LP	Linkage Pointer
111	PC	Program Counter

There are three classes of instruction formats:

Single word:

B	OP	SOURCE	DESTINATION
15			0

Double word:

B	OP	SOURCE	DESTINATION
15			0

immediate address or data	
15	0

The second word can be used by either the source or destination field of the first word.

Triple word:

B	OP	SOURCE	DESTINATION
15			0

immediate source address or data	
15	0

immediate destination address or data	
15	0

The second word is used by the source field of the first word, the third word is used by the destination field of the first word.

In addition, there is a hardware register that contains a STATUS word

PRIORITY	I	N	Z	V	C	UNDEFINED
15 13	12	11	10	9	8	7 0

where: PRIORITY The operating program priority 000 - 111 which determines the major priority level of interrupts allowed.

I Reserved for INSTRUCTION set expansion (Not implemented in basic set.)

N A condition code indicating last result was NEGATIVE.

Z A condition code indicating the last result was ZERO.

V A condition code indicating that the last arithmetic operation had a true arithmetic OVERFLOW (a change in sign when quantities of like sign are added).

C A condition code indicating that the last arithmetic operation had a CARRY out from bit 15 (bit 7 if byte). Unchanged on unary instructions: Increment (INC) and Decrement (DEC).

The syntax for the PDP-11 Assembler is as follows (Numbers are octal unless specified otherwise):

#L designates a literal L (literal data is analogous to immediate instruction)

(#100 is the quantity 100)

A designates the absolute address of a location A

(A(100 is the memory location 100)

@ designates a level of deferral

(@100 is the contents of location 100)

. designates current location

(.+10 is the memory location 10 bytes beyond the present instruction)

'A designates a value which is the octal equivalent of an ASCII A

(#'A is the quantity 101)

"AB designates a value which is the octal equivalent of an ASCII pair AB.

(#"AB is the quantity 102 101 = 041102)

must ''''''' ABCDEFG ?

a ''' A△△

%m designates a register m, where m can be an expression that has a value in the range 0 – 7 (corresponding to registers RØ-R5, LP, PC).

(m)+ designates using register m (as defined above) in an autoincrement mode. (Increment is after use.)

-(m) designates using register m (as defined above) in an autodecrement mode. (Decrement is before use.) Why? For stacks.

A(m) designates using register m (as defined above) in an indexed mode (A is added to the register in computing address).

/ designates that a comment field follows; must be repeated on continuation lines.

= equates symbol to the value specified (K1=07123)

: defines the absolute address of a symbolically expressed location.

The symbology for description of the instruction operations is defined as:

(PC)+1 → (RØ) The contents of the register named PC are incremented and become the new contents of the register named RØ. The contents of PC do not change.

(PC)→((RØ)) The contents of the register named PC become the contents of the memory cell whose address is the contents of the register named RØ. The contents of PC and RØ do not change.

(PC)+n This is symbolic of the program counter pointing to the next instruction. If there is no immediate data or address PC is incremented by 2. If there is an immediate byte or word data, or word address, the PC is incremented by 4. If there are 2 immediate references, the PC is incremented by 6.

E	Effective Address
SE	Source Effective Address
DE	Destination Effective Address

Refer to Appendix B for a complete discussion of address modes.

Because of the nature of the possible stack operations in the PDP-11 instructions, stacks usually are built to expand towards zero; a push is an autodecrement and a pop is an autoincrement.

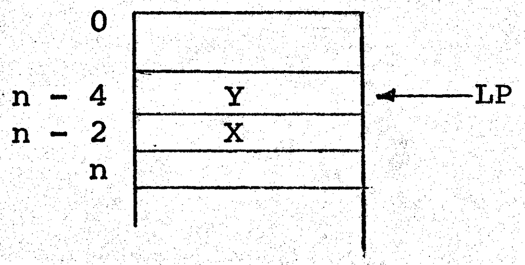
```
MOV  A,   -(RØ)       /Push A onto RØ stack

MOV  (RØ)+,  A        /Pop A from RØ stack
```

The linkage pointer stack (pointed to by LP) must be built to expand towards zero since the interrupt stacking process decrements LP during the push operation and increments LP during the pop operation.

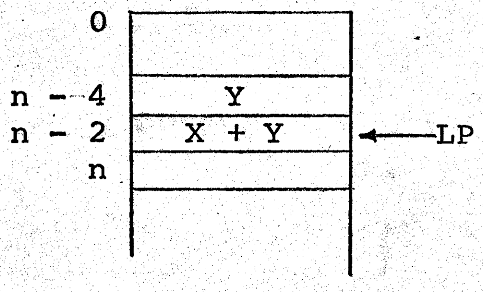
Note because of this:

```
1:  MOV  #X,   -(RØ)     /Push X onto stack

2:  MOV  #Y,   -(RØ)     /Push Y onto stack

3:  ADD  (RØ)+, @%RØ     /Pop last on stack and

                         /add to former next to last, leave

                         /result in top of

                         /stack
```

Stack After 2:



Stack After 3:



Both instruction and data words are constrained to even address boundaries. Bytes (except a byte immediate reference) may be on even or odd address boundaries. This is a hardware restriction. It will result in faster and uniform execution times.

The following assumptions are common to all instructions, unless otherwise noted:

1. The B bit determines that the operand is a 16-bit word (B = 0) or an 8-bit byte (B = 1). Condition codes are set accordingly.

2. Two's complement arithmetic.

3. Condition codes are unchanged.

BINARY GROUP

MOV

MOV		A	B
B	001	SOURCE	DESTINATION
15	14 12	11 6	5 0

(SE) → (DE)
(PC)+n → (PC)

Condition code operation same as in ADD, noted below. There is no carry or overflow, however, as a MOV is an ADD to zero.

not changed?

ADD

ADD		A	B
B	010	SOURCE	DESTINATION
15	14 12	11 6	5 0

(SE) + (DE) → (DE)
(PC)+n → PC

If result negative 1 → N, otherwise Ø → N
If zero 1 → Z, otherwise Ø → Z
If carry 1 → C, otherwise Ø → C.
If overflow 1 → V, otherwise Ø → V.

SUBTRACT

not transitive op.

SUB		A	B
B	011	SOURCE	DESTINATION
15	14 12	11 6	5 0

(DE) - (SE) → (DE)
(PC)+n → PC
Condition codes N, Z, and V operate as in ADD.
If carry 0 → C, otherwise 1 → C.

COMPARE

CMP		A	B
B	100	SOURCE	DESTINATION
15	14 12	11 6	5 0

(SE) - (DE), Set Condition Codes
(PC)+n → (PC)

Condition codes operate as in SUB.

AND

AND		A	B
B	100	SOURCE	DESTINATION
15	14 12	11 6	5 0

$$(SE) \cdot (DE) \longrightarrow (DE)$$
$$(PC)+n \longrightarrow (PC)$$

Condition codes operate as in ADD. No carry or overflow results.

MULTIPLY, DIVIDE, ETC. (OP Codes 1100 - 1111)

Not fully defined at this time; not implemented in basic machine.

<u>GOTO (Unconditional Jump)</u>

GOTO

0	001	0	SOURCE	ØØØ111
15	14 12	11	6	5 0

This is a special case of the MOV instruction (move source word to PC).

The source field is the location to which the jump is desired.

GOTO A ≡ MOV #A, %PC

This assembles as a two-word instruction unless A is held in a register:

if (RØ) = A

then GOTO @%RØ

assembles as MOV %RØ, %PC

Condition codes operate as in ADD.

BRANCH B--

X	0000	Y	T	OFFSET
15	14 11	10 9	8	7 0

Test condition and, if met, branch to a location which is from +127 to -128 words from location of Branch instruction. Condition codes are not changed.
If condition met: (PC) + OFFSET → (PC), where OFFSET is an 8-bit 2's complement quantity.
If condition not met: (PC) +2 → (PC)

X	Y	CONDITION*	MNEMONICS T=1	T=0	
0	01	$=,(Z\cdot\overline{V})$	BEQ	BNE	
0	10	$<,(N\oplus V)$	BLT	BGE	Arithmetic Results Tests
0	11	$\leq,(Z\cdot\overline{V}+N\oplus V)$	BLE	BGT	
1	ØØ	N	BNS	BNC	
1	Ø1	Z	BZS	BZC	
1	10	V	BVS	BVC	Condition Code Flag Tests
1	11	C	BCS	BCC	

* If T is set the branch takes place if stated condition is met.

UNARY GROUP

B	000101	OP	DESTINATION
15	14 9	8 6	5 0

Eight byte-word instructions which perform operations on the operand as designated by DESTINATION. Byte, word and DESTINATION operation same as noted before. Condition codes are set as noted.

CLEAR CLR A

OP = ØØØ

Ø → (DE)

(PC)+n → (PC)

Condition codes operate as in ADD.

COMPLEMENT COM A

OP = ØØ1

1's complement of (DE) → (DE)
(PC)+n → (PC)
Condition codes operate as in SUB.

INCREMENT INC A

OP = Ø1Ø

(DE) + 1 → (DE)
(PC)+n → (PC)
Condition codes Z, N, V operate as in ADD; C is not changed.

DECREMENT DEC A

OP = Ø11

(DE) - 1 → (DE)
(PC)+n → (PC)
Condition codes Z, N, V operate as in ADD; C is not changed.

NEGATE NEG A

OP = 1ØØ

2's complement of (DE) → (DE)
(PC)+n → (PC)
Condition codes operate as in SUB.

ADD CARRY ADC A

OP = 1Ø1

(DE) + (C) → (DE)
(PC)+n → (PC)
Condition codes operate as in ADD.

DECREMENT IF CARRY DIC A

OP = 11Ø

(DE) - (C) → (DE)
(PC)+n → (PC)
Condition codes operate as in SUB.

TEST TST A

OP = 111

(PC)+n → (PC)
Condition codes operate as in ADD.

ROTATE/SHIFT

B	000110	OP	DESTINATION
15	14 9	8 6	5 0

Rotates include carry to expand register to 9 or 17 bits.

Shift left fills bit Ø with Ø
Shift right fills left bits with sign (arithmetic shift)

Condition codes N and Z operate as in ADD.
Condition code C receives the bit shifted or rotated out.
Condition code V is set if C changes, cleared otherwise.

OP:		Operation	Mnemonics
OP:	Bit 8 = 0	Rotate	ROT
	8 = 1	Shift	SHF
	Bit 7 = 0	Right	+
	7 = 1	Left	-
	Bit 9 = 0	Once	1
	9 = 1	Multiple	(unimplemented in basic machine)

Really?

Example: ROT+1 (Rotate right once) = ØØØ
(DE) Rotated right once → (DE)
(PC)+n → (PC)

SUBROUTINE CALL JSR R A

0 000 100	REG	DESTINATION
15 9	8 6	5 0

JSR, Jump and Store Return

(LP)-2 → (LP)
(REG) → ((LP))
(PC)+n → (REG)
DE → (PC)

DE cannot be REG undeferred. This is illegal and will trap to location 0.

EXECUTE

EXC		A
1 000 100	0	ADDRESS

15 9 8 7 0

The instruction at ADDRESS of the low 256 memory words is executed. For all but subroutine linkage the effective PC becomes ADDRESS; for JSR the PC is that of the EXC instruction. Condition codes are a function of the executed instruction.

EXC BRANCH

TRAP

1 000 100	1	ADDRESS

15 9 8 7 0

Trap to ADDRESS of low 256 memory words is undefined and unimplemented in basic machine.

OPERATE GROUP

0 000 000	SG	OP	X

15 9 8 7 5 4 0

Combinations of SG, OP and X provide the miscellaneous instructions noted:

BRANCH ALWAYS

0 000 000	1	OFFSET

15 9 8 7 0

(PC) + OFFSET → (PC)

where OFFSET is an 8-bit, 2's complement quantity allowing a +127, -128 word relative branch.

CONDITION CODES OPERATES

OP = 101, SG = 0

0 000 000	0	101	S	N	Z	V	C

15 9 8 7 5 4 3 2 1 0

Micro-coding of the X, bits 4 - 0, allows a variety of instructions:

Bit 4 = 0	Clear codes noted
Bit 4 = 1	Set codes noted
Bit 3 = 0	N code unaffected by bit 4
Bit 3 = 1	N code affected by bit 4
Bit 2 = 0	Z code unaffected by bit 4
Bit 2 = 1	Z code affected by bit 4
Bit 1 = 0	V code unaffected by bit 4
Bit 1 = 1	V code affected by bit 4
Bit 0 = 0	C code unaffected by bit 4
Bit 0 = 1	C code affected by bit 4

RETURN

OP = 100, SG = 0

RTS					A
0 000 000	0	100	0	D	REG
15 9	8	7 5	4	3	2 0

Return from subroutine either direct or indirect on register noted.

For D = 0: (REG) → (PC), ((LP)) → (REG), (LP)+2 → (LP)

For D = 1: ((REG)) → (PC), ((LP)) → (REG), (LP)+2 → (LP)

Reminder of this OP unimplemented on basic machine.

PUSH/POP

OP = 000, SG = 0

0 000 000	0	000	X
15 9	8	7 5	4 0

Direct coding of X provides the Halt, Wait and basic Push/Pop instructions. The remainder of this OP is unimplemented on the basic machine.

MNEMONIC	X	OPERATION
HALT	00 000	$(PC)+2 \rightarrow (PC)$, Halt
WAIT	00 001	$(PC)+2 \rightarrow (PC)$, Pause and stop cycling, wait for interrupt.
PUS	00 010	Push ST on Linkage Pointer Stack
POS	00 011	Pop ST off Linkage Pointer Stack
PUSP	00 100	Push ST, PC on Linkage Pointer Stack
RTI	00 101	Pop PC, ST off Linkage Pointer Stack

RTI instruction is used for return from interrupt.

Typical operation: RTI

$((LP)) \rightarrow (PC)$
$(LP)+2 \rightarrow (LP)$
$((LP)) \rightarrow (ST)$
$(LS)+2 \rightarrow (LP)$

The remainder of OPERATE GROUP, SG = 0, is unimplemented on the basic machine.

The group code (bits 14-9) 000111 is unimplemented on the basic machine.

Traps on the Basic Machine:

While no specific trap instruction is provided on the basic machine, two trap locations exist. These are: location Ø for time-out and illegal instructions; and location 4 for unimplemented instructions. The trap results in ST and PC being pushed on the Linkage Pointer Stack and a new PC and ST obtained from the trap locations.

How about SE, DE

APPENDIX A: Instruction Summary (Basic Set)

NOTE: All instruction numbers in octal representation except where digit is underlined it is a single binary digit. (First digit is always single digit and underline is implied.)

i.e. 1051 (8 bit displacement)

```
  1    0     5    1      X
+---+-----+-----+---+---------+
| 1 | 000 | 101 | 1 |         |
+---+-----+-----+---+---------+
 15  14       9   8  7        0
```

Binary

			2 octal digits 6 bits (Source)	2 octal digits 6 bits (Destination)
MOV	A, B	01		
MOVB	A, B	11		
ADD	A, B	02		
ADDB	A, B	12		
SUB	A, B	03		
SUBB	A, B	13		
CMP	A, B	04		
CMPB	A, B	14		
AND	A, B	05		
ANDB	A, B	15		

Branch

BEQ	Q	0011	(Q = 8 bit displacement)
BNE	Q	0010	
BLT	Q	0021	
BGE	Q	0020	
BLE	Q	0031	
BGT	Q	0030	
BCS	Q	1031	
BCC	Q	1030	
BVS	Q	1021	
BVC	Q	1020	
BZS	Q	1011	
BZC	Q	1010	
BNS	Q	1001	
BNC	Q	1000	
BR	Q	0001	

Execute

EXC	A	1040	(8 bit address of 256)

Unary Group

		6 bits
CLR A	0050	(Destination)
CLRB A	1050	
COM A	0051	
COMB A	1051	
INC A	0052	
INCB A	1052	
DEC A	0053	
DECB A	1053	
NEG A	0054	
NEGB A	1054	
ADC A	0055	
ADCB A	1055	
DIC A	0056	
DICB A	1056	
TST A	0057	
TSTB A	1057	

Rotate/Shift

		6 bits
ROT+1 A	0060	(Destination)
ROTB+1 A	1060	
ROT-1 A	0062	
ROTB-1 A	1062	
SHF+1 A	0064	
SHFB+1 A	1064	
SHF-1 A	0066	
SHFB-1 A	1066	

Subroutine Call

		3 bits	6 bits
JSR A, A	004	(Reg.)	(Destination)

Subroutine Return

		3 bits
RTS R	00020	(Reg.)
RTS R	00021	(Reg.)

Operate Group, Condition Codes

CLC	000241
SEC	000251
CLV	000242
SEV	000252
CLZ	000243
SEZ	000253
CLN	000250
SEN	000270
CNZ	000254
CCC	000257
SCC	000277

Operate Group, Push/Pop

HALT	000000
WAIT	000001
PUS	000002
POS	000003
PUSP	000004
RTI	000005

Exec mode?

APPENDIX B:

Typical source and destination configurations are shown below together with the M, D, and REG bit patterns.
D

CODE	SYMBOL	MEANING	
M D REG			
00 0 000	%RØ	E = RØ	RØ contains data
00 1 000	@%RØ	E = (RØ)	RØ contains address
01 0 000	+(RØ)	E = (RØ), (RØ)+2 → (RØ)	Autoincrement
01 1 000	@+(RØ)	E = ((RØ)), (RØ)+2 → (RØ)	Autoincrement, defer.
10 0 000	-(RØ)	(RØ)-2 → (RØ), E = (RØ)	Autodecrement
10 1 000	@-(RØ)	(RØ)-2 → (RØ), E = ((RØ))	Autodecrement, defer.
01 0 111	#L	E = (PC)+2	Immediate literal
01 1 111	@#L A	E = ((PC)+2)	Defer thru immediate
11 0 001	A(R1)	E = (R1)+((PC)+2)	Indexed by R1 A is next word
11 1 001	A(R1)	E = (R1)+((PC)+2))	Index defer R1
11 0 111	A	E = (PC)+((PC)+2) where((PC)+2)=A-.	Alternate form of defer. thru immediate mode
11 1 111	@A	E = ((PC)+((PC)+2)) where((PC)+2)=A-.	Above with additional defer. level

NOTES:
1) A can be an expression.
2) RØ, R1 designate registers RØ, R1. These can be named with expressions.
3) Note the ease of counting instruction length with this symbology. The base instruction is 2 bytes, and each expression not enclosed by parenthesis or preceeded by % adds another word.

APPENDIX B (continued):

```
          MOV     %RØ, %R1          1 word
          MOV     #L,  %R1          2 words
          MOVB    #L,  %R1          2 words
          MOV     A,   B            3 words
          MOV     A,  (RØ)+         2 words
          MOV     A(RØ), B(R1)      3 words
```

BASIC PDP11 INSTRUCTIONS

BINARY GROUP

B	OP	SOURCE	DESTINATION
15	14 12	11 6	5 0

B	
0-WORD	
1-BYTE	

OP	
001	MOV : S→D
010	ADD : S+D→D
011	SUB : D-S→D
100	CMP : TEST S-D
101	AND : S·D→D
110 111	RESERVED

SOURCE DESTINATION

M	D	REG	M	D	REG
11 10	9	8 6	5 4	3	2 0

M	ADDRESS MODE
00	REGISTER
01	AUTO-INC (AFTER)
10	AUTO-DEC (BEFORE)
11	INDEX

D	
0	DIRECT
1	DEFERRED

REG	
000	RΦ
001	R1
010	R2
011	R3
100	R4
101	R5
110	LP
111	PC

NOTE: SOURCE AND DESTINATION CODES ARE SIMILIAR.

NOTES:

1. INSTRUCTION LENGTH:

   ONE WORD : OPERATE, BRANCH, EXECUTE ; OTHERS IF SOURCE AND DESTINATION NOT INDEXED OR AUTO-INC ON PC.

   TWO WORDS: EITHER SOURCE OR DESTINATION IS INDEXED BY NEXT WORD OR THE AUTO-INC OF PC PROVIDES AN IMMEDIATE OPERAND. A WORD IS REQUIRED FOR THIS DATA.

   THREE WORDS: BOTH SOURCE AND DESTINATION ARE INDEXED OR HAVE AN IMMEDIATE OPERAND. NEXT WORD AFTER INSTRUCTION REFERS TO SOURCE; NEXT WORD REFERS TO DESTINATION.

2. GROUP CODE 000 111 IS RESERVED.
3. RESERVED INSTRUCTION CODES ARE TRAPPED TO LOCATION 4; ILLEGAL INSTRUCTIONS ARE TRAPPED TO LOCATION Ø.
4. TRAP: ST AND PC ARE PUSHED ON LP, NEW PC AND ST FROM TRAP LOCATION.

OPERATE GROUP

GROUP CODE	SG	OP	X
15 9	8	7 5	4 0

GROUP CODE	SG	OP	X	INSTRUCTION
0 000 000	0	00 0	00 000	HALT
			001	WAIT
0 0 0	001		~~010~~	~~PUSH ST ON LP~~
			~~011~~	~~POP ST, ON LP~~
			~~100~~	~~PUSH ST, PC ON LP~~
			~~101~~ 010	POP PC, ST ~~OF~~ ON LP
			REMAINDER	RESERVED

10 0 0

D	REG
3	2 0

D=0, RTS: REG → PC
(LP)+→ REG

D=1, RTS: @REG→PC
(LP)+→ REG

REMAINDER RESERVED

10 1

S	N	Z	V	C
4	3	2	1	0

S=0, CLEAR
S=1, SET

MICRO-PROGRAMMED OPERATES ON CONDITION CODES

REMAINDER RESERVED

0 000 000 1 OFFSET

BRANCH ALWAYS (BR) OFFSET IS ± 128 WORDS.

0001 add JMP

BRANCH ON CONDITION

GROUP CODE	T	OFFSET
15 9	8	7 0

GROUP CODE	COND.
0 000 001	=
0 000 010	<
0 000 011	≤
1 000 000	N
1 000 001	Z
1 000 010	V
1 000 011	C

T	OFFSET
0 - FALSE	±128 WORDS
1 - TRUE	

OPERATION OF INSTR.

COND. MET: PC+OFFSET→PC

OTHERWISE: PC+2→PC

SUBROUTINE CALL

GROUP CODE	REG	DESTINATION
15 9	8 6	5 0

GROUP CODE	REG	DESTINATION
0 000 100		

JSR : REG→-(LP), PC+n→REG, DE→PC
(M=00, D=0 ILLEGAL)

EXECUTE

GROUP CODE	SG	ADDRESS
15 9	8	7 0

GROUP CODE	SG	INSTR	ADDRESS
1 000 100	0	EXC	LOW 256 WORDS
1 000 100	1	RESERVED	

EXC: EXECUTE CONTENTS OF ADDRESS

UNARY GROUP

B	GROUP CODE	OP	DESTINATION
15	14 9	8 6	5 0

GROUP CODE	OP	INSTR.	DESTINATION
000 101	000	CLR: 0→D	AS NOTED ABOVE.
	001	COM: $\overline{D}$→D	
	010	INC : D+1→D	
	011	DEC: D-1→D	
	100	NEG: $\overline{D}$+1→D	
	101	ADC: D+C→D	
	110	DIC: D-C→D	
	111	TST; TEST D-0	

B	
0-WORD	
1-BYTE	

ROTATE/SHIFT

B	GROUP CODE	R/S	R/L	1/M	DESTINATION
15	14 9	8	7	6	5 0

GROUP CODE	R/L	DESTINATION
000 110	0-RIGHT	AS NOTED ABOVE
	1-LEFT	

B	R/S	1/M
0-WORD	0-ROTATE	0-ONE
1-BYTE	1-SHIFT	1-RESERVED

JFOL 3/28/69