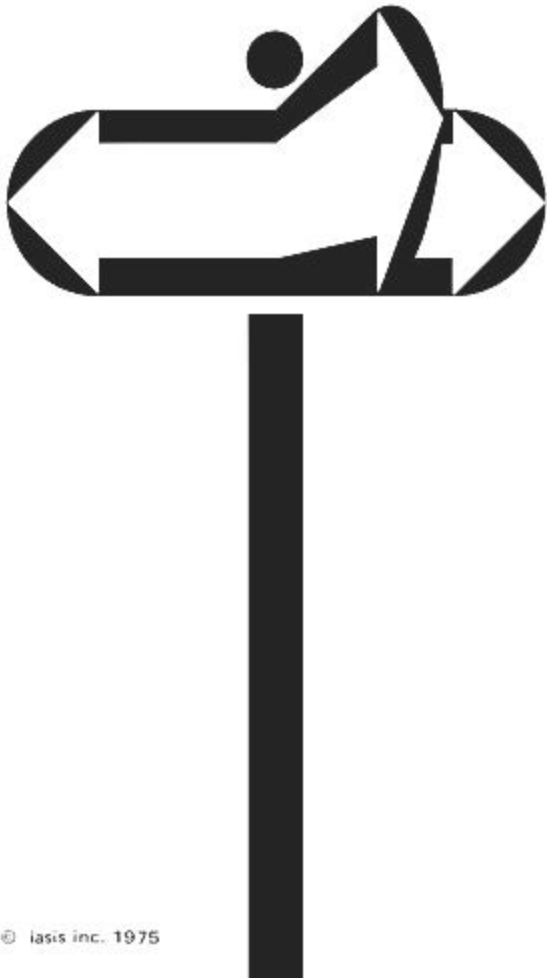
IASIS MCS 8/80
POCKET REFERENCE BOOK



Data and Instruction Formats

Data in the 8008 is stored in the form of 8-bit binary integers. All data transfers to the system data bus will be in the same format.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

DATA WORD

The program instructions may be one, two, or three bytes in length. Multiple byte instructions must be stored in successive words in program memory. The instruction formats then depend on the particular operation executed.

One Byte Instructions

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	OP CODE

Two Byte Instructions

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	OP CODE
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	OPERAND

Three Byte Instructions

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	OP CODE
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	LOW ADDRESS
X	X	D_5	D_4	D_3	D_2	D_1	D_0	HIGH ADDRESS*

TYPICAL INSTRUCTIONS

Register to register, memory reference, I/O arithmetic or logical, rotate or return instructions.

Immediate mode instructions

JUMP or CALL instructions

*For the third byte of this instruction, D_6 and D_7 are "don't care" bits.

For the MCS-8™ a logic "1" is defined as a high level and a logic "0" is defined as a low level.

Index Register Instructions

The load instructions do not affect the flag flip-flops. The increment and decrement instructions affect all flip-flops except the carry.

MNEMONIC	MINIMUM STATES REQUIRED	INSTRUCTION CODE D_7 D_6	D_5 D_4 D_3	D_2 D_1 D_0	DESCRIPTION OF OPERATION
(1)MOV r_1, r_2	(5)	1 1	D D D	S S S	Load index register r_1 with the content of index register r_2.
(2)MOV r, M	(8)	1 1	D D D	1 1 1	Load index register r with the content of memory register M.
MOV M, r	(7)	1 1	1 1 1	S S S	Load memory register M with the content of index register r.
(3)MVI r	(8)	0 0 B B	D D D B B B	1 1 0 B B B	Load index register r with data B . . . B.
MVI M	(9)	0 0 B B	1 1 1 B B B	1 1 0 B B B	Load memory register M with data B . . . B.
INR r	(5)	0 0	D D D	0 0 0	Increment the content of index register r (r ≠ A).
DCR r	(5)	0 0	D D D	0 0 1	Decrement the content of index register r (r ≠ A).

Accumulator Group Instructions

The result of the ALU instructions affect all of the flag flip-flops. The rotate instructions affect only the carry flip-flop.

ADD r	(5)	1	0	0	0	0	S	S	S	Add the content of index register r, memory register M, or data B . . . B to the accumulator. An overflow (carry) sets the carry flip-flop.
ADD M	(8)	1	0	0	0	0	1	1	1	
ADI	(8)	0	0	0	0	0	1	0	0	
		B	B	B	B	B	B	B	B	
ADC r	(5)	1	0	0	0	1	S	S	S	Add the content of index register r, memory register M, or data B . . . B from the accumulator with carry. An overflow (carry) sets the carry flip-flop.
ADC M	(8)	1	0	0	0	1	1	1	1	
ACI	(8)	0	0	0	0	1	1	0	0	
		B	B	B	B	B	B	B	B	
SUB r	(5)	1	0	0	1	0	S	S	S	Subtract the content of index register r, memory register M, or data B . . . B from the accumulator. An underflow (borrow) sets the carry flip-flop.
SUB M	(8)	1	0	0	1	0	1	1	1	
SUI	(8)	0	0	0	1	0	1	0	0	
		B	B	B	B	B	B	B	B	
SBB r	(5)	1	0	0	1	1	S	S	S	Subtract the content of index register r, memory register M, or data B . . . B from the accumulator with borrow. An underflow (borrow) sets the carry flip-flop.
SBB M	(8)	1	0	0	1	1	1	1	1	
SBI	(8)	0	0	0	1	1	1	0	0	
		B	B	B	B	B	B	B	B	

BASIC INSTRUCTION SET

MNEMONIC	MINIMUM STATES REQUIRED	INSTRUCTION CODE D_7 D_6	D_5 D_4 D_3	D_2 D_1 D_0	DESCRIPTION OF OPERATION
ANA r	(5)	1 0	1 0 0	S S S	Compute the logical AND of the content of index register r, memory register M, or data B . . . B with the accumulator.
ANA M	(8)	1 0	1 0 0	1 1 1	
ANI	(8)	0 0 B B	1 0 0 B B B	1 0 0 B B B	
XRA r	(5)	1 0	1 0 1	S S S	Compute the EXCLUSIVE OR of the content of index register r, memory register M, or data B . . . B with the accumulator.
XRA M	(8)	1 0	1 0 1	1 1 1	
XRI	(8)	0 0 B B	1 0 1 B B B	1 0 0 B B B	
ORA r	(5)	1 0	1 1 0	S S S	Compute the INCLUSIVE OR of the content of index register r, memory register M, or data B . . . B with the accumulator.
ORA M	(8)	1 0	1 1 0	1 1 1	
ORI	(8)	0 0 B B	1 1 0 B B B	1 0 0 B B B	
CMP r	(5)	1 0	1 1 1	S S S	Compare the content of index register r, memory register M, or data B . . . B with the accumulator. The content of the accumulator is unchanged.
CMP M	(8)	1 0	1 1 1	1 1 1	
CPI	(8)	0 0 B B	1 1 1 B B B	1 0 0 B B B	
RLC	(5)	0 0	0 0 0	0 1 0	Rotate the content of the accumulator left.
RRC	(5)	0 0	0 0 1	0 1 0	Rotate the content of the accumulator right.
RAL	(5)	0 0	0 1 0	0 1 0	Rotate the content of the accumulator left through the carry.
RAR	(5)	0 0	0 1 1	0 1 0	Rotate the content of the accumulator right through the carry.

Program Counter and Stack Control Instructions

(4) JMP	(11)	0 1 B_2 B_2 X X	X X X B_2 B_2 B_2 B_3 B_3 B_3	1 0 0 B_2 B_2 B_2 B_3 B_3 B_3	Unconditionally jump to memory address $B_3 \ldots B_3B_2 \ldots B_2$.
(5) JNC, JNZ, JP, JPO	(9 or 11)	0 1 B_2 B_2 X X	0 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 0 0 B_2 B_2 B_2 B_3 B_3 B_3	Jump to memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop is true. Otherwise, execute the next instruction in sequence.
JC, JZ JM, JPE	(9 or 11)	0 1 B_2 B_2 X X	1 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 0 0 B_2 B_2 B_2 B_3 B_3 B_3	Jump to memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop is true. Otherwise, execute the next instruction in sequence.
CALL	(11)	0 1 B_2 B_2 X X	X X X B_2 B_2 B_2 B_3 B_3 B_3	1 1 0 B_2 B_2 B_2 B_3 B_3 B_3	Unconditionally call the subroutine at memory address $B_3 \ldots B_3B_2 \ldots B_2$. Save the current address (up one level in the stack).
CNC, CNZ, CP, CPO	(9 or 11)	0 1 B_2 B_2 X X	0 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 1 0 B_2 B_2 B_2 B_3 B_3 B_3	Call the subroutine at memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop is false, and save the current address (up one level in the stack). Otherwise, execute the next instruction in sequence.
CC, CZ, CM, CPE	(9 or 11)	0 1 B_2 B_2 X X	1 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 1 0 B_2 B_2 B_2 B_3 B_3 B_3	Call the subroutine at memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop is false, and save the current address (up one level in the stack). Otherwise, execute the next instruction in sequence.
RET	(5)	0 0	X X X	1 1 1	Unconditionally return (down one level in the stack).
RNC, RNZ, RP, RPO	(3 or 5)	0 0	0 C_4 C_3	0 1 1	Return (down one level in the stack) if the condition flip-flop is false. Otherwise, execute the next instruction in sequence.
RC, RZ RM, RPE	(3 or 5)	0 0	1 C_4 C_3	0 1 1	Return (down one level in the stack) if the condition flip-flop is true. Otherwise, execute the next instruction in sequence.
RST	(5)	0 0	A A A	1 0 1	Call the subroutine at memory address AAA000 (up one level in the stack).

Input/Output Instructions

IN	(8)	0 1	0 0 M	M M 1	Read the content of the selected input port (MMM) into the accumulator.
OUT	(6)	0 1	R R M	M M 1	Write the content of the accumulator into the selected output port (RRMMM, RR ≠ 00).

Machine Instruction

HLT	(4)	0 0	0 0 0	0 0 X	Enter the STOPPED state and remain there until interrupted.
	(4)	1 1	1 1 1	1 1 1	

NOTES:

(1) SSS = Source Index Register
DDD = Destination Index Register

These registers, r_1, are designed A(accumulator—000), B(001), C(010), D(011), E(100), H(101), L(110).

(2) Memory registers are addressed by the contents of registers H & L.

(3) Additional bytes of instruction are designated by BBBBBBB.

(4) X = "Don't Care".

(5) Flag flip-flops are defined by C_4C_3: carry (00-overflow or underflow), zero (01-result is zero), sign (10-MSB of result is "1"), parity (11-parity is even).

8080 INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
$MOV_{r1, r2}$	Move register to register	0	1	D	D	D	S	S	S	5
MOV M, r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r, M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
ANA r	And register with A	1	0	1	0	0	S	S	S	4
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Nump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11

Mnemonic	Description	Instruction Code[1] D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	Clock[2]
RZ	Return on zero	1 1 0 0 1 0 0 0	5/11
RNZ	Return on no zero	1 1 0 0 0 0 0 0	5/11
RP	Return on positive	1 1 1 1 0 0 0 0	5/11
RM	Return on minus	1 1 1 1 1 0 0 0	5/11
RPE	Return on parity even	1 1 1 0 1 0 0 0	5/11
RPO	Return on parity odd	1 1 1 0 0 0 0 0	5/11
RST	Restart	1 1 A A A 1 1 1	11
IN	Input	1 1 0 1 1 0 1 1	10
OUT	Output	1 1 0 1 0 0 1 1	10
LXI B	Load immediate register Pair B & C	0 0 0 0 0 0 0 1	10
LXI D	Load immediate register Pair D & E	0 0 0 1 0 0 0 1	10
LXI H	Load immediate register Pair H & L	0 0 1 0 0 0 0 1	10
LXI SP	Load immediate stack pointer	0 0 1 1 0 0 0 1	10
PUSH B	Push register Pair B & C on stack	1 1 0 0 0 1 0 1	11
PUSH D	Push register Pair D & E on stack	1 1 0 1 0 1 0 1	11
PUSH H	Push register Pair H & L on stack	1 1 1 0 0 1 0 1	11
PUSH PSW	Push A and Flags on stack	1 1 1 1 0 1 0 1	11
POP B	Pop register pair B & C off stack	1 1 0 0 0 0 0 1	10
POP D	Pop register pair D & E off stack	1 1 0 1 0 0 0 1	10
POP H	Pop register pair H & L off stack	1 1 1 0 0 0 0 1	10
POP PSW	Pop A and Flags off stack	1 1 1 1 0 0 0 1	10
STA	Store A direct	0 0 1 1 0 0 1 0	13
LDA	Load A direct	0 0 1 1 1 0 1 0	13
XCHG	Exchange D & E, H & L Registers	1 1 1 0 1 0 1 1	4
XTHL	Exchange top of stack, H & L	1 1 1 0 0 0 1 1	18
SPHL	H & L to stack pointer	1 1 1 1 1 0 0 1	5
PCHL	H & L to program counter	1 1 1 0 1 0 0 1	5
DAD B	Add B & C to H & L	0 0 0 0 1 0 0 1	10
DAD D	Add D & E to H & L	0 0 0 1 1 0 0 1	10
DAD H	Add H & L to H & L	0 0 1 0 1 0 0 1	10
DAD SP	Add stack pointer to H & L	0 0 1 1 1 0 0 1	10
STAX B	Store A indirect	0 0 0 0 0 0 1 0	7
STAX D	Store A indirect	0 0 0 1 0 0 1 0	7
LDAX B	Load A indirect	0 0 0 0 1 0 1 0	7
LDAX D	Load A indirect	0 0 0 1 1 0 1 0	7
INX B	Increment B & C registers	0 0 0 0 0 0 1 1	5
INX D	Increment D & E registers	0 0 0 1 0 0 1 1	5
INX H	Increment H & L registers	0 0 1 0 0 0 1 1	5
INX SP	Increment stack pointer	0 0 1 1 0 0 1 1	5
DCX B	Decrement B & C	0 0 0 0 1 0 1 1	5
DCX D	Decrement D & E	0 0 0 1 1 0 1 1	5
DCX H	Decrement H & L	0 0 1 0 1 0 1 1	5
DCX SP	Decrement stack pointer	0 0 1 1 1 0 1 1	5
CMA	Compliment A	0 0 1 0 1 1 1 1	4
STC	Set carry	0 0 1 1 0 1 1 1	4
CMC	Compliment carry	0 0 1 1 1 1 1 1	4
DAA	Decimal adjust A	0 0 1 0 0 1 1 1	4
SHLD	Store H & L direct	0 0 1 0 0 0 1 0	16
LHLD	Load H & L direct	0 0 1 0 1 0 1 0	16
EI	Enable Interrupts	1 1 1 1 1 0 1 1	4
DI	Disable interrupt	1 1 1 1 0 0 1 1	4
NOP	No-operation	0 0 0 0 0 0 0 0	4

NOTES:

1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

8080 INSTRUCTION MACHINE CODES

00	NOP		2B	DCX	H	56	MOV	D,M
01	LXI	B,D16	2C	INR	L	57	MOV	D,A
02	STAX	B	2D	DCR	L	58	MOV	E,B
03	INX	B	2E	MVI	L,D8	59	MOV	E,C
04	INR	B	2F	CMA		5A	MOV	E,D
05	DCR	B	30	- - -		5B	MOV	E,E
06	MVI	B,D8	31	LXI	SP,D16	5C	MOV	E,H
07	RLC		32	STA	Adr	5D	MOV	E,L
08	- - -		33	INX	SP	5E	MOV	E,M
09	DAD	B	34	INR	M	5F	MOV	E,A
0A	LDAX	B	35	DCR	M	60	MOV	H,B
0B	DCX	B	36	MVI	M,D8	61	MOV	H,C
0C	INR	C	37	STC		62	MOV	H,D
0D	DCR	C	38	- - -		63	MOV	H,E
0E	MVI	C,D8	39	DAD	SP	64	MOV	H,H
0F	RRC		3A	LDA	Adr	65	MOV	H,L
10	- - -		3B	DCX	SP	66	MOV	H,M
11	LXI	D,D16	3C	INR	A	67	MOV	H,A
12	STAX	D	3D	DCR	A	68	MOV	L,B
13	INX	D	3E	MVI	A,D8	69	MOV	L,C
14	INR	D	3F	CMC		6A	MOV	L,D
15	DCR	D	40	MOV	B,B	6B	MOV	L,E
16	MVI	D,D8	41	MOV	B,C	6C	MOV	L,H
17	RAL		42	MOV	B,D	6D	MOV	L,L
18	- - -		43	MOV	B,E	6E	MOV	L,M
19	DAD	D	44	MOV	B,H	6F	MOV	L,A
1A	LDAX	D	45	MOV	B,L	70	MOV	M,B
1B	DCX	D	46	MOV	B,M	71	MOV	M,C
1C	INR	E	47	MOV	B,A	72	MOV	M,D
1D	DCR	E	48	MOV	C,B	73	MOV	M,E
1E	MVI	E,D8	49	MOV	C,C	74	MOV	M,H
1F	RAR		4A	MOV	C,D	75	MOV	M,L
20	- - -		4B	MOV	C,E	76	HLT	
21	LXI	H,D16	4C	MOV	C,H	77	MOV	M,A
22	SHLD	Adr	4D	MOV	C,L	78	MOV	A,B
23	INX	H	4E	MOV	C,M	79	MOV	A,C
24	INR	H	4F	MOV	C,A	7A	MOV	A,D
25	DCR	H	50	MOV	D,B	7B	MOV	A,E
26	MVI	H,D8	51	MOV	D,C	7C	MOV	A,H
27	DAA		52	MOV	D,D	7D	MOV	A,L
28	- - -		53	MOV	D,E	7E	MOV	A,M
29	DAD	H	54	MOV	D,H	7F	MOV	A,A
2A	LHLD	Adr	55	MOV	D,L	80	ADD	B

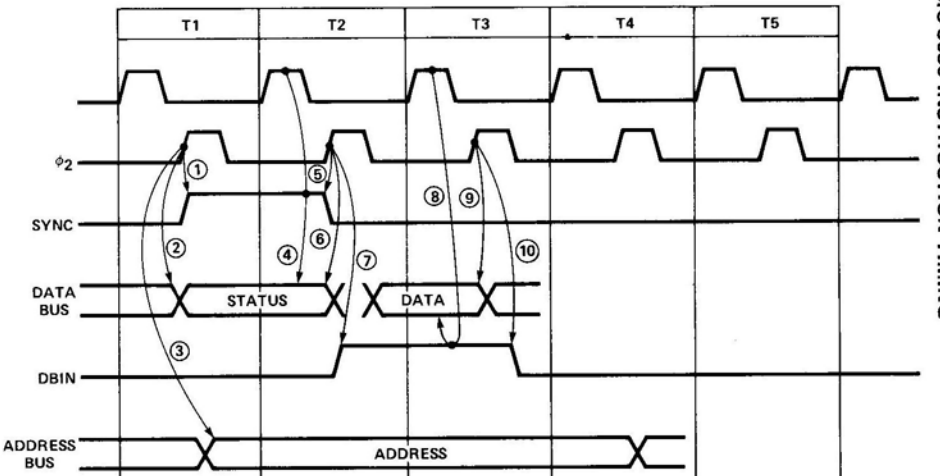
D8 = constant, or logical/arithmetic expression that evaluates to an 8 bit data quantity.

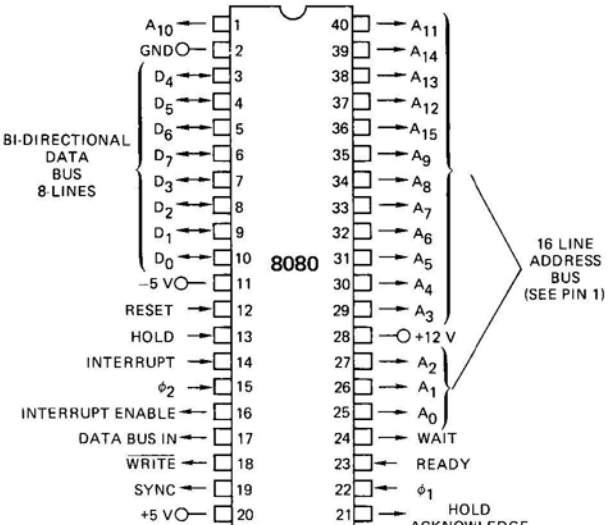
81	ADD	C	AC	XRA	H		D7	RST	2		
82	ADD	D	AD	XRA	L		D8	RC			
83	ADD	E	AE	XRA	M		D9	---			
84	ADD	H	AF	XRA	A		DA	JC		Adr	
85	ADD	L	B0	ORA	B		DB	IN		D8	
86	ADD	M	B1	ORA	C		DC	CC		Adr	
87	ADD	A	B2	ORA	D		DD	---			
88	ADC	B	B3	ORA	E		DE	SBI		D8	
89	ADC	C	B4	ORA	H		DF	RST	3		
8A	ADC	D	B5	ORA	L		E0	RPO			
8B	ADC	E	B6	ORA	M		E1	POP	H		
8C	ADC	H	B7	ORA	A		E2	JPO		Adr	
8D	ADC	L	B8	CMP	B		E3	XTHL			
8E	ADC	M	B9	CMP	C		E4	CPO		Adr	
8F	ADC	A	BA	CMP	D		E5	PUSH	H		
90	SUB	B	BB	CMP	E		E6	ANI		D8	
91	SUB	C	BC	CMP	H		E7	RST	4		
92	SUB	D	BD	CMP	L		E8	RPE			
93	SUB	E	BE	CMP	M		E9	PCHL			
94	SUB	H	BF	CMP	A		EA	JPE		Adr	
95	SUB	L	C0	RNZ			EB	XCHG			
96	SUB	M	C1	POP	B		EC	CPE		Adr	
97	SUB	A	C2	JNZ		Adr	ED	---			
98	SBB	B	C3	JMP		Adr	EE	XRI		D8	
99	SBB	C	C4	CNZ		Adr	EF	RST	5		
9A	SBB	D	C5	PUSH	B		F0	RP			
9B	SBB	E	C6	ADI		D8	F1	POP		PSW	
9C	SBB	H	C7	RST	0		F2	JP		Adr	
9D	SBB	L	C8	RZ			F3	DI			
9E	SBB	M	C9	RET		Adr	F4	CP		Adr	
9F	SBB	A	CA	JZ			F5	PUSH		PSW	
A0	ANA	B	CB	---			F6	ORI		D8	
A1	ANA	C	CC	CZ		Adr	F7	RST	6		
A2	ANA	D	CD	CALL		Adr	F8	RM			
A3	ANA	E	CE	ACI		D8	F9	SPHL			
A4	ANA	H	CF	RST	1		FA	JM		Adr	
A5	ANA	L	D0	RNC			FB	EI			
A6	ANA	M	D1	POP	D		FC	CM		Adr	
A7	ANA	A	D2	JNC		Adr	FD	---			
A8	XRA	B	D3	OUT		D8	FE	CPI		D8	
A9	XRA	C	D4	CNC		Adr	FF	RST	7		
AA	XRA	D	D5	PUSH	D						
AB	XRA	E	D6	SUI		D8					

D16 = constant, or logical/arithmetic expression that evaluates to a 16 bit data quantity.

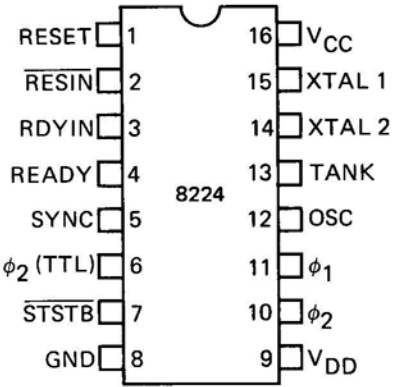
Notes:

1. Rising edge of ϕ_2 clock in T1 STATE triggers sync pulse.
2. Rising edge of ϕ_2 clock in T1 STATE causes status word to be sent out on data bus.
3. Rising edge of ϕ_2 in T1 STATE causes address to be sent out on address bus.
4. External logic reads status word when both ϕ_1 and sync are high.
5. Rising edge of ϕ_2 clock in T2 STATE terminates sync pulse.
6. Rising edge of ϕ_2 clock in T2 STATE causes status word to be withdrawn from the data bus. Shortly thereafter the incoming data word from the memory arrives.
7. Rising edge of ϕ_2 clock in T2 STATE triggers beginning of DBIN pulse.
8. Data bus read when ϕ_1 and DBIN high during STATE T3. Data, at this point, consists of instruction fetched from memory.
9. Rising edge of ϕ_2 clock in T3 STATE marks the end of the data stable period.
10. Rising edge of ϕ_2 clock in T3 STATE terminates the DBIN pulse.

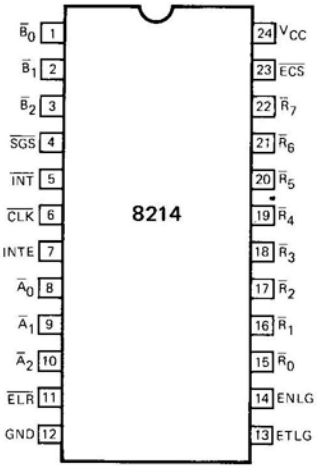




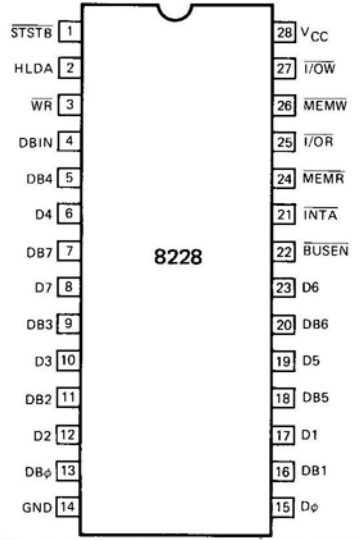
8080 8-BIT CENTRAL PROCESSOR UNIT



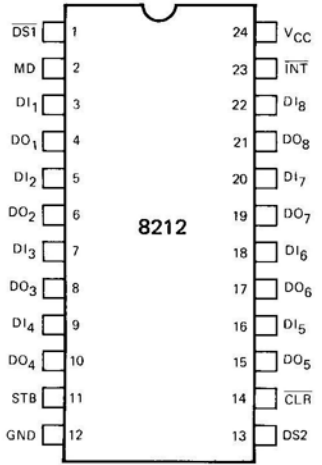
8224 CLOCK GENERATOR



8214 PRIORITY INTERRUPT CONTROL UNIT



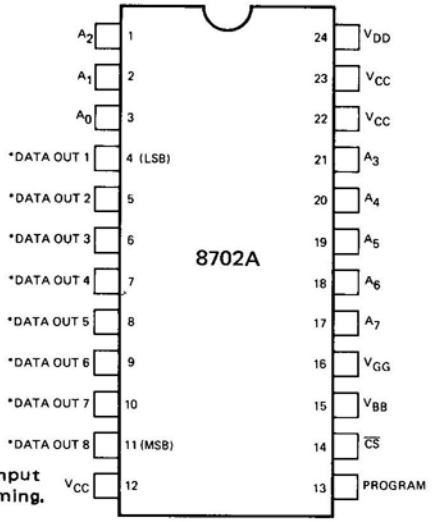
8228
SYSTEM CONTROLLER AND BUS DRIVER FOR 8080 CPU



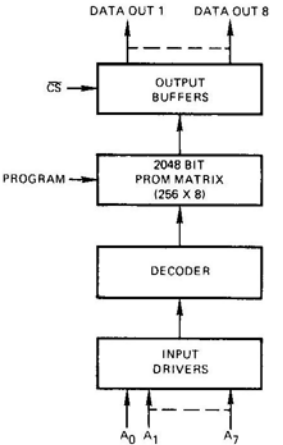
DI_1–DI_8	DATA IN
DO_1–DO_8	DATA OUT
$\overline{DS_1}$–DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

8212 8-BIT INPUT/OUTPUT PORT

This 256 x 8-bit ROM is electrically programmable. Fast turn around and pattern experimentation make it well suited for microcomputer system development. Entirely static, no clocks required, 1.3 μsec access.

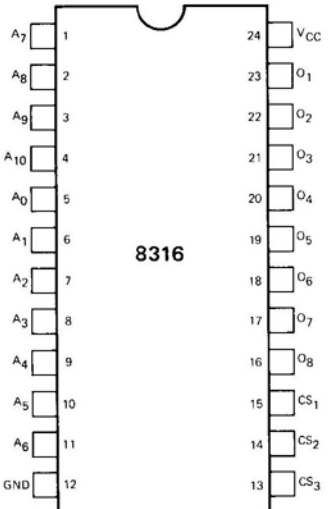


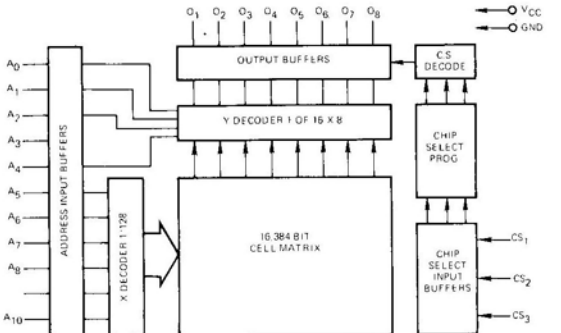
*This pin is the data input lead during programming.



8702A 2K REPROGRAMMABLE PROM

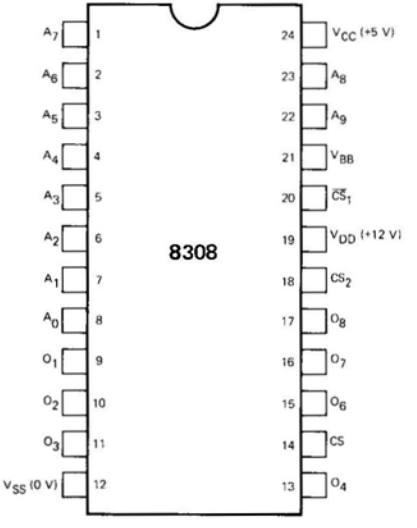
This 16384 bit ROM is organized as 2048 words x 8 bits. High performance, large bit storage, and simple interfacing make the 8316 ideal for microcomputer memory applications. Access time is 2 μsec. I/O are TTL compatible.

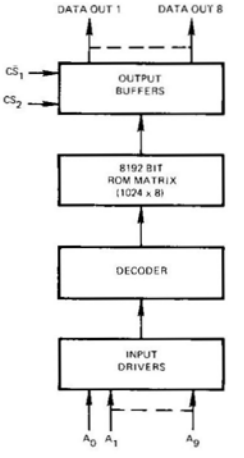




8316 16K STATIC MOS ROM

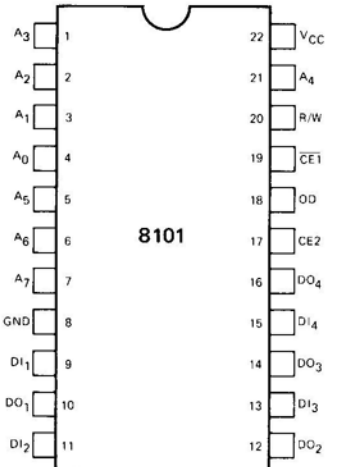
This 8192 bit ROM is organized as 1024 words x 8-bits. High performance, large bit storage and simple interfacing make the 8308 ideal for 8080 systems application. Access time is 450 ns, I/O are TTL compatible.

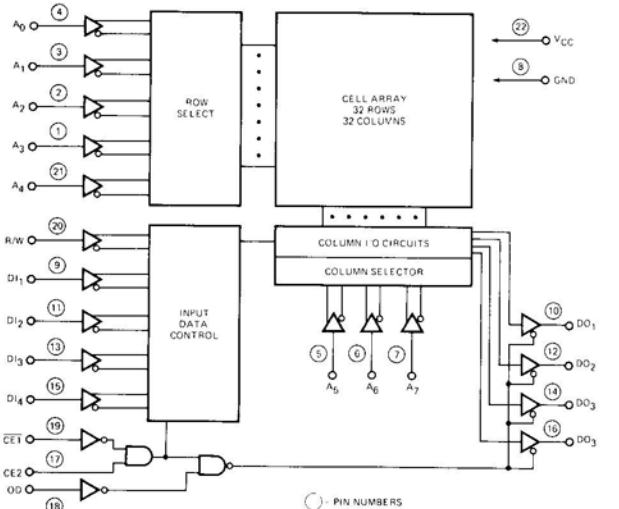




8308 8K STATIC MOS ROM

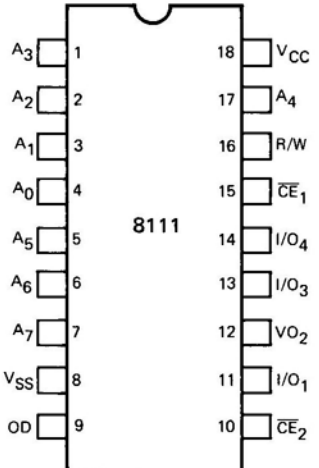
This low cost 256 word by 4 bits MOS RAM uses normally off N-channel MOS devices. With full DC stable (static circuitry, no clocks or refreshing is needed. Data is read out nondestructively. TTL compatible, 850 ns access time.

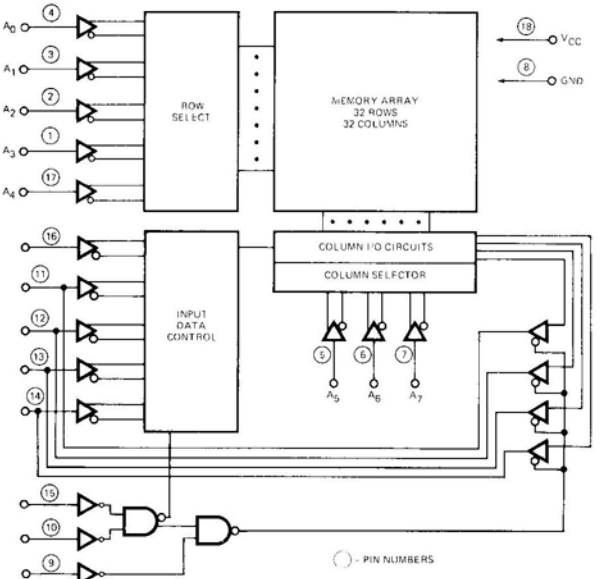




8101 1024 BIT (256 X 4) STATIC MOS RAM WITH SEPARATE I/O

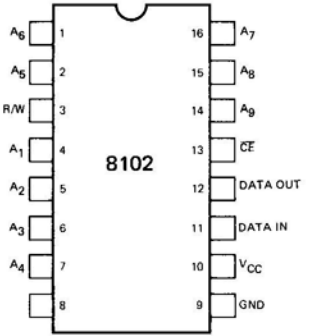
This low cost 256 x 4 bit RAM uses normally off N-channel MOS devices with full DC stable (static) circuitry, no clocks or refreshing is needed. Data is read out non-destructively. TTL compatible, 850 ns access time.

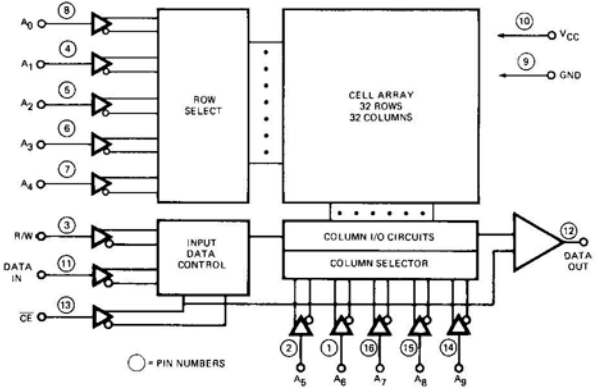




8111 1024 BIT (256 X 4) STATIC MOS RAM WITH COMMON I/O AND OUTPUT DISABLE

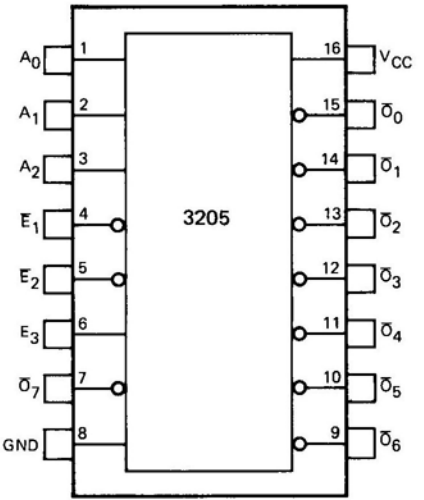
This low cost 1024 x 1 bit RAM has an access time of 1300 ns, with DC stable (static) circuitry, the 8102 needs no clocks or refreshing. Output data is nondestructive and has same polarity as input data.





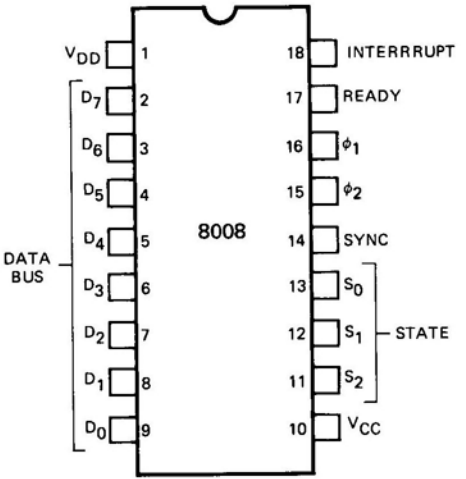
8102 1024 BIT (1K X 1) STATIC MOS RAM

Used for simple system expansion with memory components that have active low chip select input. When the 3205 is enabled one of the eight outputs goes low, thus a single row of a memory system is selected. The 3205 can be cascaded for very large memory systems. 18 ns max. delay over 0°C to 75°C, DTL and TTL compatible.



3205 HIGH SPEED 1 OUT OF 8 BINARY DECODER

This CPU consists of 48 instructions sets with an instruction cycle time of 20 μs. The 8008 directly addresses 16 K x 8 bits of memory (RAM, ROM or SR). The 8008 has interrupt capability for slow I/O peripheral devices. The address stack contains eight 14 bit registers for nesting of up to seven levels of subroutines.



iasis inc.
110 First Street
Los Altos, CA 94022

Printed in U.S.A. PA006-1075/5K