Second Edition

MICROPROCESSORS and MICROCOMPUTER-BASED SYSTEM DESIGN

DR. M. RAFIQUZZAMAN

Second Edition

MICROPROCESSORS and MICROCOMPUTER-BASED SYSTEM DESIGN

DR. M. RAFIQUZZAMAN

2

INTEL 8085

This chapter describes hardware, software, and interfacing aspects of the Intel 8085. Topics include 8085 register architecture, addressing modes, instruction set, input/output, and system design.

2.1 Introduction

The Intel 8085 is an 8-bit microprocessor. The 8085 is designed using NMOS in 40-in DIP (Dual In-line Package). The 8085 can be operated from either 3.03 MHz maximum (8085A) or 5 MHz maximum (8085A-2) internal clock frequency.

The 8085 has three enhanced versions, namely, the 8085AH, 8085AH-2, and 8085AH-1. These enhanced processors are designed using the HMOS (High-density MOS) technology. Each is packaged in a 40-pin DIP like the 8085. These enhanced microprocessors consume 20% lower power than the 8085A. The internal clock frequencies of the 8085AH, 8085AH-2, and 8085AH-1 are 3, 5, 6 MHz, respectively. These HMOS 8-bit microprocessors are expensive compared to the NMOS 8-bit 8085A.

Figure 2.1 shows a simplified block diagram of the 8085 microprocessor. The accumulator connects to the data bus and the Arithmetic and Logic Unit (ALU). The ALU performs all data manipulation, such as incrementing a number or adding two numbers.

The temporary register feeds the ALU's other input. This register is invisible to the programmer and is controlled automatically by the microprocessor's control circuitry.

The flags are a collection of flip-flops that indicate certain characteristics of the result of the most recent operation performed by the ALU. For example, the zero flag is set if the result of an operation is zero. The zero flag is tested by the JZ instruction.

The instruction register, instruction decoder, program counter, and control and timing logic are used for fetching instructions from memory and directing their execution.

2.2 Register Architecture

The 8085 registers and status flags are shown in Figure 2.2.

The accumulator (A) is an 8-bit register. Most arithmetic and logic operations are performed using the accumulator. All I/O data transfers between the 8085 and the I/O devices are performed via the accumulator. Also, there are a number of instructions that move data between the accumulator and memory.

The B, C, D, E, H, and L are each 8 bits long. Registers H and L are the memory address register or data counter. This means that these two registers are used to store the 16-bit address

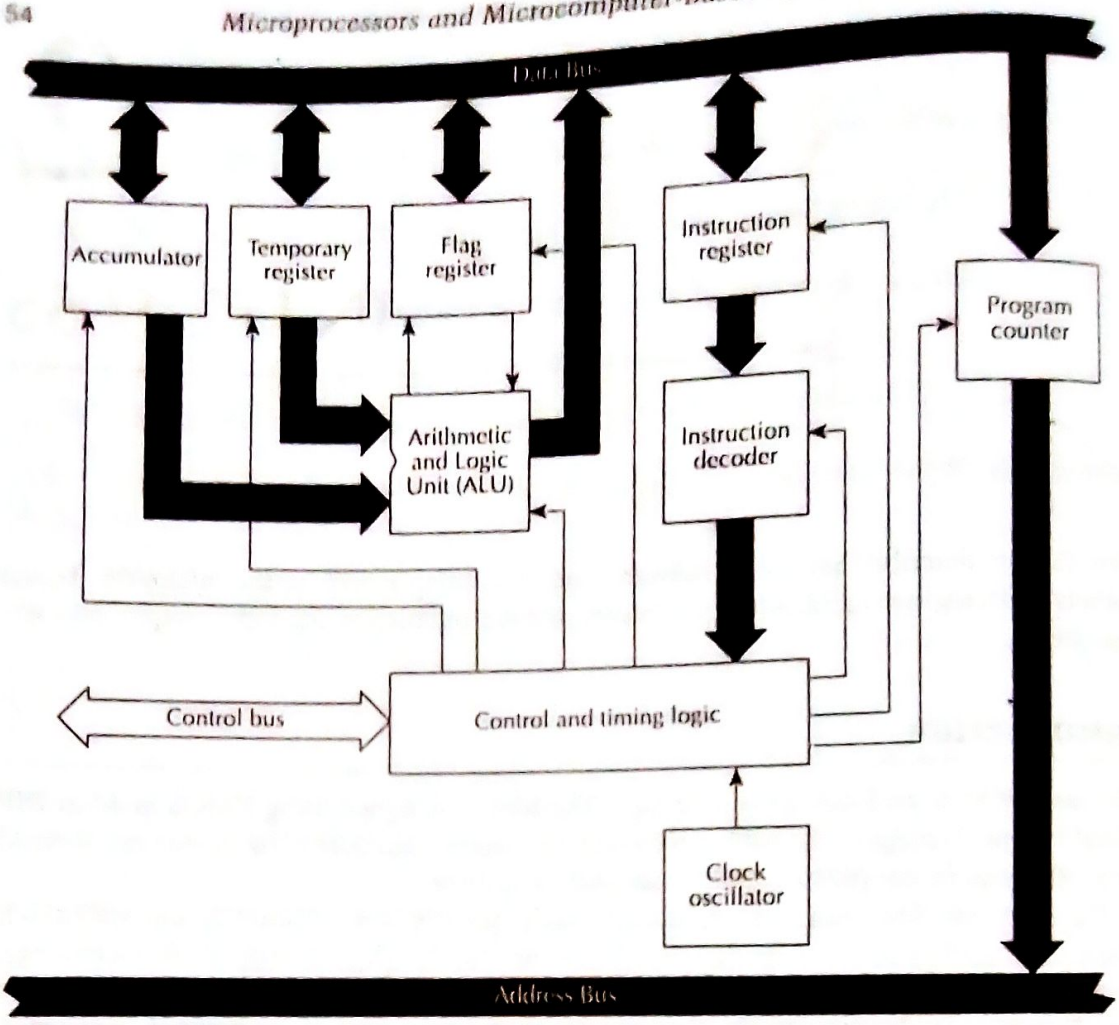


FIGURE 2.1 Simplified 8085 block diagram.

of 8-bit data being accessed from memory. This is the implied or register indirect addressing mode. There are a number of instructions, such as MOV reg, M, and MOV M, reg, which move data between any register and memory location addressed by H and L. However, using any other memory reference instruction, data transfer takes place between a memory location and the only 8085 register, the accumulator. The instruction LDAX B is a typical example.

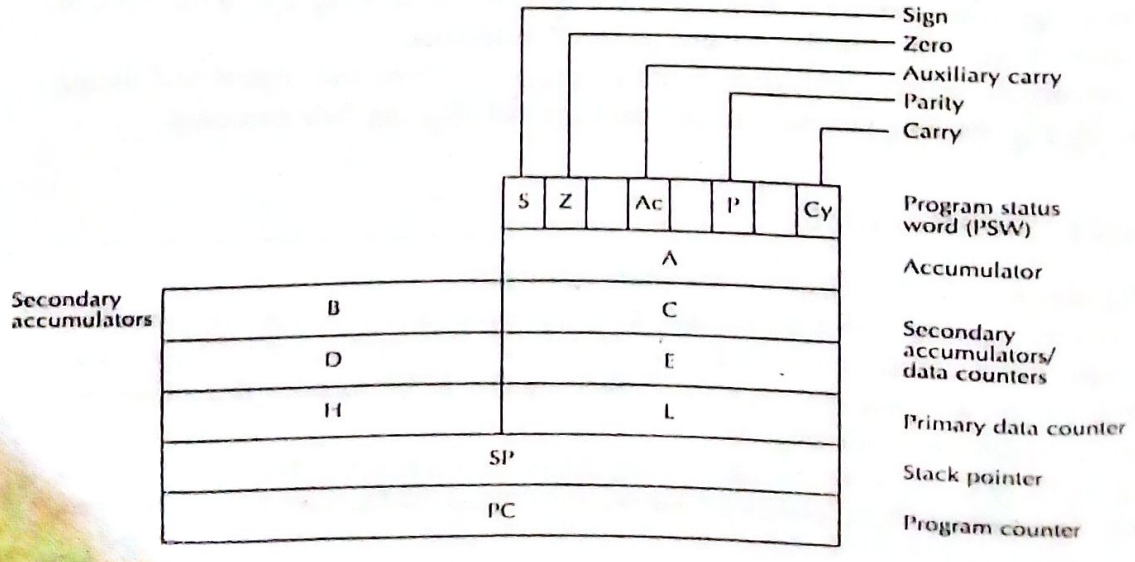


FIGURE 2.2 8085 microprocessor registers and status flags.

B, C, D, and E are secondary accumulators or data counters. There are a number of instructions to move data between any two registers. There are also a few instructions that combine registers B and C or D and E, as a 16-bit data counter with high byte of a pair contained in the first register and low byte in the second. These instructions typically include LDAX B, LDAX D, STAX B, and STAX D, which transfer data between memory and the accumulator.

Each of these 8-bit registers can be incremented and decremented by a single byte instruction. There are a number of instructions which combine two of these 8-bit registers to form 16-bit register pairs as follows:

A	and	PSW
B	and	C
D	and	E
H	and	L
high-order byte		low-order byte

The 16-bit register pair obtained by combining the accumulator and the program status word (PSW) is used only for stack operations. Sixteen bit arithmetic operations use B and C, D and E, or H and L as 16-bit data registers.

The **program status word** consists of five status flags. These are described below.

The **carry flag** (Cy) reflects the final carry out of the most significant bit of any arithmetic operation. Any logic instruction resets or clears the carry flag. This flag is also used by the shift and rotate instructions. The 8085 does not have any CLEAR CARRY instruction. One way of clearing the carry will be by ORing or ANDing the accumulator with itself.

The **parity status flag** (P) is set to 1 if an arithmetic or logic instruction generates an answer with even parity, that is, containing an even number of 1 bits. This flag is 0 if the arithmetic or logic instruction generates an answer with odd parity, that is, containing an odd number of 1s.

The **auxiliary carry flag** (Ac) reflects any carry from bit 3 to bit 4 (assuming 8-bit data with bit 0 as the least significant bit and bit 7 as the most significant bit) due to an arithmetic operation. This flag is useful for BCD operations.

The **zero flag** (Z) is set to 1 whenever an arithmetic or logic operation produces a result of 0. The zero flag is cleared to zero for a nonzero result due to arithmetic or logic operation.

The **sign status flag** (S) is set to the value of the most significant bit of the result in the accumulator after an arithmetic or logic operation. This provides a range of -128_{10} to $+127_{10}$ (with 0 being considered positive) as the 8085's data-handling capacity.

The 8085 does not have an overflow flag. Note that execution of arithmetic or logic instructions in the 8085 affects the flags. All conditional instructions in the 8085 instruction set use one of the status flags as the required condition.

The **stack pointer** (SP) is 16 bits long. All stack operations with the 8085 use 16-bit register pairs. The stack pointer contains the address of the last data byte written into the stack. It is decremented by 2 each time 2 bytes of data are written or pushed onto the stack and is incremented by 2 each time 2 bytes of data are read from or pulled (popped) off the stack, that is, the top of the stack has the lowest address in the stack that grows downward.

The **program counter** (PC) is 16 bits long to address up to 64K of memory. It usually addresses the next instruction to be executed.

2.3 Memory Addressing

When addressing a memory location, the 8085 uses either register indirect or direct memory addressing. With register indirect addressing, the H and L registers perform the function of the memory address register or data counter; that is, the H, L pair holds the address of the data.

With this mode, data transfer may occur between the addressed memory location and any one of the registers A, B, C, D, E, H, or L.

Also, some instructions, such as LDAX B, LDAX D, STAX B, and STAX D, use registers B and C or D and E to hold the address of data. These instructions transfer data between the accumulator and the memory location addressed by registers B and C or D and E using the register indirect mode.

There are also a few instructions, such as the STA ppqq, which use the direct-memory addressing mode to move data between the accumulator and the memory. These instructions use 3 bytes, with the first byte as the OP code followed by 2 bytes of address.

The stack is basically a part of the RAM. Therefore, PUSH and POP instructions are memory reference instructions.

All 8085 JUMP instructions use direct or absolute addressing and are 3 bytes long. The first byte of this instruction is the OP code followed by a 2-byte address. This address specifies the memory location to which the program would branch.

2.4 8085 Addressing Modes

The 8085 has five addressing modes:

1. **Direct** — Instructions using this mode specify the effective address as a part of the instruction. These instructions contain 3 bytes, with the first byte as the OP code followed by 2 bytes of address of data (the low-order byte of the address in byte 2, the high-order byte of the address in byte 3). Consider LDA 2035H. This instruction loads accumulator with the contents of memory location 2035_{16}. This mode is also called the absolute mode.
2. **Register** — This mode specifies the register or register pair that contains data. For example, MOV B, C moves the contents of register C to register B.
3. **Register Indirect** — This mode contains a register pair which stores the address of data (the high-order byte of the address in the first register of the pair, and the low-order byte in the second). As an example, LDAX B loads the accumulator with the contents of a memory location addressed by B, C register pair.
4. **Implied or Inherent** — The instructions using this mode have no operands. Examples include STC (Set the Carry Flag).
5. **Immediate** — For an 8-bit datum, this mode uses 2 bytes, with the first byte as the OP code, followed by 1 byte of data. On the other hand, for 16-bit data, this instruction contains 3 bytes, with the first byte as the OP code followed by 2 bytes of data. For example, MVI B, 05 loads register B with the value 5, and LXI H, 2050H loads H with 20H and L with 50H.

A JUMP instruction interprets the address that it would branch to in the following ways:

1. **Direct** — The JUMP instructions, such as JZ ppqq, use direct addressing and contain 3 bytes. The first byte is the OP code, followed by 2 bytes of the 16-bit address where it would branch to unconditionally or based on a condition if satisfied. For example, JMP 2020 unconditionally branches to location 2020H.
2. **Implied or Inherent Addressing** — This JUMP instruction using this mode is 1 byte long. A 16-bit register pair contains the address of the next instruction to be executed. The instruction PCHL unconditionally branches to a location addressed by the H, L pair.

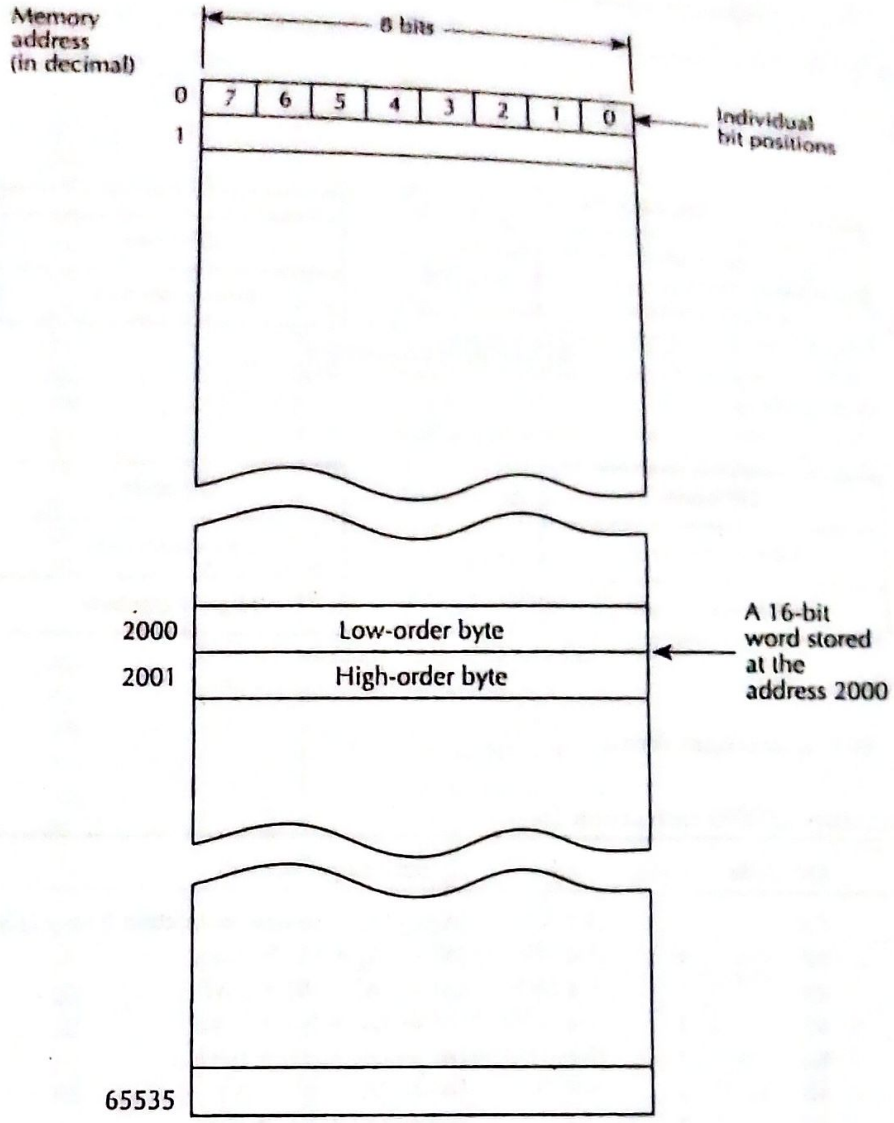


FIGURE 2.3 8085 addressing structures.

2.5 8085 Instruction Set

As mentioned before, the 8085 uses a 16-bit address. Since the 8085 is a byte-addressable machine, it follows that it can directly address 65,536 (2^{16}) distinct memory locations. The addressing structure of the 8085 processor is shown in Figure 2.3.

From this figure, we notice that two consecutive memory locations may be used to represent a 16-bit data item. However, according to the Intel convention, the high-order byte of a 16-bit quantity is always assigned to the high memory address.

The 8085 instructions are 1 to 3 bytes long and these formats are shown in Figure 2.4. The 8085 instruction set contains 74 basic instructions and supports conventional addressing modes such as immediate, register, absolute, and register indirect addressing modes.

Table 2.1 lists the 8085 instructions in alphabetical order; the object codes and instruction cycles are also included. When two instruction cycles are shown, the first is for "condition not met", while the second is for "condition met". Table 2.2 provides the 8085 instructions affecting the status flags. Note that not all 8085 instructions affect the status flags. The 8085 arithmetic and logic instructions normally affect the status flags.

In describing the 8085 instruction set, we will use the symbols in Table 2.3.

The 8085 move instruction transfers 8-bit data from one register to another, register to memory, and vice versa. A complete summary of these instructions is presented in Table 2.4.

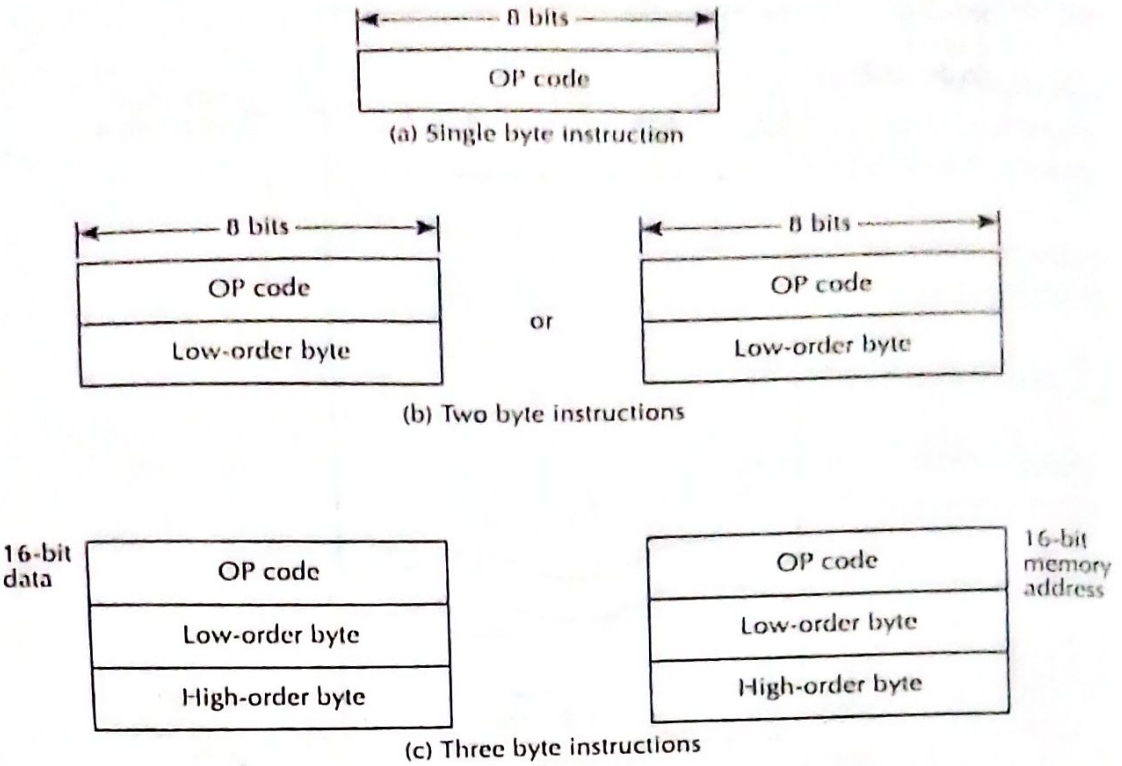


FIGURE 2.4 8085 instructions format.

Table 2.1 Summary of 8085 Instruction Set

Instruction	OP Code	Bytes	Cycles	Operations performed
ACI DATA	CE	2	7	[A] ← [A] + second instruction byte + [Cy]
ADC A	8F	1	4	[A] ←[A] + [A] + [Cy]
ADC B	88	1	4	[A] ←[A] + [B] + [Cy]
ADC C	89	1	4	[A] ← [A] + [C] + [Cy]
ADC D	8A	1	4	[A] ← [A] + [D] + [Cy]
ADC E	8B	1	4	[A] ← [A] + [E] + [Cy]
ADC H	8C	1	4	[A] ← [A] + [H] + [Cy]
ADC L	8D	1	4	[A] ← [A] + [L] + [Cy]
ADC M	8E	1	7	[A] ← [A] + [[H L]] + [Cy]
ADD A	87	1	4	[A] ← [A] + [A]
ADD B	80	1	4	[A] ← [A] + [B]
ADD C	81	1	4	[A] ← [A] + [C]
ADD D	82	1	4	[A] ← [A] + [D]
ADD E	83	1	4	[A] ← [A] + [E]
ADD H	84	1	4	[A] ← [A] + [H]
ADD L	85	1	4	[A] ← [A] + [L]
ADD M	86	1	7	[A] ← [A] + [[H L]]
ADI DATA	C6	2	7	[A] ← [A] + second instruction byte
ANA A	A7	1	4	[A] ← [A] ∧ [A]
ANA B	A0	1	4	[A] ← [A] ∧ [B]
ANA C	A1	1	4	[A] ← [A] ∧ [C]
ANA D	A2	1	4	[A] ← [A] ∧ [D]
ANA E	A3	1	4	[A] ← [A] ∧ [E]
ANA H	A4	1	4	[A] ← [A] ∧ [H]
ANA L	A5	1	4	[A] ← [A] ∧ [L]
ANA M	A6	1	4	[A] ← [A] ∧ [[H L]]
ANI DATA	E6	2	7	[A] ← [A] ∧ second instruction byte
CALL ppqq	CD	3	18	Call A subroutine addressed by ppqq
CC ppqq	DC	3	9/18	Call a subroutine addressed by ppqq if Cy = 1
CM ppqq	FC	3	9/18	Call a subroutine addressed by ppqq if S = 1
CMA	2F	1	4	[A] ← 1's complement of [A]
CMC	3F	1	4	[Cy] ← 1's complement of [Cy]
CMP A	BF	1	4	[A] – [A] and affects flags
CMP B	B8	1	4	[A] – [B] and affects flags

Table 2.1 Summary of 8085 Instruction Set (*continued*)

Instruction	OP Code	Bytes	Cycles	Operations performed
CMP C	B9	1	4	[A] – [C] and affects flags
CMP D	BA	1	4	[A] – [D] and affects flags
CMP E	BB	1	4	[A] – [E] and affects flags
CMP H	BC	1	4	[A] – [H] and affects flags
CMP L	BD	1	4	[A] – [L] and affects flags
CMP M	BE	1	7	[A] – [[H L]] and affects flags
CNC ppqq	D4	3	9/18	Call a subroutine addressed by ppqq if Cy = 0
CNZ ppqq	C4	3	9/18	Call a subroutine addressed by ppqq if Z = 0
CP ppqq	F4	3	9/18	Call a subroutine addressed by ppqq if S = 0
CPE ppqq	EC	3	9/18	Call a subroutine addressed by ppqq if P = 1
CPI DATA	FE	2	7	[A] – second instruction byte and affects flags
CPO ppqq	E4	3	9/18	Call a subroutine addressed by ppqq if P = 0
CZ ppqq	CC	3	9/18	Call a subroutine addressed by ppqq if Z = 1
DAA	27	1	4	Decimal adjust accumulator
DAD B	09	1	10	[HL] ← [HL] + [BC]
DAD D	19	1	10	[HL] ← [HL] + [DE]
DAD H	29	1	10	[HL] ← [HL] + [HL]
DAD SP	39	1	10	[HL] ← [HL] + [SP]
DCR A	3D	1	4	[A] ← [A] – 1
DCR B	05	1	4	[B] ← [B] – 1
DCR C	0D	1	4	[C] ← [C] – 1
DCR D	15	1	4	[D] ← [D] – 1
DCR E	1D	1	4	[E] ← [E] – 1
DCR H	25	1	4	[H] ← [H] – 1
DCR L	2D	1	4	[L] ← [L] – 1
DCR M	35	1	4	[[HL]] ← [[HL]] – 1
DCX B	0B	1	6	[BC] ← [BC] – 1
DCX D	1B	1	6	[DE] ← [DE] – 1
DCX H	2B	1	6	[HL] ← [HL] – 1
DCX SP	3B	1	6	[SP] ← [SP] – 1
DI	F3	1	4	Disable interrupts
EI	FB	1	4	Enable interrupts
HLT	76	1	5	Halt
IN PORT	DB	2	10	[A] ← [specified port]
INR A	3C	1	4	[A] ← [A] + 1
INR B	04	1	4	[B] ← [B] + 1
INR C	0C	1	4	[C] ← [C] + 1
INR D	14	1	4	[D] ← [D] + 1
INR E	1C	1	4	[E] ← [E] + 1
INR H	24	1	4	[H] ← [H] + 1
INR L	2C	1	4	[L] ← [L] + 1
INR M	34	1	4	[[HL]] ← [[HL]] +1
INX B	03	1	6	[BC] ← [BC] +1
INX D	13	1	6	[DE] ← [DE] + 1
INX H	23	1	6	[HL] ← [HL] + 1
INX SP	33	1	6	[SP] ← [SP] + 1
JC ppqq	DA	3	7/10	Jump to ppqq if Cy = 1
JM ppqq	FA	3	7/10	Jump to ppqq if S = 1
JMP ppqq	C3	3	10	Jump to ppqq
JNC ppqq	D2	3	7/10	Jump to ppqq if Cy = 0
JNZ ppqq	C2	3	7/10	Jump to ppqq if Z = 0
JP ppqq	F2	3	7/10	Jump to ppqq if S = 0
JPE ppqq	EA	3	7/10	Jump to ppqq if P = 1
JPO ppqq	E2	3	7/10	Jump to ppqq if P = 0
JZ ppqq	CA	3	7/10	Jump to ppqq if Z = 1
LDA ppqq	3A	3	13	[A] ← [ppqq]
LDAX B	0A	1	7	[A] ← [[BC]]
LDAX D	1A	1	7	[A] ← [[DE]]
	2A	3	16	[L] ← [ppqq],[H] ← [ppqq + 1]

Instruction	OP Code	Bytes	Cycles	Operations perfor
LXI B	01	3	10	[BC] ← second and third instruction bytes
LXI D	11	3	10	[DE] ← second and third instruction bytes
LXI H	21	3	10	[HL] ← second and third instruction bytes
LXI SP	31	3	10	[SP] ← second and third instruction bytes
MOV A,A	7F	1	4	[A] ← [A]
MOV A,B	78	1	4	[A] ← [B]
MOV A,C	79	1	4	[A] ← [C]
MOV A,D	7A	1	4	[A] ← [D]
MOV A,E	7B	1	4	[A] ← [E]
MOV A,H	7C	1	4	[A] ← [H]
MOV A,L	7D	1	4	[A] ← [L]
MOV A,M	7E	1	7	[A] ← [[HL]]
MOV B,A	47	1	4	[B] ← [A]
MOV B,B	40	1	4	[B] ← [B]
MOV B,C	41	1	4	[B] ← [C]
MOV B,D	42	1	4	[B] ← [D]
MOV B,E	43	1	4	[B] ← [E]
MOV B,H	44	1	4	[B] ← [H]
MOV B,L	45	1	4	[B] ← [L]
MOV B,M	46	1	7	[B] ← [[HL]]
MOV C,A	4F	1	4	[C] ← [A]
MOV C,B	48	1	4	[C] ← [B]
MOV C,C	49	1	4	[C] ← [C]
MOV C,D	4A	1	4	[C] ← [D]
MOV C,E	4B	1	4	[C] ← [E]
MOV C,H	4C	1	4	[C] ← [H]
MOV C,L	4D	1	4	[C] ← [L]
MOV C,M	4E	1	7	[C] ← [[HL]]
MOV D,A	57	1	4	[D] ← [A]
MOV D,B	50	1	4	[D] ← [B]
MOV D,C	51	1	4	[D] ← [C]
MOV D,D	52	1	4	[D] ← [D]
MOV D,E	53	1	4	[D] ← [E]
MOV D,H	54	1	4	[D] ← [H]
MOV D,L	55	1	4	[D] ← [L]
MOV D,M	56	1	7	[D] ← [[HL]]
MOV E,A	5F	1	4	[E] ← [A]
MOV E,B	58	1	5	[E] ← [B]
MOV E,C	59	1	4	[E] ← [C]
MOV E,D	5A	1	4	[E] ← [D]
MOV E,E	5B	1	4	[E] ← [E]
MOV E,H	5C	1	4	[E] ← [H]
MOV E,L	5D	1	4	[E] ← [L]
MOV E,M	5E	1	7	[E] ← [[HL]]
MOV H,A	67	1	4	[H] ← [B]
MOV H,B	60	1	4	[H] ← [A]
MOV H,C	61	1	4	[H] ← [C]
MOV H,D	62	1	4	[H] ← [D]
MOV H,E	63	1	4	[H] ← [E]
MOV H,H	64	1	4	[H] ← [H]
MOV H,L	65	1	4	[H] ← [L]
MOV H,M	66	1	7	[H] ← [[HL]]
MOV L,A	6F	1	4	[L] ← [A]
MOV L,B	68	1	4	[L] ← [B]
MOV L,C	69	1	4	[L] ← [C]
MOV L,D	6A	1	4	[L] ← [D]
MOV L,E	6B	1	4	[L] ← [E]
MOV L,H	6C	1	4	[L] ← [H]
MOV L,L	6D	1	4	[L] ← [L]

Table 2.1 Summary of 8085 Instruction Set (*continued*)

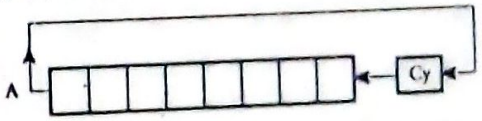
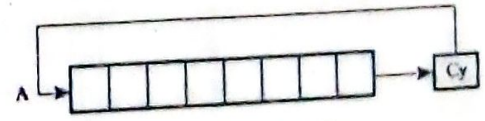
Instruction	OP Code	Bytes	Cycles	Operations performed
MOV L,M	6E	1	7	[L] ← [[HL]]
MOV M,A	77	1	7	[[HL]] ← [A]
MOV M,B	70	1	7	[[HL]] ← [B]
MOV M,C	71	1	7	[[HL]] ← [C]
MOV M,D	72	1	7	[[HL]] ← [D]
MOV M,E	73	1	7	[[HL]] ← [E]
MOV M,H	74	1	7	[[HL]] ← [H]
MOV M,L	75	1	7	[[HL]] ← [L]
MVI A, DATA	3E	2	7	[A] ← second instruction byte
MVI B, DATA	06	2	7	[B] ← second instruction byte
MVI C, DATA	0E	2	7	[C] ← second instruction byte
MVI D, DATA	16	2	7	[D] ← second instruction byte
MVI E, DATA	1E	2	7	[E] ← second instruction byte
MVI H, DATA	26	2	7	[H] ← second instruction byte
MVI L, DATA	2E	2	7	[L] ← second instruction byte
MVI M, DATA	36	2	10	[[HL]] ← second instruction byte
NOP	00	1	4	No operation
ORA A	B7	1	4	[A] ← [A] ∨ [A]
ORA B	B0	1	4	[A] ← [A] ∨ [B]
ORA C	B1	1	4	[A] ← [A] ∨ [C]
ORA D	B2	1	4	[A] ← [A] ∨ [D]
ORA E	B3	1	4	[A] ← [A] ∨ [E]
ORA H	B4	1	4	[A] ← [A] ∨ [H]
ORA L	B5	1	4	[A] ← [A] ∨ [L]
ORA M	B6	1	7	[A] ← [A] ∨ [[HL]]
ORI DATA	F6	2	7	[A] ← [A] ∨ second instruction byte
OUT PORT	D3	2	10	[specified port] ← [A]
PCHL	E9	1	6	[PCH]* ← [H], [PCL]* ← [L]
POP B	C1	1	10	[C] ← [[SP]], [SP] ← [SP] + 2 [B] ← [[SP] + 1]
POP D	D1	1	10	[E] ← [[SP]], [SP] ← [SP] + 2 [D] ← [[SP] + 1]
POP H	E1	1	10	[L] ← [[SP]], [SP] ← [SP] + 2 [H] ← [[SP] + 1]
POP PSW	F1	1	10	[A] ← [[SP] + 1], [PSW] ←[[SP]], [SP] ← [SP] + 2
PUSH B	C5	1	12	[[SP]– 1] ← [B], [SP] ← [SP] – 2 [[SP] – 2] ← [C]
PUSH D	D5	1	12	[[SP] – 1] ← [D], [[SP] – 2] ← [E] [SP] ← [SP] – 2
PUSH H	E5	1	12	[[SP] – 1] ← [H], [SP] ← [SP] – 2 [[SP] – 2] ← [L]
PUSH PSW	F5	1	12	[[SP] – 1] ← [A], [SP] ← [SP] – 2 [[SP] – 2] ← [PSW]
RAL	17	1	4	
RAR	1F	1	4	
RC	D8	1	6/12	Return if carry; [PC] ← [SP]]
RET	C9	1	10	[PCL]* ← [[SP]], [SP] ← [SP] + 2 [PCH]* ← [[SP] + 1]
RIM	20	1	4	Read interrupt mask
RLC	07	1	4	
RM	F8	1	6/12	Return if minus; [PC] ← [[SP]]

Table 2.1 Summary of 8085 Instruction Set (continued)

Instruction	OP Code	Bytes	Cycles	Operations performed
RNC	D0	1	6/12	Return if no carry; [PC] ← [[SP]]
RNZ	C0	1	6/12	Return if result not zero; [PC] ← [[SP]]
RP	F0	1	6/12	Return if positive; [PC] ← [[SP]], [SP] ← [SP] + 2
RPE	E8	1	6/12	Return if parity even; [PC] ← [[SP]], [SP] ← [SP] + 2
RPO	E0	1	6/12	Return if parity odd; [PC] ← [[SP]], [SP] ← [SP] + 2
RRC	0F	1	4	
RST0	C7	1	12	Restart
RST1	CF	1	12	Restart
RST2	D7	1	12	Restart
RST3	DF	1	12	Restart
RST4	E7	1	12	Restart
RST5	EF	1	12	Restart
RST6	F7	1	12	Restart
RST7	FF	1	12	Restart
RZ	C8	1	6/12	Return if zero; [PC] ← [[SP]]
SBB A	9F	1	4	[A] ← [A] − [A] − [Cy]
SBB B	98	1	4	[A] ← [A] − [B] − [Cy]
SBB C	99	1	4	[A] ← [A] − [C] − [Cy]
SBB D	9A	1	4	[A] ← [A] − [D] − [Cy]
SBB E	9B	1	4	[A] ← [A] − [E] − [Cy]
SBB H	9C	1	4	[A] ← [A] − [H] − [Cy]
SBB L	9D	1	4	[A] ← [A] − [L] − [Cy]
SBB M	9E	1	7	[A] ← [A] − [[HL]] − [Cy]
SBI DATA	DE	2	7	[A] ← [A] − second instruction byte − [Cy]
SHLD ppqq	22	3	16	[ppqq] ← [L], [ppqq + 1] ← [H]
SIM	30	1	4	Set interrupt mask
SPHL	F9	1	6	[SP] ← [HL]
STA ppqq	32	3	13	[ppqq] ← [A]
STAX B	02	1	7	[[BC]] ← [A]
STAX D	12	1	7	[[DE]] ← [A]
STC	37	1	4	[Cy] ← 1
SUB A	97	1	4	[A] ← [A] − [A]
SUB B	90	1	4	[A] ← [A] − [B]
SUB C	91	1	4	[A] ← [A] − [C]
SUB D	92	1	4	[A] ← [A] − [D]
SUB E	93	1	4	[A] ← [A] − [E]
SUB H	94	1	4	[A] ← [A] − [H]
SUB L	95	1	4	[A] ← [A] − [L]
SUB M	96	1	7	[A] ← [A] − [[HL]]
SUI DATA	D6	2	7	[A] ← [A] − second instruction byte
XCHG	EB	1	4	[D] ↔ [H], [E] ↔ [L]
XRA A	AF	1	4	[A] ← [A] ⊕ [A]
XRA B	A8	1	4	[A] ← [A] ⊕ [B]
XRA C	A9	1	4	[A] ← [A] ⊕ [C]
XRA D	AA	1	4	[A] ← [A] ⊕ [D]
XRA E	AB	1	4	[A] ← [A] ⊕ [E]
XRA H	AC	1	4	[A] ← [A] ⊕ [H]
XRA L	AD	1	4	[A] ← [A] ⊕ [L]
XRA M	AE	1	7	[A] ← [A] ⊕ [[HL]]
XRI DATA	EE	2	7	[A] ← [A] ⊕ second instruction byte
XTHL	E3	1	16	[[SP]] ↔ [L], [[SP] + 1]] ↔ [H]

*PCL — program counter low byte; PCH — program counter high byte.

Table 2.2 8085 Instructions Affecting the Status Flags

Instructions[a]	Status flags[b]				
	CY	AC	Z	S	P
ACI DATA	+	+	+	+	+
ADC reg	+	+	+	+	+
ADC M	+	+	+	+	+
ADD reg	+	+	+	+	+
ADD M	+	+	+	+	+
ADI DATA	+	+	+	+	+
ANA reg	0	1	+	+	+
ANA M	0	1	+	+	+
ANI DATA	0	1	+	+	+
CMC	+				
CMP reg	+	+	+	+	+
CMP M	+	+	+	+	+
CPI DATA	+	+	+	+	+
DAA	+	+	+	+	+
DAD rp	+				
DCR reg		+	+	+	+
DCR M		+	+	+	+
INR reg		+	+	+	+
INR M		+	+	+	+
ORA reg	0	0	+	+	+
ORA M	0	0	+	+	+
ORI DATA	0	0	+	+	+
RAL	+				
RAR	+				
RLC	+				
RRC	+				
SBB reg	+	+	+	+	+
SBB M	+	+	+	+	+
SBI DATA	+	+	+	+	+
STC	+				
SUB reg	+	+	+	+	+
SUB M	+	+	+	+	+
SUI DATA	+	+	+	+	+
XRA reg	0	0	+	+	+
XRA M	0	0	+	+	+
XRI DATA	0	0	+	+	+

[a]reg — 8-bit register; M — memory; rp — 16-bit register pair.

[b]Note that instructions which are not shown in the table do not affect the flags; + indicates that the particular flag is affected; 0 or 1 indicates that these flags are always 0 or 1 after the corresponding instructions are executed.

Table 2.3 Symbols to be Used in 8085 Instruction Set

Symbol	Interpretation
r1, r2	8-bit register
rp	Register pair
data8	8-bit data
data16	16-bit data
M	Memory location indirectly addressed through the register pair H,L
addr16	16-bit memory address

Table 2.4 8085 MOVE Instructions

Instruction	Symbolic description	Addressing mode		Illustration	
		Source	Destination	Example	Comments
MOV r1, r2	(r1) ← (r2)	Register	Register	MOV A, B	Copy the contents of the register B into the register A
MOV r, M	(r) ← M((HL))	Register indirect	Register	MOV A, M	Copy the contents of the memory location whose address is specified in the register pair H,L into the A register
MVI r, data8	(r) ← data8	Immediate	Register	MVI A, 08	Initialize the A register with the value 08
MOV M, r	M((HL)) ← (r)	Register	Register indirect	MOV M, B	Copy the contents of the B register into the memory location addressed by H,L pair
MVI M, data8	M((HL)) ← data8	Immediate	Register indirect	MVI M, 07	Initialize the memory location whose address is specified in the register pair H,L with the value 07

...quired. Now the program for the 5-s delay can be

```
        LXI  SP,  5000H   ;  Set stack pointer
        MVI  C,   32H     ;  Do DELAY loop 50(10) times by loading C
START   LXI  D,   30D3H   ;  with count 32(16)
        CALL DELAY        ;  Load initial count
        DCR  C            ;  Call DELAY loop
                          ;  Decrement C and check if zero: if
        JNZ  START        ;  not, do another delay
        HLT               ;  Loop back
                          ;  STOP
```

In the above, since execution times of DCR C and JNZ START are very small compared to 5 s, they are not considered in computing the delay.

2.7 8085 Pins and Signals

The 8085 is housed in a 40-pin dual in-line package (DIP). Figure 2.8 shows the 8085 pins and signals.

The low-order address byte and data lines AD0 to AD7 are multiplexed. These lines are bidirectional. The beginning of an instruction is indicated by the rising edge of the ALE signal. At the falling edge of ALE, the low byte of the address is automatically latched by some of the 8085 support chips such as 8155 and 8355: AD0 to AD7 lines can then be used as data lines. Note that ALE is an input to these support chips. However, if the support chips do not latch AD0 to AD7, then external latches are required to generate eight separate address lines A7 to A0 at the falling edge of ALE.

Pins A8 to A15 are unidirectional and contain the high byte of the address.

Table 2.11 lists the 8085 pins along with a brief description of each.

The $\overline{RD}$ pin signal is output LOW by the 8085 during a memory or I/O READ operation. Similarly, the $\overline{WR}$ pin signal is output LOW during a memory or I/O WRITE.

Next, we explain the purpose of IO/$\overline{M}$, S0, and S1 signals. The IO/$\overline{M}$ signal is output HIGH by the 8085 to indicate execution of an I/O instruction such as IN or OUT. This pin is output LOW during execution of a memory instruction such as LDA 2050H.

The IO/$\overline{M}$, S0, and S1 are output by the 8085 during its internal operations, which can be interpreted as follows:

IO/$\overline{M}$	S1	S0	Operation performed by the 8085
0	0	1	Memory WRITE
0	1	0	Memory READ
1	0	1	I/O WRITE
1	1	0	I/O READ
0	1	1	OP code fetch
1	1	1	Interrupt acknowledge

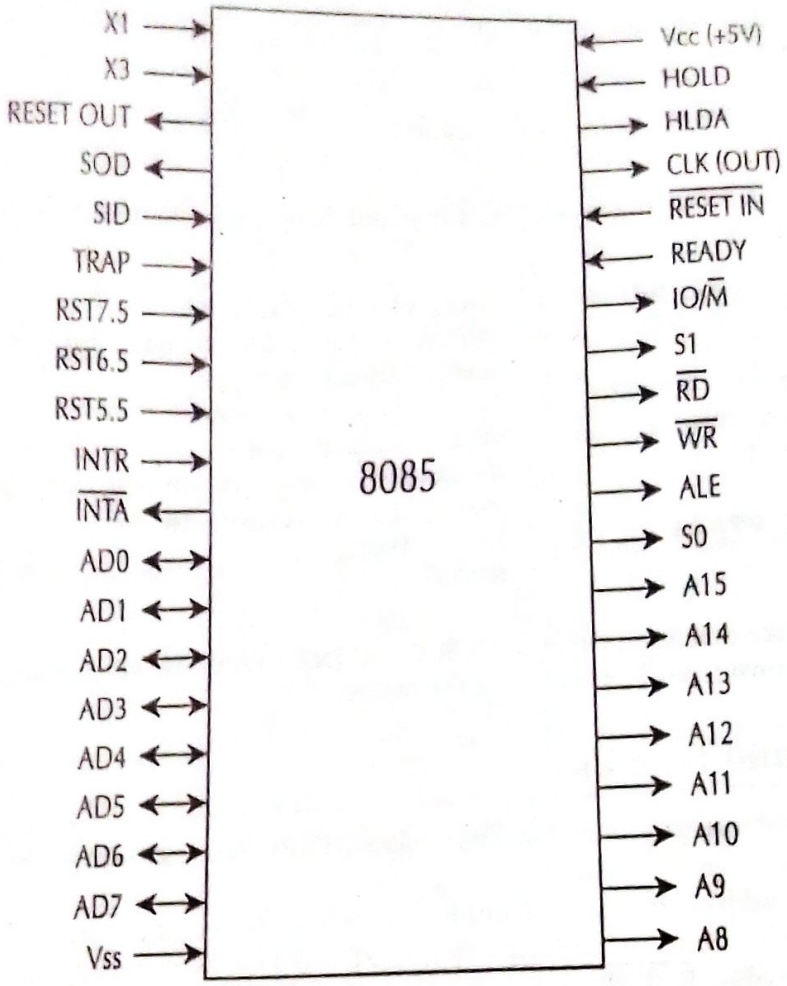
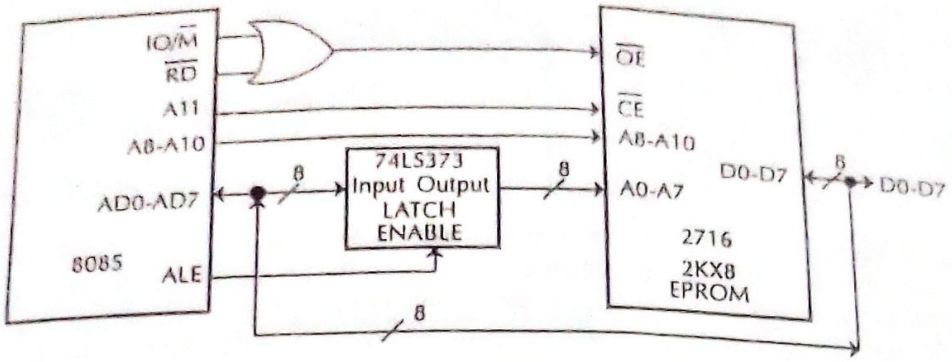


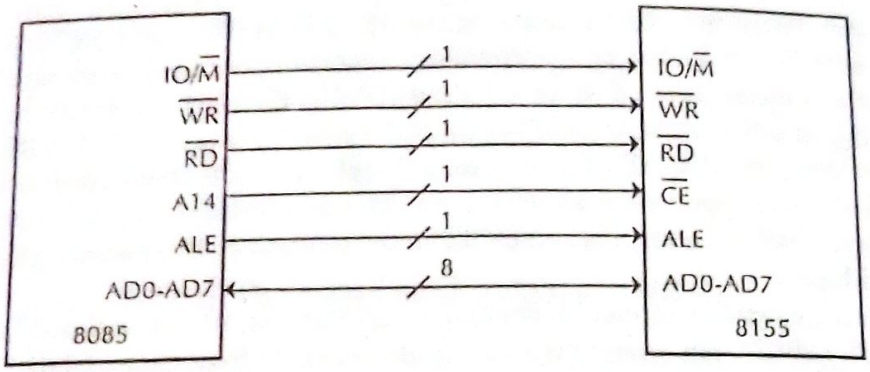
FIGURE 2.8 8085 microprocessor signals and pin assignments.

Table 2.11 8085 Signal Description Summary

Pin name	Description	Type
AD0-AD7	Address/data bus	Bidirectional, tristate
A8-A15	Address bus	Output, tristate
ALE	Address latch enable	Output, tristate
RD	Read control	Output, tristate
WR	Write control	Output, tristate
IO / M	I/O or memory indicator	Output, tristate
S0, S1	Bus state indicators	Output
READY	Wait state request	Input
SID	Serial data input	Input
SOD	Serial data output	Output
HOLD	Hold request	Input
HLDA	Hold acknowledge	Output
INTR	Interrupt request	Input
TRAP	Nonmaskable interrupt request	Input
RST5.5	Hardware vectored	Input
RST6.5	Hardware vectored interrupt request	Input
RST7.5	Hardware vectored	Input
INTA	Interrupt acknowledge	Output
RESET IN	System reset	Input
RESET OUT	Peripherals reset	Output
X1, X2	Crystal or RC connection	Input
CLK (OUT)	Clock signal	Output
Vcc, Vss	Power, ground	



(a) 8155 - 2716 interface using internal latches.



(b) 8085 - 8155 interface using ALE and AD0-AD7

FIGURE 2.9 8085's interface to external device using ALE and the multiplexed AD0 to AD7 pins.

Figure 2.9 illustrates the utilization of ALE and AD0 to AD7 signals for interfacing an EPROM and a RAM.

The 2716 is a 2K × 8 EPROM with separate address and data lines without any built-in latches. This means that a separate latch such as the 74LS373 must be used to isolate the 8085 low byte address and D0-D7 data lines at the falling edge of ALE (Figure 2.9a).

The 8155 contains 256-byte static RAM, three user ports, and a 14-bit timer. The 8155 is designed for 8085 in the sense that it has built-in latches with ALE as input along with multiplexed address (low byte) and data lines, AD0 to AD7. Therefore, as shown in Figure 2.9b, external latches are not required.

The READY input can be used by the slower external devices for obtaining extra time in order to communicate with the 8085. The READY signal (when LOW) can be utilized to provide wait-state clock periods in the 8085 machine. If READY is HIGH during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If not used, it must be tied high.

The Serial Input Data (SID) and Serial Output Data (SOD) lines are associated with the 8085 serial I/O transfer. The SOD line can be used to output the most significant bit of the accumulator. The SID signal can be input into the most significant bit of the accumulator.

The HOLD and HLDA signals are used for the Direct Memory Access (DMA) type of data transfer. The external devices place a HIGH on HOLD line in order to take control of the system bus. The HOLD function is acknowledged by the 8085 by placing a HIGH output on the HLDA pin and driving the tristate outputs high impedance.

The signals on the TRAP, RST7.5, RST6.5, RST5.5, INTR, and $\overline{\text{INTA}}$ are related to the 8085 interrupt signals. TRAP is a nonmaskable interrupt; that is, it cannot be enabled or disabled

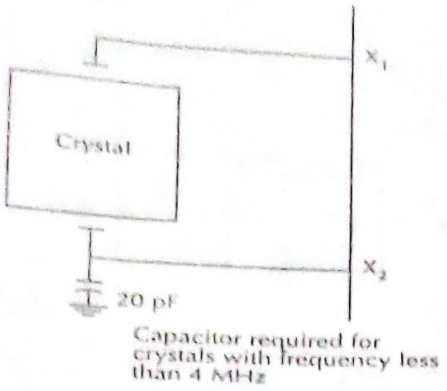


FIGURE 2.10 Crystal connection to X1 and X2 pins.

by an instruction. The TRAP has the highest priority. RST7.5, RST6.5, and RST5.5 are maskable interrupts used by the devices whose vector addresses are generated automatically. $\overline{\text{INTA}}$ is an interrupt acknowledge signal which is pulsed LOW by the 8085 in response to the interrupt INTR request. In order to service INTR, one of the eight OP codes (RST0 to RST7) has to be provided on the 8085 AD0-AD7 bus by external logic. The 8085 then executes this instruction and vectors to the appropriate address to service the interrupt.

All unused control pins such as interrupts and HOLD must be disabled by grounding them. (READY must be tied high).

The 8085 has the clock generation circuit on the chip and, therefore, no external oscillators need to be designed. The 8085A can operate with a maximum clock frequency of 3.03 MHz and the 8085A-2 can be driven with a maximum of 5 MHz clock. The 8085 clock frequency can be generated by a crystal, an LC tuned circuit, or an external clock circuit. The frequency at X1X2 is divided by 2 internally. This means that in order to obtain 3.03 MHz, a clock source of 6.06 MHz must be connected to X1X2. For crystals of less than 4 MHz, a capacitor of 20 pF should be connected between X2 and a ground to ensure the starting up of the crystal at the right frequency (Figure 2.10).

There is a TTL signal which is output on pin 37, called the CLK (OUT) signal. This signal can be used by other external microprocessors or support chips.

The $\overline{\text{RESET IN}}$ signal, when pulsed LOW then high, causes the 8085 to execute the first instruction at the 0000_{16} location. In addition, the 8085 resets instruction register, interrupt mask (RST5.5, RST6.5, and RST7.5) bits, and other registers. The $\overline{\text{RESET IN}}$ must be held LOW for at least three clock periods. A typical 8085 reset circuit is shown in Figure 2.11. In this circuit, when the switch is activated, $\overline{\text{RESET IN}}$ is driven to LOW with a large time constant providing adequate time to reset the system.

The 8085 requires a minimum operating voltage of 4.75 V. Upon applying power, the 8085A attains this voltage after 500 μs. The reset circuit of Figure 2.11 resets the 8085 upon activation

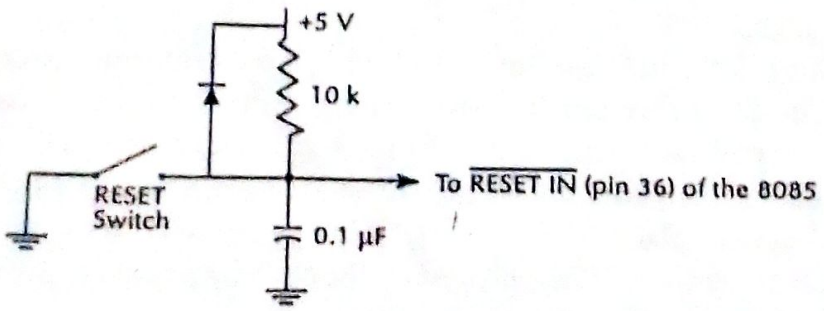


FIGURE 2.11 8085 reset circuit.

FIGURE 2.12 8085 machine cycles.

of the switch. The voltage across the 0.1-μF capacitor is zero on power-up. The capacitor then charges Vcc after a definite time determined by the time constant RC. The chosen values of RC in the figure will drive the $\overline{\text{RESET IN}}$ pin to low for at least three clock periods. In this case, after activating the switch, $\overline{\text{RESET IN}}$ will be low (assuming capacitance charge time is equal to the discharge time) for 10K*0.1 μF = 1 ms, which is greater than three clock periods (3*1/3 μs = 1 μs) of the 3-MHz 8085A. During normal operation of the 8085, activation of the switch will short the capacitor to ground and will discharge it. When the switch is opened, the capacitor charges and the $\overline{\text{RESET IN}}$ pin becomes HIGH. Upon hardware reset, the 8085 clears PC, IR, HALT flip-flop, and some other registers; the 8085 registers PSW, A, B, C, D, E, H, and L are unaffected. Upon activation of the $\overline{\text{RESET IN}}$ to low, the 8085 outputs HIGH at the RESET OUT pin which can be used to reset the memory and I/O chips connected to the 8085. Note that since hardware reset initializes PC to 0, the 8085 fetches the first instruction for address 0000_{16} after reset.

2.8 8085 Instruction Timing and Execution

An 8085's instruction execution consists of a number of machine cycles (MCs). These cycles vary from one to five (M1 to M5) depending on the instruction. Each machine cycle contains a number of 320-ns clock periods. The first machine cycle will be executed in either four or six clock periods, and the machine cycles that follow will have three clock periods. This is shown in Figure 2.12.

The shaded MCs indicate that these machine cycles are required by certain instructions. Similarly, the shaded clock periods (T5 and T6) mean that they are needed in M1 by some instructions.

The clock periods within a machine cycle can be illustrated as shown in Figure 2.13. Note that the beginning of a new machine cycle is indicated on the 8085 by outputting the Address Latch Enable (ALE) signal HIGH. During this time, lines AD0 to AD7 are used for placing the low byte of the address.

When the ALE signal goes LOW, the low byte of the address is latched so that the AD0 to AD7 lines can be used for transferring data.

We now discuss the timing diagrams for instruction fetch, READ, and WRITE.

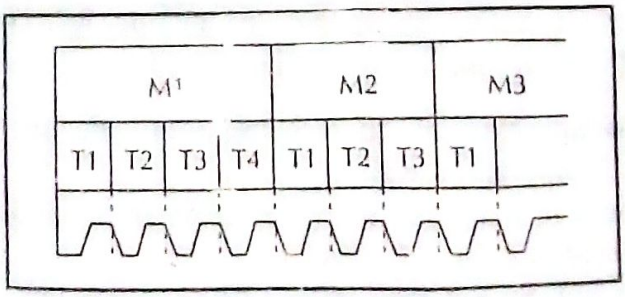


FIGURE 2.13 Clock period within a machine cycle.

Figure 2.14 shows the 8085 basic system timing. An instruction execution includes two operations: OP code fetch and execution.

The OP code fetch cycle requires either four (for one-byte instructions such as MOV A,B) or six cycles (for 3 byte instructions such as LDA 2030H). The machine cycles that follow will need three clock periods.

The purpose of an instruction fetch is to read the contents of a memory location containing an instruction addressed by the program counter and to place it in the instruction register. The 8085 instruction fetch timing diagram shown in Figure 2.14 can be explained in the following way:

1. The 8085 puts a LOW on the $IO/\overline{M}$ line of the system bus, indicating a memory operation.
2. The 8085 sets S0 = 1 and S1 = 1 on the system bus, indicating the memory fetch operation.
3. The 8085 places the program counter high byte on the A8 to A15 lines and the program counter low byte on the AD0 to AD7 lines of the system bus. The 8085 also sets the ALE signal to HIGH. As soon as the ALE signal goes to LOW, the program counter low byte on the AD0 to AD7 is latched automatically by some 8085 support chips such as 8155 (if 8085 support chips are not used, these lines must be latched using external latches), since these lines will be used as data lines for reading the OP code.
4. At the beginning of T2 in M1, the 8085 puts the $\overline{RD}$ line to LOW indicating a READ operation. After some time, the 8085 loads the OP code (the contents of the memory location addressed by the program counter) into the instruction register.
5. During the T4 clock period in M1, the 8085 decodes the instruction.

The Machine Cycle M2 of Figure 2.14 shows a memory (or I/O) READ operation as seen by the external logic, and the status of the S0 and S1 signals indicates whether the operation is instruction fetch or memory READ; for example, S1 = 1, S0 = 1 during instruction fetch and S1 = 1, S0 = 0 during memory READ provided $IO/\overline{M} = 0$.

The purpose of the memory READ is to read the contents of a memory location addressed by a register pair, such as the H,L pair, or a memory location specified with the instruction and the data placed in a microprocessor register such as the accumulator. In contrast, the purpose of the memory fetch is to read the contents of a memory location addressed by PC into IR. The machine cycle M3 of Figure 2.14 indicates a memory (or I/O) write operation. In this case, S1 = 0 and S0 = 1 indicate a memory write operation when $IO/\overline{M} = 0$ and an I/O write operation when $IO/\overline{M} = 1$.

2.8.2 8085 Memory READ ($IO/\overline{M} = 0$, $\overline{RD} = 0$) and I/O READ ($IO/\overline{M} = 1$, $\overline{RD} = 0$)

Figure 2.15a shows an 8085A clock timing diagram. The machine cycle of M2 of Figure 2.14 shows a memory READ timing diagram.

The purpose of the memory READ is to read the contents of a memory location addressed by a register pair, such as HL. Let us explain the 8085 memory READ timing diagram of Figure 2.15b along with the READ timing signals of Figure 2.14:

1. The 8085 uses machine cycle M1 to fetch and decode the instruction. It then performs the memory READ operation in M2.
2. The 8085 continues to maintain $IO/\overline{M}$ at LOW in M2 indicating a memory READ operation (or $IO/\overline{M} = 1$ for I/O READ).
3. The 8085 puts S1 = 1, S0 = 0, indicating a READ operation.
4. The 8085 places the contents of the high byte of the memory address register, such as the contents of the H register, on lines A8 to A15.

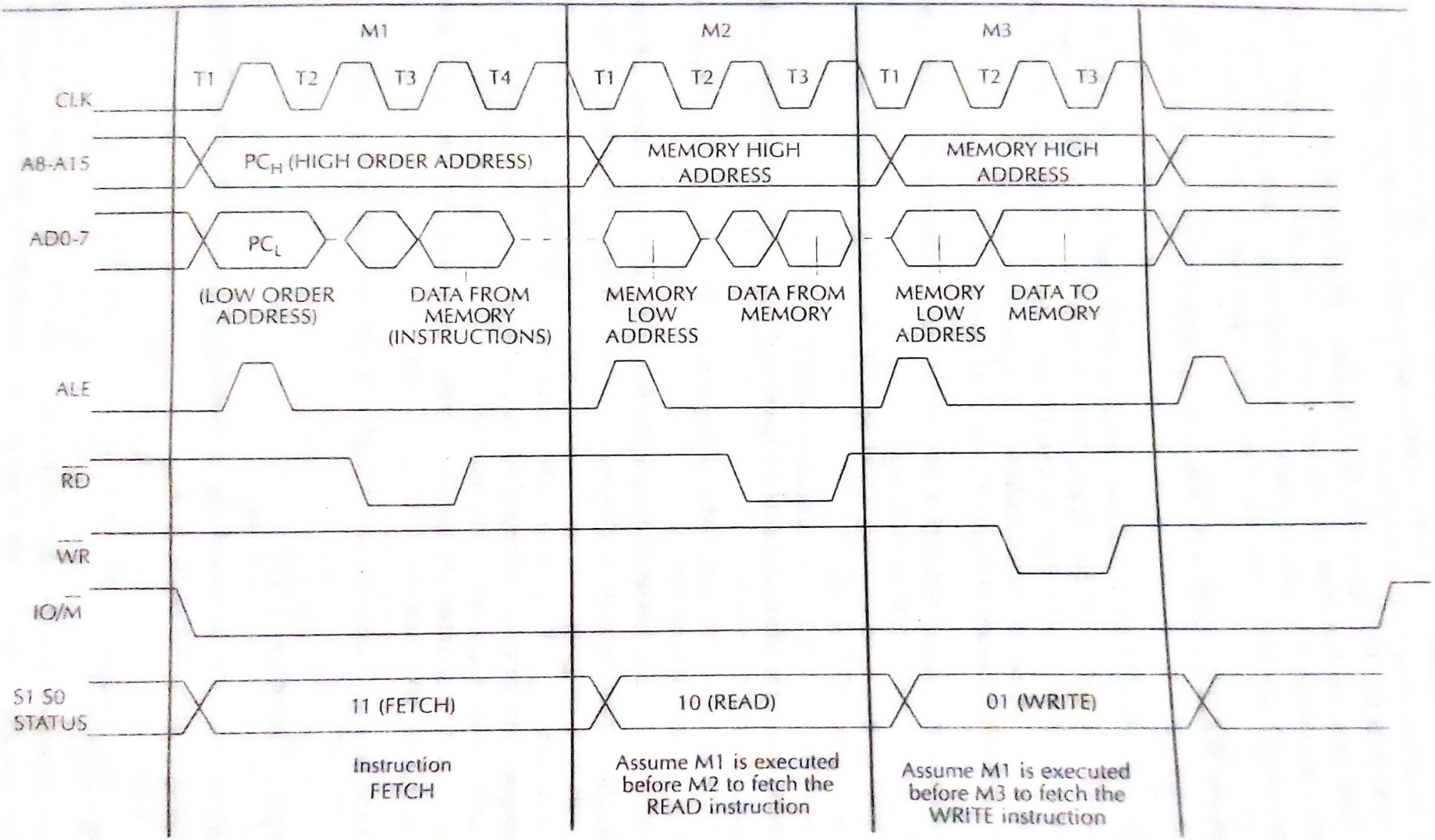


FIGURE 2.14 8085 Basic system timing.

5. The 8085 places the contents of the low byte of the memory address register, such as the contents of the L register, in lines AD0 to AD7.
6. The 8085 sets ALE to high, indicating the beginning of M2. As soon as ALE goes to low, the memory or support chip must latch the low byte of the address lines, since the same lines are going to be used as data lines.
7. The 8085 puts the $\overline{RD}$ signal to LOW, indicating a READ operation.
8. The 8085 gets the data from the memory location addressed by the memory address register, such as the H,L pair, and places the data into a register such as the accumulator. In case of I/O, the 8085 inputs data from the I/O port into the accumulator.

2.8.3 8085 Memory WRITE ($IO/\overline{M} = 0$, $\overline{WR} = 0$) and I/O WRITE ($IO/\overline{M} = 1$, $\overline{WR} = 0$)

The machine cycle M3 of Figure 2.14 shows a memory WRITE timing diagram. As seen by the external logic, the signals S1 = 0, S0 = 1, and $\overline{WR} = 0$ indicate a memory WRITE operation.

The purpose of a memory WRITE is to store the contents of the 8085 register, such as the accumulator, into a memory location addressed by a pair, such as H,L.

The WRITE timing diagram of Figure 2.14 can be explained as follows:

1. The 8085 uses machine cycle M1 to fetch and decode the instruction. It then executes the memory WRITE instruction in M3.
2. The 8085 continues to maintain $IO/\overline{M}$ at LOW, indicating a memory operation (or $IO/\overline{M} = 1$ for I/O WRITE).
3. The 8085 puts S1 = 0, S0 = 1, indicating a WRITE operation.
4. The 8085 places the Memory Address Register high byte, such as the contents of the H register, on lines A8 to A15.
5. The 8085 places the Memory Address Register low byte, such as the contents of L register, on lines AD0 to AD7.
6. The 8085 sets ALE to HIGH, indicating the beginning of M3. As soon as ALE goes to LOW, the memory or support chip must latch the low byte of the address lines, since the same lines are going to be used as data lines.
7. The 8085 puts the $\overline{WR}$ signal to LOW, indicating a WRITE operation.
8. It also places the contents of the register, say, accumulator, on data lines AD0 to AD7.
9. The external logic gets data from the lines AD0 to AD7 and stores the data in the memory location addressed by the Memory Address Register, such as the H,L pair. In case of I/O, the 8085 outputs [A] to an I/O port.

Figures 2.15a through c show the 8085A clock and read and write timing diagrams.

2.9 8085 Input/Output (I/O)

The 8085 I/O transfer techniques are discussed. The 8355/8755 and 8155/8156 I/O ports and 8085 SID and SOD lines are also included.

2.9.1 8085 Programmed I/O

There are two I/O instructions in the 8085, namely, IN and OUT. These instructions are 2 bytes long. The first byte defines the OP code of the instruction and the second byte specifies the I/O port number. Execution of the IN PORT instruction causes the 8085 to receive one byte of data into the accumulator from a specified I/O port. On the other hand, the OUT PORT instruction, when executed, causes the 8085 to send one byte of data from the accumulator into a specified I/O port.

FIGURE 2.15a 8085A clock.

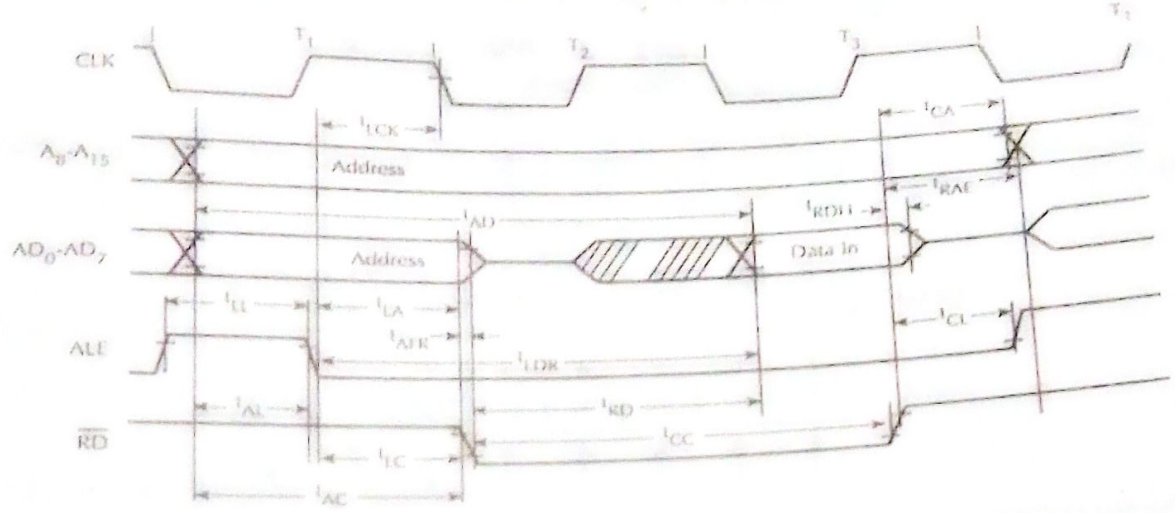


FIGURE 2.15b 8085 Read Timing diagram.

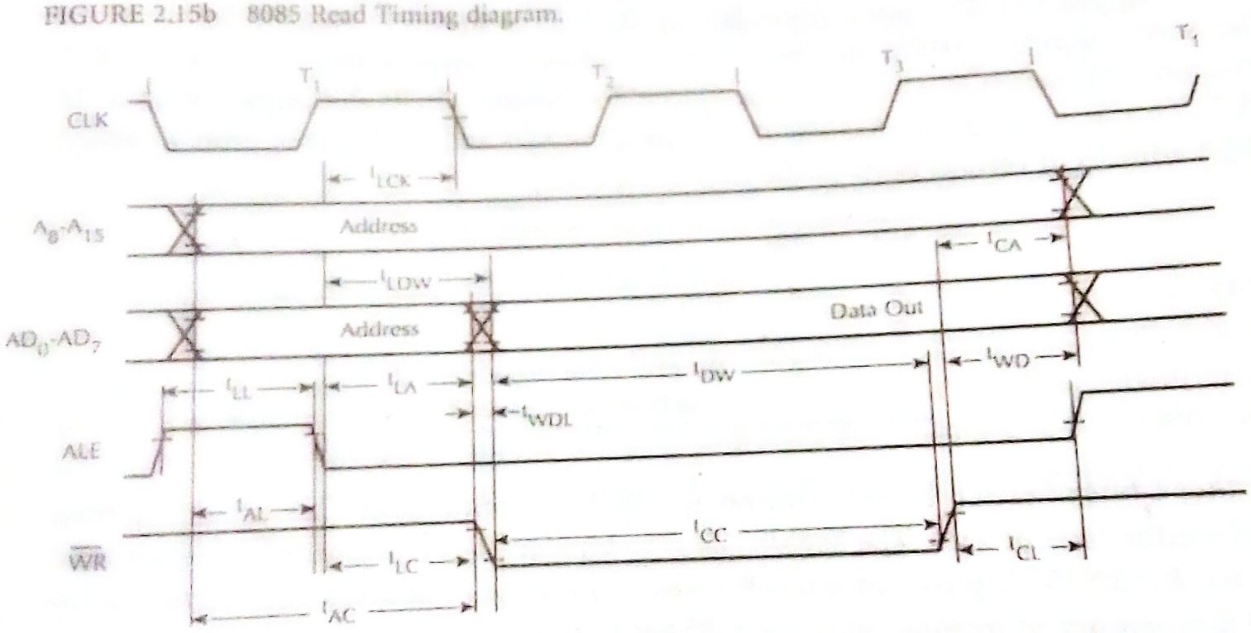


FIGURE 2.15c 8085 Write Timing diagram.

The 8085 can access I/O ports using either standard I/O or memory-mapped I/O.

In standard I/O, the 8085 inputs or outputs data using IN or OUT instructions.

In memory-mapped I/O, the 8085 maps I/O ports as memory addresses. Hence, LDA addr or STA addr instructions are used to input or output data to or from the 8085. The 8085's programmed I/O capabilities are obtained via the support chips, namely, 8355/8755 and 8155/8156. The 8355/8755 contains a 2K-byte ROM/EPROM and two 8-bit I/O ports (ports A and B).

The 8155/8156 contains 256-byte RAM, two 8-bit and one 6-bit I/O ports, and a 14-bit programmable timer. The only difference between the 8155 and 8156 is that chip enable is LOW on the 8155 and HIGH on the 8156.