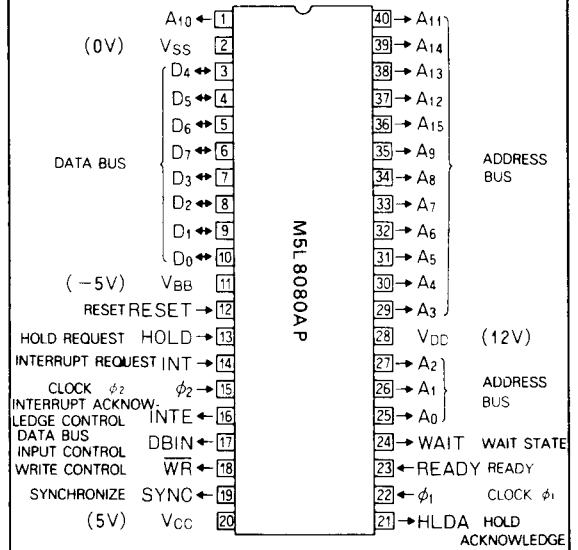


The M5L 8080A P, S is an 8-bit parallel central processing unit (CPU) fabricated on a single chip using a high-speed N-channel silicon-gate MOS process, in a ceramic DIL package.

**FEATURES**

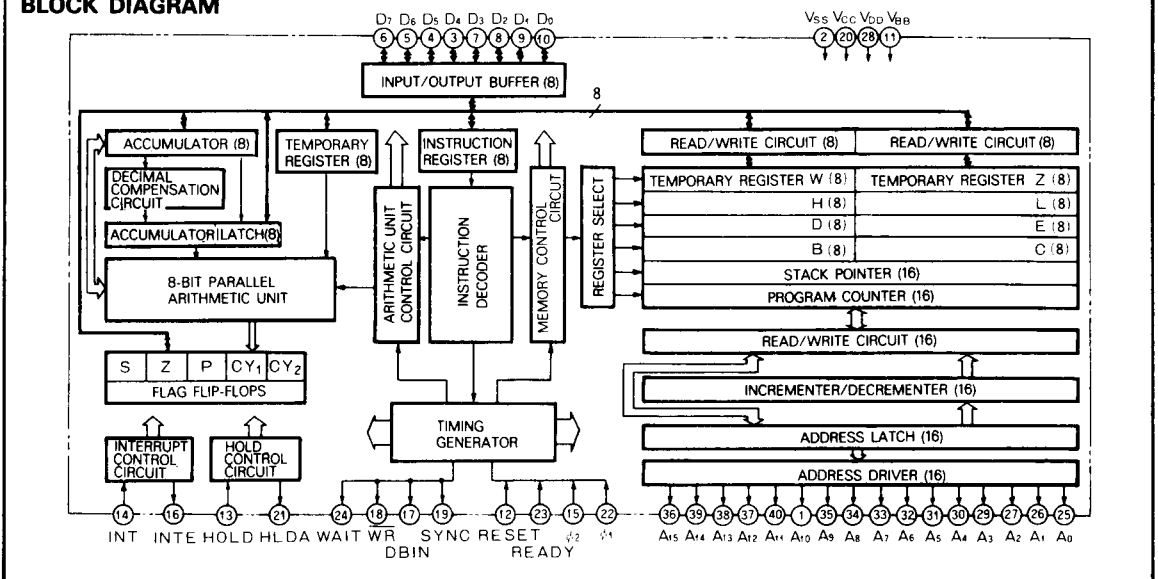
- Basic machine instructions: 78
- Execution time (at clock frequency 2MHz): 2 $\mu$ s
- Directly accessible memory capacity: 65 536 bytes
- Number of input/output ports: 256 each
- Multi-level interruption
- Direct memory access (DMA) operation
- All outputs are fully TTL-compatible; I<sub>OL</sub> = 1.9mA
- Unlimited subroutine nesting
- Interchangeable with the Intel's 8080A in pin-to-pin connections and machine instructions

**PIN CONFIGURATION (TOP VIEW)**



**Outline 40P1 (M5L 8080AP)  
 40S1 (M5L 8080AS)**

**BLOCK DIAGRAM**



**MITSUBISHI MICROCOMPUTERS**  
**M5L 8080A P, S**

**8-BIT PARALLEL CPU**

**PIN DESCRIPTIONS**

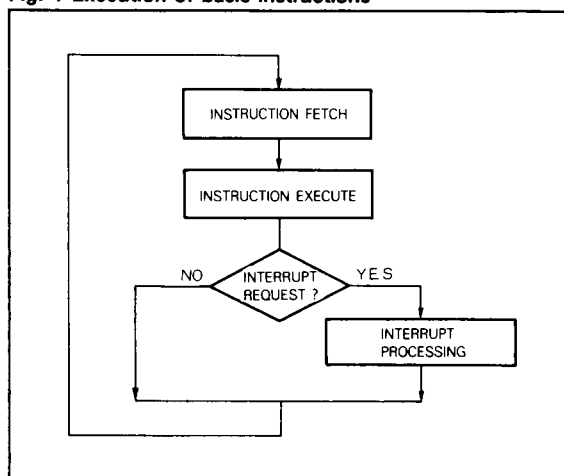
Pin	Name	Input or output	Function significance
A <sub>0</sub> } A <sub>15</sub>	Address bus	Out	Provides the address signal to external memory up to 65 536 bytes or denotes the I/O device number for up to 256 input and 256 output devices. Address terminals are three-state, and remain in the floating state during the HLT instruction execute cycle T <sub>WH</sub> or in the hold state.
D <sub>0</sub> } D <sub>7</sub>	Data bus	In/Out	Provides bidirectional transfer of instructions and data between CPU and the external memory or the I/O ports. Status signals are also transferred. When $\overline{WR}$ is low, data goes to memory or output ports. When DBIN is high, the data is received by the CPU. The status signals are sent on the data bus, synchronizing with SYNC. Like the address bus, the data bus maintains the floating state during the HLT instruction execute cycle (T <sub>WH</sub> ) and in the hold state.
SYNC	Synchronizing signal	Out	Indicates the beginning of machine cycles M <sub>1</sub> through M <sub>5</sub> . The status signals for each cycle are sent out on the data bus while SYNC is active, and are latched in the external registers during SYNC.
DBIN	Data bus input control signal	Out	Indicates to the external circuits that the data bus is in the input mode, in which the CPU receives instructions or data from memory or input ports. The CPU receives instructions or data on the data bus when DBIN is high.
READY	Ready signal	In	Indicates to the CPU that data from memory or input/output ports is valid on the data bus. When the READY signal is not high in the T <sub>2</sub> state, the CPU will enter a waiting state (T <sub>W</sub> ) and the WAIT signal goes high. When READY is high, its state advances from T <sub>2</sub> or T <sub>W</sub> to T <sub>3</sub> . This READY signal is used to synchronize the CPU with slower memory or input/output ports.
WAIT	Wait state signal	Out	Indicates that the CPU has entered a waiting state. When the WAIT signal is high, the CPU is in a waiting state (T <sub>W</sub> ) and the output on the address bus and the data bus is kept stable.
$\overline{WR}$	Write control signal	Out	Indicates timing of a data write-in operation to memory or output ports. When $\overline{WR}$ is low, data on the data bus is valid; when the WAIT signal is high, it is kept low.
HOLD	Hold request signal	In	When READY is high, the CPU enters the hold state provided that: <ul style="list-style-type: none"> <li>· the CPU is in the HLD instruction execute state (T<sub>WH</sub>).</li> <li>· the CPU is in the T<sub>2</sub> or T<sub>W</sub> state and the READY signal is high.</li> </ul> When the CPU is in the hold state, the data bus and the address bus will be in the floating state, and will be used with the memory or input/output ports regardless of CPU operation.
HLDA	Hold acknowledge signal	Out	When high, indicates that the CPU is in the hold state and the address bus and the data bus will be in the floating state.
INTE	Interrupt enable control signal	Out	When high, indicates that an interruption will be accepted by the CPU. It is set to high by instruction EI and is reset to low by instruction DI. It is automatically reset to low at state T <sub>1</sub> of machine cycle M <sub>1</sub> when an interrupt is accepted, and is also reset by the RESET signal.
INT	Interrupt request signal	In	Indicates to the CPU M5L 8080AP that an interrupt is being requested. When the INT is high, the interrupt request will be accepted by the CPU unless HLDA is high or INTE is low. If INT is accepted, INTE will go low and status information INTA will be transferred to the data bus as an interrupt request signal.
RESET	Reset signal	In	When high, the program counter is reset to '0' and instruction NOP is set to the instruction register. INTE is reset to low, and the CPU will not accept interrupts. While RESET is high, the address bus and the data bus remain in the floating state; when RESET goes low, the program will start at location 0. The data registers, accumulator, stack pointer and flag flip-flops are not reset by this signal. No synchronization is necessary for the RESET signal, but the high level must be kept for a minimum of 3 clock cycles.

**BASIC TIMING**

Execution of instructions proceeds in two stages: 1) fetch, and 2) analyze and execute.

Fig. 1 shows the consecutive relationship between stages 1 and 2, after which it is determined whether or not there has been an interrupt request. If there has not, the next instruction is fetched immediately; if there has, it is fetched after completing the necessary interrupt processing. One cycle of this loop completes the execution of one instruction.

**Fig. 1 Execution of basic instructions**



There are five machine cycles ( $M_1, M_2, M_3, M_4$  and  $M_5$ ) and the fetching, analysis, and execution of a single instruction requires from 1 to 5 machine cycles.

Each cycle consists of from three to five states ( $T_1, T_2, T_3, T_4$  and  $T_5$ ), the actual number depending upon the instruction being executed. The duration of one state is defined by successive low-to-high transitions of the  $\phi_1$  clock. (500ns at a clock frequency of 2MHz).

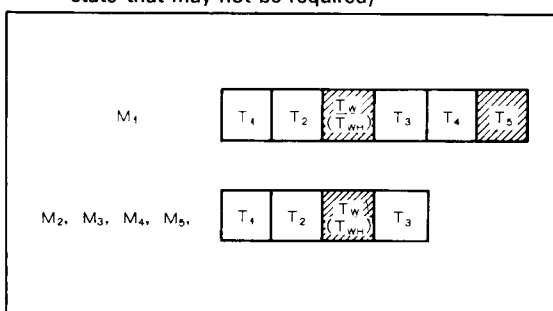
There is also another state,  $T_W$ , situated between  $T_2$  and  $T_3$  (see Fig. 2) and controlled by the external signals READY and HOLD, and instruction HLT. The duration of  $T_W$  is an integral multiple of the clock cycle.

The first machine cycle ( $M_1$ ) in every instruction cycle is a fetch cycle, and the address for memory read is sent on the address bus.  $M_1$  is composed of states  $T_1 \sim T_4$  or  $T_1 \sim T_5$ , as shown in Fig. 2. Machine cycles  $M_2, M_3, M_4$  and  $M_5$  are

usually composed of three states ( $T_1 \sim T_3$ ), with the exception of the instruction XTHL, which requires five states:  $T_1 \sim T_5$ .

When the clock period is 500ns and there is no  $T_W$ ,  $M_1$  requires  $2\mu s$  or  $2.5\mu s$ , and the other machine cycles require  $1.5\mu s$  to execute an instruction. When  $T_W$  exists, the execution time increases accordingly. Since the minimum instruction cycle requires four states ( $T_1 \sim T_4$ ) of machine cycle  $M_1$ , the minimum instruction execution time is  $2\mu s$ .

**Fig. 2 Machine cycle states (hatched blocks indicate a state that may not be required)**



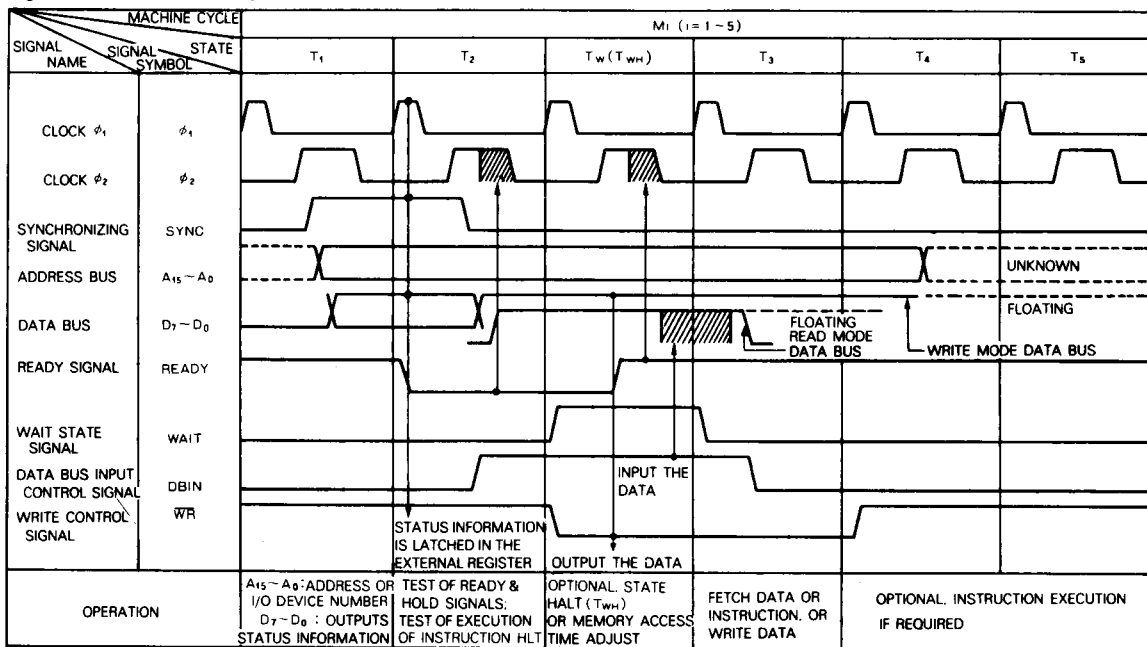
**INTERRUPT**

When an interrupt is requested, the decision whether to accept it or not is taken after the instruction in progress is completed; that is, during the last state of the last machine cycle.

When interrupt is requested and the CPU is in the interrupt-enable state (signal INTE is high), the CPU accepts the interrupt and begins a special interrupt machine cycle  $M_1$  in which the program counter is not incremented and the CPU sends out status information INTA (the interrupt acknowledge signal). During state  $T_3$  of special interrupt machine cycle  $M_1$ , the external interrupt control circuit sends the interrupt instruction corresponding to interrupt factors on the data bus, and the CPU fetches and executes this instruction. This instruction is a special one-byte call (instruction RST) or a special three-byte call (instruction CALL) which facilitates the processing of interrupts.

**8-BIT PARALLEL CPU**

**Fig. 3 Basic instruction cycle**



- Besides the states shown in Fig.3, there is a state T<sub>H</sub>, in which the CPU stays in the hold state after the machine cycle.
- States T<sub>w</sub>, T<sub>4</sub> and T<sub>5</sub> are optional.

**Table 1 Status information**

Data bus	Signal symbol	Status information designation	Function
D <sub>0</sub>	INTA	Interrupt acknowledge	Goes high when the CPU accepts the interrupt request signal from the INT terminal.
D <sub>1</sub>	W <sub>0</sub>	Write mode	Goes high when the current machine cycle is in a read mode, and falls when in a write (output) mode.
D <sub>2</sub>	STACK	Stack	Goes high when the address bus holds the pushdown stack address from the stack pointer.
D <sub>3</sub>	HLTA	HLT instruction acknowledge	Goes high when the CPU executes the HLT instruction and maintains the halt state.
D <sub>4</sub>	OUT	Output instruction acknowledge	Goes high when the address bus contains the address of an output device and the data bus contains the output data. (The address of an output device is contained simultaneously in the upper 8 bits and the lower 8 bits of the address bus.)
D <sub>5</sub>	M <sub>i</sub>	M <sub>i</sub> status	Goes high when the CPU is in the fetch cycle for the first byte of an instruction.
D <sub>6</sub>	INP	Input instruction acknowledge	Goes high when the address bus contains the address of an input device and the data bus receives the input data. (The address of an output device is contained simultaneously in the upper 8 bits and the lower 8 bits of the address bus.)
D <sub>7</sub>	MEMR	Memory read	Goes high when the data bus is used for memory read data.

- Hatched portions indicate periods during which input data should be kept stable.
- The address data is valid during the period designated by solid lines.
- The period of T<sub>w</sub> depends on the condition of the READY signal.

**STATUS INFORMATION**

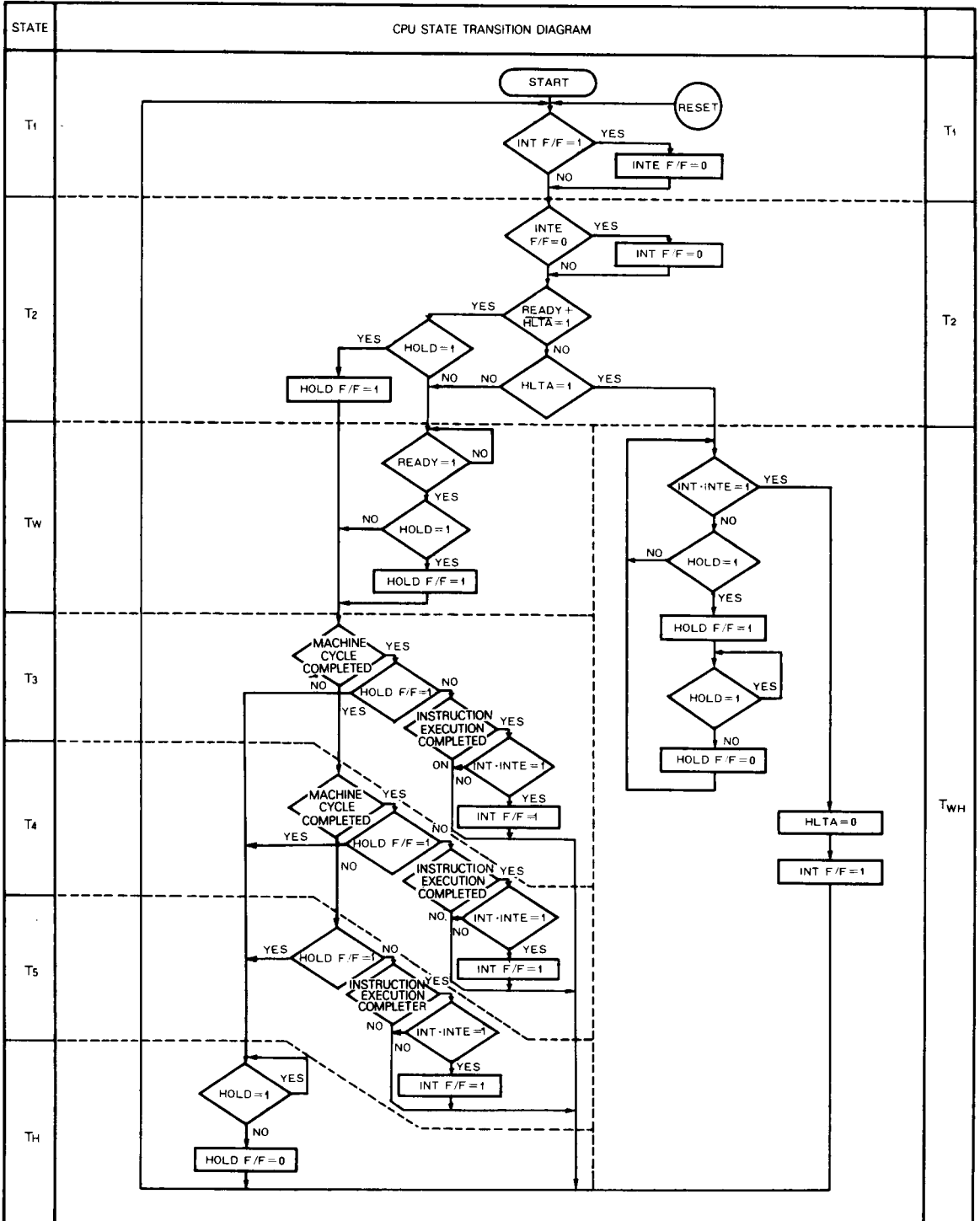
The M58710S sends out 8 bits of status information on data bus (D<sub>7</sub>~D<sub>0</sub>) at the first state of each machine cycle (M<sub>i</sub>·T<sub>1</sub>) synchronizing with signal SYNC that indicates the function of each machine cycle. The status signal will be latched in the external register by signal SYNC· $\phi_1$ . Table 1 gives the functions of the status information that will be sent out on the data bus.

**Table 2 Status**

Status information	Mode No.										
		1	2	3	4	5	6	7	8	9	10
Data bus bit	Status signal name	Instruction fetch	Memory read	Stack read	Input read	Interrupt acknowl.	Halt acknowledge	Interrupt acknowl. while halt	Memory write	Stack write	Output write
D <sub>0</sub>	INTA	0	0	0	0	1	0	1	0	0	0
D <sub>1</sub>	W <sub>0</sub>	1	1	1	1	1	1	1	0	0	0
D <sub>2</sub>	STACK	0	0	1	0	0	0	0	0	0	1
D <sub>3</sub>	HLTA	0	0	0	0	0	1	1	0	0	0
D <sub>4</sub>	OUT	0	0	0	0	0	0	0	0	0	1
D <sub>5</sub>	M <sub>i</sub>	1	0	0	0	1	0	1	0	0	0
D <sub>6</sub>	INP	0	0	0	1	0	0	0	0	0	0
D <sub>7</sub>	MEMR	1	1	1	0	0	1	0	0	0	0

**8-BIT PARALLEL CPU**

**CPU STATE TRANSITION DIAGRAM**



8-BIT PARALLEL CPU

MACHINE INSTRUCTIONS

Item class	Item	Mnemonic	Instruction code				Format notation	No. of bytes	No. of states	No. of cycles	Functions	Flags			Address bus		Data bus					
			D7 D6	D5 D4 D3	D2 D1 D0	Format						S	Z	P	CY2	CY1	Contents	Mach cycle	Contents	I/O	Mach cycle†	
Data transfer	MOV	r1, r2	01	DDD	SSS	8	5	1	1	(r1) ← (r2)	Where, M = (H) (L)	X	X	X	X	M	M4	(r1)	0	M4		
	MOV	M, r	01	110	SSS	8	7	1	2	(M) ← (r)	Where, M = (H) (L)	X	X	X	X	M	M4	(M)	1	M4		
	MOV	r, M	01	DDD	110	8	7	1	2	(r) ← (M)	Where, M = (H) (L)	X	X	X	X	M	M4	(M)	1	M4		
	MVI	r, n	00	DDD	110	10	7	2	2	(r) ← n	Where, M = (H) (L)	X	X	X	X	M	M4	(M)	1	M4		
	MVI	M, n	00	110	110	10	3	6	2	(M) ← n	Where, M = (H) (L)	X	X	X	X	M	M5	(B2)	1	M5		
	LXI	B, m	00	000	001	10	0	1	3	(C) ← (B2) (B) ← (B3)	Where, m = (B3) (B2)	X	X	X	X			(B2)	1	M2		
	LXI	D, m	00	010	001	10	1	1	3	(E) ← (B2) (D) ← (B3)	Where, m = (B3) (B2)	X	X	X	X			(B2)	1	M2		
	LXI	H, m	00	100	001	10	2	1	3	(L) ← (B2) (H) ← (B3)	Where, m = (B3) (B2)	X	X	X	X			(B2)	1	M2		
	LXI	SP, m	00	110	001	10	3	1	3	(SP) ← m	Where, m = (B3) (B2)	X	X	X	X			(B2)	1	M2		
	SPHL		11	111	001	10	F	9	5	1	(SP) ← (H) (L)		X	X	X	X			(B2)	1	M2	
	STAX	B	00	000	010	10	0	2	7	1	2	((B) (C)) ← (A)		X	X	X	X	(B) (C)	M4	(A)	0	M4
	STAX	D	00	010	010	10	1	2	7	1	2	((D) (E)) ← (A)		X	X	X	X	(D) (E)	M4	(A)	0	M4
	LDA	B	00	001	010	10	0	A	7	1	2	(A) ← ((B) (C))		X	X	X	X	(B) (C)	M4	((B) (C))	1	M4
	LDA	D	00	011	010	10	1	A	7	1	2	(A) ← ((D) (E))		X	X	X	X	(D) (E)	M4	((D) (E))	1	M4
	STA	m	00	110	010	10	3	2	13	3	4	(m) ← (A)		X	X	X	X	m	M4	(A)	0	M4
LDA	m	00	111	010	10	3	A	13	3	4	(A) ← (m)		X	X	X	X	m	M4	(m)	1	M4	
SHLD	m	00	100	010	10	2	2	16	3	5	(m) ← (L) (m+1) ← (H)		X	X	X	X	m	M4	(L)	0	M4	
LHLD	m	00	101	010	10	2	A	16	3	5	(L) ← (m) (H) ← (m+1)		X	X	X	X	m	M4	(H)	0	M4	
XCHG		11	101	011	10	E	4	1	1	1	(H) (L) ↔ (D) (E)		X	X	X	X						
XTHL		11	100	011	10	E	3	18	1	1	(H) (L) ↔ ((SP) + 1) ((SP))		X	X	X	X	(SP)	M2	((SP))	1	M2	
ADD	r	10	000	SSS	8	4	1	1	1	1	(A) ← (A) + (r)		0	0	0	0						
ADD	M	10	000	110	8	6	7	1	2	2	(A) ← (A) + (M)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
ADI	n	11	000	110	8	C	6	7	2	2	(A) ← (A) + n	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
ADC	r	10	001	SSS	8	E	7	1	2	2	(A) ← (A) + (r) + (CY2)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
ADC	M	10	001	110	8	E	7	1	2	2	(A) ← (A) + (M) + (CY2)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
ACI	n	11	001	110	8	C	6	7	2	2	(A) ← (A) + n + (CY2)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
DAD	B	00	011	001	10	0	9	10	1	3	(H) (L) ← (H) (L) + (B) (C)		X	X	X	X						
DAD	D	00	011	001	10	1	9	10	1	3	(H) (L) ← (H) (L) + (D) (E)		X	X	X	X						
DAD	H	00	101	001	10	2	9	10	1	3	(H) (L) ← (H) (L) + (H) (L)		X	X	X	X						
DAD	SP	00	111	001	10	3	9	10	1	3	(H) (L) ← (H) (L) + (SP)		X	X	X	X						
SUB	r	10	010	SSS	8	4	1	1	1	1	(A) ← (A) - (r)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
SUB	M	10	010	110	8	6	7	1	2	2	(A) ← (A) - (M)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
SUB	n	11	010	110	8	D	6	7	2	2	(A) ← (A) - n	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
SBB	r	10	011	SSS	8	4	1	1	1	1	(A) ← (A) - (r) - (CY2)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
SBB	M	10	011	110	8	6	7	1	2	2	(A) ← (A) - (M) - (CY2)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
SBI	n	11	011	110	8	D	E	7	2	2	(A) ← (A) - n - (CY2)	Where, M = (H) (L)	0	0	0	0	M	M4	(B2)	1	M4	
ANA	r	10	100	SSS	8	4	1	1	1	1	(A) ← (A) ∧ (r)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
ANA	M	10	100	110	8	6	7	1	2	2	(A) ← (A) ∧ (M)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
ANI	n	11	100	110	8	E	6	7	2	2	(A) ← (A) ∧ n	Where, M = (H) (L)	0	0	0	0	M	M4	(B2)	1	M4	
XRA	r	10	101	SSS	8	4	1	1	1	1	(A) ← (A) ⊕ (r)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
XRA	M	10	101	110	8	A	E	7	1	2	(A) ← (A) ⊕ (M)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
XRI	n	11	101	110	8	E	E	7	2	2	(A) ← (A) ⊕ n	Where, M = (H) (L)	0	0	0	0	M	M4	(B2)	1	M4	
ORA	r	10	110	SSS	8	4	1	1	1	1	(A) ← (A) ∨ (r)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
ORA	M	10	110	110	8	6	7	1	2	2	(A) ← (A) ∨ (M)	Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
ORI	n	11	110	110	8	F	6	7	2	2	(A) ← (A) ∨ n	Where, M = (H) (L)	0	0	0	0	M	M4	(B2)	1	M4	
CMP	r	10	111	SSS	8	4	1	1	1	1	(A) - (r)	Compare: Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
CMP	M	10	111	110	8	6	7	1	2	2	(A) - (M)	Compare: Where, M = (H) (L)	0	0	0	0	M	M4	(M)	1	M4	
CPI	n	11	111	110	8	F	E	7	2	2	(A) - n	Compare: Where, M = (H) (L)	0	0	0	0	M	M4	(B2)	1	M4	
Register increment/decrement	INR	r	00	DDD	100	3	4	5	1	1	(r) ← (r) + 1	Where, M = (H) (L)	0	0	0	X	M	M4	(M)	1	M4	
	INR	M	00	110	100	3	4	10	1	3	(M) ← (M) + 1	Where, M = (H) (L)	0	0	0	X	M	M4	(M)	1	M4	
	DCR	r	00	DDD	101	3	5	5	1	1	1	(r) ← (r) - 1	Where, M = (H) (L)	0	0	0	X	M	M4	(M)	1	M4
	DCR	M	00	110	101	3	5	10	1	3	3	(M) ← (M) - 1	Where, M = (H) (L)	0	0	0	X	M	M4	(M)	1	M4
	INX	B	00	000	011	10	0	3	5	1	1	(B) (C) ← (B) (C) + 1		X	X	X	X					
	INX	D	00	010	011	10	1	3	5	1	1	(D) (E) ← (D) (E) + 1		X	X	X	X					
	INX	H	00	100	011	10	2	3	5	1	1	(H) (L) ← (H) (L) + 1		X	X	X	X					
INX	SP	00	110	011	10	3	3	5	1	1	(SP) ← (SP) + 1		X	X	X	X						
DCX	B	00	001	011	10	0	B	5	1	1	(B) (C) ← (B) (C) - 1		X	X	X	X						
DCX	D	00	011	011	10	1	B	5	1	1	(D) (E) ← (D) (E) - 1		X	X	X	X						
DCX	H	00	101	011	10	2	B	5	1	1	(H) (L) ← (H) (L) - 1		X	X	X	X						
DCX	SP	00	111	011	10	3	B	5	1	1	(SP) ← (SP) - 1		X	X	X	X						
Rotate & shift contents of accumulator	RLC		00	000	111	0	7	4	1	1	Left shift Cy2		X	X	X	X						
	RRC		00	001	111	0	F	4	1	1	Right shift Cy2		X	X	X	X						
	RAL		00	010	111	0	7	4	1	1	Left shift Cy2		X	X	X	X						
	RAR		00	011	111	0	F	4	1	1	Right shift Cy2		X	X	X	X						
Accumulator complement	CMA		00	101	111	2	F	4	1	1	(A) ← (A)		X	X	X	X						
	DAA		00	100	111	2	7	4	1	1	Results of binary addition are adjusted to BCD		X	X	X	X						
Carry set	STC		00	110	111	3	7	4	1	1	(CY2) ← 1		X	X	X	X						
	CMC		00	111	111	3	F	4	1	1	(CY2) ← (CY2)		X	X	X	X						

\*: State is T1 †: State is T2

Item Instr. class.	Mnemonic	Instruction code					16- bit format	No. of states	No. of bytes	No. of cycles	Functions	Flags		Address bus		Data bus							
		D7 D6	D5 D4 D3	D2 D1 D0	Imaginary format	S						Z	P	CY2 CY1	Contents	Mach. cycle	Contents	I/O	Mach. cycle**				
Jump	JMP	m	1 1	0 0 0	0 1 1	C 3	10	3	3	(PC) ← m	x	x	x	x			<B2> <B3>	1 1	M2 M3				
	PCHL	m	1 1	1 0 1	0 0 1	E 9	5	1	1	(PC) ← (H) (L)	x	x	x	x									
	JC	m	1 1	0 1 1	0 1 0	D A	10	3	3	(CY2) = 1	x	x	x	x									
	JNC	m	1 1	0 1 0	0 1 0	D 2	10	3	3	(CY2) = 0	x	x	x	x		If condition is true.							
	JZ	m	1 1	0 0 1	0 1 0	C A	10	3	3	(Z) = 1	x	x	x	x			<B2> <B3>	1 1	M2 M3				
	JNZ	m	1 1	0 0 0	0 1 0	C 2	10	3	3	(Z) = 0	x	x	x	x									
	JP	m	1 1	1 1 0	0 1 0	F 2	10	3	3	(S) = 0	x	x	x	x		If condition is false.							
	JM	m	1 1	1 1 1	0 1 0	F A	10	3	3	(S) = 1	x	x	x	x									
	JPE	m	1 1	1 0 1	0 1 0	E A	10	3	3	(P) = 1	x	x	x	x									
JPO	m	1 1	1 0 0	0 1 0	E 2	10	3	3	(P) = 0	x	x	x	x										
Subroutine call	CALL	m	1 1	0 0 1	1 0 1	C D	17	3	5	((SP) - 1) ((SP) - 2) ← (PC) + 3, (PC) ← m (SP) ← (SP) - 2	x	x	x	x			<B2> <B3>	1 1	M2 M3				
	RST	n	1 1	A A A	1 1 1		11	1	3	((SP) - 1) ((SP) - 2) ← (PC) + 1, (PC) ← n × 8, (SP) ← (SP) - 2 Where 0 ≤ n ≤ 7	x	x	x	x			(SP) - 1 (SP) - 2 (SP) - 1 (SP) 2	M4 M5 M4 M5	<(PC) + D7 - 4> <(PC) + D7 - 1> <(PC) + D7 - 0> <(PC) + D7 - 0>	0 0 0 0	M4 M5 M4 M5		
	CC	m	1 1	0 1 1	1 0 0	D C	17	11	3	5/3 (CY2) = 1	x	x	x	x									
	CNC	m	1 1	0 1 0	1 0 0	D 4	17	11	3	5/3 (CY2) = 0	x	x	x	x		If condition is true.							
	CZ	m	1 1	0 0 1	1 0 0	C C	17	11	3	5/3 (Z) = 1	x	x	x	x		((SP) - 1) ((SP) - 2) ← (PC) + 3			<B2> <B3>	1 1	M2 M3		
	CNZ	m	1 1	0 0 0	1 0 0	C 4	17	11	3	5/3 (Z) = 0	x	x	x	x		(PC) ← m (SP) ← (SP) - 2			(SP) - 1 (SP) - 2	M4 M5	<(PC) + D7 - 4> <(PC) + D7 - 4>	0 0	M4 M5
	CP	m	1 1	1 1 0	1 0 0	F 4	17	11	3	5/3 (S) = 0	x	x	x	x									
	CM	m	1 1	1 1 1	1 0 0	F C	17	11	3	5/3 (S) = 1	x	x	x	x		If condition is false							
	CPE	m	1 1	1 0 1	1 0 0	E C	17	11	3	5/3 (P) = 1	x	x	x	x		(PC) ← (PC) + 3							
CPO	m	1 1	1 0 0	1 0 0	E 4	17	11	3	5/3 (P) = 0	x	x	x	x										
Return	RET	n	1 1	0 0 1	0 0 1	C B	10	1	3	(PC) ← ((SP) + 1) ((SP)), (SP) ← (SP) + 2	x	x	x	x			(SP) + 1 (SP) + 1	M4 M5	((SP) + 1) ((SP) + 1)	1 1	M4 M5		
	RC	n	1 1	0 1 1	0 0 0	D B	11	5	1	3/1 (CY2) = 0	x	x	x	x		If condition is true							
	RNC	n	1 1	0 1 0	0 0 0	D 0	11	5	1	3/1 (CY2) = 0	x	x	x	x		If condition is true							
	RZ	n	1 1	0 0 1	0 0 0	C B	11	5	1	3/1 (Z) = 1	x	x	x	x		(PC) ← ((SP) + 1) ((SP))			(SP) M4 (SP) + 1 M5	M4 M5	((SP) + 1) ((SP) + 1)	1 1	M4 M5
	RNZ	n	1 1	0 0 0	0 0 0	C 0	11	5	1	3/1 (Z) = 0	x	x	x	x		(SP) ← (SP) + 2							
	RP	n	1 1	1 1 0	0 0 0	F 0	11	5	1	3/1 (S) = 0	x	x	x	x									
	RM	n	1 1	1 1 1	0 0 0	F B	11	5	1	3/1 (S) = 1	x	x	x	x		If condition is false							
	RPE	n	1 1	1 0 1	0 0 0	E B	11	5	1	3/1 (P) = 1	x	x	x	x		(PC) ← (PC) + 1							
	RPO	n	1 1	1 0 0	0 0 0	E 0	11	5	1	3/1 (P) = 0	x	x	x	x		If condition is false.							
Input/output control	IN	n	1 1	0 1 1	0 1 1	D B	10	2	3	(A) ← (Input buffer) ← (Input device of number n).	x	x	x	x			<B2>	M5	(Input data)	0	M4		
	OUT	n	1 1	0 1 0	0 1 1	D 3	10	2	3	(Output device of number n) ← (A) (Input data)	x	x	x	x			<B2> <B2>	M5	(Input data)	0	M4		
Interrupt control	E I	n	1 1	1 1 1	0 1 1	F B	4	1	1	(INTE) ← 1	x	x	x	x			<B2>	M5	(A)	0	M5		
	D I	n	1 1	1 1 0	0 1 1	F 3	4	1	1	(INTE) ← 0	x	x	x	x									
Stack control	PUSH PSW	n	1 1	1 1 0	1 0 1	F 5	11	1	3	((SP) - 1) ← (A), ((SP) - 2) ← (F) (SP) ← (SP) - 2	x	x	x	x			(SP) 1 (SP) - 2	M4 M5	(A) (F)	0 0	M4 M5		
	PUSH B	n	1 1	0 0 0	1 0 1	C 5	11	1	3	((SP) - 1) ← (B), ((SP) - 2) ← (C) (SP) ← (SP) - 2	x	x	x	x			(SP) - 1 (SP) - 2	M4 M5	(B) (C)	0 0	M4 M5		
	PUSH D	n	1 1	0 1 0	1 0 1	D 5	11	1	3	((SP) - 1) ← (D), ((SP) - 2) ← (E) (SP) ← (SP) - 2	x	x	x	x			(SP) - 1 (SP) - 2	M4 M5	(D) (E)	0 0	M4 M5		
	PUSH H	n	1 1	1 0 0	1 0 1	E 5	11	1	3	((SP) - 1) ← (H), ((SP) - 2) ← (L) (SP) ← (SP) - 2	x	x	x	x			(SP) - 1 (SP) - 2	M4 M5	(H) (L)	0 0	M4 M5		
	POP PSW	n	1 1	1 1 0	0 0 1	F 1	10	1	3	(F) ← ((SP)), (A) ← ((SP) + 1) (SP) ← (SP) + 2	0	0	0	0			(SP) (SP) + 1	M4 M5	((SP) + 1) ((SP) + 1)	1 1	M4 M5		
	POP B	n	1 1	0 0 0	0 0 1	C 1	10	1	3	(B) ← ((SP)), (C) ← ((SP) + 1) (SP) ← (SP) + 2	x	x	x	x			(SP) (SP) + 1	M4 M5	((SP) + 1) ((SP) + 1)	1 1	M4 M5		
POP D	n	1 1	0 1 0	0 0 1	D 1	10	1	3	(E) ← ((SP)), (D) ← ((SP) + 1) (SP) ← (SP) + 2	x	x	x	x			(SP) (SP) + 1	M4 M5	((SP) + 1) ((SP) + 1)	1 1	M4 M5			
POP H	n	1 1	1 0 0	0 0 1	E 1	10	1	3	(L) ← ((SP)), (H) ← ((SP) + 1) (SP) ← (SP) + 2	x	x	x	x			(SP) (SP) + 1	M4 M5	((SP) + 1) ((SP) + 1)	1 1	M4 M5			
Others	HLT	n	0 0	1 1 0	1 1 0	7 6	7	1	1	(PC) ← (PC) + 1	x	x	x	x									
	NOP	n	0 0	0 0 0	0 0 0	0 0	4	1	1	(PC) ← (PC) + 1	x	x	x	x									

\*: State is T1. \*\*: State is T2.



Symbol	Meaning	Symbol	Meaning	Symbol	Meaning
r	Register	S S S	Bit pattern designating register or memory.	←	Data is transferred in direction shown
m	Two-byte data			( )	Contents of register or memory location
n	One-byte data			v	Inclusive OR
<B2>	Second byte of instruction			∨	Exclusive OR
<B3>	Third byte of instruction	∧	Logical AND		
AAA	Binary representation for RST instruction n	—	1's complement		
F	8-bit data from the most to the least significant bit S, Z, 0, CY1, 0, P, 1, CY2	D D D	Where, M = (H) (L)	x	Content of flag is not changed after execution
PC	Program counter	L	1 0 1	○	Content of flag is set or reset after execution
SP	Stack pointer	M	1 1 0	⊖	Input mode
		A	1 1 1	○	Output mode

**8-BIT PARALLEL CPU**

**INSTRUCTION CODE LIST**

D <sub>7</sub> - D <sub>4</sub> Hexadecimal notation	D <sub>3</sub> - D <sub>0</sub>	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NOP	(NOP)	(NOP)	(NOP)	MOV B, B	MOV D, B	MOV H, B	MOV M, B	ADD B	SUB B	ANA B	ORA B	RNZ	RNC	RFD	RP
0001	1	LXI B	LXI D	LXI H	LXI SP	MOV B, C	MOV D, C	MOV H, C	MOV M, C	ADD C	SUB C	ANA C	ORA C	PCP B	POP D	PCP H	POP PSW
0010	2	STAX B	STAX D	SHLD	STA	MOV B, D	MOV D, D	MOV H, D	MOV M, D	ADD D	SUB D	ANA D	ORA D	JNZ	JNC	JPO	JP
0011	3	INX B	INX D	INX H	INX SP	MOV B, E	MOV D, E	MOV H, E	MOV M, E	ADD E	SUB E	ANA E	ORA E	JMP	OUT	XT-L	DI
0100	4	INR B	INR D	INR H	INR M	MOV B, H	MOV D, H	MOV H, H	MOV M, H	ADD H	SUB H	ANA H	ORA H	CNZ	CNC	CPO	CP
0101	5	DCR B	DCR D	DCR H	DCR M	MOV B, L	MOV D, L	MOV H, L	MOV M, L	ADD L	SUB L	ANA L	ORA L	PUSH B	PUSH D	PUSH H	PUSH PSW
0110	6	MVI B	MVI D	MVI H	MVI M	MOV B, M	MOV D, M	MOV H, M	HLT	ADD M	SUB M	ANA M	ORA M	API	SUI	ANI	ORI
0111	7	RLC	RAL	DAA	STC	MOV B, A	MOV D, A	MOV H, A	MOV M, A	ADD A	SUB A	ANA A	ORA A	RST 0	RST 2	RST 4	RST 6
1000	8	(NOP)	(NOP)	(NOP)	(NOP)	MOV C, B	MOV E, B	MOV L, B	MOV A, B	ADC B	SBB B	XRA B	CMP B	RZ	RC	RPE	RM
1001	9	DAD B	DAD D	DAD H	DAD SP	MOV C, C	MOV E, C	MOV L, C	MOV A, C	ADC C	SBB C	XRA C	CMP C	RET	(RET)	PCHL	SPHL
1010	A	LDAX B	LDAX D	LHLD	LDA	MOV C, D	MOV E, D	MOV L, D	MOV A, D	ADC D	SBB D	XRA D	CMP D	JZ	JC	JPE	JM
1011	B	DCX B	DCX D	DCX H	DCX SP	MOV C, E	MOV E, E	MOV L, E	MOV A, E	ADC E	SBB E	XRA E	CMP E	(JMP)	IN	XCHG	EI
1100	C	INR C	INR E	INR L	INR A	MOV C, H	MOV E, H	MOV L, H	MOV A, H	ADC H	SBB H	XRA H	CMP H	CZ	CC	CPE	CM
1101	D	DCR C	DCR E	DCR L	DCR A	MOV C, L	MOV E, L	MOV L, L	MOV A, L	ADC L	SBB L	XRA L	CMP L	CALL	(CALL)	(CALL)	(CALL)
1110	E	MVI C	MVI E	MVI L	MVI A	MOV C, M	MOV E, M	MOV L, M	MOV A, M	ADC M	SBB M	XRA M	CMP M	ACI	SBI	XRI	CPI
1111	F	RRC	RAR	CMA	CMC	MOV C, A	MOV E, A	MOV L, A	MOV A, A	ADC A	SBB A	XRA A	CMP A	RST 1	RST 3	RST 5	RST 7

This list shows the machine codes and corresponding machine instructions. D<sub>3</sub>~D<sub>0</sub> indicate the lower 4 bits of the machine code and D<sub>7</sub>~D<sub>4</sub> indicate the upper 4 bits. Hexadecimal numbers are also used to indicate this code. The instruction may consist of one, two, or three bytes, but only the first byte is listed.

 indicates a three-byte instruction.  
 indicates a two-byte instruction.  
 ( ) is not a formal instruction, but if this code is accessed, the instruction in parentheses may be executed. This is not, however, guaranteed.

**ABSOLUTE MAXIMUM RATINGS**

Symbol	Parameter	Conditions	Limits	Unit
V <sub>DD</sub>	Supply voltage	With respect to V <sub>BB</sub> (substrate)	-0.3 ~ 20	V
V <sub>CC</sub>	Supply voltage		-0.3 ~ 20	V
V <sub>SS</sub>	Supply voltage		-0.3 ~ 20	V
V <sub>I</sub>	Input voltage		-0.3 ~ 20	V
P <sub>d</sub>	Maximum power dissipation	T <sub>a</sub> = 25°C	1500	mW
T <sub>opr</sub>	Operating free-air temperature range		0 ~ 70	°C
T <sub>stg</sub>	Storage temperature range		-65 ~ 150	°C

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Limits			Unit
		Min	Nom	Max	
V <sub>BB</sub>	Supply voltage	-4.75	-5	-5.25	V
V <sub>CC</sub>	Supply voltage	4.75	5	5.25	V
V <sub>DD</sub>	Supply voltage	11.4	12	12.6	V
V <sub>SS</sub>	Supply voltage		0		V
V <sub>IH</sub>	High-level input voltage	3.3		V <sub>CC</sub> + 1	V
V <sub>IL</sub>	Low-level input voltage	-1		0.8	V
V <sub>IH</sub> (φ)	High-level clock input voltage	9		V <sub>DD</sub> + 1	V
V <sub>IL</sub> (φ)	Low-level clock input voltage	-1		0.8	V
T <sub>opr</sub>	Operating free-air temperature	0		70	°C

**ELECTRICAL CHARACTERISTICS** (T<sub>a</sub> = 0 ~ 70°C, V<sub>DD</sub> = 12V ± 5%, V<sub>CC</sub> = 5V ± 5%, V<sub>BB</sub> = -5V ± 5%, V<sub>SS</sub> = 0V, unless otherwise noted)

Symbol	Parameter	Test conditions	Limits			Unit
			Min	Typ	Max	
V <sub>OL</sub>	Low-level output voltage	I <sub>OL</sub> = 1.9mA, All output			0.45	V
V <sub>OH</sub>	High-level output voltage	I <sub>OH</sub> = -150μA	3.7			V
I <sub>BB</sub>	V <sub>BB</sub> supply current	Operating at t <sub>c</sub> (φ) = 480ns, T <sub>a</sub> = 25°C (Note 2)		-0.01	-1	mA
I <sub>CC</sub>	V <sub>CC</sub> supply current		60	75	mA	
I <sub>DD</sub>	V <sub>DD</sub> supply current		40	70	mA	
I <sub>I</sub>	Input current, except clock and data bus		0 ≤ V <sub>I</sub> ≤ V <sub>CC</sub>			± 10
I <sub>I</sub> (φ)	Clock input current	0 ≤ V <sub>I</sub> (φ) ≤ V <sub>DD</sub>			± 10	μA
I <sub>I</sub> (DB)	Input current, data bus (Note 3)	0 ≤ V <sub>I</sub> (DB) ≤ V <sub>IL</sub> V <sub>IL</sub> ≤ V <sub>I</sub> (DB) ≤ V <sub>CC</sub>			10 -100	μA
I <sub>I</sub> (HOLD)	Input current during hold, address or data bus	At hold state 0.45V ≤ V <sub>O</sub> ≤ V <sub>CC</sub>			10 -100 -2	μA mA
C <sub>I</sub> (φ <sub>1</sub> )	Input capacitance, clock input (φ <sub>1</sub> )	V(φ <sub>1</sub> ) = 0V V(φ <sub>2</sub> ) = 0V V <sub>I</sub> = 0V V <sub>O</sub> = 0V } f = 1MHz, 25mVr.m.s	20	25	25	pF
C <sub>I</sub> (φ <sub>2</sub> )	Input capacitance, clock input (φ <sub>2</sub> )		15	20	20	pF
C <sub>I</sub>	Input capacitance, any input except clock		5	10	10	pF
C <sub>O</sub>	Output capacitance		5	20	20	pF

Note 1 : Current flowing into an IC is positive; out is negative

2 : I<sub>c</sub>(φ) = I<sub>d</sub>(φ<sub>1H</sub>·φ<sub>2</sub>) + I<sub>r</sub>(φ<sub>2</sub>) + I<sub>w</sub>(φ<sub>2</sub>) + I<sub>f</sub>(φ<sub>2</sub>) + I<sub>d</sub>(φ<sub>2</sub>·φ<sub>1</sub>) + I<sub>r</sub>(φ<sub>1</sub>)

3 : Active pull-up resistors will be switched on to the data bus when DBIN is high and data input voltage is more positive than V<sub>IH</sub> min.

**8-BIT PARALLEL CPU**

**TIMING REQUIREMENTS** ( $T_a = 0 \sim 70^\circ\text{C}$ ,  $V_{DD} = 12\text{V} \pm 5\%$ ,  $V_{CC} = 5\text{V} \pm 5\%$ ,  $V_{BB} = -5\text{V} \pm 5\%$ ,  $V_{SS} = 0\text{V}$ ; unless otherwise noted)

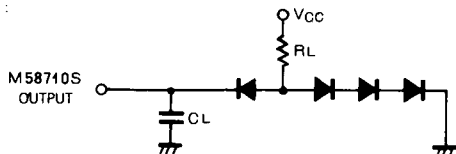
Symbol	Parameter	Limits			Unit
		Min	Typ	Max	
$t_c(\phi)$	Clock cycle time (Note 4)	480		2000	ns
$t_r(\phi)$	Clock rise time	0		50	ns
$t_f(\phi)$	Clock fall time	0		50	ns
$t_w(\phi_1)$	Clock 1 pulse width	60			ns
$t_w(\phi_2)$	Clock 2 pulse width	220			ns
$t_d(\phi_1L-\phi_2)$	Delay time, clock 1 to clock 2	0			ns
$t_d(\phi_2-\phi_1)$	Delay time, clock 2 to clock 1	70			ns
$t_d(\phi_1H-\phi_2)$	Delay time, clock 1 high to clock 2	80			ns
$t_{su}(DA-\phi_1)$	Data setup time with respect to clock 1	30			ns
$t_{su}(DA-\phi_2)$	Data setup time with respect to clock 2	150			ns
$t_{su}(HOLD)$	Hold setup time	140			ns
$t_{su}(INT)$	Interrupt setup time	120			ns
$t_{su}(RDY)$	Ready setup time	120			ns
$t_h(DA)$	Data hold time	$t_{PD}(DBI)$			ns
$t_h(HOLD)$	Hold input hold time	0			ns
$t_h(INT)$	Interrupt hold time	0			ns
$t_h(RDY)$	Ready hold time	0			ns

Note 4 :  $t_c(\phi) = t_d(\phi_1L-\phi_2) + t_r(\phi_2) + t_w(\phi_2) + t_r(\phi_2) + t_d(\phi_2-\phi_1) + t_r(\phi_1)$

**SWITCHING CHARACTERISTICS** ( $T_a = 0 \sim 70^\circ\text{C}$ ,  $V_{DD} = 12\text{V} \pm 5\%$ ,  $V_{CC} = 5\text{V} \pm 5\%$ ,  $V_{BB} = -5\text{V} \pm 10\%$ ,  $V_{SS} = 0\text{V}$ ; unless otherwise noted)

Symbol	Parameter	Test conditions (Note 5)	Limits			Unit
			Min	Typ	Max	
$t_{PD}(AD)$	Propagation delay time, clock 2 to address outputs	$R_L = 2.1\text{k}\Omega$ , $C_L = 100\text{pF}$			200	ns
$t_{PD}(DA)$	Propagation delay time, clock 2 to data bus	$R_L = 2.1\text{k}\Omega$ , $C_L = 100\text{pF}$			220	ns
$t_{PD}(CONT)$	Propagation delay time, clocks to control outputs	$R_L = 2.1\text{k}\Omega$ , $C_L = 50\text{pF}$			120	ns
$t_{PD}(DBI)$	Propagation delay time, clock 2 to DBIN output	$R_L = 2.1\text{k}\Omega$ , $C_L = 50\text{pF}$	25		140	ns
$t_{PD}(INT)$	Propagation delay time, clock 2 to INTE output	$R_L = 2.1\text{k}\Omega$ , $C_L = 50\text{pF}$			200	ns
$t_{PD}(DI)$	Time for data bus to enter input mode				$t_{PD}(DBI)$	ns
$t_{PXZ}$	Disable time to high-impedance state during hold address output and data bus				120	ns
$t_d(\overline{WR}-AD)$	Delay time, write signal to address output	$R_L = 2.1\text{k}\Omega$ , $C_L = 100\text{pF}$	$t_d(\phi_1H-\phi_2)$			ns
$t_d(AD-\overline{WR})$	Delay time, address output to write signal	$R_L = 2.1\text{k}\Omega$ , $C_L = 100\text{pF}$	Note 6			ns
$t_d(\overline{WR}-DA)$	Delay time, write signal to data output	$R_L = 2.1\text{k}\Omega$ , $C_L = 100\text{pF}$	$t_d(\phi_1H-\phi_2)$			ns
$t_d(DA-\overline{WR})$	Delay time, data output to write signal	$R_L = 2.1\text{k}\Omega$ , $C_L = 100\text{pF}$	Note 7			ns

Note 5 : Load circuit :

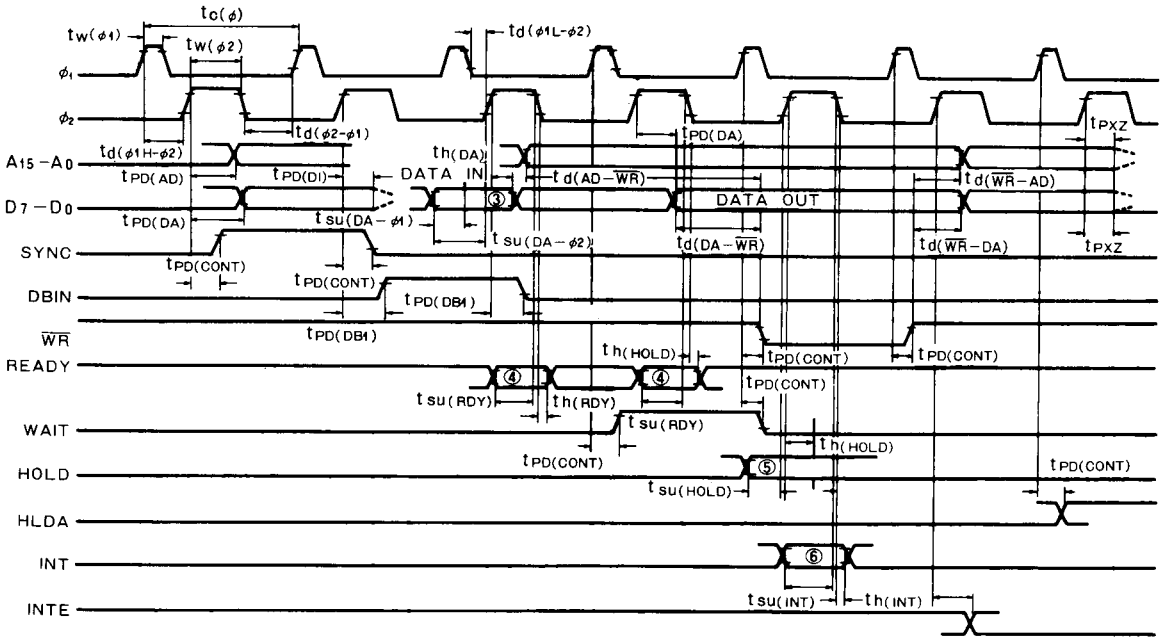


Note 6 :  $t_d(AD-\overline{WR}) = 2t_c(\phi) - t_d(\phi_1H-\phi_2) - t_r(\phi) - 140\text{ns}$

7 :  $t_d(DA-\overline{WR}) = t_c(\phi) - t_d(\phi_1H-\phi_2) - t_r(\phi) - 170\text{ns}$

**8-BIT PARALLEL CPU**

**TIMING DIAGRAM**



Note 8 : This timing diagram shows timing relationships only; it does not represent any specific machine cycle.

9 : Time measurements are made at the following reference voltages: Clock voltage H = 8.0V, L = 1.0V; input voltage, H = 3.3V, L = 0.8V; output voltage, H = 2.0V, L = 0.8V.

10 : Data on the data bus must be stable for this period in the input mode. Requirements  $t_{SU}(DA-\phi1)$ ,  $t_{SU}(DA-\phi2)$ ,  $t_h(DA)$  must be satisfied.

11 : The ready signal must be stable for this period during state  $T_2$  or  $T_W$ . External synchronization is required.

12 : The hold signal must be stable for this period during state  $T_2$  or  $T_W$  when entering the hold mode and during states  $T_3$ ,  $T_4$ ,  $T_5$ ,  $T_{WH}$  and  $T_H$  when in the hold mode. External synchronization is not required.

13 : The interrupt signal INT must be stable for the period immediately before the last state of any instruction in order to be recognized on the following machine cycle  $M_1$ . External synchronization is not required.

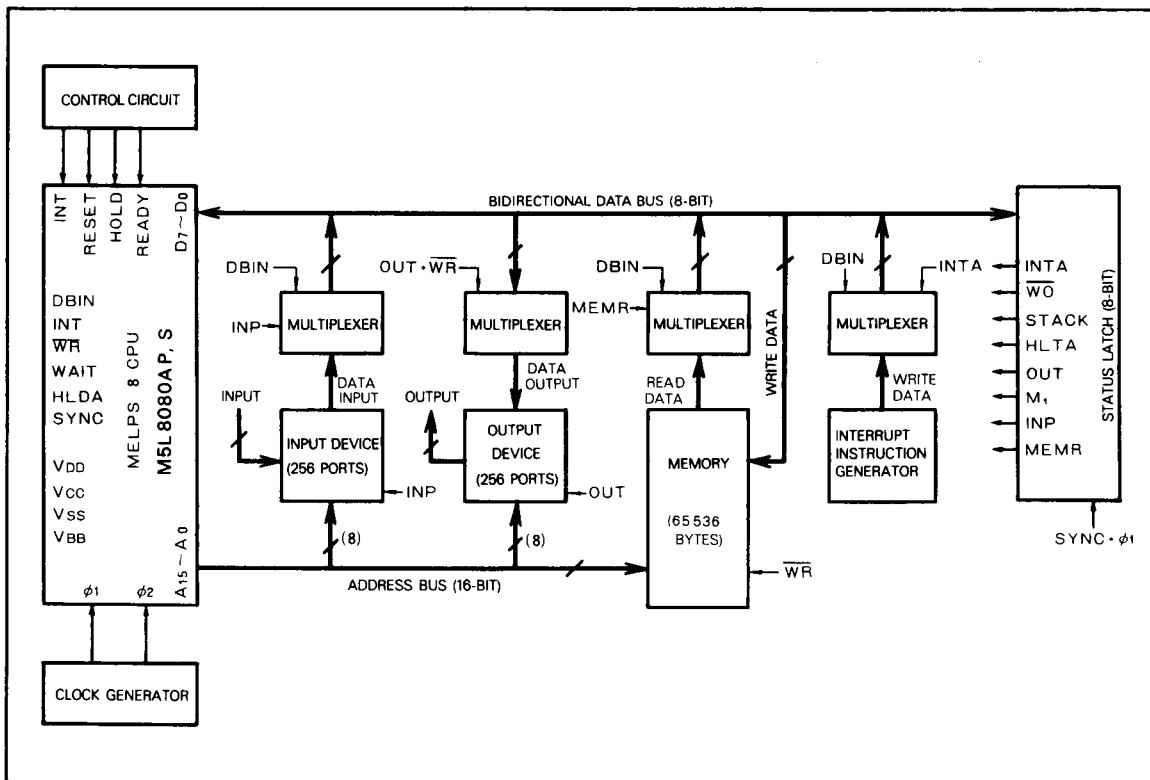
**8-BIT PARALLEL CPU**

**APPLICATIONS**

**A Basic System Using the M5L 8080A P, S**

The configuration of a system using the M5L 8080A P, S will depend on the functions of the system. A typical basic system is shown in Fig. 1, and a summary of its operation is as follows:

Fig. 1 An example of a basic system using the M5L 8080A P, S



1. After the CPU receives the two phase clocks  $\phi_1$  and  $\phi_2$  from the clock generator and the external reset signal, the address bus provides the address to memory location zero.
  2. At the same time the CPU sends out status signals, which are latched temporarily in the status latch (flip-flops in which status information is latched). The status signals alert external circuits as to the state of the machine cycle that the CPU is ready to execute. When the CPU calls for data or instructions to be read from memory, status signal MEMR is applied to the multiplexer, and the 8-bit data from memory is read into the CPU through the bidirectional data bus across the multiplexer.
  3. The 8-bit data coming from memory is decoded as an instruction. If it is a register-reference-arithmetic instruction, it is executed in the CPU; if it is a move-to-memory instruction, the CPU outputs the memory location to the address bus and data to be written on the data bus in the next machine cycle (Note 1). The memory write in operation is executed by write control signal WR.
  4. During input and output operation, the CPU outputs the I/O device number to the address bus, outputs a status signal (INP in the input mode; OUT in the output mode) and executes the read/write operation to the I/O devices using the bidirectional data bus.
  5. If there is a signal from terminal INT to the CPU, the CPU is in the interrupt enable state, and it sends out status information INTA (Note 15), and an interrupt instruction is sent to the CPU from the interrupt instruction generator across the multiplexer. By executing this interrupt instruction, the CPU can jump to the interrupt processing subroutine.
- Note 14 : Each instruction may have five machine cycles. For register-to-register transfer or arithmetic instruction, instruction fetching and execution are carried out by machine cycle M<sub>1</sub>, but memory access instructions, or 2-byte or 3-byte instructions require more than one machine cycle.
- 15 : The interrupt acknowledge signal goes high when the CPU accepts an interrupt request (INT) signal.

**Push-Down Stack Operation**

The M58710S has a last-in-first-out stack. This stack has a pointer that maintains the address of the next available stack location in memory and can be initialized to use any position of memory. Since the stack pointer has a 16-bit register, it can locate any stack location up to 65536 bytes according to memory capacity. An example of the interrupt request is shown in Fig. 2 and the operation of the stack pointer in Fig. 3.

**Fig. 2 Processing an interrupt request**

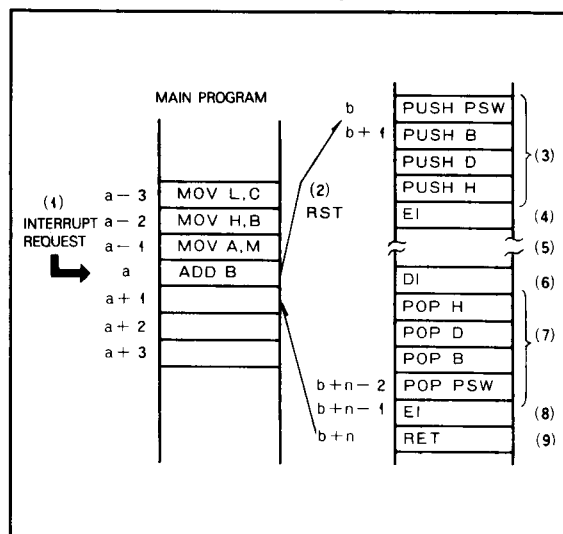


Fig. 2 is explained as follows:

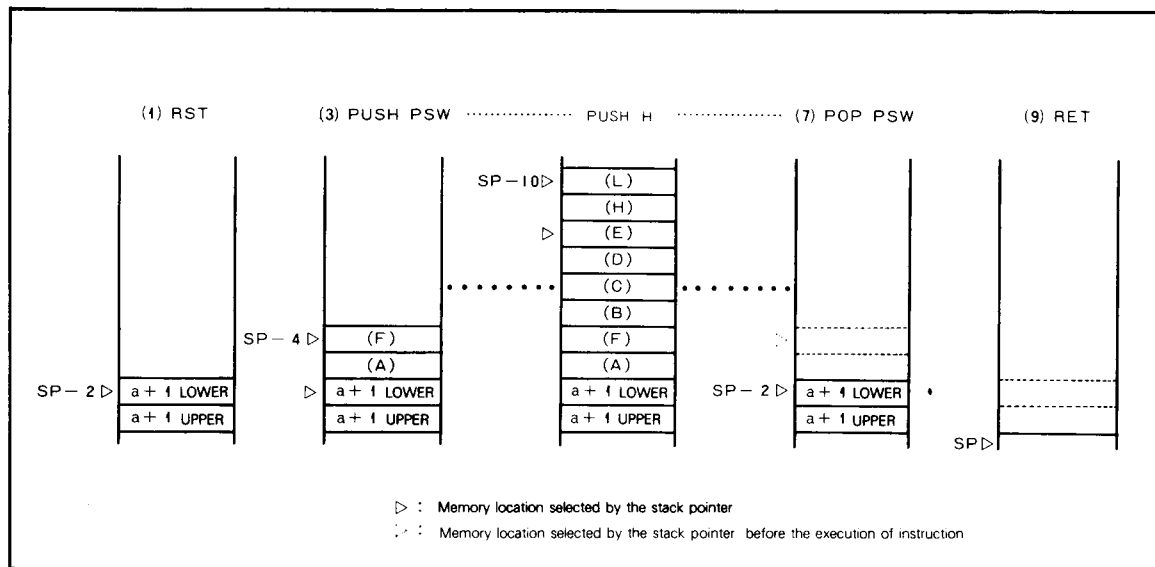
1. An external interrupt request occurs when the CPU executes the instruction stored at location a in the main program.
2. Instruction RST is fetched, the content of the program counter is incremented and pushed onto the push-down stack. Then the CPU jumps to the first location b of the interrupt operation program.
3. The contents of the register are pushed onto the stack. F (in Fig. 3) indicates 8-bit data of flag flip-flops including  $CY_2$ ,  $CY_1$ , Z, P and S. These are, from the most to the least significant bit, S, Z, 0,  $CY_1$ , 0, P, 1,  $CY_2$ .
4. Instruction EI is executed, enabling the CPU to accept the next interrupt request.
5. The interrupt operation is carried out.
6. The CPU enters the interrupt disable state.
7. The contents of registers are popped off the stack.
8. Instruction EI is executed, enabling the CPU to accept the interrupt request after return to the main program.
9. The content of the program counter is returned to location a+1 of the main program.

The operation of the push-down stack shown in Fig. 2 is described in Fig. 3, where SP indicates the content of the stack pointer before the interrupt is requested. Instruction LXI SP should be used to initialize the stack pointer.

The content of the stack pointer is SP-4 at (3), but at (9), after the execution of instruction RET, it returns to the initial state, and the content of the stack point is SP.

**8**

**Fig.3 Operation of the push-down stack**



**8-BIT PARALLEL CPU**

**EXAMPLE OF APPLICATION CIRCUIT**

