Microprocessor Architecture and Its Operations

- The microprocessor is a programmable digital device, designed with register, flip-flops, and timing elements
- The microprocessor has a set of instruction, designed internally, to manipulate data and communicate with peripherals
- This process of data manipulation and communication is determined by the logic design of the microprocessor, called the architecture.

- The microprocessor can be programmed to perform functions on given data by selecting necessary instructions from its set.
- These instructions are given to the microprocessor by writing them into its memory
- Writing/Entering instructions and data is done through an input device such as a keyboard.
- The microprocessor reads or transfers one instruction at a time, matches it with its instruction set, and performs the data manipulation indicated by the instruction.

- The result can be stored in memory or sent to such output devices as LEDs and CRT terminal.
- In addition, the microprocessor can respond to external signals.
- It can be interrupted, reset, or asked to wait to synchronize with slower peripherals.
- All the various functions performed by the microprocessor can be classified in three general categories:

1. microprocessor-initiated operations
2. Internal operations
3. Peripheral or externally initiated operations

- To perform these functions, the microprocessor requires a group of logic circuits and a set of signals called control signals.
- However, early processors did not have the necessary circuitry on one chip; the complete units were made up of more than one chip.
- Therefore, the term microprocessing unit (MPU) is defined here as a group of devices that can perform these functions with the necessary set of control signals.
- This term is similar to the term central processing unit (CPU).
- However, later microprocessors include most of the necessary circuitry to perform these operations on a single chip.
- Therefore, the terms MPU and microprocessor often are used synonymously.

The MPU performs primarily four operations:

- 1. Memory Read: Reads data (or instructions) from memory.
- Memory Write: Writes data (or instructions) into memory.
- I/0 Read: Accepts data from input devices.
- I/O Write: Sends data to output devices.

All these operations are part of the communication process between the MPU and peripheral devices (including memory). To communicate with a peripheral (or a memory location), the MPU needs to perform the following steps:

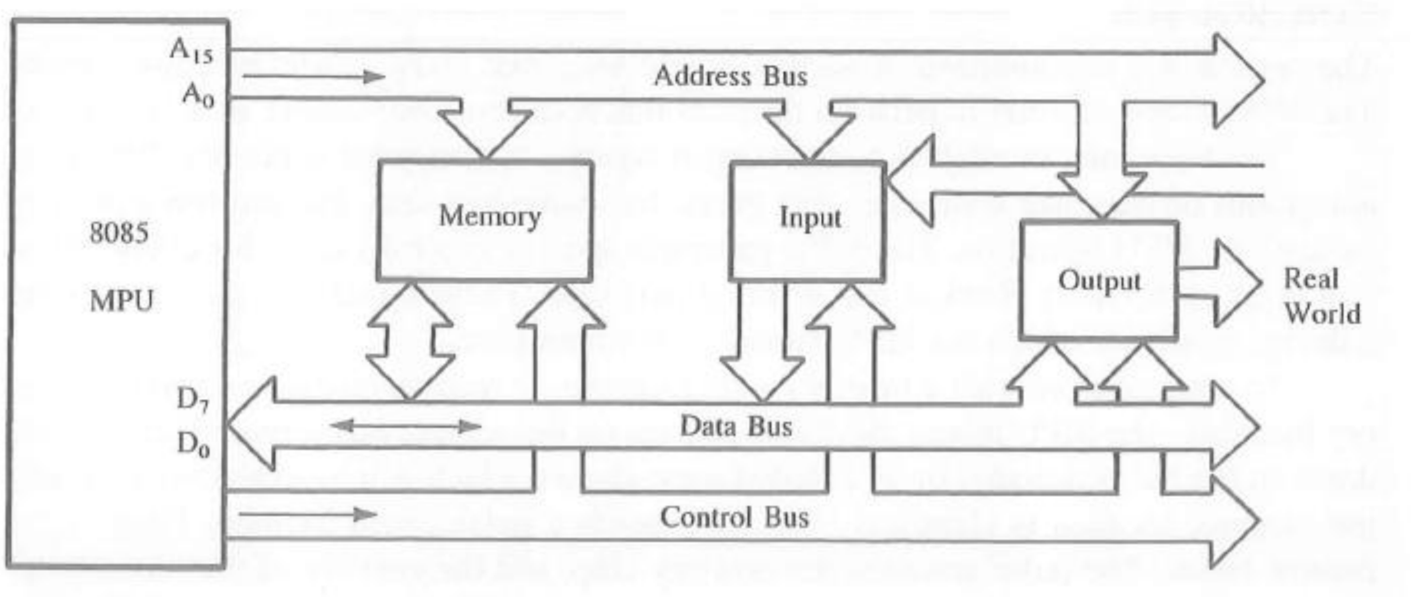
Step 1: Identify the peripheral or the memory location (with its address).

Step 2: Transfer binary information (data and instructions).

Step 3: Provide timing or synchronization signals.

The 8085 MPU performs these functions using three sets of communication lines called buses: the address bus, the data bus, and the control bus.

Address Bus

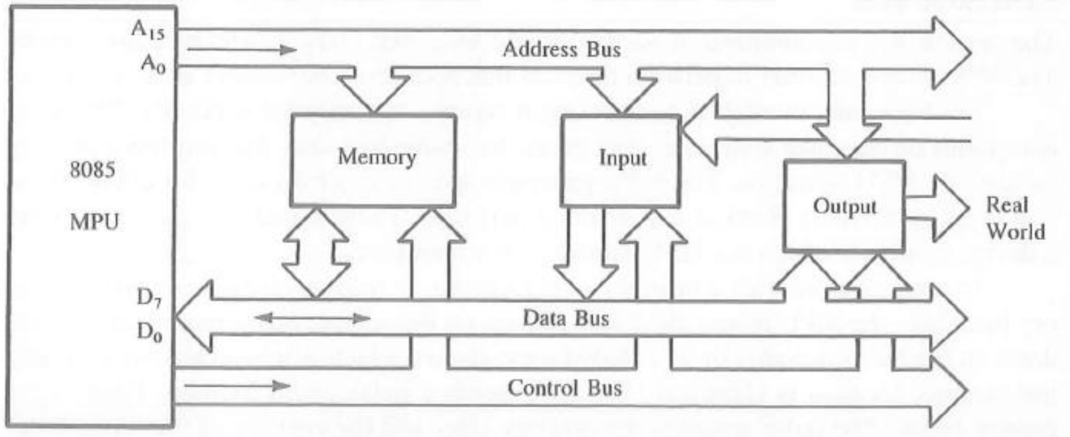


The address bus is a group of 16 lines generally identified as A0 to A15. The address bus is unidirectional: bits flow in one direction—from the MPU to peripheral devices. The MPU uses the address bus to perform the first function: identifying a peripheral or a memory location (Step I). In a computer system, each peripheral or memory location is identified by a binary number, called an address, and the address bus is used to carry a 16-bit address.

This is similar to the postal address of a house. A house can be identified by various number schemes. For example, the forty-fifth house in a lane can be identified by the two-digit number 45 or by the four-digit number 0045. The two-digit numbering scheme can identify only a hundred houses, from 00 to 99. On the other hand, the four-digit scheme can identify ten thousand houses, from 0000 to 9999. Similarly, the number of address lines of the MPU determines its capacity to identify different memory locations (or peripherals). The 8085 MPU with its 16 address lines is capable of addressing 2^16 = 65,536 (generally known as 64K) memory locations. As explained in Chapter 1, 1K memory is determined by rounding off 1024 to the nearest thousand; similarly, 65,536 is rounded off to 64,000 as a multiple of 1K.

Most 8-bit microprocessors have 16 address lines. This may explain why microcomputer systems based on 8-bit microprocessors have 64K memory. However, not every microcomputer system has 64K memory. In fact, most single-board microcomputers have less than 4K of memory, even if the MPU is capable of addressing 64K memory. The number of address lines is arbitrary; it is determined by the designer of a microprocessor based on such considerations as availability of pins and intended applications of the processor. For example, the Intel 8088 processor has 20 and the Pentium processor has 32 address lines.

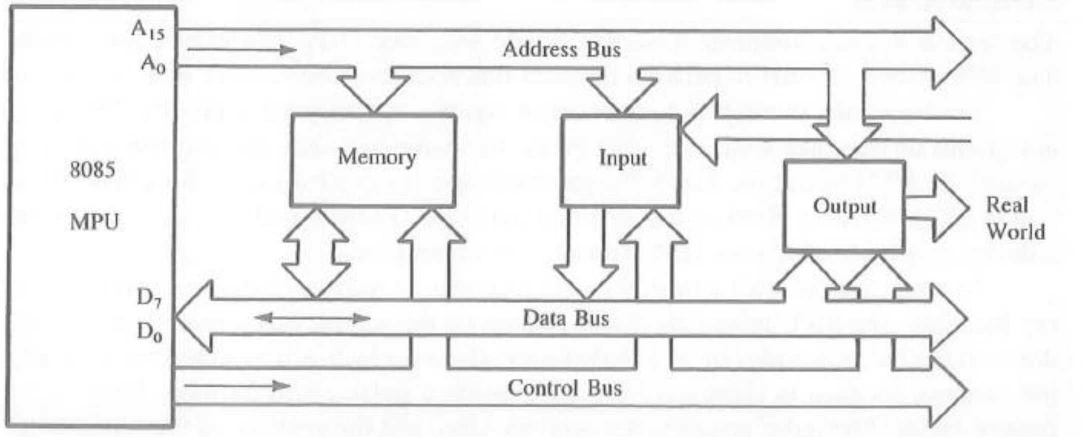
Data Bus



The data bus is a group of eight lines used for data flow These lines are bidirectional-data flow in both directions between the MPU and memory and peripheral devices. The MPU uses the data bus to perform the second function: transferring bi-nary information (Step 2).

The eight data lines enable the MPU to manipulate 8-bit data ranging from 00 to FF (2^8 = 256 numbers). The largest number that can appear on the data bus is 11111111 (255_10). The 8085 is known as an 8-bit microprocessor. Microprocessors such as the Intel 8086, Zilog Z8000, and Motorola 68000 have 16 data lines; thus they are known as 16-bit microprocessors. The Intel 80386/486 have 32 data lines; thus they are classified as 32-bit microprocessors.

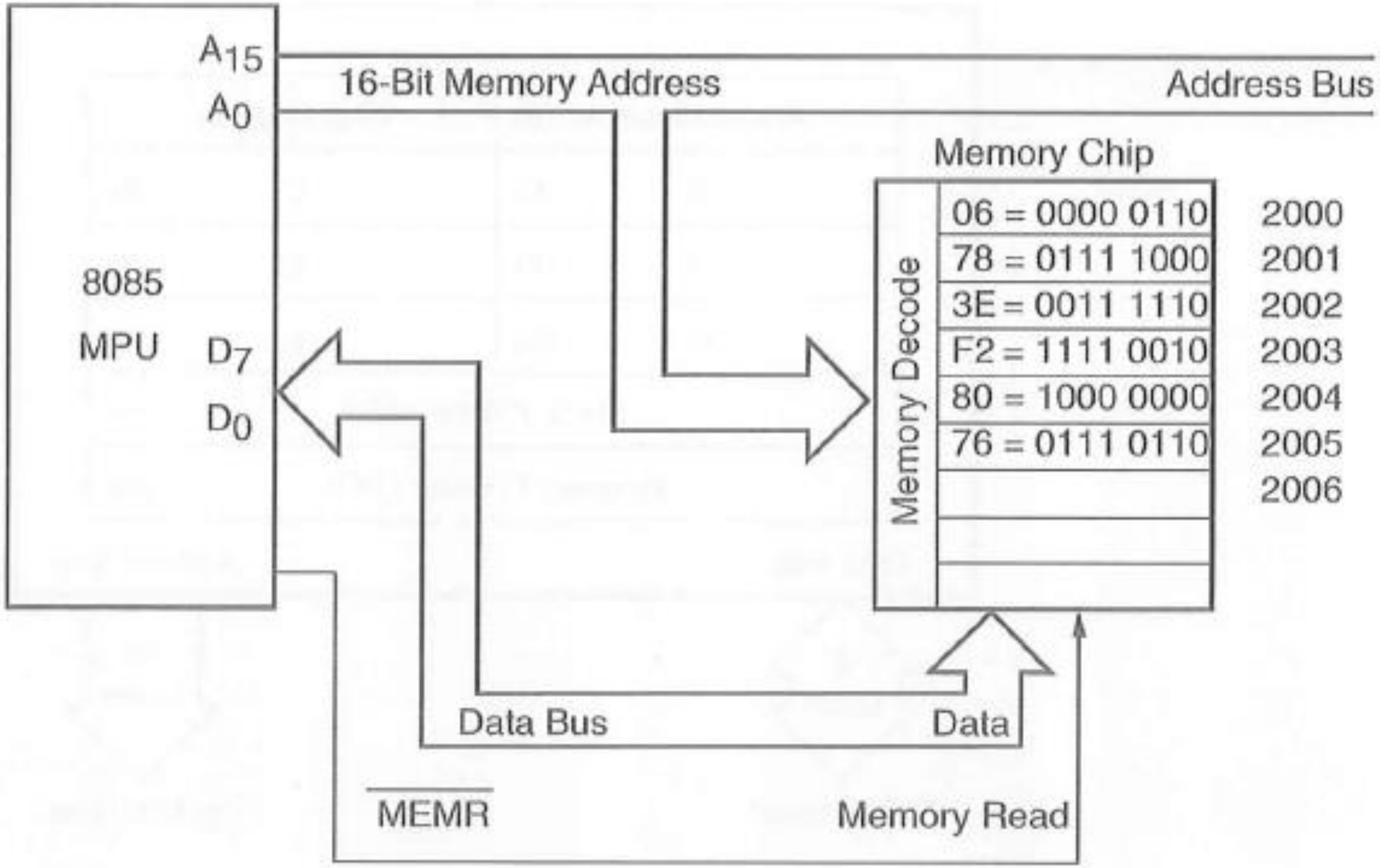
Control Bus



The control bus is comprised of various single lines that carry synchronization signals. The MPU uses such lines to perform the third function: providing timing signals (Step 3).

The term bus, in relation to the control signals, is somewhat confusing. These are not groups of lines like address or data buses, but individual lines that provide a pulse to indicate an MPU operation. The MPU generates specific control signals for every operation (such as Memory Read or I/O Write) it performs. These signals are used to identify a device type with which the MPU intends to communicate.

To communicate with a memory-for example, to read an instruction from a memory location-the MPU places the 16-bit address on the address bus. The ad-dress on the bus is decoded by an external logic circuit, which will be explained later, and the memory location is identified. The MPU sends a pulse called Memory Read as the control signal. The pulse activates the memory chip, and the contents of the memory lo-cation (8-bit data) are placed on the data bus and brought inside the microprocessor.

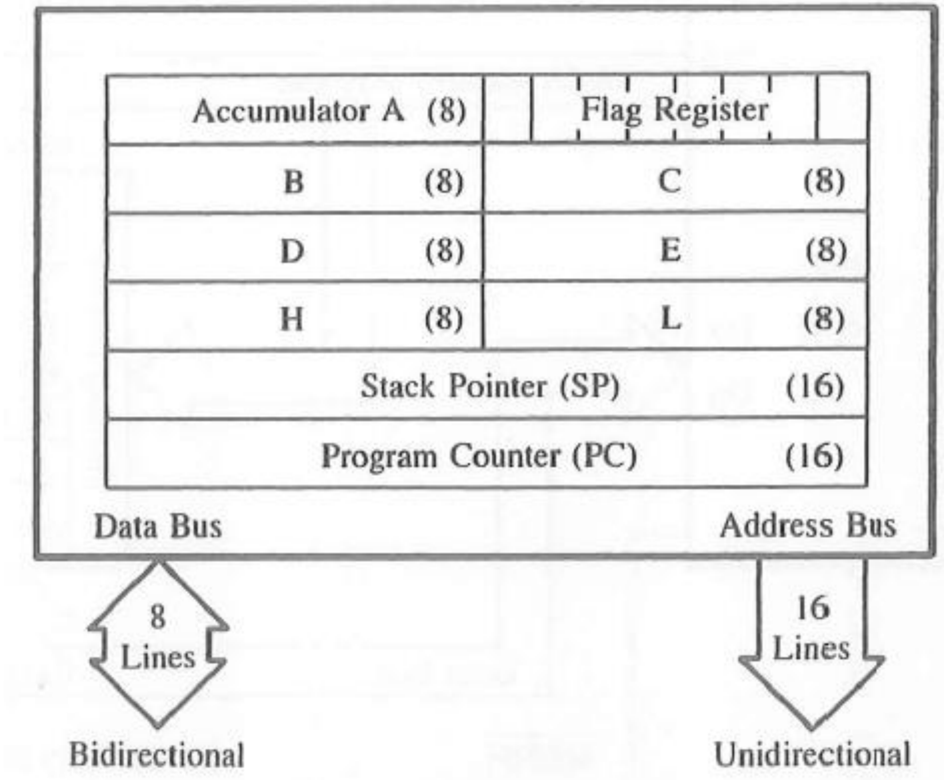

A15
A0
16-Bit Memory Address
Address Bus
Memory Chip
8085
MPU
D7
D0
Memory Decode
06 = 0000 0110
78 = 0111 1000
3E = 0011 1110
F2 = 1111 0010
80 = 1000 0000
76 = 0111 0110
2000
2001
2002
2003
2004
2005
2006
Data Bus
Data
MEMR
Memory Read

Internal data operations and the 8085 registers

The internal architecture of the 8085 microprocessor determines how and what operations can be performed with the data. These operations are:

1. Store 8-bit data.
2. Perform arithmetic and logical operations.
3. Test for conditions.
4. Sequence the execution of instructions.
5. Store data temporarily during execution in the defined R/W memory locations called the stack.

To perform these operations, the microprocessor requires registers, an arithmetic/ logic unit (ALU) and control logic, and internal buses (paths for information flow). Figure; it is repeated here (for reference) shows the programming model of the 8085 displaying the internal registers and the accumulator. The functions of these registers are described in reference to the five operations when the processor executes the following three instructions. The Hex codes of these instructions are stored in memory locations from 2000 to 2005H as shown in earlier figure (slide no 12) .



Accumulator A (8)	Flag Register
B (8)	C (8)
D (8)	E (8)
H (8)	L (8)
Stack Pointer (SP) (16)	
Program Counter (PC) (16)	

```
2000    06   MVI B, 76H (load)
2001    78
2002    3E   MVI A, F2H
2003    F2
2004    80    ADD B
2005    76    HLT  (end)
```

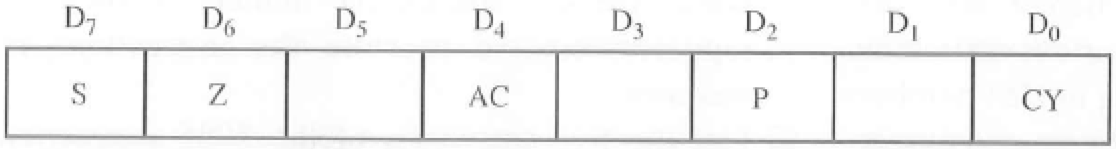
REGISTERS

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L, as shown in Figure (slide no 14). They can be combined as register pairs- BC, DE, and HL-to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions. H and L can be used as a data pointer (hold memory address)

ACCUMULATOR

The accumulator is an 8-bit register that is part of the arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. Store 8 bit data during I/O transfer The accumulator is also identified as register A.

FLAGS



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

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags; they are listed (next slide) and their bit positions in the flag register are shown in Figure. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.

These flags have critical importance in the decision-making process of the micro-processor. The conditions (set or reset) of the flags are tested through software instructions. For example, the instruction JC (Jump On Carry) is implemented to change the sequence of a program when the CY flag is set. So, thorough understanding of flags is essential in writing assembly language programs.

8085 Flags

The following flags are set or reset after the execution of an arithmetic or logic operation; data copy instructions do not affect any flags.

- Z-Zero: The Zero flag is set to 1 when the result is zero; otherwise it is reset.

$$\begin{array}{r} 10110011 \\ \underline{+\ \ 01001101} \\ 1\ \ 00000000 \end{array}$$

- CY-Carry: If an arithmetic operation results in a carry, the CY flag is set; otherwise it is reset.

$$\begin{array}{lrlr} & 10110101 & & 10110101 \\ & \underline{+\ \ 01101100} & & \underline{-\ \ 11001100} \\ \text{Carry } 1 & 00100001 & \text{Borrow } \ 1 & 11101001 \end{array}$$

- S-Sign: used for indicating the sign of the data in the accumulator. The sign flag is set if negative (1 –negative) and reset if positive (0-positive). i.e the Sign flag is set if bit D_7 of the result = 1; otherwise it is reset.
- P-Parity: If the result has an even number of 1s, the flag is set; for an odd number of 1s, the flag is reset.
- AC-Auxiliary Carry: is set if there is a carry out of bit 3. In an arithmetic operation, when a carry is generated by digit D_3 and passed to digit D_4, the AC flag is set. This flag is used internally for BCD (binary-coded decimal) operations; there is no Jump instruction associated with this flag.

Program Counter (PC) and Stack Pointer (SP)

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

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. This register always holds the address of the next instruction. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location. Since it holds an address, it must be 16 bits wide.

The stack pointer is also a 16-bit register used to point into memory as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading a 16-bit address in the stack pointer. The stack is usually accessed in a Last In First Out (LIFO).

When the user enters the memory address 2000H and pushes the execute key of the trainer, the processor places the address 2000H in the program counter (PC).

- The program counter is a 16-bit register that performs the fourth operation in the list: sequencing the execution of the instructions. When the processor begins execution, it places the address 2000H on the address bus and increments the address in the PC to 2001 for the next operation. It brings the code 06, interprets the code, places the address 2001H on the address bus, and then gets byte 78H and increments the address in PC to 2002H. The processor repeats the same process for the next instruction, MVI A, F2H. (load)
- When the processor executes the first two instructions, it uses register B to store 78H and A to store F2H in binary (Operation 1).

❑When the processor executes the instruction ADD B in the ALU (Operation 2), it adds 78H to F2H, resulting in the sum 16AH (78H + F2H = 16AH). It replaces F2H by 6AH in A and sets the Carry flag as described next.

❑In our example, the addition operation generates a carry because the sum is larger than the size of the accumulator (8 bits). To indicate the carry, the processor sets the flip-flop called Carry (CY flag) to 1 and places logic 1 in the flag register at the designated bit position for the carry.

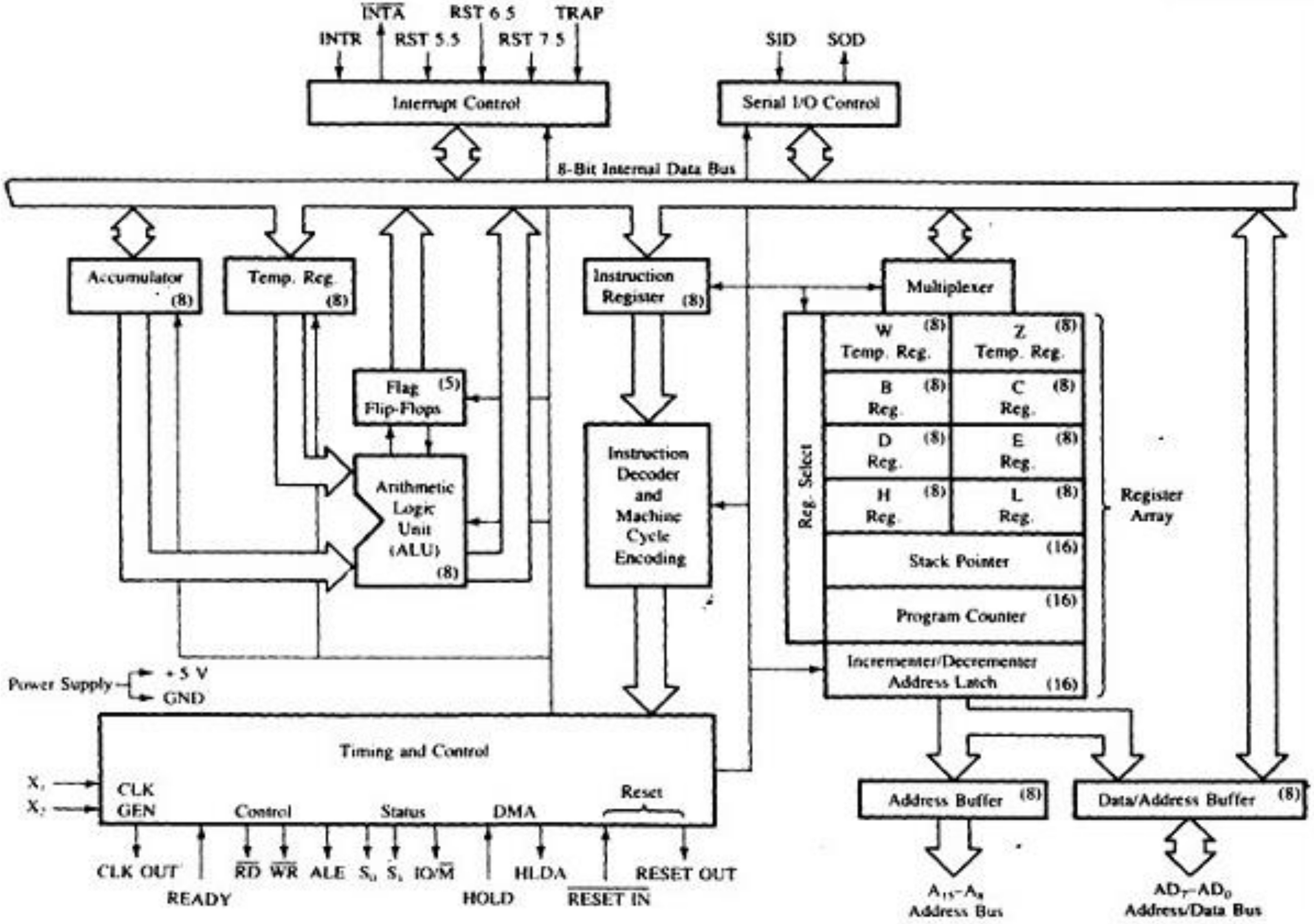
❑The fifth operation deals with the concept of the stack. The stack pointer is a 16-bit register used as a memory pointer to identify the stack, part of the R/W memory defined and used by the processor for temporary storage of data during the execution.

Peripheral or Externally initiated operations

External device (or signals) can initiate the following operations, for which individual pins on the microprocessor chip are assigned: Reset, Interrupt, Ready, Hold

- Reset: When the reset pin is activated by an external key (also called a reset key), all internal operations are suspended and the program counter is cleared (it holds 0000H). Now the program execution can again begin at the zero memory address.
- Interrupt: The microprocessor can be interrupted from the normal execution of instructions and asked to execute some other instructions called a service routine (for example, emergency procedures). The microprocessor resumes its operation after completing the service routine.

- Ready: The 8085 has a pin called READY. If the signal at this READY pin is low, the microprocessor enters into a Wait state. This signal is used primarily to synchronize slower peripherals with the microprocessor.
- Hold: When the HOLD pin is activated by an external signal, the microprocessor relinquishes (abandons/give up) control of buses and allows the external peripheral to use them. For example, the HOLD signal is used in Direct Memory Access (DMA) data transfer.


INTA
RST 6.5
TRAP
INTR
RST 5.5
RST 7.5
SID
SOD
Interrupt Control
Serial I/O Control
8-Bit Internal Data Bus
Accumulator (8)
Temp. Reg. (8)
Instruction Register (8)
Multiplexer
Flag (5) Flip-Flops
Arithmetic Logic Unit (ALU) (8)
Instruction Decoder and Machine Cycle Encoding
Reg. Select
W (8) Temp. Reg.
Z (8) Temp. Reg.
B (8) Reg.
C (8) Reg.
D (8) Reg.
E (8) Reg.
H (8) Reg.
L (8) Reg.
Stack Pointer (16)
Program Counter (16)
Incrementer/Decrementer Address Latch (16)
Register Array
Power Supply
+5 V
GND
Timing and Control
X₁
X₂
CLK GEN
Control
Status
DMA
Reset
CLK OUT
READY
RD
WR
ALE
S₀
S₁
IO/M
HOLD
HLDA
RESET IN
RESET OUT
Address Buffer (8)
Data/Address Buffer (8)
A₁₅–A₈ Address Bus
AD₇–AD₀ Address/Data Bus

Memory

Memory

Memory is an essential component of a microcomputer system; it stores binary instructions and data for the microprocessor. There are various types of memory, which can be classified in two groups: prime (or main) memory and storage memory. examples of prime memory: Read/Write memory (R/WM) and Read-Only memory (ROM). Magnetic tapes or disks can be cited as examples of storage memory.

The R/W memory is made of registers, and each register has a group of flip-flops or field-effect transistors that store bits of information; these flip-flops are called memory cells.

The number of bits stored in a register is called a memory word; memory devices (chips) are available in various word sizes. The user can use this memory to hold pro-grams and store data. On the other hand, the ROM stores information permanently in the form of diodes; the group of diodes can be viewed as a register. In a memory chip, all registers are arranged in a sequence and identified by binary numbers called memory addresses. To communicate with memory, the MPU should be able to

- Select the chip,
- Identify the register, and
- Read from or write into the register.

The MPU uses its address bus to send the address of a memory register and uses the data bus and control lines to read from (as shown slide no 12 figure) or write into that register.

Memory stores information such as instructions and data in binary format (0 and 1). It provides this information to the microprocessor whenever it is needed.

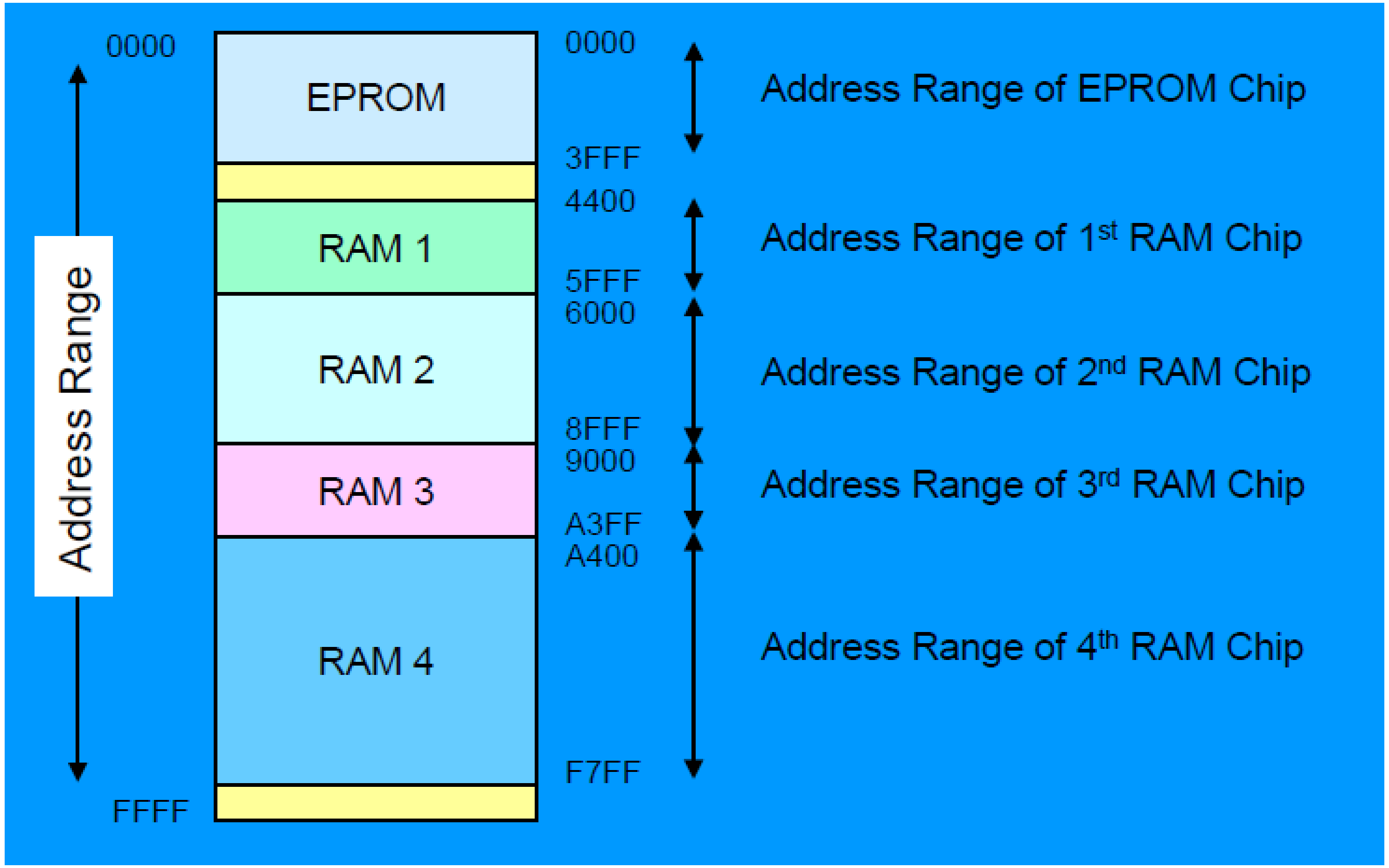
Usually, there is a memory “sub-system” in a microprocessor-based system. This sub-system includes:

- The registers inside the microprocessor
- Read Only Memory (ROM)
    - used to store information that does not change.
- Random Access Memory (RAM) (also known as Read/Write Memory).
    - used to store information supplied by the user. Such as programs and data.

Memory Mapping

- Memory Mapping 8085 has 16-bit Address Bus
- The complete address space is thus given by the range of addresses 0000H – FFFFH
- The range of addresses allocated to a memory device is known as its memory map

The memory map is a picture representation of the address range and shows where the different memory chips are located within the address range.



0000
Address Range
FFFF
EPROM
RAM 1
RAM 2
RAM 3
RAM 4
0000
3FFF
4400
5FFF
6000
8FFF
9000
A3FF
A400
F7FF
Address Range of EPROM Chip
Address Range of 1st RAM Chip
Address Range of 2nd RAM Chip
Address Range of 3rd RAM Chip
Address Range of 4th RAM Chip

To execute a program:

The user enters its instructions in binary format into the memory.

The microprocessor then reads these instructions and whatever data is needed from memory, executes the instructions and places the results either in memory or produces it on an output device.

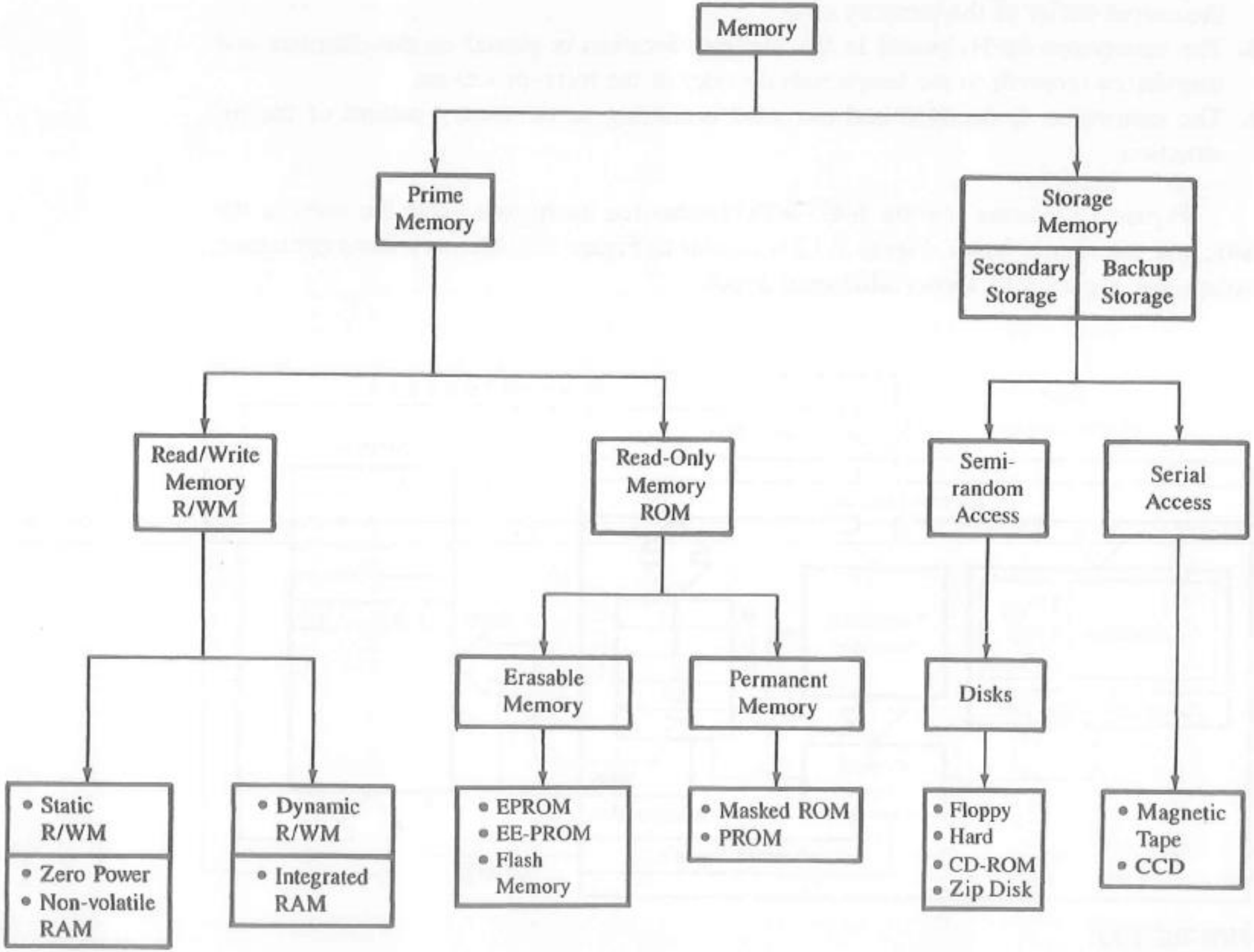
The three cycle instruction execution model

To execute a program, the microprocessor “reads” each instruction from memory, “interprets” it, then “executes” it.

To use the right names for the cycles:

- The microprocessor fetches each instruction,
- Decodes it,
- Then executes it.

This sequence is continued until all instructions are performed.


Memory
Prime Memory
Storage Memory
Secondary Storage
Backup Storage
Read/Write Memory R/WM
Read-Only Memory ROM
Semi-random Access
Serial Access
Erasable Memory
Permanent Memory
Disks
• Static R/WM
• Zero Power
• Non-volatile RAM
• Dynamic R/WM
• Integrated RAM
• EPROM
• EE-PROM
• Flash Memory
• Masked ROM
• PROM
• Floppy
• Hard
• CD-ROM
• Zip Disk
• Magnetic Tape
• CCD

INPUT AND OUTPUT (I/O) DEVICES

Input/output devices are the means through which the MPU communicates with "the out-side world," The MPU accepts binary data as input from devices such as keyboards and A/D converters and sends data to output devices such as LEDs or printers. There are two different methods by which I/O devices can be identified:

- one uses an 8-bit address
- uses a 16-bit address

I/Os with 8-Bit Addresses (Peripheral-Mapped I/O)

In this type of I/O, the MPU uses eight address lines to identify an input or an output device; this is known as peripheral-mapped I/O (also known as I/O-mapped I/O).

This is an 8-bit numbering system for I/Os used in combination with Input and Output instructions.

This is also known as I/O space, separate from memory space, which is a 16-bit numbering system.

The eight address lines can have 256 (2^8 combinations) addresses; thus, the MPU can identify 256 input devices and 256 output devices with addresses ranging from 00H to FFH.

The input and output devices are differentiated by the control signals; the MPU uses the I/O Read control signal for input devices and the I/O Write control signal for output devices.

The entire range of I/O addresses from 00 to FF is known as an I/O map, and individual addresses are referred to as I/O device addresses or I/O port numbers.

If we use LEDs as output or switches as input, we need to resolve two issues: how to assign addresses and how to connect these I/0 devices to the data bus.

In a bus architecture, these devices cannot be connected directly to the data bus or the address bus; all connections must be made through tri-state interfacing devices so they will be enabled and connected to the buses only when the MPU chooses to communicate with them.

In the case of memory, we did not have to be concerned with these problems because of the internal address decoding, Read/Write buffers, and availability of CS and control signals of the memory chip.

The steps in communicating with an I/O device are similar to those in communicating with memory and can be summarized as follows:

1. The MPU places an 8-bit address on the address bus, which is decoded by external de-code logic (explained in Chapter 4).
2. The MPU sends a control signal (I/O Read or I/O Write) and enables the I/O device.
3. Data are transferred using the data bus.

I/Os with 16-Bit Addresses (Memory-Mapped I/O)

In this type of I/O, the MPU uses 16 address lines to identify an I/O device; an I/O is connected as if it is a memory register.

This is known as memory-mapped I/O.

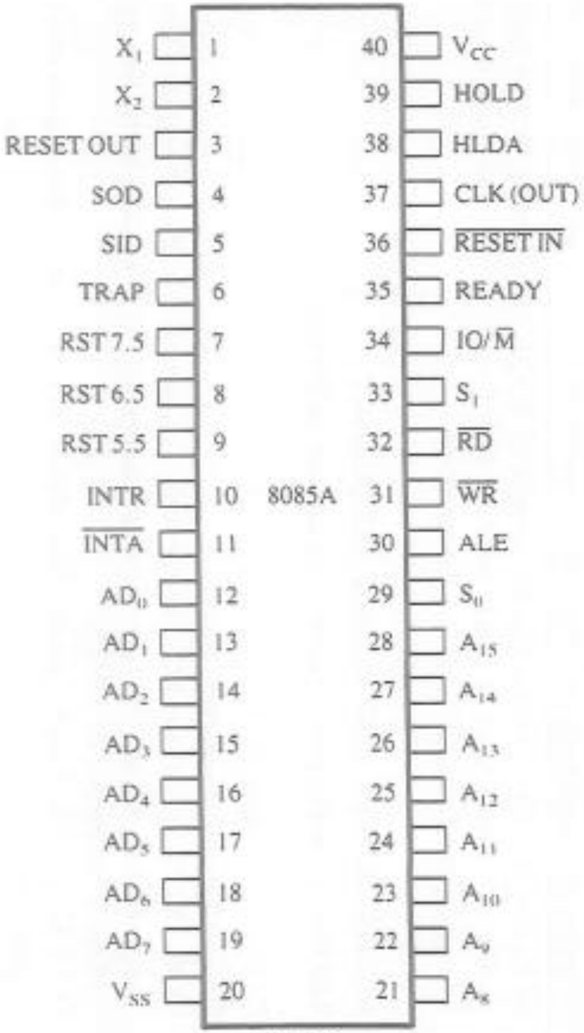
The MPU uses the same control signal (Memory Read or Memory Write) and instructions as those of memory.

In some microprocessors, such as the Motorola 6800, all I/Os have 16-bit ad-dresses; I/Os and memory share the same memory map (64K).

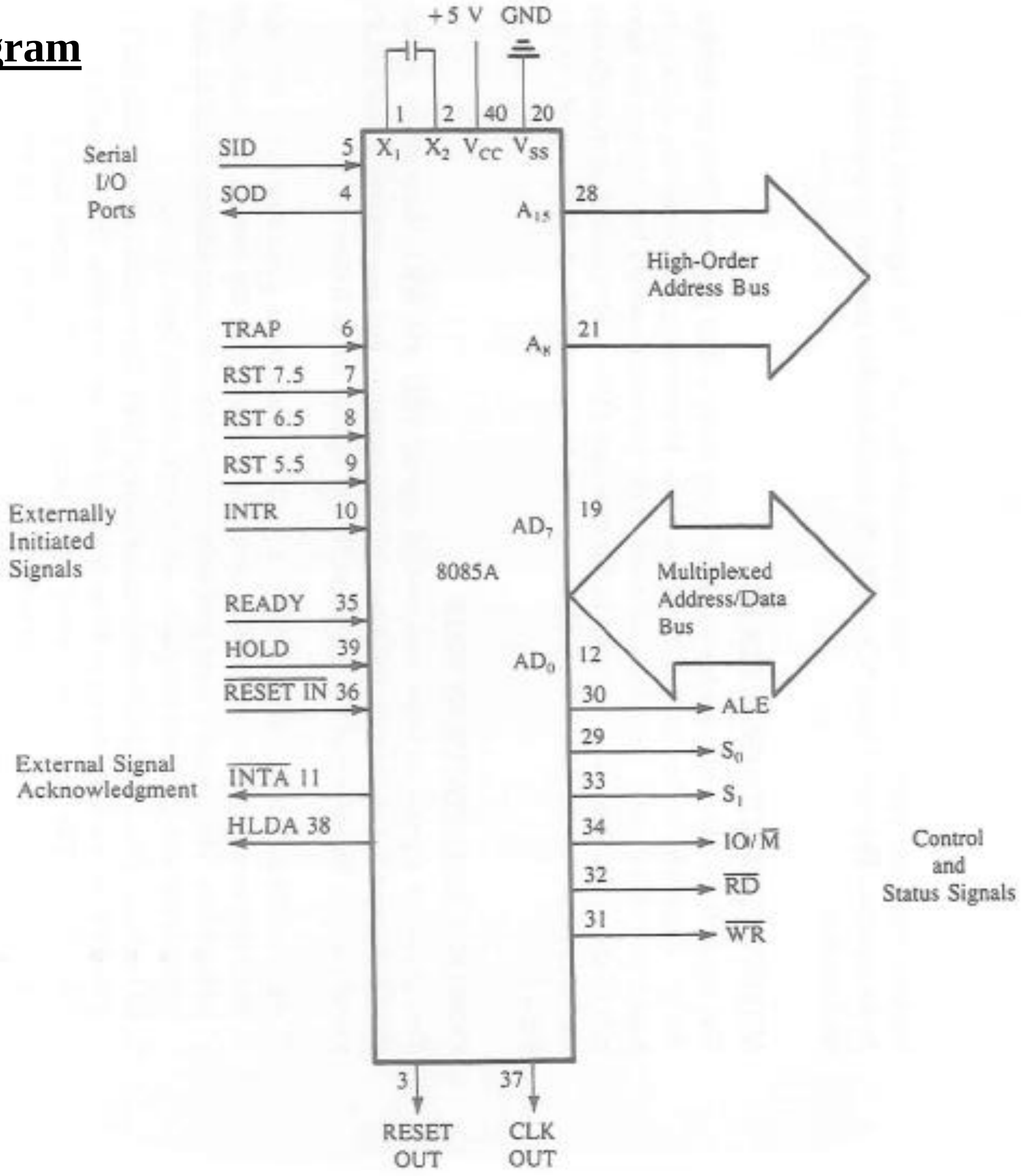
In memory-mapped I/O, the MPU follows the same steps as if it is accessing a memory register.

8085 Microprocessor Pin Description

8085 Microprocessor Pin and Signals Diagram



8085 Pinout



The signals of this 40 pin IC is grouped into 7 categories, which are given below:

1. Power supply and clock signals
2. Data bus
3. Address bus
4. Serial I/O ports
5. Control and status signals
6. Interrupts and externally generated signals
7. Direct memory access

These are the categories among which the 40 pin configuration of 8085 is divided.

Power supply and clock signals

In 40 pin configuration, 4 pins are allotted to this particular category.

- **VCC** – Pin number 40 denotes VCC, and an external power supply of + 5 V is provided at this pin.
- **VSS** – Its pin number is 20. This pin shows the grounded connection of the microprocessor.
- **X1** and **X2** – These are represented by pin number 1 and 2 respectively in the pin configuration. These 2 pins are connected with a crystal or LC network *(LC circuits are used either for generating signals at a particular frequency, or picking out a signal at a particular frequency from a more complex signal)* to maintain the internal frequency of the clock generator.
- **CLK** (OUT) – It is the 37th pin of the 8085 IC and acts as the system clock that keeps the record of time duration required by each operation to get completed.

Address Bus

This category contains 8 pins.

- The address bus has 16 lines i.e.; it can carry 16 bits at a time. However, out of 16, 8 are multiplexed with the data bus and the leftover 8 are separately shown by pin number 21 to 28 in the pin configuration.
- These are used to carry the address of data and instruction from the processor to the memory location and is unidirectional in nature. These are denoted by $\boldsymbol{A_8}$ ***to*** $\boldsymbol{A_{15}}$ that represents the 8 MSB of the memory location or input-output address.

Data Bus with multiplexed address bus

This category also contains 8 pins.

- The size of the data bus of the 8085 microprocessor is 8 bits. However, to reduce the number of bus lines these 8-bit data bus lines are multiplexed with the 8-bit address bus.
- These are shown by pin number 12 to 19. The address bus is denoted by A whereas the data bus is denoted by D. The pin configuration denotes the lower order multiplexed address and data bus bits from ***AD_0 to AD_7.***
- We know that the address bus contains the address of the desired memory location from where the data or instruction is to be fetched. While the data bus contains the data or instruction that is needed to be fetched from the memory.

Serial I/O ports

It has basically 2 pins.

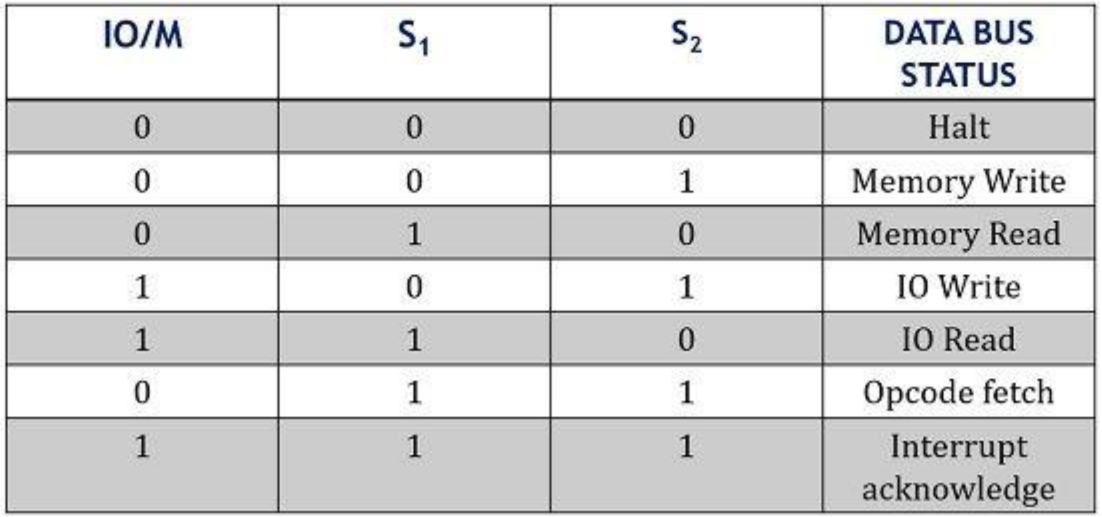
- **SID** – SID denotes serial input data pin and its pin is numbered as 5. With this pin, data is serially fed to the processor directly through the input devices.
- **SOD** – SOD denotes serial output data pin and its pin number is 4, in the pin configuration of 8085. Once the data is processed in the microprocessor then this pin represents bit by bit results at the output devices.

Control and status signals

Basically, 6 pins of the pin configuration are used by control and status signals.

- **ALE** – ALE is an acronym for address latch enable and is pin number 30 in the configuration. We know that 8 lower order bits of the 16-bit address bus are multiplexed with the 8-bit data bus. This pin gets enabled at the time when the address is present at the multiplexed address and data bus. Otherwise, it gets disabled showing the absence of an address on the bus.
- **RD** – This pin is numbered 32 in the configuration and a low signal in this pin shows the read operation either from I/O devices or from the memory unit. Thereby indicating that the data bus is now in a state or position to accept the data from the memory or I/O devices.
- **WR** – It is the 31st pin in the pin diagram and a low signal in this pin represents the write operation at the memory or I/O devices. This indicates that the data present in the data bus is to be written into the desired memory address or I/O device by the processor.
- **IO/M** – It is pin number 34 and indicates the selection of a memory address or input-output device. This shows whether the read/write operation is to be carried out at the memory location or at the I/O device. The low signal at this pin shows that operation is performing over memory location. As against, a high signal at this pin represents the operation at I/O device.

- **S_0 and S_1** – The pins S_0 and S_1 represent the status signal at pin number 29 and 33 respectively. These signals show the type of recent operation of the microprocessor. The table below represents the status of the data bus under different conditions:



IO/M	S_1	S_2	DATA BUS STATUS
0	0	0	Halt
0	0	1	Memory Write
0	1	0	Memory Read
1	0	1	IO Write
1	1	0	IO Read
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

Interrupts and Externally generated signals

Interrupts are the signals that are generated to break the sequence of an ongoing operation. When an interrupt signal is generated then CPU immediately stops its recent task under operation and switches to some other program known as interrupt service routine (ISR).

However, after handling ISR, the CPU gets back to its main program for execution.

Interrupts are the signals generated by external devices to request the microprocessor to perform a task. In the pin configuration, 5 types of interrupts are shown by 5 different pins from pin number 6 to 10. These pins are used to manage the interrupt.

Basically, there exist 2 types of interrupts:

Maskable Interrupt and **Non- maskable interrupt**

- Out of the 5 major interrupts 4 are the maskable interrupts. These are INTR, RST5.5, RST6.5, RST7.5 and are easily manageable interrupts.
- However, TRAP is a non-maskable interrupt and holds the topmost priority among all interrupts in the 8085 microprocessor.

- **RESET IN** – It is pin number 36 in the pin diagram. An active low signal at this pin resets the PC of the microprocessor to 0. Or we can say, after resetting the PC holds its initial memory address.
- **RESET OUT** – It is the 3rd pin in the pin diagram. This pin generates a signal to provide information about the resetting of the microprocessor. Also, we can say that once a processor is reset then all the connected devices must also be reset. So, enabling this signal shows the resetting of the interconnected devices.
- **INTA**: It is the 11th pin of the 8085 pin configuration. A signal at this pin acknowledges the generated interrupt.

Direct Memory Access (DMA)

We are aware of the fact that memory and I/O devices are connected with each other by the microprocessor. So, the intermediator i.e., CPU manages the data transfer between the input-output device and memory.

However, when data in a large amount is to be transferred between I/O devices and memory the CPU gets disabled by tri-stating its buses. And this transfer is manageable by external control circuits. The DMA has 2 pins.

- **HOLD** – This signal is generated at pin number 39. This pin generates a signal to notify the processor that more than one request is present to access the data and address bus. When this signal gets enabled, the CPU frees the bus after completion of the recent operation. Once the hold signal gets disabled, the processor can access the bus again.

- **HLDA -**This signal is generated at pin number 38. This signal is enabled at the time when the processor gets HOLD signal and it releases HLDA i.e., hold acknowledge signal. In order to show that the multiple requests are kept on hold and will be considered once the bus gets free after the recent operation. After the disabling of hold request, the HLDA signal becomes low.
- **READY -**This is the 35th numbered pin in the pin diagram that maintains synchronization between the processor and peripherals, memory. It is clear that a microprocessor has a much faster response than peripherals and memory. So, this pin is enabled when the processor as well as the peripherals and memory both become ready to begin the next operation. In the case when the READY pin is disabled, then the microprocessor is in the WAIT state.

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.

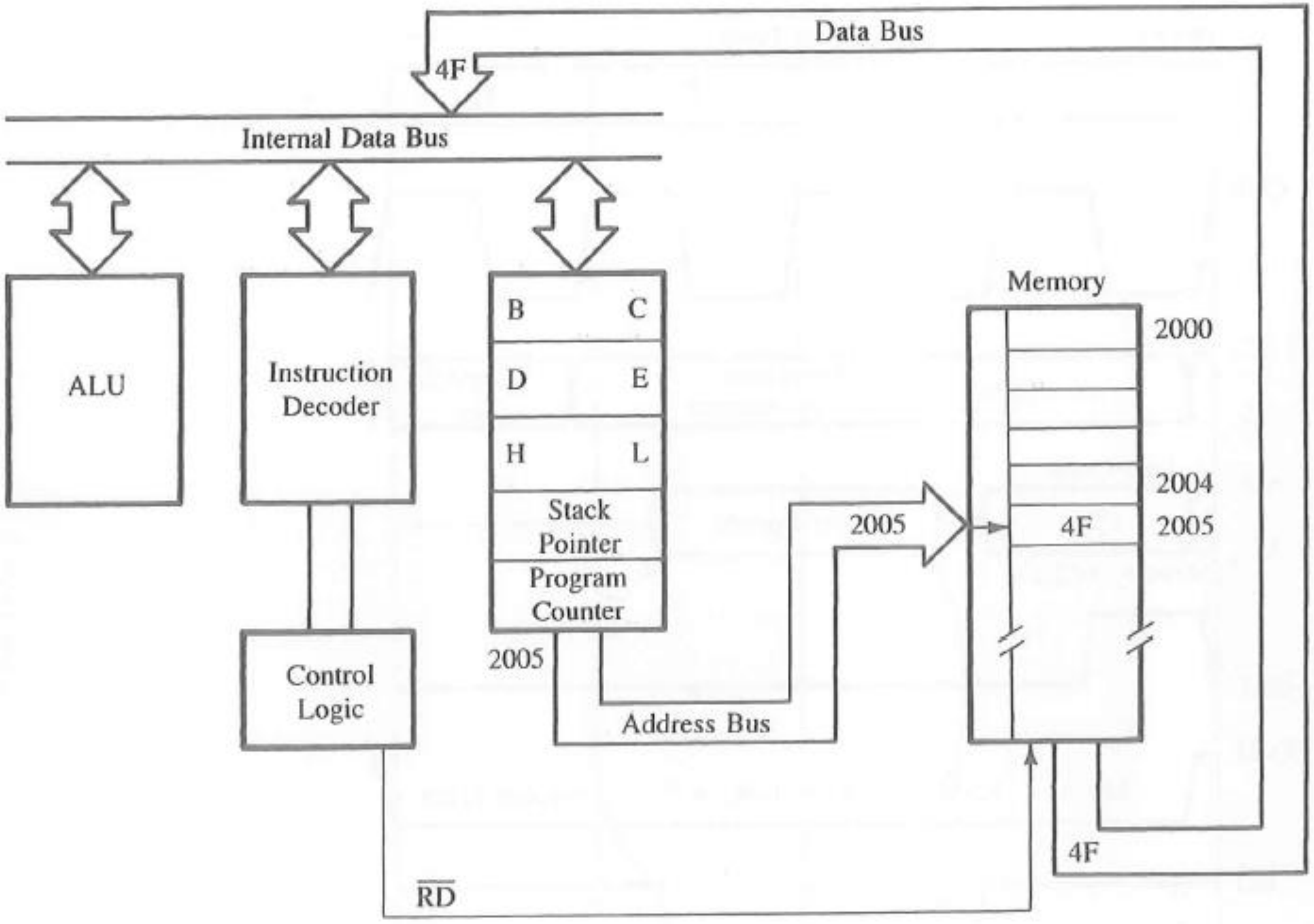
Now let us examine the steps in the following example of how the microprocessor fetches or gets a machine code from memory.

Example

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.

Ans: 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, and the timings are explained below.

Figure 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 in Figure. In our analogy, this is the equivalent of our courier getting on the road to find the address.

Step 2: The control unit sends the control signal RD to enable the memory chip. This is similar to ringing the doorbell in our analogy of a package pickup

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

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

The Address and Data Busses

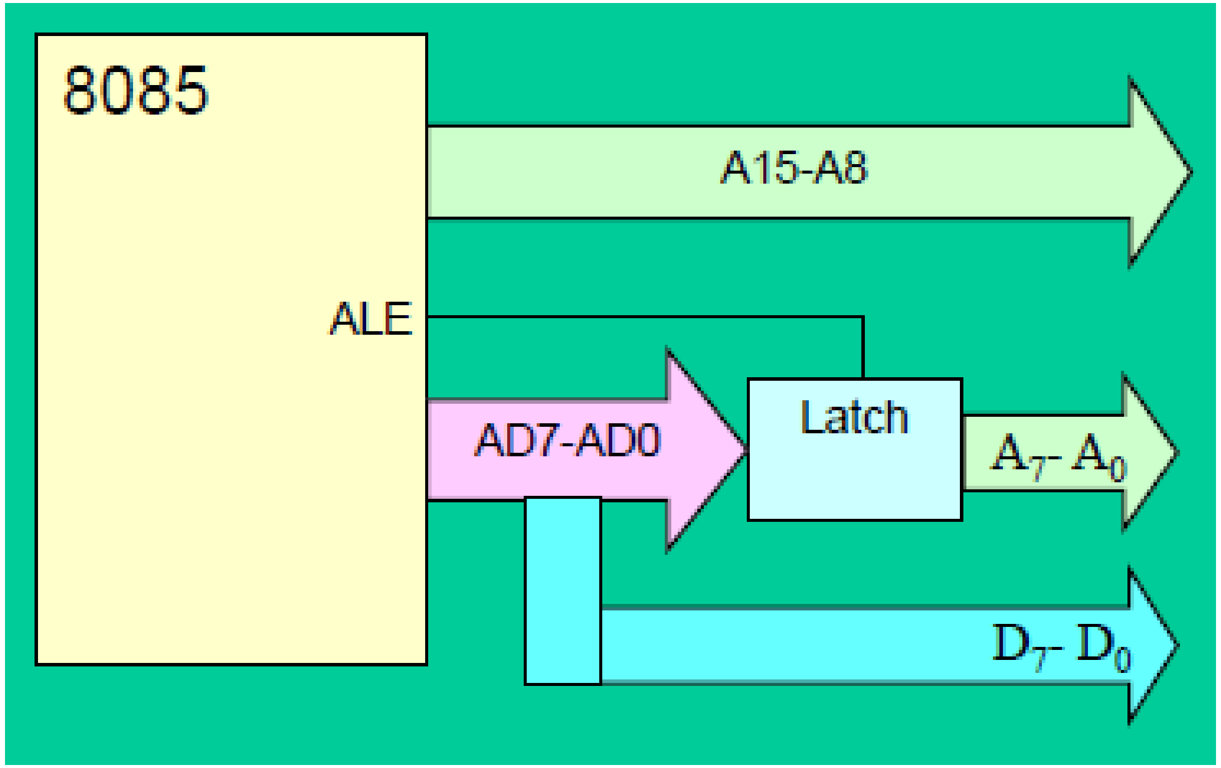
- The address bus has 8 signal lines A8 –A15which are unidirectional.
- The other 8 address bits are multiplexed(time shared) with the 8 data bits.
  - So, the bits AD0 –AD7are bi-directional and serve as A0 – A7and D0–D7at the same time.
    - During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.
- In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

Demultiplexing AD7-AD0

- From the above description, it becomes obvious that the AD7–AD0 lines are serving a dual purpose and that they need to be demultiplexed to get all the information.
- The high order bits of the address remain on the bus for three clock periods. However, the low order bits remain for only one clock period and they would be lost if they are not saved externally. Also, notice that the low order bits of the address disappear when they are needed most.
- To make sure we have the entire address for the full three clock cycles, we will use an external latch to save the value of AD7–AD0 when it is carrying the address bits. We use the ALE signal to enable this latch.

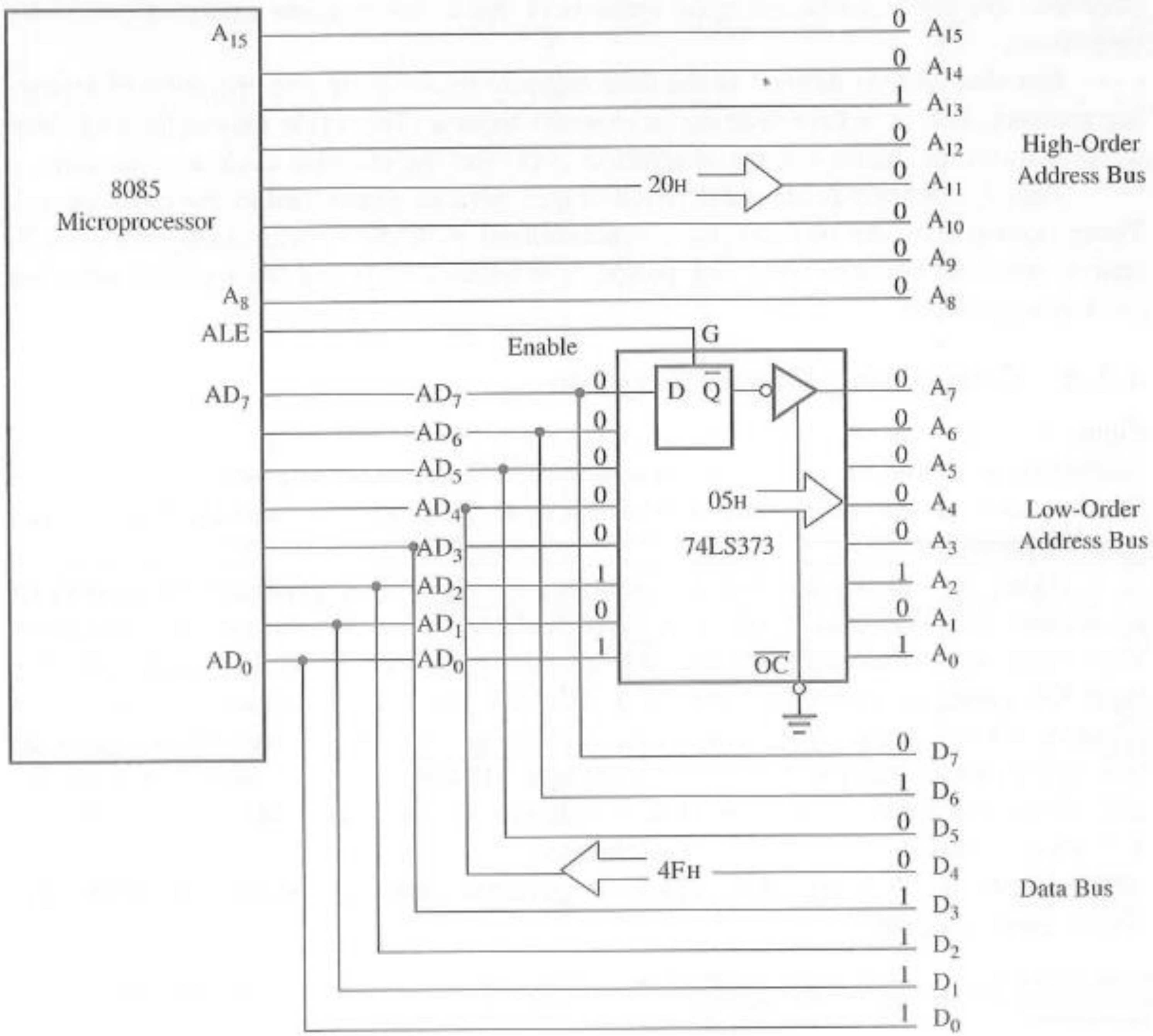
Demultiplexing AD7-AD0

- Given that ALE operates as a pulse during T1, we will be able to latch the address. Then when ALE goes low, the address is saved and the AD7–AD0 lines can be used for their purpose as the bi-directional data lines.



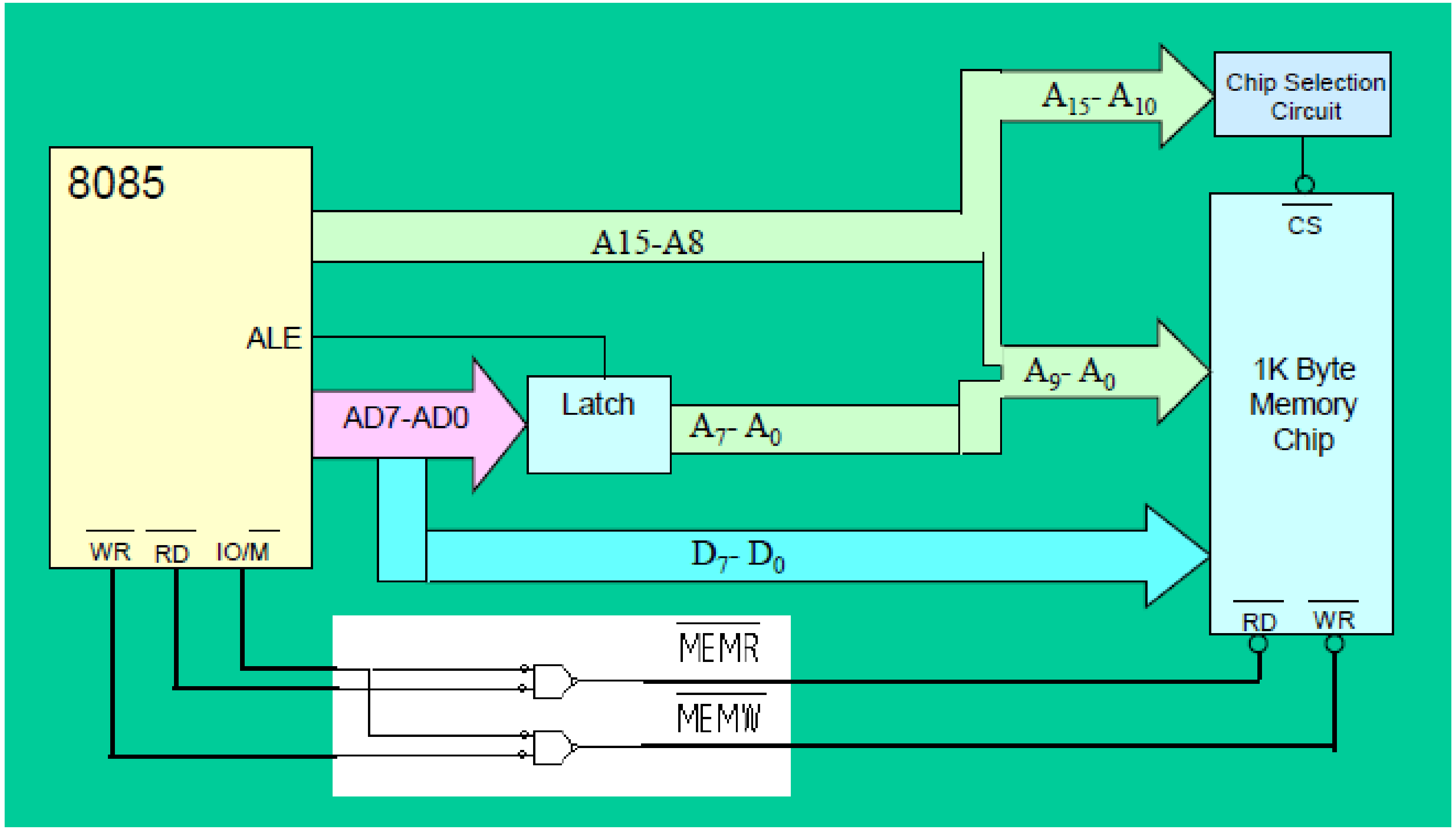
Demultiplexing the Bus AD7–AD0

- The high order address is placed on the address bus and hold for 3 clk periods,
- The low order address is lost after the first clk period, this address needs to be hold however we need to use latch
- The address AD7 –AD0 is connected as inputs to the latch 74LS373.
- The ALE signal is connected to the enable (G) pin of the latch and the OC –Output control –of the latch is grounded


8085
Microprocessor
A15
A8
ALE
AD7
AD0
0 A15
0 A14
1 A13
0 A12
20H
0 A11
0 A10
0 A9
0 A8
High-Order
Address Bus
Enable
G
D
Q
AD7
AD6
AD5
AD4
AD3
AD2
AD1
AD0
0
0
0
0
0
1
0
1
05H
74LS373
OC
0 A7
0 A6
0 A5
0 A4
0 A3
1 A2
0 A1
1 A0
Low-Order
Address Bus
0 D7
1 D6
0 D5
4FH
0 D4
1 D3
1 D2
1 D1
1 D0
Data Bus

The Overall Picture

- Putting all of the concepts together, we get:



8085
ALE
AD7-AD0
Latch
A15-A8
A_{15}- A_{10}
Chip Selection Circuit
$\overline{CS}$
A_9- A_0
A_7- A_0
1K Byte Memory Chip
D_7- D_0
$\overline{WR}$
$\overline{RD}$
$IO/\overline{M}$
$\overline{MEMR}$
$\overline{MEMW}$
$\overline{RD}$
$\overline{WR}$