# ARCHITECTURE OF 8085 MICROPROCESSOR (8 BIT)

The 8085A is an 8 bit general purpose microprocessor capable of addressing 64K of memory. The main components of the 8085 include the ALU, timing and control unit, Instruction register and decoder, Register array. These are linked by an internal data bus.

**The ALU**

The arithmetic/logic unit performs the computing functions; it includes the accumulator the temporary register, the arithmetic and logic circuits, and five flags. The temporary register is used to hold the data during an arithmetic/logic operation. The result of the operation is stored in the accumulator and flags are set or reset according to the result of operation. Flags generally reflect data conditions in the accumulator.

**Timing and Control Unit**

This unit synchronizes all the microprocessor operations with the clock and generates the control signals necessary for communication between the microprocessor and peripherals.

**Instruction Register and Decoder**

The instruction register and the decoder are part of the ALU. When an instruction is fetched from memory, it is loaded in the instruction register. The decoder decodes the instruction and establishes the sequences of events to follow (what operation is to be performed).

**Accumulator**

The accumulator (Register A) is an 8 bit register that is part of the arithmetic/logic unit(ALU). This register is accessible to user and holds one of the operands and the result itself.
8085 is accumulator based microprocessor because the operation of 8085 depends on the accumulator. Almost all the arithmetic and logic functions are performed on the data are in the accumulator

**Registers B,**
**C, D, E**

These four 8 bit registers are accessible to the programmer. These can be used individually as 8 bit individual registers or in pairs as BC and DE as 16 bit registers.

**Register H & L**

Register H & L can be used as 8 bit registers or as pairs. However, these can also be used for indirect addressing.

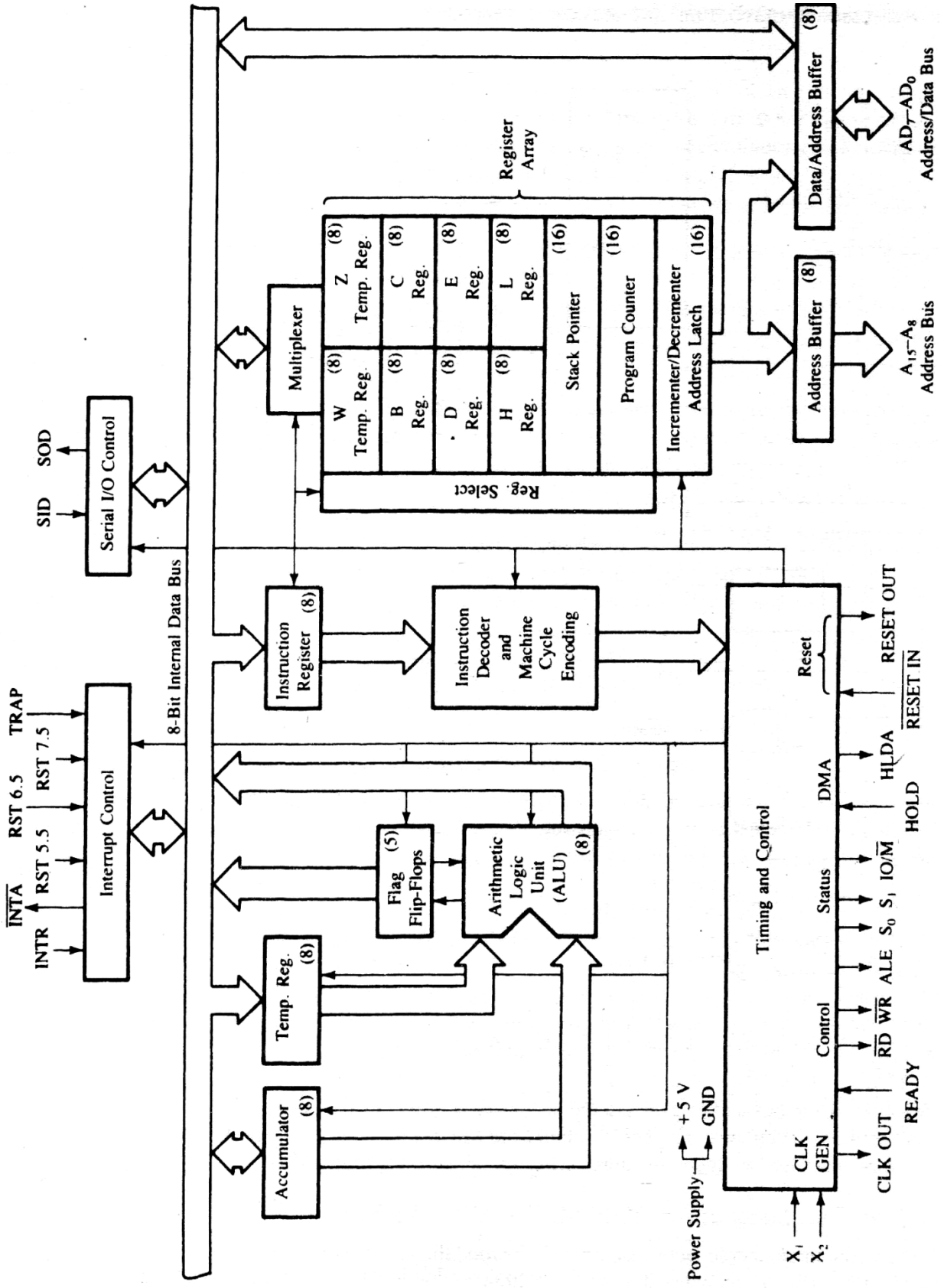



Fig: Functional Block Diagram of 8085A

**Program Counter (PC) and Stack Pointer (SP)**

These are 16 bit registers used to hold addresses. The size of these registers is 16 bits because the memory addresses are 16 bits.

The computer program consists of the sequence of coded instructions. These are stored in memory. The microprocessor uses the PC register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. PC is incremented by one when the machine code is fetched.

A stack is an area of memory set aside for the purpose of storing data. The location of the memory is defined by the 16 bit register called SP. When a data is stored its value is decreased by 1 while it is increased by 1 when the data is retrieved.

**Flags**

It consists of a register with five flip flops, each holding a specific status of the output of the operation of ALU. The register is known as Flag Register and each flip flop is called flag. The states of the flags indicate the result of arithmetic and logic operation, which in turn is used for decision making processes. There are no decision instructions for AC flag. The different flags are:

- S – Sign Flag: Is set if the MSB of the result of last operation is 1. (Also called negative, since 1 in the MSB position of two"s complement number)
- Z – Zero Flag: Is set if the ALU operation results in 0, and flag is reset if the result is not 0. Operation of the flag is affected by accumulator as well as that of other registers.
- AC – Auxiliary Carry Flag: If the arithmetic operation generates a carry from lower nibble to upper nibble then this flag is set. It is useful while performing BCD arithmetic.
- P – Parity Flag: Is set if the result has an even numbers of 1s while reset if the number of 1 is odd.
- CY – Carry Flag: Is set if the arithmetic operation results in a carry and is reset if otherwise.

The bit positions reserved for these flags in the flag register are as follows:

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| S | Z | | AC | | P | | CY |

## ADDRESSING MODES

Instructions are commands to the microprocessor to perform a certain task. The instructions consist of Op-code and data called operand. The operand may be source only, destination only or both of them.

For example:

```
MOV     A,      M
        ↑       ↑
    Destination, Source

LDA     3000H
          ↑
        Source

STA     3000H
          ↑
        Destination
```

In these instructions, the source can be register, input port and a memory location. Similarly, a destination port can be a register and output port or a memory location.The various format of specifying the operand are called addressing modes. The way operands are chosen during program execution is dependent on the addressing mode. Addressing mode specifies the rule for interpreting and modifying the address of the instruction before the operand is actually referenced. The 8085 instruction set has six different addressing modes.

- Direct Addressing
- Register Indirect Addressing
- Register Direct Addressing
- Implied Addressing
- Immediate Addressing

**Direct Addressing (Absolute addressing mode):**

This addressing mode is called direct because the effective address of the operand is specified directly in the instruction. Instructions using this mode may contain 2 or 3 bytes, with first byte as the Op-code followed by 1 or 2 bytes of address of data. Loading and storing in memory use 2 bytes of address while IN and OUT have one byte address.

For example:

LDA 9089H

This instruction loads the accumulator with the contains of the memory location having a 16 bit address of 9089H. This type of addressing is also called absolute addressing.

IN and OUT instructions are the examples of the instructions which have only 1-byte address.

For example:

IN FCH

**Register Indirect Addressing:**

In the mode, the address part of the instruction specifies the memory location whose content is the address of the operand. So in this type of addressing mode, it is the address of the address rather than the address itself is part of the

instruction. In 8085, wherever the instruction uses HL pointer the address is called indirect addressing.
For example:

MOV A, M

Here, the content of the memory location whose address is specified by the content of the register H & L is moved to the accumulator. It is called indirect addressing because the register contains the address of the memory location. If register H & L contains 90H, 89H respectively, then the content of the memory location 9089H will be transferred to Register A.

**Register Direct Addressing:**

Register direct addressing mode means that a register is the source of an operand for an instruction.
For example:

MOV A, B

This instruction moves the contents of the register B to the accumulator. Here, register B contains data rather than the address of the data. So, it is similar to direct addressing.

**Implied or Inherent Addressing:**

The instructions of this mode do not have operands.
For example:

EI (Enable Interrupt),
STC (set the carry flag),
NOP (No operation)

**Immediate Addressing:**

This is the simplest form of addressing. The operand is present in instruction in this mode is used to define and use constants of set initial values of variables. In this mode, the instruction will operate on immediate hexadecimal number. The operand may be 8 bit data or 16 bit data.
For example:

MVI B, 05H

This 2- bytes instruction with the 1$^{st}$ byte as op-code, loads Register B with 8 bit data.

LXI B, 5009H

This is 3- bytes instruction with the 1$^{st}$ byte as op-code followed by 2 bytes of data. This instruction loads Register B with 50H and Register C with 09H.

## INSTRUCTION WORD SIZE

During the execution of an instruction different actions are exercised. The preliminary action: op-code fetch is same for all the instructions however the latter action depends on the size of the instruction. A microprocessor can handle only a fixed number of instructions in case of 8085 it can handle a maximum of $2^8$ i.e. 256 instructions. The different instructions and their equivalent op-codes are available in the 8085 instruction sheet.

Depending on the byte size the instruction set is classified into the following three groups.

- One byte Instructions: A 1byte instruction includes the op-code and the operand in the same byte. The operand will be default for that instruction. These instructions require one memory locations. E.g.: MOV C,A , ADD B, CMA etc.

| Op-code | Operand |
|---|---|
| MOV | C, A |
| ADD | B |
| CMA | |

- Two byte Instructions: In a 2byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Thus, two memory locations are required. E.g.: MVI A, 32H, ADI F2H etc.

| Op-code | Operand |
|---|---|
| MVI A | 32H |
| ADI | F2H |

- Three byte Instruction: In three byte instruction first byte will be op-code where as second and third byte will be the operands. Operands may be data or address depending upon the instruction. E.g.: LXI B, 4455H, LDA 2050H, JMP 2085H etc.

| Op-code | Operand |
|---|---|
| LXI B | 4455H |
| LDA | 4050H |
| JMP | 2085H |

## INSTRUCTION GROUPS OF 8085

An "instruction" is a binary pattern designed to perform a specific function. The list of entire instructions is called the instruction set. The instruction set determines what function the microprocessor can perform. The 8085 instruction set can be classified into the following five categories:

- Data transfer group
- Arithmetic group
- Logical group
- Branching group
- Miscellaneous group

**Data transfer group instructions:**

The data transfer group is the longest group of instructions in 8085 and consists of 88 different op-codes. This group of instructions copy data from a source location to the destination location, without modifying the contents of the source. The transfer of data may be between the registers or between register and memory or between an I/O device and accumulator. None of these instructions modify the flags.

- Load register (immediate instruction): loads the designated register with the operand.

  Some of the instructions are:

  MVI A, data  MVI B, data  MVI C, data  MVI D, data

  MVI E, data  MVI H, data  MVI L, data

- Load memory (immediate): loads the memory location whose address is contained in registers H and L with the operand

  Instruction  MVI M, data

- Load register pairs (Immediate): loads the register pair with 2 bytes of data. The register pairs are B>B&C, D>D&E, H>H&L. The instructions are:

  LXI B, data  LXI D, data  LXI H, data

- Load Accumulator (direct): loads the accumulator with the contents of memory location specified by the 2 byte address. The instruction is LDA 2050H.

  Instructions  LDA, address

- Load Accumulator (indirect): load the accumulator with the contents of memory location whose address is contained in the designated register pair (only register pair B & D are used). The instructions are LDAX B & LDAX D.

  Instruction  LDAX B

- Move register: this instruction loads the first specified register with the contents of the second specified register. The instructions are MOV A, B , MOV C, D.

  Instruction  MOV A, B

- Move to memory from designated registers: load the memory location whose address is contained in register H-L with the contents of the register. The instructions are MOV M,A , MOV M,D.

  Instruction  MOV M, reg

- Move to designated registers from memory: loads the designated register with the contents of the memory location whose address is specified by the contents of register H&L. The instructions are MOV A,M , MOV D,M.

- Store Accumulator contents (indirect address): loads the memory location whose address is contained in the designated register pair (only register pair B & D are used) with the contents of the accumulator. The instructions are STAX B, STAX D.

- Store Accumulator Contents (direct): loads the memory location whose address is specified by the operand. The instruction is STA operand.

- Exchange contents of H&L with D&E: This instruction is used to exchange the contents of register H with the contents of register D and the contents of register L with the contents of register E. The instruction is XCHG.

- Load H & L directly: This instruction loads the registers H&L with the contents of the memory location whose address is the operand of the instruction. (Register L contains the value at the address and Register H contains the value at the address+1)

- Store H&L directly: This instruction is similar to load instruction and stores the contents of H & L at the designated memory address.

**Arithmetic group instructions:**

There are different instructions for arithmetic operations in 8085. The arithmetic operations performed by 8085 are addition, subtraction, increment and decrement. All the addition and subtraction operations are performed with accumulator. However, increment and decrement can be performed in any register and affect the flags except carry flag.

For addition and subtraction the following features exist:

- The instructions assume implicitly that the accumulator is one of the operands.
- The flags get modified according to the data conditions of the result.
- The result is placed in the accumulator.
- The execution of the instruction does not affect the operand register.

The different instructions are:

ADD register: Add the contents of a register with the accumulator
ADI data: Add immediate 8 bit data with the accumulator
ADC register: Add the contents of a register with the accumulator and the carry ADI data: Add immediate 8 bit data with the accumulator and the carry SUB register: Subtract the contents of register from accumulator
SBB register: Subtract the contents of register and borrow from accumulator
SUI data: Subtract the immediate data from the accumulator
SBI data: Subtract the immediate data and borrow from accumulator
INR register: Increment the contents of register by 1
DCR register: Decreases the contents of register by 1

**Logical group instructions:**

A microprocessor is programmable logic device. All the operations of the microprocessor is carried out via the hardwired logic circuits. The logical functions that 8085 microprocessor can perform are AND, OR, Ex-OR and NOT.

The different instructions are:

ANA register: Logical AND the contents of register with accumulator ANI data: Logical AND the immediate data with accumulator ORA register: Logical OR the contents of register with accumulator ORI data: Logical OR the immediate data with accumulator
XRA register: Logical exclusive OR the contents of register with accumulator
XRI data: Logical exclusive OR the data with accumulator
CMA: Complements the contents of accumulator

The following features hold true for all logic instructions:

- The accumulator is one of the operands
- Except the instruction CMA, all instruction reset carry flag
- They modify the Z, P, A, S flags according to the data conditions of result
- They place the result in accumulator
- The contents of operand registers are not affected.

**Branching group instructions:**

The branch instructions allow the microprocessor to change the sequence of a program, either on certain conditions or unconditionally. The microprocessor is a sequential machine i.e. the instructions are executed from memory location sequentially. Branch instructions cause the microprocessor to execute from another memory locations. The branch instructions can be categorized as:

- Jump Instruction
- Call and Return Instruction
- Restart Instruction

Jump Instruction:

The jump instructions specify the memory location explicitly. These are 3 byte instructions where the first byte is op-code and other bytes represent the 16 bit address. The jump may be conditional or unconditional.

An unconditional jump enables the programmer to set up continuous loops without depending on any type of conditions. The instruction is JMP data. The jump location can also be specified using a label if the exact memory location which a program sequence should be directed, is not known. However, it should be considered that a single label cannot be used for different memory locations.

A conditional jump instruction allows the microprocessor to make decision based on certain conditions indicated by the flags. After a arithmetic or logical operation, flags are set or reset to reflect the data conditions. These instructions check flag conditions and make decision to change the sequence of program. The flags tested for conditional jump are zero, carry, sign and parity. The different instructions are:

JC: Jump on carry
JNC: Jump on if no carry
JZ: Jump on zero

JNZ: Jump on no zero
JP: Jump on plus
JM: Jump on minus
JPE: Jump on even parity
JPO: Jump on odd parity

Call and return Instructions:

Call and return instructions are used for using subroutine. A subroutine is a group of instruction that performs a subtask of repeated occurrence. The subroutine is written as a separate unit, apart from the main program and the control from program is transferred to subroutine when the call instruction is encountered. After the completion of the subroutine the control is returned back to the main program.

The call instruction and return instructions can also be set conditionally or unconditionally. The different call and return instructions are:

CALL: Unconditional call instruction
RET: Unconditional return instruction
CC, CNC, CZ, CNZ, CM, CP, CPE, CPO: Conditional Call
RC, RNC, RZ, RNZ, RM, RP, RPE, RPO: Conditional Return

Restart Instruction:

The 8085 instruction set includes eight RST (Restart instructions). These are 1 byte call instructions and transfer the program execution to a specific location as listed in table below:

| Restart Instructions | Hex. Code | Call Location in Hex |
|---|---|---|
| RST0 | C7 | 0000H |
| RST1 | CF | 0008H |
| RST2 | D7 | 0010H |
| RST3 | DF | 0018H |
| RST4 | E7 | 0020H |
| RST5 | EF | 0028H |
| RST6 | F7 | 0030H |
| RST7 | FF | 0038H |

Actually these restart instructions are inserted through additional hardware and for this purpose the INTA signal of 8085A is used. These instructions are part of interrupt process.

**Miscellaneous group instruction:**

All other instructions in the instruction set fall on this group. These instructions include stack operation instructions, input/output operations and interrupt operations of 8085 microprocessor. The different instructions are.

PUSH, POP, DI, EI, HLT, etc….

- ☐ List the steps performed by the 8085 microprocessor, and identify the contents of buses when an instruction is being executed.
- ☐ Analyze a memory interfacing circuit, and specify the memory addresses of a given memory device.
- ☐ Recognize partial decoding and identify foldback (mirror) memory space.

# 4.1 THE 8085 MPU

The term **microprocessing unit** (MPU) is similar to the term **central processing unit** (CPU) used in traditional computers. We define the MPU as a device or a group of devices (as a unit) that can communicate with peripherals, provide timing signals, direct data flow, and perform computing tasks as specified by the instructions in memory. The unit will have the necessary lines for the address bus, the data bus, and the control signals, and would require only a power supply and a crystal (or equivalent frequency-determining components) to be completely functional.

Using this description, the 8085 microprocessor can almost qualify as an MPU, but with the following two limitations.

1. The low-order address bus of the 8085 microprocessor is **multiplexed** (time-shared) with the data bus. The buses need to be demultiplexed.
2. Appropriate control signals need to be generated to interface memory and I/O with the 8085. (Intel has some specialized memory and I/O devices that do not require such control signals.)

This section shows how to demultiplex the bus and generate the control signals after describing the 8085 microprocessor and illustrates the bus timings.

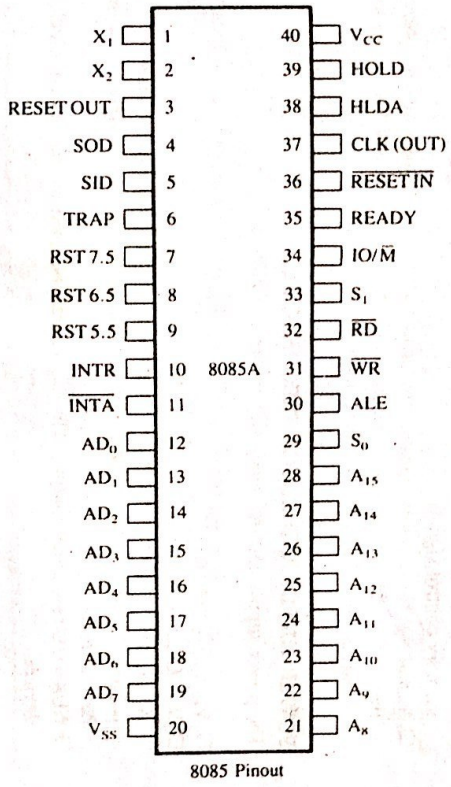
## 4.1.1 The 8085 Microprocessor

The 8085A (commonly known as the 8085) is an 8-bit general-purpose microprocessor capable of addressing 64K of memory. The device has forty pins, requires a +5 V single power supply, and can operate with a 3-MHz single-phase clock. The 8085A-2 version can operate at the maximum frequency of 5 MHz. The 8085 is an enhanced version of its predecessor, the 8080A; its instruction set is upward-compatible with that of the 8080A, meaning that the 8085 instruction set includes all the 8080A instructions plus some additional ones.

Figure 4.1 shows the logic pinout of the 8085 microprocessor. All the signals can be classified into six groups: (1) address bus, (2) data bus, (3) control and status signals, (4) power supply and frequency signals, (5) externally initiated signals, and (6) serial I/O ports.

### ADDRESS BUS

The 8085 has 16 signal lines (pins) that are used as the address bus; however, these lines are split into two segments: $A_{15}$–$A_8$ and $AD_7$–$AD_0$. The eight signal lines, $A_{15}$–$A_8$, are



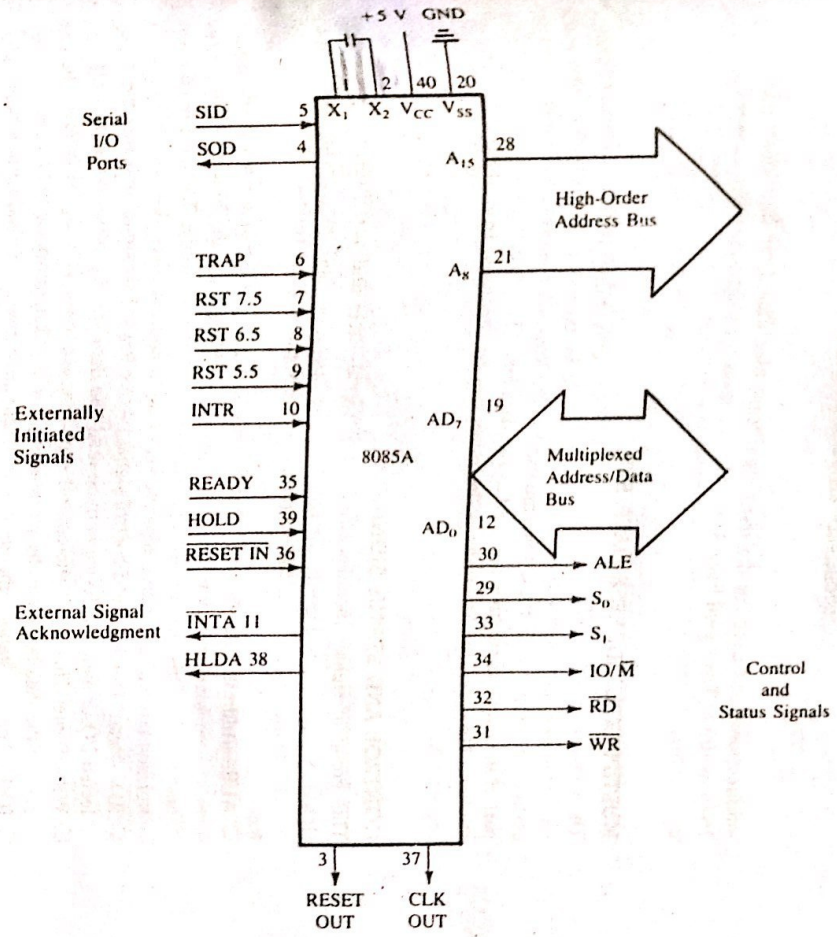




FIGURE 4.1
The 8085 Microprocessor Pinout and Signals

NOTE: The 8085A is commonly known as the 8085.
SOURCE (Pinout): Intel Corporation, *Embedded Microprocessors* (Santa Clara, Calif.: Author, 1994), pp. 1-11.

unidirectional and used for the most significant bits, called the high-order address, of a 16-bit address. The signal lines $AD_7$–$AD_0$ are used for a dual purpose, as explained in the next section.

### MULTIPLEXED ADDRESS/DATA BUS

The signal lines $AD_7$–$AD_0$ are bidirectional: they serve a dual purpose. They are used as the low-order address bus as well as the data bus. In executing an instruction, during the earlier part of the cycle, these lines are used as the low-order address bus. During the later part of the cycle, these lines are used as the data bus. (This is also known as multiplexing the bus.) However, the low-order address bus can be separated from these signals by using a latch.

### CONTROL AND STATUS SIGNALS

This group of signals includes two control signals ($\overline{RD}$ and $\overline{WR}$), three status signals ($IO/\overline{M}$, $S_1$, and $S_0$) to identify the nature of the operation, and one special signal (ALE) to indicate the beginning of the operation. These signals are as follows:

- ALE—Address Latch Enable: This is a positive going pulse generated every time the 8085 begins an operation (machine cycle); it indicates that the bits on $AD_7$–$AD_0$ are address bits. This signal is used primarily to latch the low-order address from the multiplexed bus and generate a separate set of eight address lines, $A_7$–$A_0$.
- $\overline{RD}$—Read: This is a Read control signal (active low). This signal indicates that the selected I/O or memory device is to be read and data are available on the data bus.
- $\overline{WR}$—Write: This is a Write control signal (active low). This signal indicates that the data on the data bus are to be written into a selected memory or I/O location.
- $IO/\overline{M}$: This is a status signal used to differentiate between I/O and memory operations. When it is high, it indicates an I/O operation; when it is low, it indicates a memory operation. This signal is combined with $\overline{RD}$ (Read) and $\overline{WR}$ (Write) to generate I/O and memory control signals.
- $S_1$ and $S_0$: These status signals, similar to $IO/\overline{M}$, can identify various operations, but they are rarely used in small systems. (All the operations and their associated status signals are listed in Table 4.1 for reference.)

### POWER SUPPLY AND CLOCK FREQUENCY

The power supply and frequency signals are as follows:

- $V_{CC}$: +5 V power supply.
- $V_{SS}$: Ground Reference.
- $X_1$, $X_2$: A crystal (or RC, LC network) is connected at these two pins. The frequency is internally divided by two; therefore, to operate a system at 3 MHz, the crystal should have a frequency of 6 MHz.
- CLK (OUT)—Clock Output: This signal can be used as the system clock for other devices.

TABLE 4.1
8085 Machine Cycle Status and Control Signals

| Machine Cycle | Status | | | Control Signals |
|---|---|---|---|---|
| | IO/$\overline{M}$ | $S_1$ | $S_0$ | |
| Opcode Fetch | 0 | 1 | 1 | $\overline{RD} = 0$ |
| Memory Read | 0 | 1 | 0 | $\overline{RD} = 0$ |
| Memory Write | 0 | 0 | 1 | $\overline{WR} = 0$ |
| I/O Read | 1 | 1 | 0 | $\overline{RD} = 0$ |
| I/O Write | 1 | 0 | 1 | $\overline{WR} = 0$ |
| Interrupt Acknowledge | 1 | 1 | 1 | $\overline{INTA} = 0$ |
| Halt | Z | 0 | 0 | $\overline{RD}$, $\overline{WR}$ = Z and $\overline{INTA} = 1$ |
| Hold | Z | X | X | |
| Reset | Z | X | X | |

NOTE: Z = Tri-state (high impedance)
X = Unspecified

## EXTERNALLY INITIATED SIGNALS, INCLUDING INTERRUPTS

The 8085 has five interrupt signals (see Table 4.2) that can be used to interrupt a program execution. One of the signals, INTR (Interrupt Request), is identical to the 8080A microprocessor interrupt signal (INT); the others are enhancements to the 8080A. The microprocessor acknowledges an interrupt request by the $\overline{INTA}$ (Interrupt Acknowledge) signal. (The interrupt process is discussed in Chapter 12.)

In addition to the interrupts, three pins—RESET, HOLD, and READY—accept the externally initiated signals as inputs. To respond to the HOLD request, the 8085 has one

TABLE 4.2
8085 Interrupts and Externally Initiated Signals

| | |
|---|---|
| ☐ INTR (Input) | Interrupt Request: This is used as a general-purpose interrupt; it is similar to the INT signal of the 8080A. |
| ☐ $\overline{INTA}$ (Output) | Interrupt Acknowledge: This is used to acknowledge an interrupt. |
| ☐ RST 7.5 (Inputs)<br>RST 6.5<br>RST 5.5 | Restart Interrupts: These are vectored interrupts that transfer the program control to specific memory locations. They have higher priorities than the INTR interrupt. Among these three, the priority order is 7.5, 6.5, and 5.5. |
| ☐ TRAP (Input) | This is a nonmaskable interrupt and has the highest priority. |
| ☐ HOLD (Input) | This signal indicates that a peripheral such as a DMA (Direct Memory Access) controller is requesting the use of the address and data buses. |
| ☐ HLDA (Output) | Hold Acknowledge: This signal acknowledges the HOLD request. |
| ☐ READY (Input) | This signal is used to delay the microprocessor Read or Write cycles until a slow-responding peripheral is ready to send or accept data. When this signal goes low, the microprocessor waits for an integral number of clock cycles until it goes high. |

signal called HLDA (Hold Acknowledge). The functions of these signals were previously discussed in Section 3.1.3. The RESET is again described below, and others are listed in Table 4.2 for reference.

- $\overline{\text{RESET IN}}$: When the signal on this pin goes low, the program counter is set to zero, the buses are tri-stated, and the MPU is reset.
- RESET OUT: This signal indicates that the MPU is being reset. The signal can be used to reset other devices.

SERIAL I/O PORTS

The 8085 has two signals to implement the serial transmission: SID (Serial Input Data) and SOD (Serial Output Data). In serial transmission, data bits are sent over a single line, one bit at a time, such as the transmission over telephone lines. This will be discussed in Chapter 16 on serial I/O.

In this chapter, we will focus on the first three groups of signals; others will be discussed in later chapters.

### 4.1.2 Microprocessor Communication and Bus Timings

To understand the functions of various signals of the 8085, we should examine the process of communication (reading from and writing into memory) between the microprocessor and memory and the timings of these signals in relation to the system clock. The first step in the communication process is reading from memory or fetching an instruction. This can be easily understood using an analogy of how a package is picked up from your house by a shipping company such as Federal Express. The steps are as follows:

1. A courier gets the address from the office; he or she drives the pickup van, finds the street, and looks for your house number.
2. The courier rings the bell.
3. Somebody in the house opens the door and gives the package to the courier, and the courier returns to the office with the package.
4. The internal office staff disposes the package according to the instructions given by the customer.

Now let us examine the steps in the following example of how the microprocessor fetches or gets a machine code from memory.

---

**Example 4.1** Refer to Example 3.5 in the last chapter (Section 3.2.6): Illustrate the steps and the timing of data flow when the instruction code 0100 1111 (4FH—MOV C,A), stored in location 2005H, is being fetched.

**Solution** To fetch the byte (4FH), the MPU needs to identify the memory location 2005H and enable the data flow from memory. This is called the Fetch cycle. The data flow is shown in Figure 4.2, and the timings are explained below.

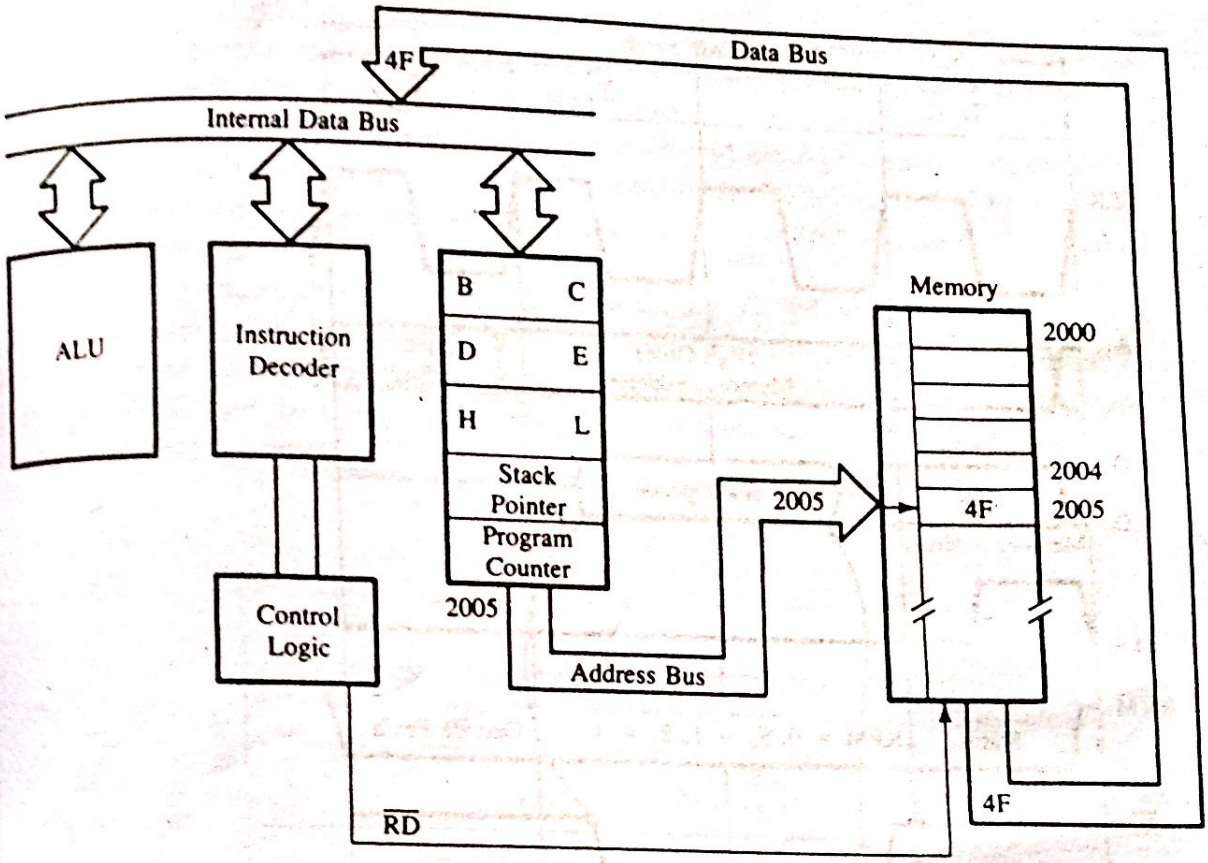



FIGURE 4.2
Data Flow from Memory to the MPU

Figure 4.3 shows the timing of how a data byte is transferred from memory to the MPU; it shows five different groups of signals in relation to the system clock. The address bus and data bus are shown as two parallel lines. This is a commonly used practice to represent logic levels of groups of lines; some lines are high and others are low. The crossover of the lines indicates that a new byte (information) is placed on the bus, and a dashed straight line indicates the high impedance state. To fetch the byte, the MPU performs the following steps:

**Step 1:** The microprocessor places the 16-bit memory address from the program counter (PC) on the address bus (Figure 4.2). In our analogy, this is the equivalent of our courier getting on the road to find the address.

Figure 4.3 shows that at $T_1$ the high-order memory address 20H is placed on the address lines $A_{15}$–$A_8$, the low-order memory address 05H is placed on the bus $AD_7$–$AD_0$, and the ALE signal goes high. Similarly, the status signal IO/$\overline{M}$ goes low, indicating that this is a memory-related operation. (For the sake of clarity, the other two status signals, $S_1$ and $S_0$, are not shown in Figure 4.3; they will be discussed in the next section.)

**Step 2:** The control unit sends the control signal $\overline{RD}$ to enable the memory chip (Figure 4.2). This is similar to ringing the doorbell in our analogy of a package pickup.

The control signal $\overline{RD}$ is sent out during the clock period $T_2$, thus enabling the memory chip (Figure 4.3). The $\overline{RD}$ signal is active during two clock periods.

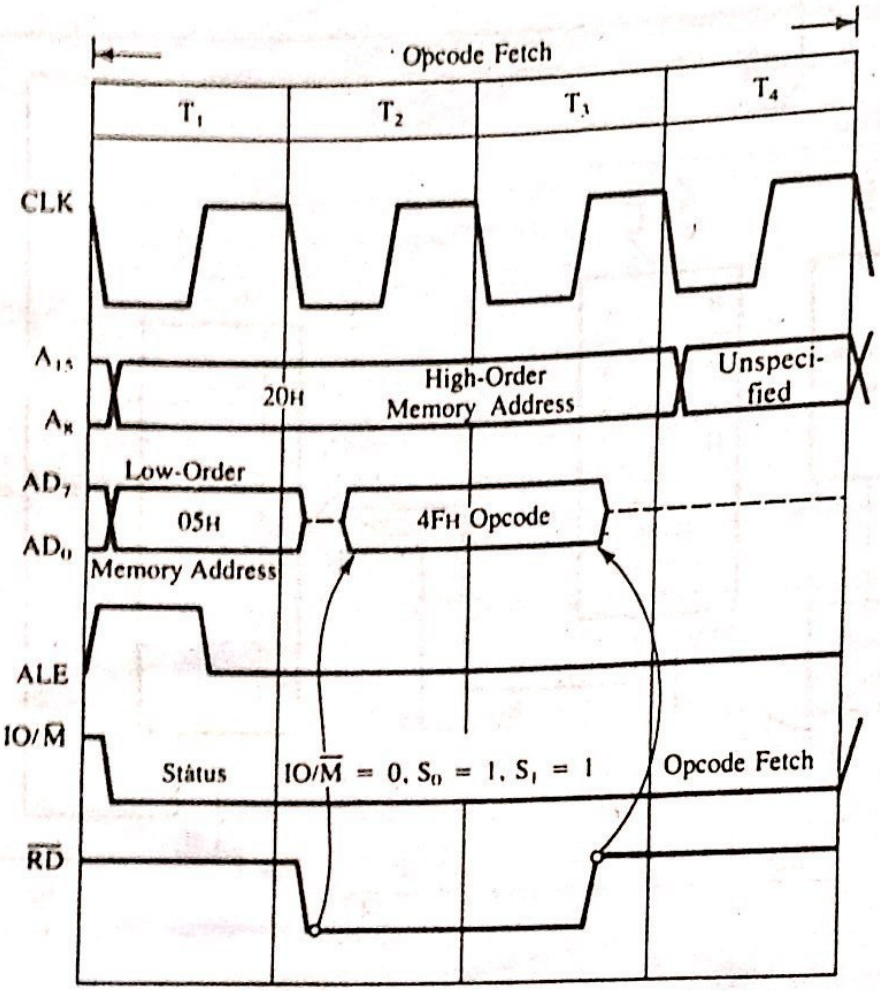



FIGURE 4.3
Timing: Transfer of Byte from Memory to MPU

**Step 3:** The byte from the memory location is placed on the data bus.

When the memory is enabled, the instruction byte (4FH) is placed on the bus $AD_7$–$AD_0$ and transferred to the microprocessor. The $\overline{RD}$ signal causes 4FH to be placed on bus $AD_7$–$AD_0$ (shown by the arrow), and when $\overline{RD}$ goes high, it causes the bus to go into high impedance.

**Step 4:** The byte is placed in the instruction decoder of the microprocessor, and the task is carried out according to the instruction.

The machine code or the byte (4FH) is decoded by the instruction decoder, and the contents of the accumulator are copied into register C. This task is performed during the period $T_4$ in Figure 4.3.

The above four steps are similar to the steps listed in our analogy of the package pickup.

### 4.1.3 Demultiplexing the Bus $AD_7$–$AD_0$

The need for demultiplexing the bus $AD_7$–$AD_0$ becomes easier to understand after examining Figure 4.3. This figure shows that the address on the high-order bus (20H) remains on the bus for three clock periods. However, the low-order address (05H) is lost after the