8080 MICROPROCESSOR

1. INTRODUCTION

The 8080 is a complete 8-bit parallel central processing unit (CPU) for use in general purpose digital computer systems. It is fabricated on a single LSI chip using Intel's n-channel silicon gate MOS process, thus offering much higher performance than conventional microprocessors (2μs instruction cycle). A complete micro computer system is formed when the 8080 CPU is interfaced with I/O ports (up to 256 input and 256 output ports) and any type or speed of semi-conductor memory.

Although significantly higher in performance than existing microprocessors, the 8080 has been designed to be software compatible at the source code level with Intel's 8008 microprocessor. Like the 8008, the 8080 contains six 8-bit data registers, an 8-bit accumulator, four 8-bit temporary registers, four testable flag bits, and an 8-bit parallel binary arithmetic unit. The 8080 also provides decimal arithmetic capability, and it includes sixteen bit arithmetic and immediate operators which greatly simplify memory address calculations, and high speed arithmetic operations.

The 8080 has a stack architecture wherein any portion of the external memory can be used as a last in/first out stack to store/retrieve the contents of the accumulator, the flags, or any of the data registers.

The 8080 also contains a 16-bit stack pointer to control the addressing of this external stack. One of the major advantages of the stack is that multiple level interrupts can easily be handled since complete system status can easily be saved when an interrupt occurs and then be restored after the interrupt. Another major advantage is that almost unlimited subroutine nesting is possible.

This processor has been designed to greatly simplify system design. Separate 16-line address and 8-line bidirectional data busses are used to allow direct interface to memories and I/O ports. Control signals, which require no decoding, are provided directly by the processor. All busses, including control, are TTL compatible.

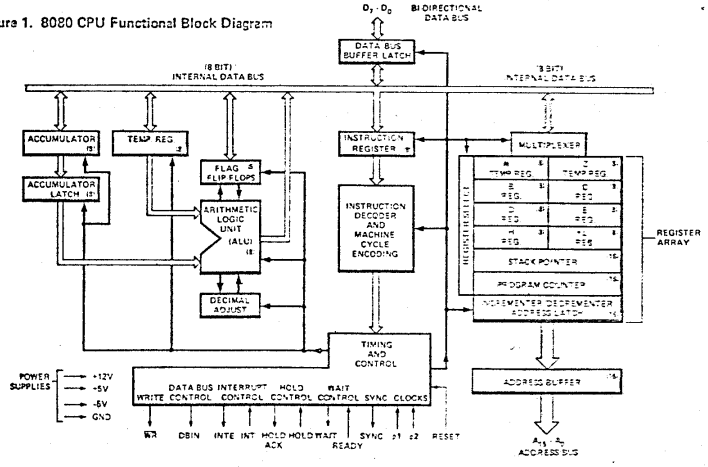
Communication on the address lines and the data lines can be interlocked by using the HOLD input. When the HLDA (Hold Acknowledge) signal is issued by the CPU, CPU operation is suspended and the address and data lines are forced to be in the FLOATING state. This permits "OR-tying" the address and data busses with other devices such as direct memory access channels (DMA).

The 8080 has many instructions which are extremely useful and extend the range of applicability of the CPU. The instruction groups are as follows:

- Data register and memory transfers
- Conditional or unconditional branches and subroutine calls
- I/O operations
- Direct Load/Store Accumulator
- Save, Restore Data Registers, Accumulator and Flags
- Double Length Operation in Data Registers
  - Increment/Decrement/Addition
  - Direct Load/Store (H and L)
  - Load Immediate
  - Index Register Modification
- Indirect Jump
- Stack Pointer Modification
- Logical Operations
- Binary Arithmetic
- Decimal Arithmetic
- Set and reset interrupt enable flip-flop
- Increment/Decrement Memory of data registers

8080 ADDRESSING MODES:
- DIRECT
- REGISTER
- REGISTER INDIRECT
- IMMEDIATE

Figure 1. 8080 CPU Functional Block Diagram



2. PROCESSOR TIMING

2-1. 8080 FUNCTIONAL PIN DEFINITION

The following describes the function of all of the 8080 I/O pins. Several of the descriptions refer to internal timing periods.

A_{15}-A_0 (output three-state)
ADDRESS BUS; the address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. A_0 is the least significant address bit.

D_7-D_0 (input/output three-state)
DATA BUS; the data bus provides bidirectional communication between the CPU, memory, and I/O devices for instructions and data transfers. D_0 is the least significant bit.

SYNC (output)
SYNCHRONIZING SIGNAL; the SYNC pin provides a signal to indicate the beginning of each machine cycle.

DBIN (output)
DATA BUS IN; the DBIN signal indicates to external circuits that the data bus is in the input mode. This signal should be used to enable the gating of data onto the 8080 data bus from memory or I/O.

READY (input)
READY; the READY signal indicates to the 8080 that valid memory or input data is available on the 8080 data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080 does not receive a READY input, the 8080 will enter a WAIT state for as long as the READY line is low. (READY can also be used to single step the CPU.)

WAIT (output)
WAIT; the WAIT signal acknowledges that the CPU is in a WAIT state.

$\overline{WR}$ (output)
WRITE; the $\overline{WR}$ signal is used for memory WRITE or I/O output control. The data on the data bus is stable while the $\overline{WR}$ signal is active low ($\overline{WR}$ = 0).

HOLD (input)
HOLD; the HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080 address and data bus as soon as the 8080 has completed its use of these buses for the current machine cycle. It is recognized under the following conditions:

- the CPU is in the HALT state.
- the CPU is in the T2 or TW state and the READY signal is active.

As a result of entering the HOLD state the CPU ADDRESS BUS (A_{15}-A_0) and DATA BUS (D_7-D_0) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.

HLDA (output)
HOLD ACKNOWLEDGE; the HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to the high impedance state. The HLDA signal begins at:

- T3 for READ memory or input.
- The Clock Period following T3 for WRITE memory or OUTPUT operation.

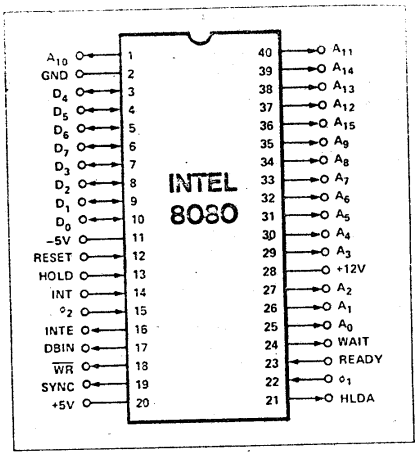


Figure 2. Pin Configuration

In either case, the HLDA signal appears after the rising edge of ϕ_1 and high impedance occurs after the rising edge of ϕ_2.

INTE (output)[1]
INTERRUPT ENABLE; indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the Enable and Disable Interrupt instructions and inhibits interrupts from being accepted by the CPU when it is reset. It is automatically reset (disabling further interrupts) at time T1 of the instruction fetch cycle (M1) when an interrupt is accepted and is also reset by the RESET signal.

INT (input)
INTERRUPT REQUEST; the CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the Interrupt Enable flip/flop is reset it will not honor the request.

RESET (input)
RESET; while the RESET signal is activated, the content of the program counter is cleared. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset. Note that the flags, accumulator, stack pointer, and registers are not cleared.

V_{SS}	Ground Reference.	V_{cc}	+5 ± 5% Volts.
V_{dd}	+12 ± 5% Volts.	V_{bb}	-5 ±5% Volts (substrate bias).

Note 1: After the EI instruction, the CPU will accept interrupts on the second instruction following the EI.
This is to allow proper processing of the RET instruction if an interrupt is pending after the service routine.

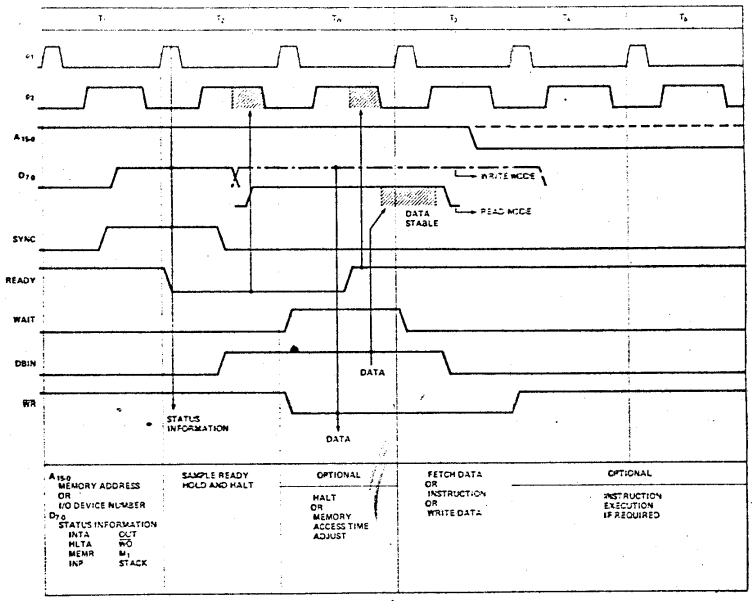


Figure 3. Basic 8080 Instruction Cycle

2-2. TIMING

Instructions in the 8080 contain one to three bytes. Each instruction requires from one to five machine or memory cycles for fetching and execution. Machine cycles are called M1, M2, . . . , M5. Each machine cycle requires from three to five states T1, T2, . . . , T5 for its completion. Each state has the duration of one clock period (0.5 micro-second). There are three other states (WAIT, HOLD, and HALT) which last one to an indefinite number of clock periods, as controlled by external signals. Machine cycle M1 is always the operation-code fetch cycle and lasts four or five clock periods. Machine cycles M2, M3, M4, and M5 normally last three clock periods each.

To understand the basic operation of the 8080, refer to the simplified state diagram shown in Figure 4 and the timing diagram of Figure 3.

During T1 the content of the program counter is sent to the address bus, SYNC is true, and the data bus contains the status information pertaining to the cycle that is currently being initiated. T1 is always followed by another state, T2, during which the condition of the READY, HOLD and HALT Acknowledge Signals are tested. If READY is true, T3 can be entered; otherwise, the CPU will go into the wait state (TW) and stay there for as long as READY is false.

READY thus allows the CPU speed to by synchronized to a memory with any access time or to any input device. Furthermore, by properly controlling the READY line, the user can single-step through his program.

During T3, the data coming from memory is available on the data bus and is transferred into the instruction register (during M1 only) as shown in the 8080 block diagram of Figure 4. The instruction decoder and control sections then generate the basic signals to control the internal data transfers, the timing, and the machine cycle requirements of the new instructions.

At the end of T4 (if the cycle is complete, or else at the end of T5, the 8080 goes back to T1 and enters machine cycle M2, unless the instruction required only one machine cycle for its execution. In such cases, a new M1 cycle is entered. The loop is repeated for as many cycles and states as required by the instruction.

It is only during the last state of the last machine cycle that the interrupt request line is tested and a special M1 cycle is entered, during which no program-counter increment takes place and INTERRUPT ACKNOWLEDGE status is sent out. During this cycle, one of eight possible restart instructions will be sent to the CPU by the interrupting device.

Instruction state requirements range from a minimum of four states for non-memory referencing instructions, like register and accumulator arithmetic instructions, up to a maximum of 18 states for the most complex instructions (exchange the contents of registers H and L with the content of the top two locations of the stack). At the maximum clock frequency of 2 megahertz, this means that all instructions will be executed in intervals ranging from 2 μs to 9 μs. If a HALT instruction is executed, the processor enters a WAIT state and remains there until an interrupt is received.

2-3. STATUS INFORMATION

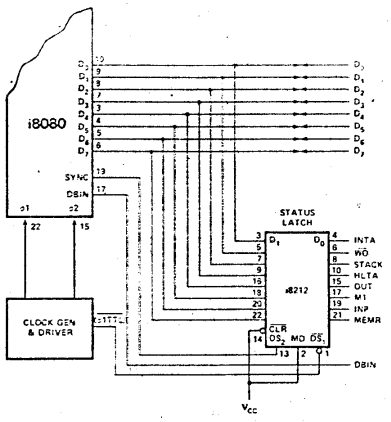
Instructions for the 8080 require from one to five machine cycles for complete execution. The 8080 sends out 8 bit of status information on the data bus at the beginning of each machine cycle (during SYNC time). The following table defines the status information.

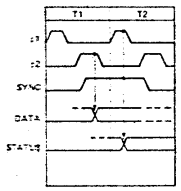
STATUS INFORMATION DEFINITION

Symbols	Data Bus Bit	Definition
INTA*	D_0	Acknowledge signal for INTERRUPT request. Signal should be used to gate a restart instruction onto the data bus when DBIN is active.
$\overline{WO}$	D_1	Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function ($\overline{WO}$ = 0). Otherwise, a READ memory or INPUT operation will be executed.
STACK	D_2	Indicates that the address bus holds the pushdown stack address from the Stack Pointer.
HLTA	D_3	Acknowledge signal for HALT instruction.
OUT	D_4	Indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{WR}$ is active.
M_1	D_5	Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction.
INP*	D_6	Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active.
MEMR*	D_7	Designates that the data bus will be used for memory read data.

*These three status bits can be used to control the flow of data onto the 8080 data bus.

8080 STATUS LATCH





STATUS WORD CHART

TYPE OF MACHINE CYCLE

DATA BUS BIT	STATUS INFORMATION	(1) INSTRUCTION FETCH	(2) MEMORY READ	(3) MEMORY WRITE	(4) STACK READ	(5) STACK WRITE	(6) INPUT READ	(7) OUTPUT WRITE	(8) INTERRUPT ACKNOWLEDGE	(9) HALT ACKNOWLEDGE	(10) INTERRUPT ACKNOWLEDGE WHILE HALT
											← (N) STATUS WORD
D_0	INTA	0	0	0	0	0	0	0	1	0	1
D_1	$\overline{WO}$	1	1	0	1	0	1	0	1	1	0
D_2	STACK	0	0	0	1	1	0	0	0	0	0
D_3	HLTA	0	0	0	0	0	0	0	0	1	1
D_4	OUT	0	0	0	0	0	0	1	0	0	0
D_5	M_1	1	0	0	0	0	0	0	1	0	1
D_6	INP	0	0	0	0	0	1	0	0	0	0
D_7	MEMR	1	1	0	1	0	0	0	0	1	0

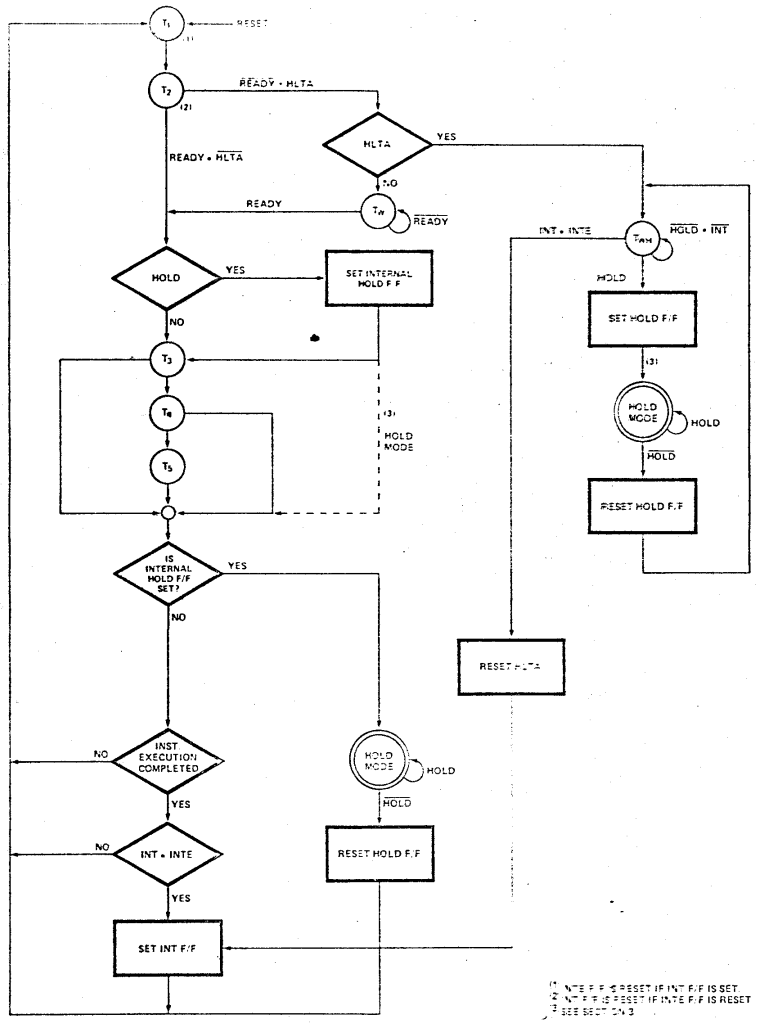


Figure 4. CPU State Transition Diagram

3. PROCESSOR INSTRUCTION SET

3-1. COMPLETE FUNCTIONAL DEFINITION

The following pages present a detailed description of the complete 8080 Instruction Set.

Symbols	Meaning
<B2>	Second byte of the instruction
<B3>	Third byte of the instruction
r	One of the scratch pad register references: A, B, C, D, E, H, L
c	One of the following flag flip-flop references: flag flip-flops. Condition for True carry — Overflow, underflow zero — Result is zero sign — MSB of result is "1" parity — Parity of result is even
M	Memory location indicated by the contents of registers H and L
( )	Contents of location or register
∧	Logical product
∀	Exclusive "or"
V	Inclusive "or"
r_m	Bit m of register r
SP	Stack Pointer
PC	Program Counter
←	Is transferred to
XXX	A "don't care"
SSS	Source register for data
DDD	Destination register for data

Register # (SSS or DDD)	Register Name
000	B
001	C
010	D
011	E
100	H
101	L
110	Memory
111	ACC

8080 INSTRUCTION SET

Mnemonic	Bytes	Cycles	Description of Operation
MOV r1, r2	1	1	$(r_1) \leftarrow (r_2)$ Load register r_1 with the content of r_2. The content of r_2 remains unchanged.
MOV r, M	1	2	(r) ← (M) Load register r with the content of the memory location addressed by the contents of registers H and L.
MOV M, r	1	2	(M) ← (r) Load the memory location addressed by the contents of registers H and L with the content of register r.
MVI r <B2>	2	2	$(r) \leftarrow <B_2>$ Load byte two of the instruction into register r.
MVI M <B2>	2	3	$(M) \leftarrow <B_2>$ Load byte two of the instruction into the memory location addressed by the contents of registers H and L.

Mnemonic	Bytes	Cycles	Description of Operation
INR r	1	1	(r) ← (r) + 1 The content of register r is incremented by one. All the condition flip-flops except carry are affected by the result.
DCR r	1	1	(r) ← (r) − 1 The content of register r is decremented by one. All of the condition flip-flops except carry are affected by the result.
ADD r	1	1	(A) ← (A) + (r) Add the content of register r to the content of register A and place the result into register A. (All flags affected.)
ADC r	1	1	(A) ← (A) + (r) + (carry) Add the content of register r and the contents of the carry flip-flop to the content of the A register and place the result into Register A. (All flags affected.)
SUB r	1	1	(A) ← (A) − (r) Subtract the content of register r from the content of register A and place the result into register A. Two's complement subtraction is used. (All flags affected.)
SBB r	1	1	(A) ← (A) − (r) − (borrow) Subtract the content of register r and the content of the carry flip-flop from the content of register A and place the result into register A. (All flags affected.)
ANA r	1	1	(A) ← (A) ∧ (r) Place the logical product of the register A and register r into register A. (Resets carry.)
XRA r	1	1	(A) ← (A) ∀ (r) Place the "exclusive - or" of the content of register A and register r into register A. (Resets carry.)
ORA r	1	1	(A) ← (A) ∨ (r) Place the "inclusive - or" of the content of register A and register r into register A. (Resets carry.)
CMP r	1	1	(A) − (r) Compare the content of register A with the content of register r. The content of register A remains unchanged. The flag flip-flops are set by the result of the subtraction. Equality (A = r) is indicated by the zero flip-flop set to "1." Less than (A < r) is indicated by the carry flip-flop, set to "1."
ADD M	1	2	(A) ← (A) + (M) ADD
ADC M	1	2	(A) ← (A) + (M) + (carry) ADD with carry
SUB M	1	2	(A) ← (A) − (M) SUBTRACT
SBB M	1	2	(A) ← (A) − (M) − (borrow) SUBTRACT with borrow
ANA M	1	2	(A) ← (A) ∧ (M) Logical AND
XRA M	1	2	(A) ← (A) ∀ (M) Exclusive OR
ORA M	1	2	(A) ← (A) ∨ (M) Inclusive OR
CMP M	1	2	(A) − (M) COMPARE
ADI $<B_2>$	2	2	(A) ← (A) + $<B_2>$ ADD
ACI $<B_2>$	2	2	(A) ← (A) + $<B_2>$ + (carry) ADD with carry
SUI $<B_2>$	2	2	(A) ← (A) − $<B_2>$ SUBTRACT
SBI $<B_2>$	2	2	(A) ← (A) − $<B_2>$ − (borrow) SUBTRACT with borrow
ANI $<B_2>$	2	2	(A) ← (A) ∧ $<B_2>$ Logical AND
XRI $<B_2>$	2	2	(A) ← (A) $<B_2>$ Exclusive OR
ORI $<B_2>$	2	2	(A) ← (A) $<B_2>$ Inclusive OR
CPI $<B_2>$	2	2	(A) − $<B_2>$ COMPARE
RLC	1	1	$A_{n+1} \leftarrow A_n$, $A_0 \leftarrow A_7$, (carry) ← A_7 Rotate the content of register A left one bit. Rotate A_7 into A_0 and into the carry flip-flop
RRC	1	1	$A_n \leftarrow A_{n+1}$, $A_7 \leftarrow A_0$, (carry) ← A_0 Rotate the content of register A right one bit. Rotate A_0 into A_7 and into the carry flip-flop

(M) addressed by the contents of registers H and L.

Flags affected are same as non-memory reference instructions.

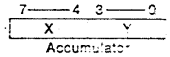
Mnemonic	Bytes	Cycles	Description of Operation
RAL	1	1	$A_{n+1} \leftarrow A_n$, $A_0 \leftarrow$ (carry), (carry) $\leftarrow A_7$ Rotate the content of Register A left one bit. Rotate the content of the carry flip-flop into A_0. Rotate A_7 into the carry flip-flop.
RAR	1	1	$A_n \leftarrow A_{n+1}$, $A_7 \leftarrow$ (carry), (carry) $\leftarrow A_0$ Rotate the content of register A right one bit. Rotate the content of the carry flip-flop into A_7. Rotate A_0 into the carry flip-flop.
JMP <B_2> <B_1>	3	3	(PC) $\leftarrow$ <B_3> <B_2> Jump unconditionally to the instruction located in memory location addressed by byte two and byte three.
JC <B_2> <B_1>	3	3	If (Carry) = 1 (PC) $\leftarrow$ <B_1> <B_2> Otherwise (PC) = (PC) + 3
JNC <B_2> <B_1>	3	3	If (Carry) = 0 (PC) $\leftarrow$ <B_3> <B_2> Otherwise (PC) = (PC) + 3
JZ <B_2> <B_1>	3	3	If (Zero) = 1 (PC) $\leftarrow$ <B_3> <B_2> Otherwise (PC) = (PC) + 3
JNZ <B_2> <B_1>	3	3	If (Zero) = 0 (PC) $\leftarrow$ <B_3> <B_2> Otherwise (PC) = (PC) + 3
JP <B_2> <B_1>	3	3	If (Sign) = 0 (PC) $\leftarrow$ <B_3> <B_2> Otherwise (PC) = (PC) + 3
JM <B_2> <B_1>	3	3	If (Sign) = 1 (PC) $\leftarrow$ <B_3> <B_2> Otherwise (PC) = (PC) + 3
JPE <B_2> <B_1>	3	3	If (Parity) = 1 (PC) $\leftarrow$ <B_3> <B_2> Otherwise (PC) = (PC) + 3
JPO <B_2> <B_1>	3	3	If (Parity) = 0 (PC) $\leftarrow$ <B_3> <B_2> Otherwise (PC) = (PC) + 3
HLT	1	1	On receipt of the Halt Instruction, the activity of the processor is immediately suspended in the STOPPED state. The content of all registers and memory is unchanged and the PC has been updated.
CALL <B_2> <B_1>	3	5	[SP − 1] [SP − 2] $\leftarrow$ (PC), (SP) = (SP) − 2 (PC) $\leftarrow$ <B_3> <B_2> Transfer the content of PC to the pushdown stack in memory addressed by the register SP. The content of SP is decremented by two. Jump unconditionally to the instruction located in memory location addressed by byte two and byte three of the instruction.
CC <B_2> <B_1>	3	3/5	If (carry) = 1 [SP − 1] [SP − 2] $\leftarrow$ PC, (SP) = (SP) − 2, (PC) $\leftarrow$ <B_3> <B_2>; otherwise (PC) = (PC) + 3
CNC <B_2> <B_1>	3	3/5	If (carry) = 0 [SP − 1] [SP − 2] $\leftarrow$ PC, (SP) = (SP) − 2, (PC) $\leftarrow$ <B_3> <B_2>; otherwise (PC) = (PC) + 3
CZ <B_2> <B_1>	3	3/5	If (zero) = 1 [SP − 1] [SP − 2] $\leftarrow$ PC, (SP) = (SP) − 2, (PC) $\leftarrow$ <B_3> <B_2>; otherwise (PC) = (PC) + 3
CNZ <B_2> <B_1>	3	3/5	If (zero) = 0 [SP − 1] [SP − 2] $\leftarrow$ PC, (SP) = (SP) − 2, (PC) $\leftarrow$ <B_3> <B_2>; otherwise (PC) = (PC) + 3
CP <B_2> <B_1>	3	3/5	If (sign) = 0 [SP − 1] [SP − 2] $\leftarrow$ PC, (SP) = (SP) − 2, (PC) $\leftarrow$ <B_3> <B_2>; otherwise (PC) = (PC) + 3

Mnemonic	Bytes	Cycles	Description of Operation
CM <B_2> <B_3>	3	3/5	If (sign) = 1 [SP − 1] [SP − 2] ← PC, (SP) = (SP) − 2, (PC) ← <B_2> <B_3>; otherwise (PC) = (PC) + 3
CPE <B_2> <B_3>	3	3/5	If (parity) = 1 [SP − 1] [SP − 2] ← PC, (SP) = (SP) − 2, (PC) ← <B_2> <B_3>; otherwise (PC) = (PC) + 3
CPO <B_2> <B_3>	3	3/5	If (parity) = 0 [SP − 1] [SP − 2] ← PC, (SP) = (SP) − 2, (PC) ← <B_2> <B_3>; otherwise (PC) = (PC) + 3
RET	1	3	(PC) ← [SP] [SP + 1] (SP) = (SP) + 2. Return to the instruction in the memory location addressed by the last values shifted into the pushdown stack addressed by SP. The content of SP is incremented by two.
RC	1	1/3	If (carry) = 1 (PC) ← [SP], [SP + 1], (SP) = (SP) + 2; otherwise (PC) = (PC) + 1
RNC	1	1/3	If (carry) = 0 (PC) ← [SP], [SP + 1], (SP) = (SP) + 2; otherwise (PC) = (PC) + 1
RZ	1	1/3	If (zero) = 1 (PC) ← [SP], [SP + 1], (SP) = (SP) + 2; otherwise (PC) = (PC) + 1
RNZ	1	1/3	If (zero) = 0 (PC) ← [SP], [SP + 1], (SP) = (SP) + 2; otherwise (PC) = (PC) + 1
RP	1	1/3	If (sign) = 0 (PC) ← [SP], [SP + 1], (SP) = (SP) + 2; otherwise (PC) = (PC) + 1
RM	1	1/3	If (sign) = 1 (PC) ← [SP], [SP + 1], (SP) = (SP) + 2; otherwise (PC) = (PC) + 1
RPE	1	1/3	If (parity) = 1 (PC) ← [SP], [SP + 1], (SP) = (SP) + 2; otherwise (PC) = (PC) + 1
RPO	1	1/3	If (parity) = 0 (PC) ← [SP], [SP + 1], (SP) = (SP) + 2; otherwise (PC) = (PC) + 1
RST	1	3	[SP − 1] [SP − 2] ← (PC), (SP) = (SP) − 2 (PC) ← (00000000 00AAA000)
IN <B_2>	2	3	(A) ← (Input data) At T_1 time of third cycle, byte two of the instruction, which denotes the I/O device number, is sent to the I/O device through the address lines*, and the INP status information, instead of MEMR, is sent out at sync time. New data for the accumulator is loaded from the data bus when DBIN control signal is active. The condition flip-flops are not affected.
OUT <B_2>	2	3	(Output data) ← (A) At T_1 time of the third cycle, byte two of the instruction, which denotes the I/O device number, is sent to the I/O device through the address lines*, and the OUT status information is sent out at sync time. The content of the accumulator is made available on the data bus when the $\overline{WR}$ control signal is 0.
LXI B <B_2> <B_3>	3	3	(C) ← <B_2>; (B) ← <B_3> Load byte two of the instruction into C. Load byte three of the instruction into B.
LXI D <B_2> <B_3>	3	3	(E) ← <B_2>, (D) ← <B_3> Load byte two of the instruction into E. Load byte 3 of the instruction into D.
LXI H <B_2> <B_3>	3	3	(L) ← <B_2>, (H) ← <B_3> Load byte two of the instruction into L. Load byte three of the instruction into H.

*The device address appears on $A_0 - A_7$ and $A_8 - A_{15}$

Mnemonic	Bytes	Cycles	Description of Operation
LXI SP <B_2> <B_3>	3	3	(SP) ← <B_3><B_2>, (SP)$_{-1}$ ← <B_1> Load byte two of the instruction into the lower order 8-bit of the stack pointer and byte three into the higher order 8-bit of the stack pointer.
PUSH PSW	1	3	[SP − 1] ← (A), [SP − 2] ← (F), (SP) = (SP) − 2 Save the contents of A and F (5-flags) into the pushdown stack addressed by the SP register. The content of SP is decremented by two. The flag word will appear as follows: D_7: CY_1 (Carry) D_6: 1 D_5: Parity (even) D_4: 0 D_3: CY_1 D_2: 0 D_1: Zero D_0: MSB (sign)
PUSH B	1	3	[SP − 1] ← (B) [SP − 2] ← (C), (SP) = (SP) − 2
PUSH D	1	3	[SP − 1] ← (D) [SP − 2] ← (E), (SP) = (SP) − 2
PUSH H	1	3	[SP − 1] ← (H) [SP − 2] ← (L), (SP) = (SP) − 2
POP PSW	1	3	(F) ← [SP], (A) ← [SP + 1], (SP) = (SP) + 2 Restore the last values in the pushdown stack addressed by SP into A and F. The content of SP is incremented by two.
POP B	1	3	(C) ← [SP], (B) ← [SP + 1], (SP) = (SP) + 2
POP D	1	3	(E) ← [SP], (D) ← [SP + 1], (SP) = (SP) + 2
POP H	1	3	(L) ← [SP], (H) ← [SP + 1], (SP) = (SP) + 2
STA <B_2> <B_3>	3	4	[<B_3> <B_2>] ← (A) Store the accumulator content into the memory location addressed by byte two and byte three of the instruction.
LDA <B_2> <B_3>	3	4	(A) ← [<B_3> <B_2>] Load the accumulator with the content of the memory location addressed by byte two and byte three of the instruction.
XCHG	1	1	(H) ←→ (D) (E) ←→ (L) Exchange the contents of registers H and L and registers D and E.
XTHL	1	5	(L) ←→ [SP], (H) ←→ [SP + 1] Exchange the contents of registers H, L and the last values in the pushdown stack addressed by registers SP. The SP register itself is not changed. (SP) = (SP)
SPHL	1	1	(SP) ← (H) (L) Transfer the contents of registers H and L into register SP.
PCHL	1	1	(PC) ← (H) (L) JUMP INDIRECT
DAD SP	1	3	(H) (L) ← (H) (L) + (SP) Add the content of register SP to the content of registers H and L and place the result into registers H and L. If the overflow is generated, the carry flip-flop is set; otherwise, the carry flip-flop is reset. The other condition flip-flops are not affected. This is useful for addressing data in the stack.
DAD B	1	3	(H) (L) ← (H) (L) + (B) (C)
DAD H	1	3	(H) (L) ← (H) (L) + (H) (L) (double precision shift left H and L)
DAD D	1	3	(H) (L) ← (H) (L) + (D) (E)
STAX B	1	2	[(B) (C)] ← (A) Store the accumulator content in the memory location addressed by the content of registers B and C.
STAX D	1	2	[(D) (E)] ← (A) Store the accumulator content into the memory location addressed by the content of register D and E.
LDAX B	1	2	(A) ← [(B) (C)] Load the accumulator with the content of the memory location addressed by the content of registers B and C.

Mnemonic	Bytes	Cycles	Description of Operation
LDAX D	1	2	(A) ← [(D) (E)] Load the accumulator with the content of memory location addressed by the content of register D and E.
INX B	1	1	(B) (C) ← (B) (C) + 1 The content of register pair B and C is incremented by one. All of the condition flip-flops are not affected.
INX H	1	1	(H) (L) ← (H) (L) + 1 The content of register H and L is incremented by one. All of the condition flip-flops are not affected.
INX D	1	1	(D) (E) ← (D) (E) + 1
INX SP	1	1	(SP) ← (SP) + 1
DCX B	1	1	(B) (C) ← (B) (C) − 1
DCX H	1	1	(H) (L) ← (H) (L) − 1
DCX D	1	1	(D) (E) ← (D) (E) − 1
DCX SP	1	1	(SP) ← (SP) − 1
CMA	1	1	(A) ← $\overline{(A)}$ The content of accumulator is complemented. The condition flip-flops are not affected.
STC	1	1	(Carry) ← 1 Set the carry flip-flop to 1. The other condition flip-flops are not affected.
CMC	1	1	(carry) ← $\overline{(carry)}$ The content of carry is complemented. The other condition flip-flops are not affected.

DAA	1	1	Decimal Adjust Accumulator
			The 8-bit value in the accumulator containing the result from an arithmetic operation on decimal operands is adjusted to contain two valid BCD digits by adding a value according to the following rules:
			 
			
			If (Y ≥ 10) or (carry from bit 3) then Y = Y + 6 with carry to X digit.
			If (X ≥ 10) or (carry from bit 7) or [(Y ≥ 10) and (X = 9)] then X = X + 6 (which sets the carry flip-flop).
			Two carry flip-flops are used for this instruction. CY_4 represents the carry from bit 3 (the fourth bit) and is accessible as a flag. CY_8 is the carry from bit 7 and is the usual carry bit.
			All condition flip-flops are affected by this instruction.

SHLD $<B_2>$ $<B_3>$	3	5	[$<B_3>$ $<B_2>$] ← (L), [$<B_3>$ $<B_2>$ + 1] ← (H) Store the contents of registers H and L into the memory location addressed by byte two and byte three of the instruction.
LHLD $<B_2>$ $<B_3>$	3	5	(L) ← [$<B_3>$ $<B_2>$], (H) ← [$<B_3>$ $<B_2>$ + 1] Load the registers H and L with the contents of the memory location addressed by byte two and byte three of the instruction.
EI	1	1	Interrupt System Enable
DI	1	1	Interrupt System Disable The Interrupt Enable flip-flop (INTE) can be set or reset by using the above mentioned instructions. The INT signal will be accepted if the INTE is set. When the INT signal is accepted by the CPU, the INTE will be reset immediately. During interrupt enable or disable instruction executions, an interrupt will not be accepted.
INR M	1	3	[M] ← [M] + 1. The content of memory designated by registers H and L is incremented by one. All of the condition flip-flops except carry are affected by the result.
DCR M	1	3	[M] ← [M] − 1. The content of memory designated by registers H and L is decremented by one. All of the condition flip-flops except carry are affected by the result.

3-2. DATA AND INSTRUCTION FORMATS

Data in the 8080 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

TYPICAL INSTRUCTIONS

Register to register, memory reference, arithmetic or logical, rotate return, push, pop, enable or disable interrupt instructions

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

Immediate mode or I/O instructions

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 OR OPERAND 1
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	HIGH ADDRESS OR OPERAND 2

Jump, call or direct load and store instructions

For the 8080 a logic "1" is defined as a high level and a logic "0" is defined as a low level.

3-3. INSTRUCTION SET

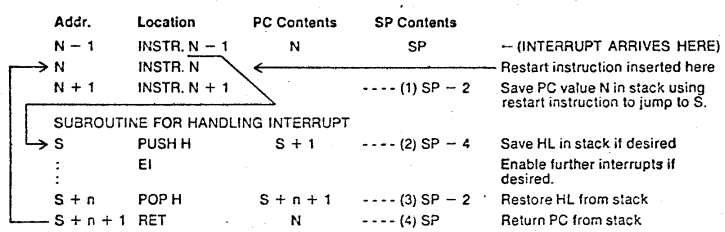
Summary of Processor Instructions

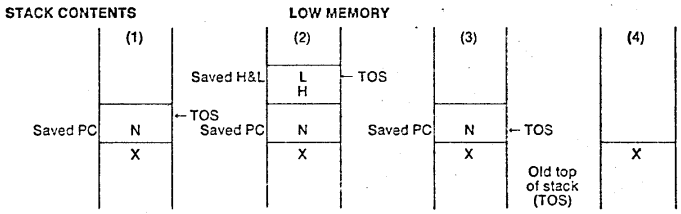
Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
$MOV_{r1,r2}$	Move register to register	0	1	D	D	D	S	S	S	5
MOV M, r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r, M	Move memory to register	0	1	D	D	D	1	1	0	7
HLT	Halt	0	1	1	1	0	1	1	0	7
MVI r	Move immediate register	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	Jump 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

Mnemonic	Description	Instruction Code[1]								Clock Cycles
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
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
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 interrupts	1	1	1	1	0	0	1	1	4
NOP	No operation	0	0	0	0	0	0	0	0	4

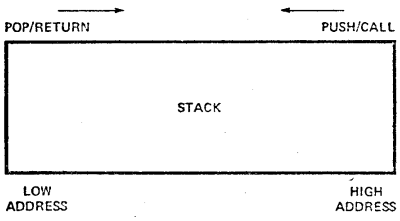
NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 101 Memory – 111 A.
2. Two possible cycle times (5/11) indicate instruction cycles dependent on condition flags.

4. HOW TO USE THE PUSHDOWN STACK





NOTE: The user can initialize the stack point SP register with a LXI SP instruction to use any section of read-write memory as a stack. The SP is decremented when data is pushed onto the stack, and incremented when data is popped (that is the stack "grows downward").



5. PROGRAMMING EXAMPLES

(Decimal operation)

a. Decimal Addition:

Memory address of Augend; D and E is (ALPHA)
Memory address of Addend; H and L is (BETA)

Mnemonic	Operand	Explanation	Bytes	Comment
LXI	D, ALPHA	Load D and E Immediate	3	Set address to DE
LXI	H, BETA	Load H and L Immediate	3	Set address to HL
MVI	C, 8	Load C with "8"		
XRA		Exclusive or A with A	1	Clear Carry
LOOP: LDAX	D	Load A with (DE)	1	Load Augend to Acc
ADC	M	Add M to A (HL)	1	Add Addend to Augend
DAA		Decimal Adjust	1	
STAX	D	Store A to (DE)	1	Replace Result
INX	H	Increment HL	1	Renew address HL
INX	D	Increment DE	1	Renew address DE
DCR	C	Decrement C	1	Check end of calculation
JNZ	LOOP	If not zero go to loop	3	

Calculation time (16 digits) ~ 230 μsec

b. Decimal Subtraction

Memory address of Minuend; D and E (ALPHA)
Memory address of Subtrahend; H and L (BETA)

Mnemonic	Operand	Explanation	Bytes	Comment
LXI	D, ALPHA	Load D and E Immediate	3	Set address to DE
LXI	H, BETA	Load H and L Immediate	3	Set address to HL
MVI	C, 8	Load C with "8"	2	
STC		Set Carry	1	
LOOP: MVI	A, 99H	Load A with 99 HEX	2	$99_{16} - 1 = 98_{16}$
ACI	0	Add with carry	2	
SUB	M	Subtract M from A	1	
XCHG		Exchange DE and HL	1	Actually
ADD	M	Add M to A	1	
DAA		Decimal Adjust	1	3 − 2 = 10 − 2 + 3 = 11
MOV	M, A	Load A to M	1	
XCHG		Exchange DE and HL	1	No borrow occurs here
INX	D	Increment DE	1	
INX	H	Increment HL	1	
DCR	C	Decrement C	1	
JNZ	LOOP		3	

Calculation time (16 digits) ~ 330 μsec

c. Binary Multiplication Loop

A contains Multiplier, D and E is Multiplicand, H and L are Partial Product

Mnemonic	Operand	Explanation	Bytes
LXI	H, 0	Initialize Partial Product to 0	3
MVI	B, 8	8 → B to control loop	2
LOOP: DAD	H	Shift partial product left and into carry	1
RAL		Rotate multiplier bit to carry	1
JNC	DEC	Test multiplier at carry	3
DAD	D	Add multiplicand to partial product if carry = 1	1
ACI	0		
DEC: DCR	B	Decrement B loop counter	1
JNZ	LOOP	Test to see if B = 0 to iterate 8 times	3

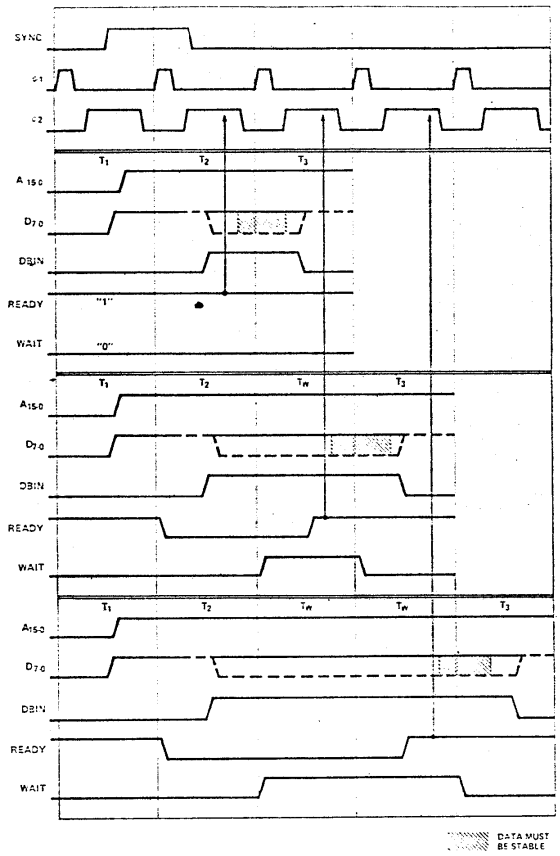
Calculation time for 8 x 16 multiply ~ 230 μsec

d. Accumulator Loading

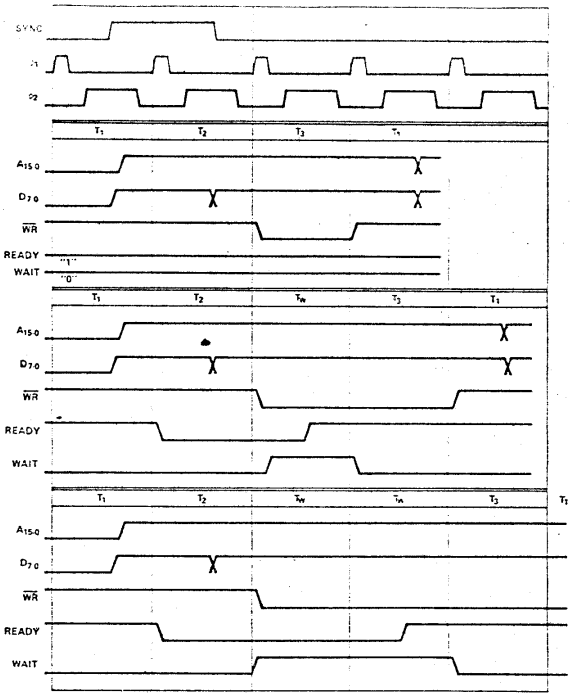
Mnemonic	Operand	Explanation	Bytes
MOV	A, B	Load A with Register B	1
MVI	A, 23	Load A with Data Immediate "23"	2
LDA	4093	Load A with contents of memory LOC 4093	3
MOV	A, M	Load A using H and L as address	1
LDAX	B	Load A using B and C as address	1
LDAX	D	Load A using D and E as address	1
LHLD	4093	Load A indirect using LOC 4093	4
MOV	A, M		
POP	A	Load A with data from stack	1
IN	10	Load A with data from Device = 10	2

6. TIMING DIAGRAMS

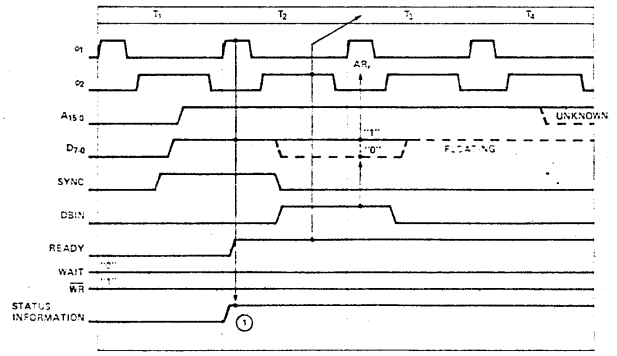
a. Relation between READY and DBIN



Relation between READY, WAIT and $\overline{WR}$

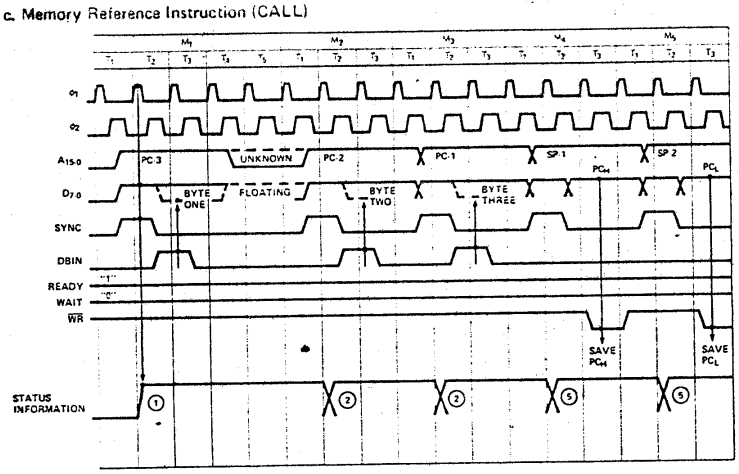


b. Non-Memory Reference Instruction (AR_r)



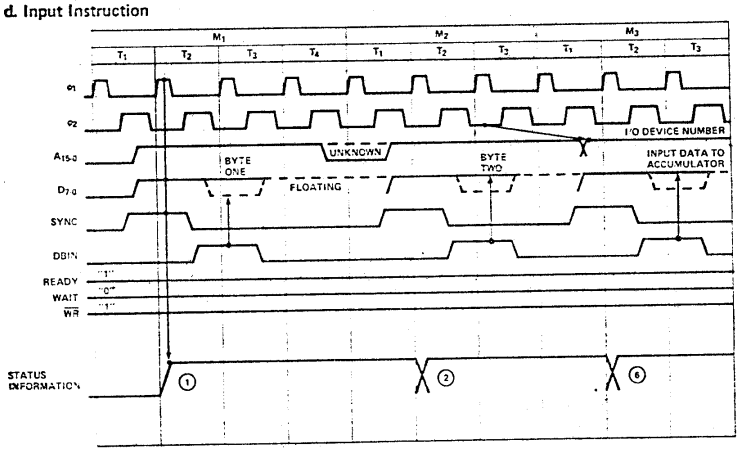
NOTE (1) Refer to Status Word Chart on Page 4

c. Memory Reference Instruction (CALL)



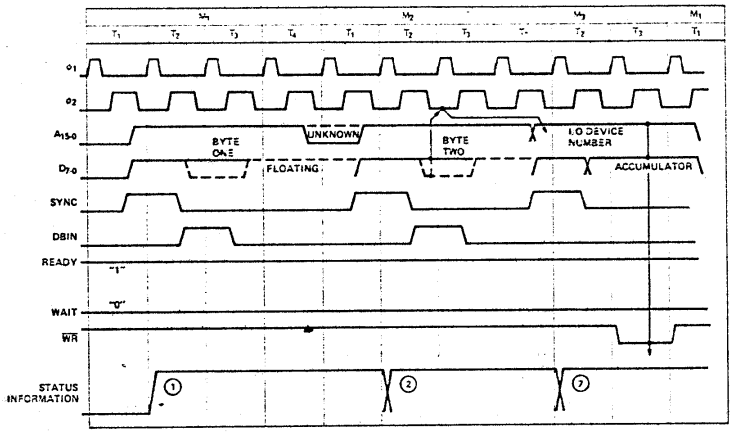
NOTE: (N) Refer to Status Word Chart on Page 4.

d. Input Instruction



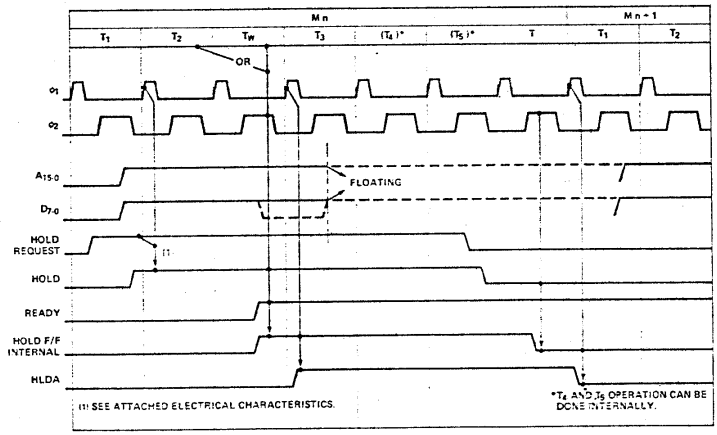
NOTE (N) Refer to Status Word Chart on Page 4.

e. OUTPUT INSTRUCTION

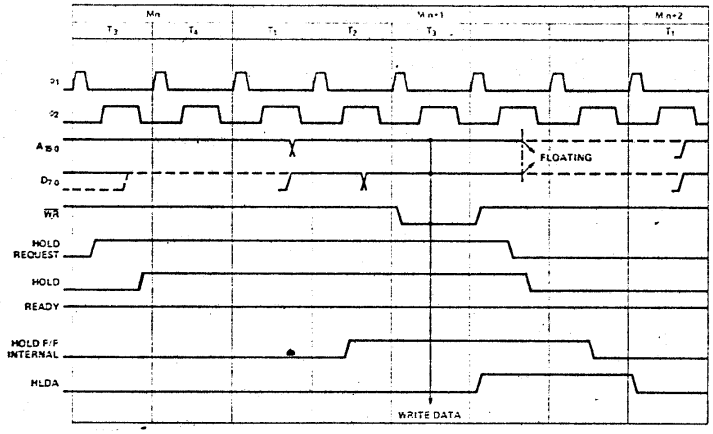


NOTE: (N) Refer to Status Word Chart on Page 4.

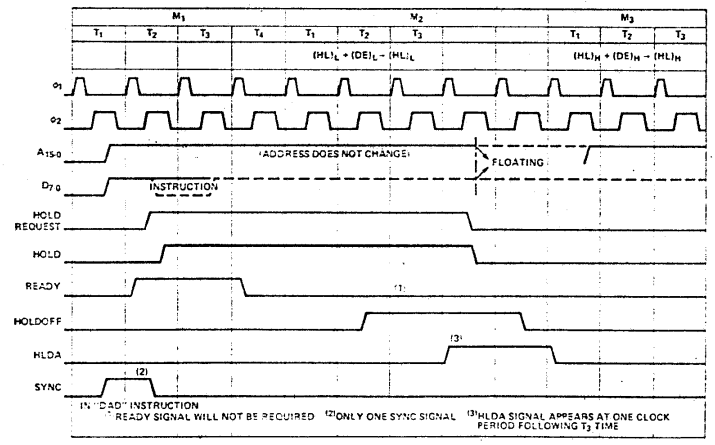
f. HOLD OPERATION (READ MODE)



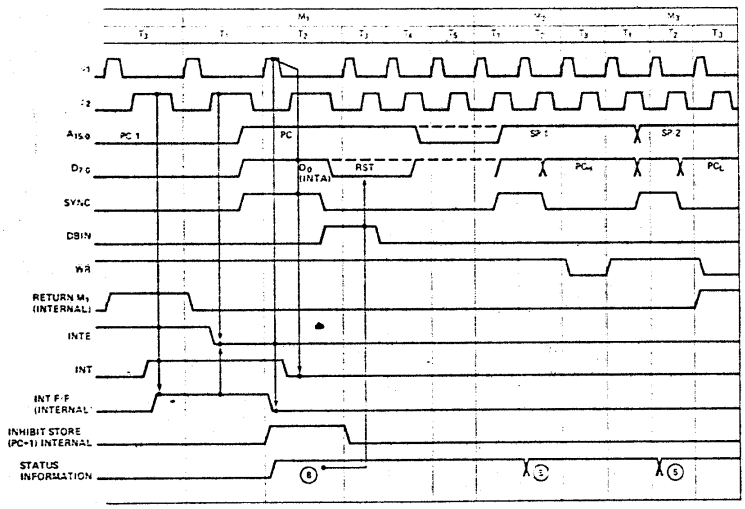
HOLD Operation (Write mode)



HOLD Operation (DAD)

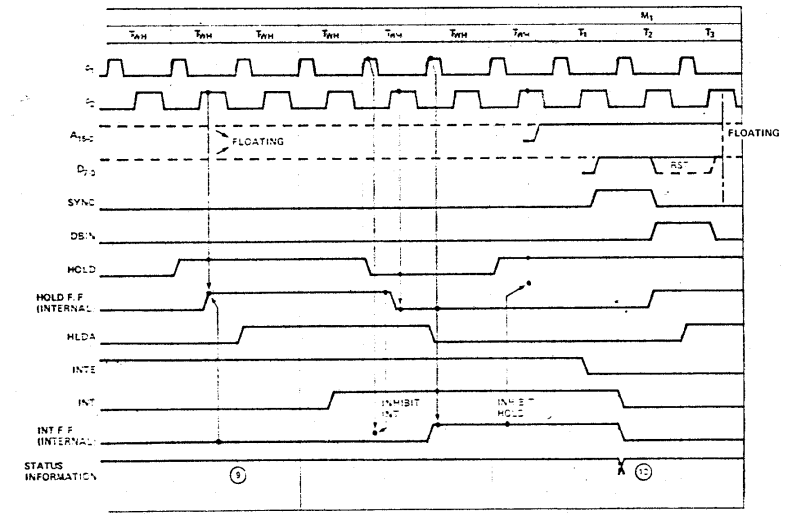


g. Interrupt



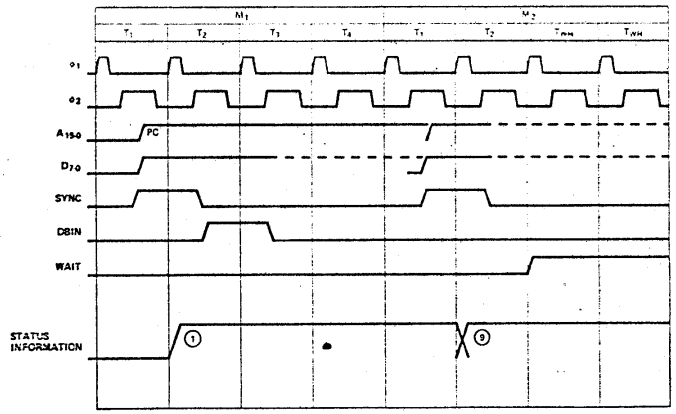
NOTE: (N) Refer to Status Word Chart on Page 4.

h. Relation between HOLD and INT in the HALT state



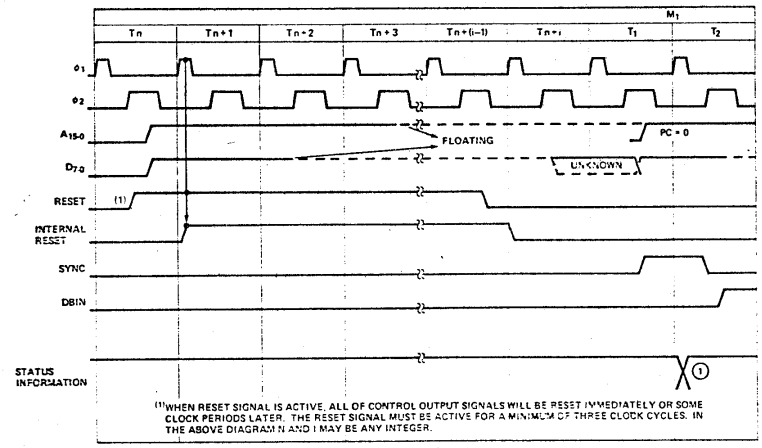
NOTE: (N) Refer to Status Word Chart on Page 4.

i. HALT Instruction



NOTE: (N) Refer to Status Word Chart on Page 4.

j. RESET



(1) WHEN RESET SIGNAL IS ACTIVE, ALL OF CONTROL OUTPUT SIGNALS WILL BE RESET IMMEDIATELY OR SOME CLOCK PERIODS LATER. THE RESET SIGNAL MUST BE ACTIVE FOR A MINIMUM OF THREE CLOCK CYCLES. IN THE ABOVE DIAGRAM N AND I MAY BE ANY INTEGER.

NOTE (N) Refer to Status Word Chart on Page 4.

7. MINIMUM 8080 SYSTEMS

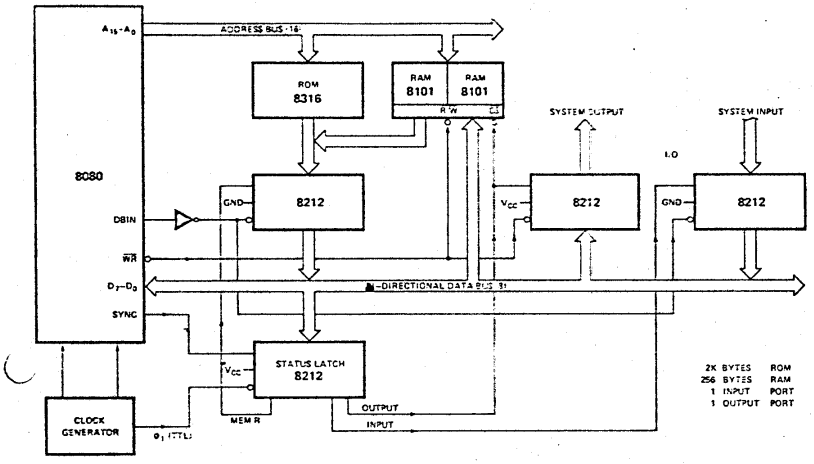


Figure 5. Minimum 8080 System.

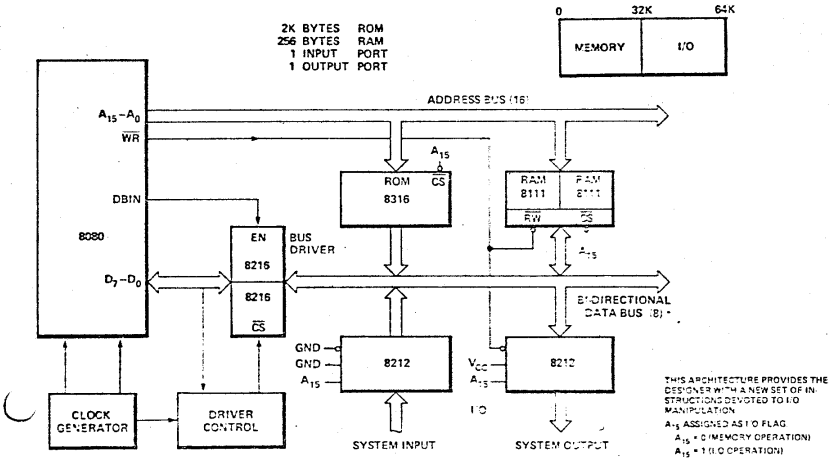


Figure 6. Minimum 8080 System (MEMORY MAPPED I/O).

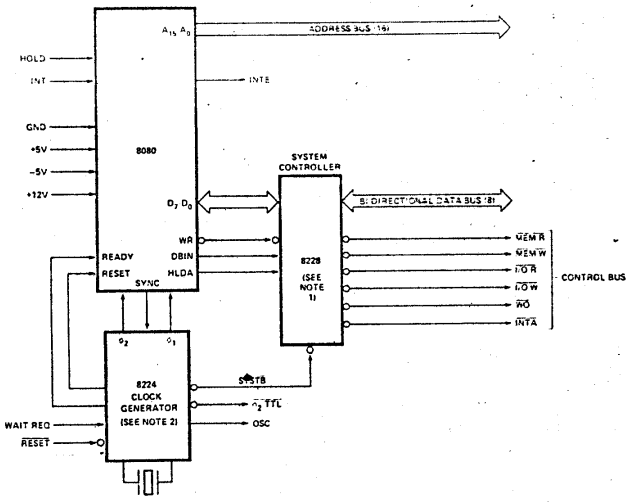


Figure 7. 8080 Standard Interface.

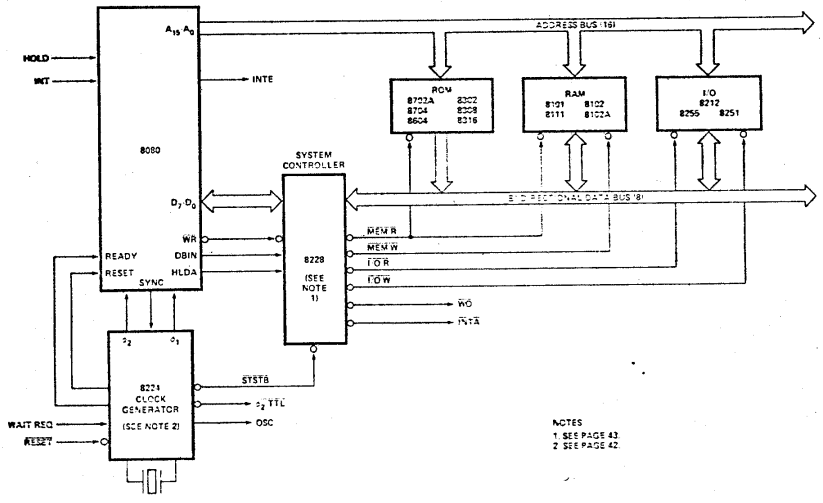


Figure 8. 8080 Standard System Architecture.

8. ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to +70° C
Storage Temperature . . . . . -65°C to +150°C
All Input or Output Voltages
With Respect to V_{BB} . . . . . -0.3V to +20V
V_{CC}, V_{DD} and V_{SS} With Respect to V_{BB} -0.3V to +20V
Power Dissipation 1.5W

COMMENT:

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

8-1. D.C. CHARACTERISTICS

T_A = 0°C, to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	V_{SS}−1		V_{SS}+0.6	V	
V_{IHC}	Clock Input High Voltage	V_{DD}−1		V_{DD}+1	V	
V_{IL}	Input Low Voltage	V_{SS}−1		V_{SS}+0.8	V	
V_{IH}	Input High Voltage	3.3		V_{CC}+1	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.7mA on the Data Bus I_{OL} = .75mA on all other outputs
V_{OH}	Output High Voltage	3.7			V	I_{OH} = 100μA.
$I_{DD\ (AV)}$	Avg. Power Supply Current (V_{DD})		40	67	mA	Operation
$I_{CC\ (AV)}$	Avg. Power Supply Current (V_{CC})		60	75	mA	T_A = 25°C
$I_{BB\ (AV)}$	Avg. Power Supply Current (V_{BB})		.01	1	mA	T_{CY} = .48 μsec
I_{IL}	Input Leakage			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{CL}	Clock Leakage			±10	μA	$V_{SS} \leq V_{CLOCK} \leq V_{DD}$
I_{DL} [3]	Data Bus Leakage in Input Mode			-100	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{FL}	Address and Data Bus Leakage During HOLD			+10 -100	μA	$V_{ADDR/DATA} = V_{CC}$ $V_{ADDR/DATA} = V_{SS}$

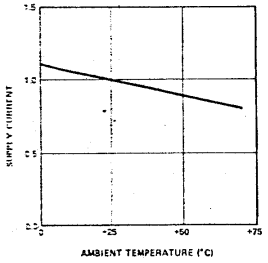
8-2. CAPACITANCE

T_A = 25°C $V_{CC} = V_{DD} = V_{SS}$ = 0V, V_{SS} = -5V ±5%

Symbol	Parameter	Typ.	Max.	Unit	Test Condition
C_ϕ	Clock Capacitance	10	20	pf	f_c = 1 MHz
C_{IN}	Input Capacitance	5	10	pf	Unmeasured Pins
C_{OUT}	Output Capacitance	10	20	pf	Returned to V_{SS}

NOTES:
1. The RESET signal must be active for a minimum of 3 clock cycles.
2. When DBIN is high and $V_{IN} > V_{IH}$ an active pull up of nominally 2kΩ will be switched onto the Data Bus.
3. ΔI supply / ΔT_A = -0.45%/°C.

TYPICAL SUPPLY CURRENT VS. TEMPERATURE, NORMALIZED. [4]



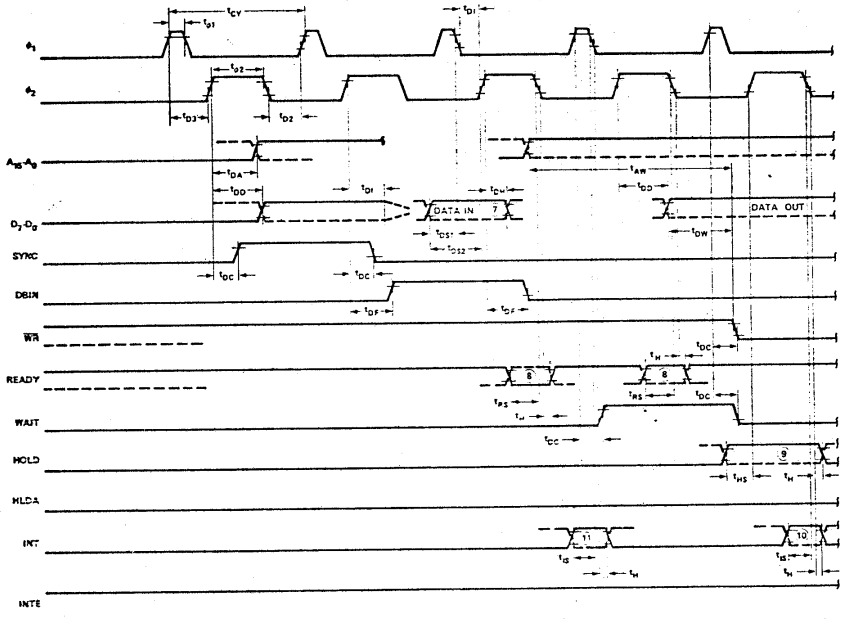
8-3. A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{CY}[3]	Clock Period	0.48	2.0	μsec	
t_r, t_f	Clock Rise and Fall Time	5	50	nsec	
$t_{\phi 1}$	ϕ_1 Pulse Width	60		nsec	
$t_{\phi 2}$	ϕ_2 Pulse Width	220		nsec	
t_{D1}	Delay ϕ_1 to ϕ_2	0		nsec	
t_{D2}	Delay ϕ_2 to ϕ_1	70		nsec	
t_{D3}	Delay ϕ_1 to ϕ_2 Leading Edges	130		nsec	
t_{DA}[2]	Address Output Delay From ϕ_2		200	nsec	R_L = 4.5kΩ, C_L = 100pf
t_{DD}[2]	Data Output Delay From ϕ_2		220	nsec	R_L = 2.1kΩ, C_L = 100pf
t_{DC}[2]	Signal Output Delay From ϕ_1, or ϕ_2 (SYNC, $\overline{WR}$ WAIT HLDA)		120	nsec	R_L = 4.5kΩ, C_L = 50pf
t_{DF}[2]	DBIN Delay From ϕ_2	25	140	nsec	R_L = 2.1kΩ, C_L = 50pf
t_{DI}[1]	Delay for Input Bus to Enter Input Mode During DBIN		t_{DF}	nsec	
t_{DS1}	Data "Setup Time" During ϕ_1 and DBIN	50		nsec	

TIMING WAVEFORMS [12]

(Note: Timing measurements are made at the following reference voltages: CLOCK "1" = 9.5V, "0" = 1.0V; INPUTS "1" = 3.3V, "0" = 0.8V; OUTPUTS "1" = 2.0V, "0" = 0.8V.)

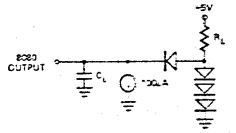


A.C. CHARACTERISTICS (Continued)

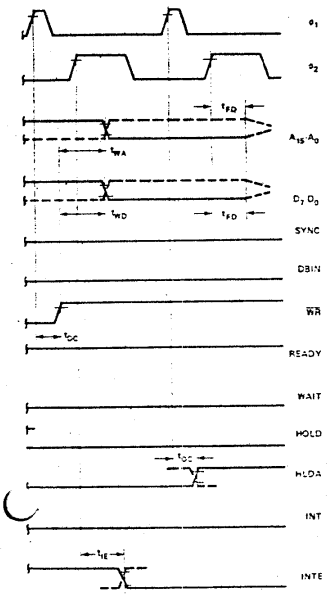
$T_A = 0°C$ to $70°C$, $V_{DD} = +12V \pm 5\%$, $V_{CC} = -5V \pm 5\%$, $V_{BB} = -5V \pm 5\%$, $V_{SS} = 0V$, Unless Otherwise Noted

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{DS2}	Data "Setup Time" to ϕ_2 During DBIN	150		nsec	
t_{DH} [1]	Data "Hold Time" From ϕ_2 During DBIN	t_{DF}		nsec	
t_{IE} [2]	INTE Output Delay From ϕ_2		200	nsec	$R_L = 4.5k\Omega$, $C_L = 50pf$
t_{RS}	Ready "Setup Time" During ϕ_2	120		nsec	
t_{HS}	Hold "Setup Time" to ϕ_2	140		nsec	
t_{IS}	INT "Setup Time" During ϕ_2 (During ϕ_1 in Halt Mode)	180		nsec	
t_H	"Hold Time" From ϕ_2 (Ready, INT, Hold)	0		nsec	
t_{FD}	Delay to Float During Hold (Address and DATA BUS)		120	nsec	
t_{WA} [2]	Address Stable From $\overline{WR}$	t_{D3}		nsec	$R_L = 4.5k\Omega$, $C_L = 100pf$
t_{AW} [2]	Address Stable Prior to $\overline{WR}$	[5]		nsec	$R_L = 4.5k\Omega$, $C_L = 100pf$
t_{WD} [2]	Output Data Stable From $\overline{WR}$	t_{D3}		nsec	$R_L = 2.1k\Omega$, $C_L = 100pf$
t_{DW} [2]	Output Data Stable Prior to $\overline{WR}$	[6]		nsec	$R_L = 2.1k\Omega$, $C_L = 100pf$

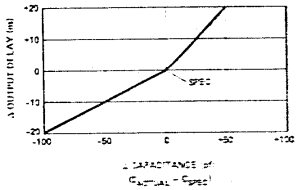
NOTES: 1. Data input should be enabled with DBIN status. No bus conflict can then occur and data hold time is assured.
2. Load circuit



3. $t_{CY} = t_{D3} + t_{o2} + t_{D2} + t_{h2} + t_{v\phi1} + t_{h\phi2} \geq 480$ ns



TYPICAL Δ OUTPUT DELAY VS. Δ CAPACITANCE



4. The following are relevant when interfacing the 8080 to devices having $V_{IH} = 3.3V$:
   a) Maximum output rise time from .8V to 3.0V = 140 ns @ C_L = SPEC.
   b) Output delay when measured to 3.0V = SPEC + 60 ns @ C_L = SPEC.
   c) If C_L < SPEC add 6 ns/pf if $C_L > C_{SPEC}$ subtract 2 ns/pf ("from modified delay" if $C_L < C_{SPEC}$.
5. $t_{AW} = 2\,t_{CY} + t_{D3} + t_{h\phi2} + 120$ nsec
6. $t_{DW} = t_{CY} + t_{D3} + t_{h\phi2} + 150$ nsec
7. Data in must be stable for this period during T_3, T_4, T_5. Both t_{DS1} and t_{DS2} must be satisfied.
8. Ready signal must be stable for this period during T_2 or T_W. (Must be externally synchronized.)
9. Hold signal must be stable for this period during T_2 or T_W when entering hold mode, and during T_3, T_4, T_5 and T_{WH} when in hold mode. (Must be externally synchronized.)
10. Interrupt signal must be stable during this period of the last clock cycle of any instruction to be recognized on the following instruction. (External synchronization is not required.)
11. During halt mode only, timing is with respect to the falling edge.
12. This timing diagram shows timing relationships only. It does not represent any specific machine cycle.

9. MCS-80™ COMPONENT SUPPORT FAMILY

RAMs

8101	Static 256 x 4
8111	Static 256 x 4 (Common I/O)
8102	Static 1K x 1
8102A	Static 1K x 1 (High Speed)
8107A	Dynamic 4K x 1

ROMs

8302	256 x 8	Mask
8308	1K x 8 (High Speed)	
8316	2K x 8	
8702A	256 x 8 (Erasable)	
8704	512 x 8 (Erasable)	
8604	512 x 8 (High Speed)	

PERIPHERALS

8205	1 of 8 Decoder
8210	Driver 8107A
8214	Priority Interrupt Control Unit
8216	Bi-Directional Bus Driver
8224	Clock Generator – 8080
8228	System Controller – 8080

INPUT/OUTPUT

8212	8-Bit I/O Port
8255	Programmable Peripheral Interface
8251	Universal Communication Interface