COSMAC
Elf

1802 Microprocessor Instructions

1802 instruction encodings for quick reference.

- 1802 instructions grouped by function, extracted from the 1802 manual.
- 1802 instructions ordered by opcodes (something I didn't find anywhere else on the web).

1802 Instructions Grouped By Function

```
INTERRUPT ACTIONS

RESET:                                          0->I,  0->N,  0->Q,  0->X,  0->P,  0->R(0),  1->IE
INTERRUPT:                                      X,P->T,  2->X,  1->P,  0->IE
DMA IN:                                         bus->M(R(0)),  ++R(0)
DMA OUT:                                        M(R(0))->bus,  ++R(0)

----------------------------------------
LOAD OPERATIONS

LOAD VIA N                              LDN     0N      M(R(N)) -> D; FOR N
LOAD ADVANCE                            LDA     4N      M(R(N)) -> D; R(N) + 1 -> R(N)
LOAD VIA X                              LDX     F0      M(R(X)) -> D
LOAD VIA X AND ADVANCE                  LDXA    72      M(R(X)) -> D; R(X) + 1 -> R(X)
LOAD IMMEDIATE                          LDI     F8      M(R(P)) -> D; R(P) + 1 -> R(P)
STORE VIA N                             STR     5N      D -> M(R(N))
STORE VIA X AND DECREMENT               STXD    73      D -> M(R(X)); R(X) - 1 -> R(X)

----------------------------------------
REGISTER OPERATIONS

INCREMENT REG N                         INC     1N      R(N) + 1 -> R(N)
DECREMENT REG N                         DEC     2N      R(N) - 1 -> R(N)
INCREMENT REG X                         IRX     60      R(X) + 1 -> R(X)
GET LOW REG N                           GLO     8N      R(N).0 -> D
PUT LOW REG N                           PLO     AN      D -> R(N).0
GET HIGH REG N                          GHI     9N      R(N).1 -> D
PUT HIGH REG N                          PHI     BN      D -> R(N).1

----------------------------------------
LOGIC OPERATIONS (Note 1)

OR                                      OR      F1      M(R(X)) OR D -> D
OR IMMEDIATE                            ORI     F9      M(R(P)) OR D -> D; R(P) + 1 -> R(P)
EXCLUSIVE OR                            XOR     F3      M(R(X)) XOR D -> D
EXCLUSIVE OR IMMEDIATE                  XRI     FB      M(R(P)) XOR D -> D; R(P) + 1 -> R(P)
AND                                     AND     F2      M(R(X)) AND D -> D
AND IMMEDIATE                           ANI     FA      M(R(P)) AND D -> D; R(P) + 1 -> R(P)
SHIFT RIGHT                             SHR     F6      SHIFT D RIGHT, LSB(D) -> DF, 0 -> MSB(D)
SHIFT RIGHT WITH CARRY                  SHRC    76      SHIFT D RIGHT, LSB(D) -> DF, DF -> MSB(D) (Note 2)
RING SHIFT RIGHT                        RSHR    76      SHIFT D RIGHT, LSB(D) -> DF, DF -> MSB(D) (Note 2)
SHIFT LEFT                              SHL     FE      SHIFT D LEFT, MSB(D) -> DF, 0 -> LSB(D)
SHIFT LEFT WITH CARRY                   SHLC    7E      SHIFT D LEFT, MSB(D) -> DF, DF -> LSB(D) (Note 2)
RING SHIFT LEFT                         RSHL    7E      SHIFT D LEFT, MSB(D) -> DF, DF -> LSB(D) (Note 2)

----------------------------------------
ARITHMETIC OPERATIONS (Note 1)

ADD                                     ADD     F4      M(R(X)) + D -> DF, D
ADD IMMEDIATE                           ADI     FC      M(R(P)) + D -> DF, D; R(P) + 1 -> R(P)
ADD WITH CARRY                          ADC     74      M(R(X)) + D + DF -> DF, D
ADD WITH CARRY, IMMEDIATE               ADCI    7C      M(R(P)) + D + DF -> DF, D; R(P) + 1 -> R(P)
SUBTRACT D                              SD      F5      M(R(X)) - D -> DF, D
SUBTRACT D IMMEDIATE                    SDI     FD      M(R(P)) - D -> DF, D; R(P) + 1 -> R(P)
SUBTRACT D WITH BORROW                  SDB     75      M(R(X)) - D - (NOT DF) -> DF, D
SUBTRACT D WITH BORROW, IMMEDIATE       SDBI    7D      M(R(P)) - D - (Not DF) -> DF, D; R(P) + 1 -> R(P)
SUBTRACT MEMORY                         SM      F7      D-M(R(X)) -> DF, D
SUBTRACT MEMORY IMMEDIATE               SMI     FF      D-M(R(P)) -> DF, D; R(P) + 1 -> R(P)
SUBTRACT MEMORY WITH BORROW             SMB     77      D-M(R(X))-(NOT DF) -> DF, D
SUBTRACT MEMORY WITH BORROW, IMMEDIATE  SMBI    7F      D-M(R(P))-(NOT DF) -> DF, D; R(P) + 1 -> R(P)
```

BRANCH INSTRUCTIONS - SHORT BRANCH

SHORT BRANCH	BR	30	M(R(P)) -> R(P).0
NO SHORT BRANCH (See SKP)	NBR	38	R(P) + 1 -> R(P) (Note 2)
SHORT BRANCH IF D = 0	BZ	32	IF D = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF D NOT 0	BNZ	3A	IF D NOT 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF DF = 1	BDF	33	IF DF = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P) (Note 2)
SHORT BRANCH IF POS OR ZERO	BPZ		
SHORT BRANCH IF EQUAL OR GREATER	BGE		
SHORT BRANCH IF DF = 0	BNF	3B	IF DF = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P) (Note 2)
SHORT BRANCH IF MINUS	BM		
SHORT BRANCH IF LESS	BL		
SHORT BRANCH IF Q = 1	BQ	31	IF Q = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF Q = 0	BNQ	39	IF Q = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF1 = 1 (EF1 = VSS)	B1	34	IF EF1 =1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF1 = 0 (EF1 = VCC)	BN1	3C	IF EF1 = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF2 = 1 (EF2 = VSS)	B2	35	IF EF2 = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF2 = 0 (EF2 = VCC)	BN2	3D	IF EF2 = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF3 = 1 (EF3 = VSS)	B3	36	IF EF3 = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF3 = 0 (EF3 = VCC)	BN3	3E	IF EF3 = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF4 = 1 (EF4 = VSS)	B4	37	IF EF4 = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF4 = 0 (EF4 = VCC)	BN4	3F	IF EF4 = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)

BRANCH INSTRUCTIONS - LONG BRANCH

LONG BRANCH	LBR	C0	M(R(P)) -> R(P). 1, M(R(P) + 1) -> R(P).0
NO LONG BRANCH (See LSKP)	NLBR	C8	R(P) = 2 -> R(P) (Note 2)
LONG BRANCH IF D = 0	LBZ	C2	IF D = 0, M(R(P)) -> R(P).1, M(R(P) +1) -> R(P).0, ELSE R(P) + 2 -> R(P)
LONG BRANCH IF D NOT 0	LBNZ	CA	IF D Not 0, M(R(P))-> R(P).1, M(R(P) + 1)-> R(P).0, ELSE R(P) + 2 -> R(P)
LONG BRANCH IF DF = 1	LBDF	C3	IF DF = 1, M(R(P))-> R(P).1, M(R(P) + 1)-> R(P).0, ELSE R(P) + 2 -> R(P)
LONG BRANCH IF DF = 0	LBNF	CB	IF DF = 0, M(R(P))-> R(P).1, M(R(P) + 1)-> R(P).0, ELSE R(P) + 2 -> R(P)
LONG BRANCH IF Q = 1	LBQ	C1	IF Q = 1, M(R(P)) -> R(P).1, M(R(P) + 1) -> R(P).0, ELSE R(P) + 2 -> R(P)
LONG BRANCH IF Q = 0	LBNQ	C9	IF Q = 0, M(R(P)) -> R(P).1, M(R(P) + 1) -> R(P).0 EISE R(P) + 2 -> R(P)

SKIP INSTRUCTIONS

SHORT SKIP (See NBR)	SKP	38	R(P) + 1 -> R(P) (Note 2)
LONG SKIP (See NLBR)	LSKP	C8	(Note 2) R(P) + 2 -> R(P)
LONG SKIP IF D = 0	LSZ	CE	IF D = 0, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF D NOT 0	LSNZ	C6	IF D Not 0, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF DF = 1	LSDF	CF	IF DF = 1, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF DF = 0	LSNF	C7	IF DF = 0, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF Q = 1	LSQ	CD	IF Q = 1, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF Q = 0	LSNQ	C5	IF Q = 0, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF IE = 1	LSIE	CC	IF IE = 1, R(P) + 2 -> R(P), ELSE CONTINUE

CONTROL INSTRUCTIONS

IDLE	IDL	0	(Note 3) WAIT FOR DMA OR INTERRUPT; M(R(0)) -> BUS
NO OPERATION	NOP	C4	CONTINUE
SET P	SEP	DN	N -> P
SET X	SEX	EN	N -> X
SET Q	SEQ	7B	1 -> Q
RESET Q	REQ	7A	0 -> Q
SAVE	SAV	78	T -> M(R(X))
PUSH X, P TO STACK	MARK	79	(X, P)-> T; (X, P)-> M(R(2)), THEN P-> X; R(2) - 1-> R(2)
RETURN	RET	70	M(R(X)) -> (X, P); R(X) + 1 -> R(X), 1 -> IE
DISABLE	DIS	71	M(R(X)) -> (X, P); R(X) + 1 -> R(X), 0 -> IE

INPUT - OUTPUT BYTE TRANSFER

OUTPUT 1	OUT 1	61	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 1
OUTPUT 2	OUT 2	62	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 2
OUTPUT 3	OUT 3	63	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 3
OUTPUT 4	OUT 4	64	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 4
OUTPUT 5	OUT 5	65	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 5
OUTPUT 6	OUT 6	66	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 6
OUTPUT 7	OUT 7	67	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 7
INPUT 1	INP 1	69	BUS -> M(R(X)); BUS -> D; N LINES = 1
INPUT 2	INP 2	6A	BUS -> M(R(X)); BUS -> D; N LINES = 2

INPUT 3	INP 3	6B	BUS -> M(R(X)); BUS -> D; N LINES = 3
INPUT 4	INP 4	6C	BUS -> M(R(X)); BUS -> D; N LINES = 4
INPUT 5	INP 5	6D	BUS -> M(R(X)); BUS -> D; N LINES = 5
INPUT 6	INP 6	6E	BUS -> M(R(X)); BUS -> D; N LINES = 6
INPUT 7	INP 7	6F	BUS -> M(R(X)); BUS -> D; N LINES = 7

NOTES

* 0. Nomenclature / register summary:
D : data register, accumulator (16 bits).
DF : data flag, carry (1 bit).
P : program-counter register designator (4 bits).
X : index register designator (4 bits).
I : high nibble of instruction (4-bits)
N : low nibble of instruction (4 bits).
R(d) : 1 of 16 16-bit registers as designated by d.
Q : Q flag (1 bit).
IE : interrupt enable flag (1 bit).
T : saved-state register (X,P) on interrupt (8 bits).
M(a) : memory location addressed by a (8 bits).

* 1. The arithmetic operations and the shift instructions are the only instructions that can alter the DF.
After an add instruction:
DF = 1 denotes a carry has occurred
DF = 0 Denotes a carry has not occurred
After a subtract instruction:
DF = 1 denotes no borrow. D is a true positive number
DF = 0 denotes a borrow. D is twoÕs complement
The syntax Ò-(not DF)Ó denotes the subtraction of the borrow.

* 2. This instruction is associated with more than one mnemonic. Each mnemonic is individually listed.

* 3. An idle instruction initiates a repeating S1 cycle. The processor will continue to idle until an I/O request (INTERRUPT, DMA-lN, or DMA- OUT) is activated. When the request is acknowledged, the idle cycle is terminated and the I/O request is serviced, and then normal operation is resumed.

* 4. Long-Branch, Long-Skip and No Op instructions require three cycles to complete (1 fetch + 2 execute).
Long-Branch instructions are three bytes long. The first byte specifies the condition to be tested; and the second and third byte, the branching address.

If the tested condition is met, then branching takes place; the branching address bytes are loaded in the high-and-low order bytes of the current program counter, respectively. This operation effects a branch to any memory location.
If the tested condition is not met, the branching address bytes are skipped over, and the next instruction in sequence is fetched and executed.
This operation is taken for the case of unconditional no branch (NLBR).

* 5. The short-branch instructions are two bytes long. The first byte specifies the condition to be tested, and the second specifies the branching address.

If the tested condition is met, then branching takes place; the branching address byte is loaded into the low-order byte position of the current program counter. This effects a branch within the current 256-byte page of the memory, i.e., the page which holds the branching address. If the tested condition is not met, the branching address byte is skipped over, and the next instruction in sequence is fetched and executed. This same action is taken in the case of unconditional no branch (NBR).

* 6. The skip instructions are one byte long. There is one Unconditional Short-Skip (SKP) and eight Long-Skip instructions.
The Unconditional Short-Skip instruction takes 2 cycles to complete (1 fetch + 1 execute). Its action is to skip over the byte following it.
Then the next instruction in sequence is fetched and executed. This SKP instruction is identical to the unconditional no-branch instruction (NBR) except that the skipped-over byte is not considered part of the program.
The Long-Skip instructions take three cycles to complete (1 fetch + 2 execute).

If the tested condition is met, then Long Skip takes place; the current program counter is incremented twice. Thus two bytes are skipped over, and the next instruction in sequence is fetched and executed. If the tested condition is not met, then no action is taken. Execution is continued by fetching the next instruction in sequence.

1802 Instructions By Opcode

IDLE	IDL	0	WAIT FOR DMA OR INTERRUPT; M(R(0)) -> BUS (NOTE 3)
LOAD VIA N	LDN	0N	M(R(N)) -> D; FOR N
INCREMENT REG N	INC	1N	R(N) + 1 -> R(N)
DECREMENT REG N	DEC	2N	R(N) - 1 -> R(N)
SHORT BRANCH	BR	30	M(R(P)) -> R(P).0
SHORT BRANCH IF Q = 1	BQ	31	IF Q = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)

SHORT BRANCH IF D = 0	BZ	32	IF D = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF DF = 1	BDF	33	IF DF = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P) (Note 2)
SHORT BRANCH IF EF1 = 1 (EF1 = VSS)	B1	34	IF EF1 =1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF2 = 1 (EF2 = VSS)	B2	35	IF EF2 = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF3 = 1 (EF3 = VSS)	B3	36	IF EF3 = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF4 = 1 (EF4 = VSS)	B4	37	IF EF4 = 1, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
NO SHORT BRANCH (See SKP)	NBR	38	R(P) + 1 -> R(P) (Note 2)
SHORT SKIP (See NBR)	SKP	38	R(P) + 1 -> R(P) (Note 2)
SHORT BRANCH IF Q = 0	BNQ	39	IF Q = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF D NOT 0	BNZ	3A	IF D NOT 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF DF = 0	BNF	3B	IF DF = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P) (Note 2)
SHORT BRANCH IF EF1 = 0 (EF1 = VCC)	BN1	3C	IF EF1 = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF2 = 0 (EF2 = VCC)	BN2	3D	IF EF2 = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF3 = 0 (EF3 = VCC)	BN3	3E	IF EF3 = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
SHORT BRANCH IF EF4 = 0 (EF4 = VCC)	BN4	3F	IF EF4 = 0, M(R(P)) -> R(P).0, ELSE R(P) + 1 -> R(P)
LOAD ADVANCE	LDA	4N	M(R(N)) -> D; R(N) + 1 -> R(N)
STORE VIA N	STR	5N	D -> M(R(N))
INCREMENT REG X	IRX	60	R(X) + 1 -> R(X)
OUTPUT 1	OUT 1	61	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 1
OUTPUT 2	OUT 2	62	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 2
OUTPUT 3	OUT 3	63	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 3
OUTPUT 4	OUT 4	64	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 4
OUTPUT 5	OUT 5	65	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 5
OUTPUT 6	OUT 6	66	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 6
OUTPUT 7	OUT 7	67	M(R(X)) -> BUS; R(X) + 1 -> R(X); N LINES = 7
not assigned		68	
INPUT 1	INP 1	69	BUS -> M(R(X)); BUS -> D; N LINES = 1
INPUT 2	INP 2	6A	BUS -> M(R(X)); BUS -> D; N LINES = 2
INPUT 3	INP 3	6B	BUS -> M(R(X)); BUS -> D; N LINES = 3
INPUT 4	INP 4	6C	BUS -> M(R(X)); BUS -> D; N LINES = 4
INPUT 5	INP 5	6D	BUS -> M(R(X)); BUS -> D; N LINES = 5
INPUT 6	INP 6	6E	BUS -> M(R(X)); BUS -> D; N LINES = 6
INPUT 7	INP 7	6F	BUS -> M(R(X)); BUS -> D; N LINES = 7
RETURN	RET	70	M(R(X)) -> (X, P); R(X) + 1 -> R(X), 1 -> IE
DISABLE	DIS	71	M(R(X)) -> (X, P); R(X) + 1 -> R(X), 0 -> IE
LOAD VIA X AND ADVANCE	LDXA	72	M(R(X)) -> D; R(X) + 1 -> R(X)
STORE VIA X AND DECREMENT	STXD	73	D -> M(R(X)); R(X) - 1 -> R(X)
ADD WITH CARRY	ADC	74	M(R(X)) + D + DF -> DF, D
SUBTRACT D WITH BORROW	SDB	75	M(R(X)) - D - (NOT DF) -> DF, D
SHIFT RIGHT WITH CARRY	SHRC	76	SHIFT D RIGHT, LSB(D) -> DF, DF -> MSB(D) (Note 2)
RING SHIFT RIGHT	RSHR	76	SHIFT D RIGHT, LSB(D) -> DF, DF -> MSB(D) (Note 2)
SUBTRACT MEMORY WITH BORROW	SMB	77	D-M(R(X))-(NOT DF) -> DF, D
SAVE	SAV	78	T -> M(R(X))
PUSH X, P TO STACK	MARK	79	(X, P)-> T; (X, P)-> M(R(2)), THEN P-> X; R(2) - 1-> R(2)
RESET Q	REQ	7A	0 -> Q
SET Q	SEQ	7B	1 -> Q
ADD WITH CARRY, IMMEDIATE	ADCI	7C	M(R(P)) + D + DF -> DF, D; R(P) + 1 -> R(P)
SUBTRACT D WITH BORROW, IMMEDIATE	SDBI	7D	M(R(P)) - D - (Not DF) -> DF, D; R(P) + 1 -> R(P)
SHIFT LEFT WITH CARRY	SHLC	7E	SHIFT D LEFT, MSB(D) -> DF, DF -> LSB(D) (Note 2)
RING SHIFT LEFT	RSHL	7E	SHIFT D LEFT, MSB(D) -> DF, DF -> LSB(D) (Note 2)
SUBTRACT MEMORY WITH BORROW, IMMEDIATE	SMBI	7F	D-M(R(P))-(NOT DF) -> DF, D; R(P) + 1 -> R(P)
GET LOW REG N	GLO	8N	R(N).0 -> D
GET HIGH REG N	GHI	9N	R(N).1 -> D
PUT LOW REG N	PLO	AN	D -> R(N).0
PUT HIGH REG N	PHI	BN	D -> R(N).1
LONG BRANCH	LBR	C0	M(R(P)) -> R(P). 1, M(R(P) + 1) -> R(P).0
LONG BRANCH IF Q = 1	LBQ	C1	IF Q = 1, M(R(P)) -> R(P).1, M(R(P) + 1) -> R(P).0, ELSE R(P) + 2 -> R(P)
LONG BRANCH IF D = 0	LBZ	C2	IF D = 0, M(R(P)) -> R(P).1, M(R(P) +1) -> R(P).0, ELSE R(P) + 2 -> R(P)
LONG BRANCH IF DF = 1	LBDF	C3	IF DF = 1, M(R(P))-> R(P).1, M(R(P) + 1)-> R(P).0, ELSE R(P) + 2 -> R(P)
NO OPERATION	NOP	C4	CONTINUE
LONG SKIP IF Q = 0	LSNQ	C5	IF Q = 0, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF D NOT 0	LSNZ	C6	IF D Not 0, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF DF = 0	LSNF	C7	IF DF = 0, R(P) + 2 -> R(P), ELSE CONTINUE
NO LONG BRANCH (See LSKP)	NLBR	C8	R(P) = 2 -> R(P) (Note 2)
LONG SKIP (See NLBR)	LSKP	C8	R(P) + 2 -> R(P) (Note 2)
LONG BRANCH IF Q = 0	LBNQ	C9	IF Q = 0, M(R(P)) -> R(P).1, M(R(P) + 1) -> R(P).0 EISE R(P) + 2 -> R(P)
LONG BRANCH IF D NOT 0	LBNZ	CA	IF D Not 0, M(R(P))-> R(P).1, M(R(P) + 1)-> R(P).0, ELSE R(P) + 2 -> R(P)
LONG BRANCH IF DF = 0	LBNF	CB	IF DF = 0, M(R(P))-> R(P).1, M(R(P) + 1)-> R(P).0, ELSE R(P) + 2 -> R(P)
LONG SKIP IF IE = 1	LSIE	CC	IF IE = 1, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF Q = 1	LSQ	CD	IF Q = 1, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF D = 0	LSZ	CE	IF D = 0, R(P) + 2 -> R(P), ELSE CONTINUE
LONG SKIP IF DF = 1	LSDF	CF	IF DF = 1, R(P) + 2 -> R(P), ELSE CONTINUE
SET P	SEP	DN	N -> P
SET X	SEX	EN	N -> X
LOAD VIA X	LDX	F0	M(R(X)) -> D

OR	OR	F1	M(R(X)) OR D -> D
AND	AND	F2	M(R(X)) AND D -> D
EXCLUSIVE OR	XOR	F3	M(R(X)) XOR D -> D
ADD	ADD	F4	M(R(X)) + D -> DF, D
SUBTRACT D	SD	F5	M(R(X)) - D -> DF, D
SHIFT RIGHT	SHR	F6	SHIFT D RIGHT, LSB(D) -> DF, 0 -> MSB(D)
SUBTRACT MEMORY	SM	F7	D-M(R(X)) -> DF, D
LOAD IMMEDIATE	LDI	F8	M(R(P)) -> D; R(P) + 1 -> R(P)
OR IMMEDIATE	ORI	F9	M(R(P)) OR D -> D; R(P) + 1 -> R(P)
AND IMMEDIATE	ANI	FA	M(R(P)) AND D -> D; R(P) + 1 -> R(P)
EXCLUSIVE OR IMMEDIATE	XRI	FB	M(R(P)) XOR D -> D; R(P) + 1 -> R(P)
ADD IMMEDIATE	ADI	FC	M(R(P)) + D -> DF, D; R(P) + 1 -> R(P)
SUBTRACT D IMMEDIATE	SDI	FD	M(R(P)) - D -> DF, D; R(P) + 1 -> R(P)
SHIFT LEFT	SHL	FE	SHIFT D LEFT, MSB(D) -> DF, 0 -> LSB(D)
SUBTRACT MEMORY IMMEDIATE	SMI	FF	D-M(R(P)) -> DF, D; R(P) + 1 -> R(P)