UNIT-I


An Overview of Microprocessor

- The first question that comes in one’s mind is “What is a microprocessor?”.
- Let us start with a more familiar term computer.
- A digital computer is an electronic machine capable of quickly performing a wide variety of tasks.
- They can be used to compile, correlate, sort, merge and store data as well as perform complex calculations at much faster rate than human being by means of stored instructions.
- A digital computer is different from a general purpose calculator in a sense that digital computer is capable of operating according to the instructions that are stored within the computer whereas a calculator must be given instructions on a step by step basis to perform calculations.
- By this definition a programmable calculator can be considered a computer.
- Historically, digital computers have been categorized according to the size using the words large, medium, minicomputer and microcomputer.
- In the early years of development, the emphasis was on large and more powerful computers.
- Large and medium sized computers were designed to solve complex scientific and engineering problems.
- In early stage of development these computers were accessible and affordable only to large corporations, big universities and government agencies.
- Later on, minicomputers were made available for use in office, small collage, medium size business organization, small factory etc.
- As the technology has advanced from SSI to VLSI & SLSI, the face of the computer has changed gradually and it became possible to build the entire central processing unit (CPU) on a single-chip known as microprocessor.
- A control processing unit (CPU) with its related timing functions on a single chip known as microprocessor.
- A microprocessor combined with memory and input/output devices forms a microcomputer.
- The microcomputer is making an impact on every activity of mankind.
- It is being used in almost all control applications.
For example analytical and scientific instruments, data communication, character recognition, musical instruments, household items, defense equipments, medical equipments etc.

Microcomputers or, in general, computers communicate and operate in binary numbers ‘0’ and ‘1’ also known as bits. (A bit is the abbreviation for the term binary digit).

The microprocessor design engineer selects combinations of bit patterns and gives a specific meaning to each combination by using electronic logic circuits; this is called instruction.

The bit size of a microprocessor refers to the number of bit which can be processed simultaneously by the arithmetic circuit of the microprocessor.

A number of bits taken as a group in this manner is called word.

For example, the first commercial microprocessor the Intel 4004 which was introduced in 1971 is a 4-bit machine and is said to process a 4-bit word.

A 4-bit word is commonly known as nibble and an 8-bit word is commonly known as byte.

Intel 8085A is an 8-bit microprocessor whereas Intel 8086 is a 16-bit microprocessor.

It should be noted that a processor can perform calculations involving more than its bit size but through program and takes more time to complete the operation.

For example, multi-byte data can be added byte by byte in 8085A processor which is an 8-bit processor.

The short word length requires few circuitry and interconnection in the CPU.

**Microcomputers:**

In a very general sense, a microcomputer is best regarded as a system incorporating a CPU and associated hardware whose purpose is to manipulate data in some fashion.

This is exactly what any digital circuit designed using SSI’s and MSI’s does.

Therefore, microcomputer should be regard as a general purpose logic device.

In contrast to standard SSI’s and MSI’s where the manufacturer decides what the device will do, with microcomputer it is the user who decides what the device should do by asking it to execute a proper set of instructions.

A microcomputer, from this point of view is merely an assembly of devices whose sole task is to ensure that the instruction desired are indeed carried out properly and to allow the microprocessor to communicate with the real world, i.e. the user environment.

The power of the microcomputer lies in the fact that if the application changes, the same system can be used by appropriately modifying the instructions to be executed and, if necessary, some changes in the hardware.
In contrast, a digit circuit designed using SSI’s and MSI’s for some application will need to be completely redesigned if the application changes significantly.

The objective of a microcomputer (μc) is to manipulate data in a certain fashion specified by the system designer.

A typical microcomputer achieves their objective by getting its CPU to execute a number of instructions in the proper sequence.

This sequence of instruction comprises the program that is executed by the microcomputer.

A microcomputer which does nothing other than manipulate data present within itself, will not be of much use to anyone.

In order to do something meaningful, data being manipulated should depend on inputs provided to the microprocessor by the user.

Similarly, the data manipulations being carried out by the microprocessor would be completely meaningless unless the results of these manipulations affect things outside the microcomputer itself.

Therefore, the μc should provide outputs which in some way depend on its inputs, the way inputs and outputs are related is decided by the program that gets executed.

Therefore, a microcomputer is an assembly of devices including a CPU, which manipulate data depending on one or more inputs and according to a program, in order to generate one or more output.

**Microcontrollers**

- A μP does not have enough memory for program and data storage, neither does it has any input and output devices.
- Thus when a μP is used to design a system, several other chips, such as memory chips and input/output ports, are also used to make up a complete microcomputer system.
- For many applications, these extra chips imply additional cost and increased size of the product and may not be suitable for the application.
- For example, when used inside a toy, a designer would like to minimize the size and cost of the electronic equipment inside the toy.
- Therefore, in such applications a microcontroller is used more often than a microprocessor.
- A microcontroller is a chip consisting of a microprocessor, memory and input/output ports.

**Evolution of the Microprocessors**

- The first μP was introduced in 1971 by Intel Corporation. This was the Intel 4004, a processor on a single chip. It had the capability of performing simple arithmetic
and logical operations. For example, addition, subtraction, comparison, logical AND and OR operations.

- It also had a control unit which could perform various control functions like fetching an instruction from the memory, decoding it and generating control signals to execute it.
- It was a 4 bit µP operating on 4 bits of data at a time.
- The processor was the central component in the chip set, which was called the MCS-4.
- The other components in the set were a 4001 ROM, 4002 ROM and a 4003 shift register.
- Shortly after the 4004 appeared in the commercial market place, three other general purpose microprocessors were introduced.
- These devices were the Rockwell International 4-bit PPS-4, the Intel 8-bit 8008 and the National Semiconductor 16-bit IMP-16.
- Other companies had also contributed in the development of µP.
- The first 8 bit µP, which would perform arithmetic and logic operations on 8 bit words, was introduced in 1973, by Intel.
- This was 8008 that was followed by an improved version- the 8080 from the same company.
- The µPs introduced between 1971 and 1972 were the first generation systems.
- They were designed using the PMOS technology (P-Type Metal Oxide Semiconductor Logic).
- This technology provided low cost, slow speed and low output currents and was compatible with TTL.
- After 1973, the second generation µPs such as Motorola 6800 and 6809, Intel 8085 and Zilog Z80 evolved.
- These µPs were fabricated using NMOS technology (N-Type Metal Oxide Semiconductor Logic).
- The NMOS process offered faster speed and higher density than PMOS and was TTL compatible.
- The distinction between the 1st & 2nd generation devices was primarily the use of new a semiconductor technology to fabricate the chips.
- This new technology resulted in a significant increase in instruction execution speed & higher chip densities.
- After 1978, the 3rd generation microprocessors were introduced.
- Typical µPs were Intel 8086/ 80186/ 80286 and Motorola 68000/ 68010.
- These µPs were designed using HMOS technology (High Performance N-Channel Metal Oxide Semiconductor Logic).
- HMOS provides the following advantages over NMOS.
1. **Speed power produced (SSP) of HMOS is 4 times better than that of NMOS.** That is for NMOS, SSP is 4 picojoules (PJ) and for HMOS, SSP is 1 picojoules (PJ).
   
   - Speed power product = speed * power
   - = nanoseconds * mill watt
   - = picojoules

2. **Circuit densities provided by HMOS are approximately twice those of NMOS.** That is for NMOS, it is 4128 µm² /gate and for HMOS it is 1052.5 µm² /gate.

   - Later, Intel introduced a high speed version of the 8085A called 8085AH using HMOS technology to fabricate the 8085A.
   - The third generation introduced in 1978 was typically separated by the Intel 8086 iAPX8086, iAPX80186, iAPX80286, Zilog 8000, and the Motorola 68000 which are 16-bit µPs with minicomputer like performance.
   - One of the most popular 16-bit µP introduced by Intel was 8088.
   - The 8088 has the same instruction set as the 8086. However, it has only an 8 bit data bus.
   - The 8088 is the µP used in the IBM PC and its clones.
   - A precursor to these microprocessors was the 16-bit Texas Instruments 9900 microprocessor introduced in 1976.
   - **In 1980, the fourth generation µPs were evolved.**
   - **Intel introduced the first commercial 32 bit microprocessor, Intel 432. Since 1985, more 32-bit µPs have been introduced.**
   - These include Intel iAPX80386, Intel 80486, Motorola MC68020/68030/68040, National semiconductor NS 32032.
   - **These processors were fabricated using the low power version of HMOS technology called HCMOS (High Density Complementary Metal Oxide Semiconductor),** and they include an on-chip RAM called the cache memory to speed up program execution.
   - The characteristics for few microprocessors introduced by Intel are given in the Table 1.1.
   - This shows that power of microprocessors has increased tremendously with advancement in integrated circuit technology & microprocessor systems architecture.
   - Very large scale integration, VLSI, allows extremely complex system consisting of as many as a million of transistors on a single chip to be realized.
   - The performance offered by a 32-bit µP is more comparable to that of supercomputers such as VAX 11.
   - Extensive research is being carried out for implementation of more on chip functions and for improvement of the speed of the memory and I/O devices.
Table 1.1: Evaluation of major μP characteristics from Intel

<table>
<thead>
<tr>
<th></th>
<th>4004</th>
<th>8008</th>
<th>8085A</th>
<th>8086</th>
<th>80386</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Bus</td>
<td>4-bit</td>
<td>8-bit</td>
<td>8-bit</td>
<td>16-bit</td>
<td>32-bit</td>
</tr>
<tr>
<td>Technology</td>
<td>PMOS</td>
<td>PMOS</td>
<td>NMOS</td>
<td>HMOS</td>
<td>CHMOS</td>
</tr>
<tr>
<td>Word size data/ instr.</td>
<td>4/8</td>
<td>8/8</td>
<td>8/8</td>
<td>16/16</td>
<td>32/32</td>
</tr>
<tr>
<td>Address capacity</td>
<td>4K</td>
<td>16K</td>
<td>64K</td>
<td>1M</td>
<td>4G</td>
</tr>
<tr>
<td>Clock kHz/phase</td>
<td>740/2</td>
<td>800/2</td>
<td>6250/2</td>
<td>8000/2</td>
<td>16000/2</td>
</tr>
<tr>
<td>Addition time</td>
<td>10.8μs</td>
<td>20μs</td>
<td>1.3μs</td>
<td>0.375μs</td>
<td>0.125μs</td>
</tr>
<tr>
<td>ALU/General Purpose Reg.</td>
<td>1/16</td>
<td>1/6</td>
<td>1/6</td>
<td>1/8</td>
<td>1/8</td>
</tr>
<tr>
<td>Stack size</td>
<td>3x12</td>
<td>7x14</td>
<td>RWM</td>
<td>RWM</td>
<td>RWM</td>
</tr>
<tr>
<td>Voltages</td>
<td>15-10,5*</td>
<td>-9.5v</td>
<td>+5V</td>
<td>+5V</td>
<td>+5V</td>
</tr>
<tr>
<td>Package size</td>
<td>16pin</td>
<td>18pin</td>
<td>40pin</td>
<td>40pin</td>
<td>132pin</td>
</tr>
<tr>
<td>Instructions</td>
<td>45</td>
<td>48</td>
<td>74</td>
<td>133</td>
<td>135</td>
</tr>
<tr>
<td>Transistors</td>
<td>2,300</td>
<td>2,000</td>
<td>6,200</td>
<td>29,000</td>
<td>2,75,000</td>
</tr>
<tr>
<td>Chip size(mil)</td>
<td>117x159</td>
<td>125x170</td>
<td>164x222</td>
<td>225x230</td>
<td>390x390</td>
</tr>
</tbody>
</table>

Applications of Microprocessors:

- The application of microprocessors is increasing day by day. Some of the applications are:
  1. Analytical scientific instruments
  2. Smart terminals
  3. Stacker crane controls
  4. Conveyor controls
  5. Word processor
  6. Point of scale systems
  7. Standalone electronics cash system
  8. Electronic games
9. Vending and dispensing machines
10. Market scales
11. Traffic light controls
12. Home heating and lighting controls
13. Security & fire alarm system
14. Home appliances
15. Computer aided instruction
16. On line control of lab instrumentation
17. Desktop computers
18. Check processor
19. Payroll system
20. Inventory control
21. Automatic type setting
22. Compact business machines
23. Medical instrumentation
24. Automobile diagnostics
25. Data communication processing
26. Optical character recognition
27. I/O terminal for computers.

Reference:
- [https://nptel.ac.in/courses/108107029/](https://nptel.ac.in/courses/108107029/)
**Microcomputer Organization:**

- As discussed earlier microprocessor is a central processing unit (CPU) with its related timing functions on a single chip.
- A microprocessor combined with memory and input/output devices forms a microcomputer.
- Therefore, the basic components of a microcomputer are:
  1. CPU
  2. Program memory
  3. Data memory
  4. Output ports
  5. Input ports
  6. Clock generator.
- These components are shown in below figure 1.1

![Fig. 1.1 Basic Components of Microcomputer](image)

**Central Processing Unit:**

- The CPU consists of ALU (Arithmetic and Logic Unit), register unit and control unit.
- The CPU fetches the stored instructions from the program memory, data word from data memory or from an input device and after processing the data stores the result in data memory or sends it to an output device.
  1. **ALU (Arithmetic and Logic Unit)**
     - This unit performs computing functions on m-bit data where ‘m’ is the bit size of the processor.
     - These functions are arithmetic operations such as addition, subtraction and logical operation such as AND, OR, XOR, rotate, compare etc.
• Results are stored either in registers or in memory or sent to output devices.

2. Register Unit:
• It contains various 8-bit or 16-bit registers.
• These registers are used primarily to store data temporarily during the execution of a program.
• Some of the registers are accessible to the user through instructions.
• It means there contents can be read and/or changed through instructions.
• Some of the registers are not accessible to user but they are used by the processor for the execution of an instruction.
• 8085A microprocessor contains 8-bit registers such as Accumulator (Reg. A), B, C, D, E, H, L etc and 16-bit registers such as Program Counter (PC), Stack Pointer (SP).

3. Control Unit:
• It provides necessary timing & control signals required for the operation of microcomputer.
• It controls the flow of data between the microprocessor and peripherals (input, output & memory).
• The control unit gets a clock signal which determines the speed of the microprocessor.
• In all, the CPU has the following basic functions:
  o It fetches an instructions word stored in memory.
  o It decodes the instruction to determine what the instruction is telling it to do.
  o It executes the instruction. Executing the instruction may include some of the following major tasks:
    1. Transfer of data from one register to another register in the CPU itself.
    2. Transfer of data between a CPU register & specified memory location or input/output device.
    3. Performing arithmetic and logical operations on data from a specific memory location or a designated CPU register.
    4. Directing the CPU to change the sequence of fetching instructions, if processing the data created a specific condition.
    5. Performing housekeeping function within the CPU itself in order to establish desired condition at certain registers.
  o It looks for control signal such as interrupts and provides appropriate responses.
It provides status, control, and timing signals that the memory and input/output section can use.

**Memory:**
- It stores both the instructions to be executed (i.e. program) and the data involved.
- It usually contains ROM (Read Only Memory) and RWM (Read Write Memory).
- The **ROM can only read and cannot be written into and is non volatile** that is, it **retains its contents when the power is turned off**.
- **A ROM is typically used to store instructions and data that do not change.**
- For example, it stores the monitor program of a microcomputer.
- One can either read from or write into a RWM in memory read operation or memory write operation respectively.
- The **RWM is volatile**, that is it **does not retain its contents when the power is turned off**.
- **It is used to store user programmes & data which are temporary** might change during the course of executing a program.
- During a **memory read operation, the content** of the addressed location is **not destroyed**.
- During a **write operation, the original content** of the addressed location is **destroyed**.

**Program Memory:**
- The basic task of a microcomputer system is to ensure that its CPU executes the desired instructions sequence i.e., the program properly.
- The **instructions sequence is stored in the program memory**.
- On initialization- usually on power up or manual reset the processor starts executing the instructions from a predetermined location in program memory.
- The first instruction of the program should, therefore, be in this location.
- In typical processor based system, the program to be executed is fixed one which does not change.
- Therefore these programmes are stored in non-volatile memory such as ROM, or PROM, EPROM, EEPROM.
- **In the trainer kit**, ROM contains only the monitor program which is an application program for the trainer system. It allows the user to interact with microprocessor to enter user program and execute it. The user program is not stored in ROM because it needs not to be stored permanently. The user program is stored in RWM or RAM.

**Data Memory:**
- A microcomputer manipulates data according to the algorithm given by the instruction in the program in the program memory.
- These instructions may require intermediate results to be stored.
- The functional block in the µc used for this storage is the data memory.
Microprocessors also have a small amount of memory in the form of internal registers which can also be used if available for such storage.

External data memory is needed if the storage requirement is more.

Apart from intermediate storage, the data memory may also be used to provide input data needed by the program and to store some of the results of the program.

Data memory is used for all storage purposes other than storage of program.

Therefore, they must have Read-Write capability called Read-Write Memory.

Both ROM & RWM are arranged into words, each of which has a unique address.

The address of a word in memory is different than its contents.

To refer the contents of a memory location, its address is placed in parentheses.

Therefore, X is an address and (X) is the content of that address X.

The address decoder takes an address and from the control unit and selects the proper memory location.

Finding the correct memory location and obtaining its content takes certain amount of time, this times is the access time of the memory.

The access time affects the speed of the computer since the computer must obtain the instruction and data from the memory.

Computer memories are usually RAM so that all memory location have the same access time.

The computer must wait whenever it uses its memory, typical memory access time range from few nano-secs to several µsecs.

Memory sections often subdivided into units called pages.

The entire memory section may involve million of words, whereas a page contains between 256 & 4k words.

The computer may access a memory location by first accessing a particular page and then accessing a location (or line number) on that page.

The advantage of paging is that the computer can reach several locations on the same page with just the address on the page.

The process is like describing street address by first specifying a street and then listing the house numbers.

The control section transfers data to or from memory as follows:

1. The control section reads an address to the memory.
2. The control section sends a read and write signal to the memory to indicate, the direction of the transform.
3. The control section waits until transfer has been completed. This delay precedes the actual data’s transfer in the input case and follows it in the output case.

Input / Output Ports:
The input & output ports provide the microcomputer the capability to communicate with the outside world.

The input ports allow data to pass from the outside world to the µc data which will be used in the data manipulation being done by the microcomputer to send data to output devices.

The user can enter instruction (i.e. program) and data in memory through input devices such as keyboard, or simple switches, CRT, disk devices, tape or card readers.

Computers are also used to measure and control physical quantities like temperature, pressure, speed etc.

For these purposes, transducers are used to convert physical quantise into proportional electrical signals A/D computers are used to convert electrical signals into digital signals which are sent to the compute.

The computer sends the results of the computation to the output devices e.g. LED, CRT, D/A converters, printers etc.

These I/O devices allow the computer to communicate with the outside world. I/O devices are called peripherals.

Clock Generator:

Operations inside the microprocessor as well as in other parts of the microcomputer are usually synchronous by nature.

This is done so that events in different parts of the system can proceed in a systematic fashion.

The clock needed to perform this synchronous operation is provided by the clock generator.

The clock generator generates the appropriate clock periods during which instruction executions are carried out by the microprocessor.

Some of the microprocessors have an internal clock generator circuit to generate a clock signal.

These microprocessors require an external crystal or RC network to be connected at the appropriate pins for deciding the operating frequency (e.g. 8085A).

Some microprocessors require an external clock generator (e.g. 8086). These microprocessors also provide an output clock signal which can be used by other devices in the microcomputer system for their own timing and synchronizing.

Reference

https://nptel.ac.in/courses/108107029/2
**Intel 8085 Microprocessor**

- The 8085A is an 8-bit microprocessor suitable for a wide range of application.
- It is a 40-pin DIP (Dual in package) chip, based on **NMOS technology**.
- It contains **approximately 6200 transistors on a 164 x 222 mil chip**.
- The pin configuration is shown in fig. 1.2
- All the signals of 8085 is 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
  6. Serial I/O ports
- It requires a single +5V supply between Vcc at pin no. 40 and Vss at pin no. 20.

![Fig.1.2 (a) Pin Configuration of Intel 8085A Microprocessor](image-url)
Pin Configuration of Intel 8085A Microprocessor:

- A15 – A8 at pin no. 28 to pin no. 21:
  - The microprocessor can address directly 216 memory locations or 65536 memory locations or 64k memory locations using 16- address lines (A15-A0).
- Pin no. 28 to pin no. 21 give us the higher order 8-bits of the address (A15-A8).
- These address lines are **unidirectional, tri-state address lines**.
- These address lines become tristated under three conditions namely:
  a) During DMA (direct memory access) operation.
  b) When a HALT instruction is executed.
  c) When microprocessor is being RESET.

**AD7–AD0 at pin no. 19 to pin no. 12:**
- Pin no. 19 to pin no.12 marked AD7–AD0 are used for dual purpose.
- It is **time multiplexed lower 8-bit address bus** (A7-A0) and 8-bit data bus (D7-D0).
- Because at the time when this chip was developed, the practical limit on the numbers of pins was 40.
- The only solution was to multiplex part of the address bus with the data bus.
- Before discussing, the functions of different pins, it is better to know few more points about the processor.
- The microprocessor, being a logic circuit, shall move from one state to the other state during it operation.
- There are ten (10) different possible states for the processor and the processor will be in one of these states as long as the power is ON.
- These states are:
  1. **RESET STATE**: (T_{RESET}): Whenever microprocessor is reset, it enters in reset state. The microprocessor can be in T_{RESET} state for an integral multiple of clock cycle.
  2. **WAIT STATE**: It can be in this state for an integral number of clock cycles, the duration being determined by an external content signal input marked READY.
  3. **HOLD STATE**: (T_{HOLD}): As long as HOLD signal is active, microprocessor is in HOLD state.
  4. **HALT STATE**: (T_{HALT}): Microprocessor enters in this state when an HALT instruction is executed by the processor. It remains in this state till such time when an external signal dictated by the user asked the microprocessor to perform further duties.
  5. The other states the microprocessor can be in are marked T_1, T_2, T_3, T_4, T_5 & T_6 state. Each of these states is of one clock period duration. During each of these predetermined timing slots microprocessor performs very well defined activities.
- Pin no.19 to pin no.12 are used by the microprocessor to send lower order 8-bits of the memory address during T_1 timing plot of a machine cycle.
• Therefore, the same 8-pins are utilized as bi-directional data bus for data transfer operation in the subsequent timing plots $T_2$ & $T_3$. Hence, these pins are designated as AD7 – AD0.

• These 8 lines are also tri-state lines.

• They will be tri-stated during $T_4$, $T_5$ & $T_6$ states.

• They will also be restated during DMA operation, during RESET operation & when a HALT instruction is executed.

• These lines will also be tri-stated for a very short duration of time (few neon seconds) between $T_1$ & $T_2$ states

**ALE at Pin No 30:**

• The 8085A uses a time multiplexed address-data bus.

• This is due to limited number of pins on the 8085A.

• Low-order 8-bits of the address appear on the AD bus during the first clock cycle i.e., $T_1$ state of a machine cycle.

• It then becomes the data bus during the second and third clock cycles i.e., $T_2$ and $T_3$ states.

• **ALE stands for address latch enable.**

• It is used to distinguish whether the AD7 – AD0 bus contains address bits A7 – A0 or data bits D7- D0.

• It is a single pulse issued during every $T_1$ state of the microprocessor as shown in fig.1.3.

![Fig.1.3 ALE Signal Issued in Every $T_1$ State](image)

• Since the lower 8-bits of the address information A7 to A0 is available at pin no.19 to pin no.12 only during T1 period, therefore, ALE pulse can be used to latch address A7 to A0 in an external latch.

• **ALE output is high during first half of the T1 period and its falling edge can be used to latch the address bits A7 to A0 in an external latch** e.g. 74LS373 register latch.

• Fig.1.4 (a) shows a schematic that uses a latch and the ALE signal to de-multiplex the bus.

• The bus AD7-AD0 is connected as the input to the latch 74LS373.

• The ALE signal is connected to the **enable (G) pin of the latch**, and the output control (OC) signal of the latch is grounded.
• When ALE goes high during the \( T_1 \) state of a machine cycle, the latch is transparent and the output of the latch changes according to the input.
• The CPU is putting lower-order bits of address during this time.
• When the ALE goes LOW, the address bits get latched on the output and remain so until the next ALE signal.

![Fig. 1.4 (a) Latching of Lower Order Address in External Latch](image)

• Once saved in an external latch the lower order address A7 to A0 shall be available at the output of the register latch for the subsequent states \( T_2 \), \( T_3 \), \( T_4 \), \( T_5 \) & \( T_6 \), while pin no. 19 to pin no.12 can then be utilized by the microprocessor for bi-directional operation.
• The falling edge of the ALE can also be used to store status information being output by the 8085A during each machine cycle.
• The **ALE output is never tri-stated in the 8085A**.
• **The manner of utilization of pins 19 to 12 is known as time multiplexed mode of operation.**
• The de-multiplexing of AD bus by latching lower byte of 16-bit address in external 8-bit latch 74LS373 is shown in fig.1.4 (b).
• Once the lower byte address is latched, the AD bus is available for bidirectional data transfer.
• The 8-bit higher order address issued by microprocessor in every \( T_1 \) state along with latched lower byte address constitutes unidirectional 16-bit address bus.
• The control signals put together constitutes bi-directional control bus, where some of the signals are always input and some are always output.
• The three buses, address bus, data bus and control bus together constitutes system bus.
• The fact that ALE is required is a direct consequence of having a multiplexed data/address bus.

• This is unlike the Intel 8080 microprocessor which is similar to the 8085A but where these buses are not multiplexed.

• Some of the peripheral chips 8155/8156/8355/8755A have internal multiplexing facility, therefore, ALE input pin of these peripheral chips is connected to ALE output pin of the 8085A, thus allowing a direct interface with the 8085A.

• Thus IC chips internally de-multiplex the AD bus using the ALE signal.

• Since a majority of peripheral devices do not have the internal multiplexing facility, there is external hardware necessity for it.

 الاقتصادي والـRD Control signals at pin no 32 and at pin no 31:

• The BDB (Bi-Directional Bus) at pin no 19 to 12 are used for bi-directional data transfer operation during T2 & T3 states.

• When the BDB is inputting the information from the external world into the microprocessor, we say that μp is in READ mode and operation is READ operation.

• When the μp is outputting 8-bit of information to the external world through BDB we say μp is in WRITE mode and operation is WRITE operation.

• To tell the external world that μp is in WRITE mode, μp issues a control signal output WR at pin no. 31. It is normally HIGH & active LOW.

• It goes LOW during the beginning of T2 state and goes HIGH again during middle of T3 state of the microprocessor. This is shown in fig.1.5.
During WRITE operation, the microprocessor (µP) first sends the desired address on the address bus during T₁ state, thereafter it places the desired data on BDB which is now in input mode and then issues a control signal, WR.

A low level on WR indicates that the data on data bus is to be written into the selected memory location or I/O device.

Data is setup at the trailing edge.

It is for the user to take appropriate action externally by the interfacing circuitry so that the data so placed goes to the appropriate device.

Similarly to tell the external world the microprocessor is in input mode for READ operation, it issues a control signal RD which is normally HIGH and active LOW.

RD signal goes LOW during T₂ state and goes HIGH again during T₃ state similar to WR signal.

A LOW level on RD indicates the selected memory or I/O device is to be read and the data bus is available for the data transfer.

It is for the user to keep the appropriate 8-bit data either from the memory or I/O device during this period.

Both RD and WR are never made LOW at the same time.

Both the signals are tri-stated during HOLD, HALT and RESET states.

**IO/M at pin no 34:**

IO/M is an output tri-state control signal. It is active both ways (HIGH as well as LOW).

Whenever the address issued by the µP on the address lines refers to the memory then the µP makes IO/M LOW throughout T₁, T₂, T₃, T₄, T₅ & T₆ states of the machine cycle to indicate the external world that the address so sent belongs to the memory and data on the BDB refers to the memory.

---

Fig. 1.5 Read and Write Signals Issued During T₂-T₃ State
• Whenever the address on the address lines refers to an I/O device the \( \mu p \) makes \( IO/\overline{M} \) control signal output HIGH to tell the external world that the address on the address bus refers to an I/O device and the data on the BDB refers to an I/O device.

• **Note** that \( IO/\overline{M} \) signal is LOW or HIGH as the case may be throughout six timing slots \( T_1, T_2, T_3, T_4, T_5 \) & \( T_6 \) states. It is for the user to make use of this feature to develop proper interfacing circuitry i.e., to generate the chip selected signals.

• In other words, a LOW \( IO/\overline{M} \) signal enables the memory chips and a HIGH \( IO/\overline{M} \) signal enables the I/O device.

✈ **READY at PIN NO 35:**

• This is a control signal input.

• There are many peripheral devices which are slow in operation compared to the microprocessor speed. Eg., the access time of a memory interfaced with a \( \mu p \) may be much larger than the clock period of the \( \mu p \).

• Thus, there is a need for telling the \( \mu p \) that the device so addressed by the \( \mu p \) is not ready for data transfer operation.

• The device, selected should have the ability to generate a control signal output READY which shall be LOW if the device is not ready for data transfer operation and goes HIGH when the device is READY for data transfer operation.

• This idea is summarized in fig.1.6 considering memory as the external device.

![Fig.1.6 Interfacing Slow Speed Memory Using READY Signal](image)

• The \( \mu p \) sent the address during \( T_1 \) state of the microprocessor to address the memory and then issues appropriate \( RD \) or \( WR \) signal during \( T_2 \) state either to read the memory or write into the memory.
• Having issued the appropriate \(RD\) or \(WR\) signal during \(T_2\) state of the \(\mu p\), it then samples and monitors READY control signal input in the middle of \(T_2\) state.

• If the READY control signal is found LOW the microprocessor knows that the device addressed is not ready for data transfer operation and therefore goes to WAIT state (\(T_{\text{WAIT}}\)).

• Once in WAIT state, the \(\mu p\) does not do any other works except monitoring control signal.

• The outputs remain unchanged during the wait period and they remain what they were at the end of the \(T_2\) clock cycle.

• Its internal status also remains unchanged during this time.

• This feature provides a slow peripheral device more time to respond that what it would normally have.

• As long as READY signal is LOW, \(\mu p\) remains in WAIT state.

• When the ready signal goes high \(\mu p\) realized that that the device addressed is ready for data transfer operation and comes out of the WAIT state and goes into \(T_3\) state.

• The partial state transition diagram describing the above process is shown in fig.1.7.

![Partial State Transition Diagram Showing WAIT State](image)

• The appropriate control signal output \(RD\) or \(WR\) shall remain LOW throughout the \(T_{\text{WAIT}}\) state and goes HIGH when the \(\mu p\) comes out of the \(T_{\text{WAIT}}\) state and goes to \(T_3\) state.

• The corresponding timing signals are shown below in fig.1.8.
**RESET IN** at pin no.36

- It is an **input** control signal normally **HIGH** and active **LOW**.
- It is used to **RESET** the microprocessor to its initial state.
- If **RESET** signal is held low for 600nsec (3 clock periods), this **input** forces the processor to do the following:
  
  a) Program Counter (PC) is reset to (0000)H.
  
  b) Instruction register (IR) is cleared i.e, ongoing instruction execution is discontinued.
  
  c) **All interrupts except TRAP are disabled** with RST7.5, RST6.5 and RST6.5 also masked.
  
  d) Serial Output Data line (SOD) is forced to 0.
  
  e) During RESET IN LOW data, address and control bus are floated i.e., these **buses are tri-stated**
  
  f) Because of the **asynchronous nature of the RESET**, the internal registers of the microprocessor and **flags are altered with unpredictable results**.
  
  g) The CPU remains in RESET state until the **RESET IN** goes high.

- During power-on the microprocessor must be RESET.
- The necessary circuitry for resetting the **µP** is shown in figure 1.9
Initially, the capacitor is discharged.
Therefore, to start with when the power is first put ON, \textit{RESET IN} control signal becomes LOW.
Therefore, the \( \mu P \) will be RESET to its initial state.
Upon power ON the \textit{RESET IN} must remain LOW for at least 10ms after the minimum \( V_{cc} \) has been reached.
Depending upon the time constant RC the voltage across the capacitor exponentially increases to 5V.
When the voltage across the capacitor reaches to 2.4V, \textit{RESET IN} control signal goes to logical 1; \( \mu P \) comes out of the RESET state and straight away goes to \( T_1 \) state.
The time duration to reach around 2.4V from the instant of switching the supply is around 4 to 5 clock cycles.
If this duration is not maintained the resetting action of the \( \mu P \) is not guaranteed and therefore RC combination should be selected accordingly.
The diode D is provided for the discharge path for the capacitor when the power supply is finally put OFF.
A push button switch is also provided to manually reset the \( \mu P \) as and when necessary.
The partial state diagram when \textit{RESET IN} control signal is active as shown in fig.1.10.
RESET OUT at Pin no. 3:

- It is normally low signal output. It indicates 8085A is being RESET.
- When \( \text{RESET IN} \) control signal at pin no. 36 is LOW, RESET OUT at pin no. 3 goes HIGH.
- It remains HIGH as long as \( \text{RESET IN} \) is active and LOW.
- RESET OUT control signal is provided for the user to use it to RESET all the peripheral devices to their initial states.
- The signal is synchronized with the system CLK and it remains high for an integral number of clock periods.
- After the \( \text{RESET IN} \) goes HIGH, RESET OUT goes LOW, the processor enters in the \( T_1 \) state and normal operation begins.

X1, X2 terminal at pin nos. 1 & 2 and CLK(OUT) at pin no. 37:

- The 8085A \( \mu \)P has an on-chip oscillator with all the required circuitry except for the crystal, LC or RC network that controls the operating frequency.
- The internal circuitry of on-chip oscillator is shown in fig. 1.11
- A crystal is connected across X1 and X2 to provide a crystal frequency of \( f_{\text{crystal}} \) MHz. Because of the internal \( T – \) Flip Flop the operating frequency of \( \phi 1 \) clock is \( f_{\text{crystal}}/2 \) (half of crystal frequency).
- The \( \phi 1 \) clock is also buffered and sent out through pin no. 37 to tell the outside world as CLK(OUT) signal.
- CLK(OUT) signal is used for synchronizing the peripheral devices with the \( \mu \)P operation.
- The data sheet of 8085 puts a lower limit to the operating frequency at 500 kHz.
• The maximum operating frequency for 8085A is 3.125 MHz. The maximum operating frequency for 8085A is 6.25 MHz.

![Schematic Diagram of Internal Clock Generator Circuit](image)

Fig. 1.11 Schematic Diagram of Internal Clock Generator Circuit

• Therefore, while using 8085A μp, a crystal having a frequency from 1.0 MHz to 6.25 MHz can be used; 10 MHz crystal has to be used for 8085 A-2 μp.

• The 20 pf capacitor is necessary between X2 and ground if the crystal frequency is less than 4 MHz to provide oscillation.

• It is not necessary for higher frequency.

• The data sheet also mentions a 15 pf stray capacitance across X1 and X2 internally.

• A crystal is to be used across X1 and X2 to get stable frequency of oscillation.

• This ensures a fixed clock period, needed for accurate time delays.

• If stable frequency of oscillation is not required, a parallel resonant LC network may be used as the frequency determining network provided its frequency tolerance of approximately ±10% is acceptable.

• The frequency of oscillation is given by,

\[
f = \frac{1}{2\pi \sqrt{L(C_{ext} + C_{in})}}
\]

• To minimize variations in frequency, it is recommended that the value of \(C_{ext}\) must be at least twice that of \(C_{in}\) i.e., 30 pf.
• The use of LC circuit is not recommended for frequencies higher than approximately 5 MHz.

Fig. 1.12 Use of (a) LC Oscillator (b) RC Oscillator as Clock Generator

• An RC circuit may be used as the frequency determining network for 8085 AH, if maintaining a precise clock frequency is of no importance.
• Variations in the ON chip timing generation can cause a wide variation in frequency when using the RC network.
• Its advantage is its low component cost.
• The driving frequency generated by the circuit shown is approximately 3 MHz.
• It is not recommended that frequencies higher or lower than this are to be attempted.
• If we have an external clock whose frequency is same and can be varied from 1MHz to 6.25 MHz, the external clock may be connected to X1 and X2 is left open as shown in Fig.1.13.
• If the driving frequency is 6 MHz to 12 MHz, stability of the clock generator will be improved by driving X1 and X2 with a push-pull source.
• To prevent self oscillations of the 8085, X2 should not be coupled back to X1 through the driving circuit.
• In last two cases, pull up resistors are required to assure that the high level voltage of the input is at least 4V and maximum low level voltage of 0.8V.
SID & SOD at PIN NO 5 & 4
- SID stands for SERIAL INPUT DATA and SOD stands for SERIAL OUTPUT DATA.
- These two pins are specially provided in 8085 μP for communicating with serial devices, like CRT, TTY, and Printers etc.
- Microprocessor as and when needed uses SID and SOD lines for transfer of data bit by bit along the same lines.
- The data on SID line is loaded into accumulator bit 7 whenever RIM instruction is executed.
- The output serial line SOD is set or reset as specified by the SIM instruction.

**INTERRUPT CONTROL SIGNALS:**
- TRAP at pin no. 6, RST7.5 at pin no. 7, RST6.5 at pin no. 8, RST 5.5 at pin no. 9, and INTR at pin no. 10 are interrupt control signals input provided for interrupting the μP while it is executing the programme.
- RST stands for RESTART.
- These interrupt control signal input can be broadly divided in to two categories:
  (a) Non – maskable interrupts
  (b) Maskable interrupts
- Non–maskable control signal inputs are those control signal inputs which can interrupt the μP programming execution once the power is ON.
The maskable interrupts are those control signal inputs which can be individually disabled or enabled as and when necessary.

Fig.1.14 Schematic Diagram of Interrupt Section of 8085A

The way these interrupt control signal inputs interrupt the µp can be pictorially represented as shown in fig.1.14

**TRAP at pin no.6:**
- TRAP control signal input of Intel 8085A processors is a Non Maskable (NMI) RESTART vectored interrupt.
- When the power is ON, it is enabled and no enable interrupt command is required.
- **TRAP has the highest priority of any interrupt.**
- It is both rising edge and level sensitive interrupt i.e. it becomes active at the Lo-Hi edge but must stay high until it is sampled and recognized.
- Whenever this interrupt is recognized, it forces the 8085A to perform a CALL 0024H instruction, means when the current instruction execution is over, the program counter (PC) is loaded with 0024H so that the CPU starts executing the program from 0024H.

**INTR at pin no.10:**
• This is the **lowest priority interrupt request** in the 8085A processor and is used as a general purpose interrupt.
• An input of INTR=1 implies some external device has put up an interrupt and wants the CPU to execute an appropriate service routine.
• The 8085A monitors the status of the INTR line by sampling it in the last but one clock cycle of each instruction and during HOLD & HALT states.
• If the interrupt structure of the 8085A is enabled when INTR is sampled high, the PC will not be incremented and an interrupt acknowledge signal ($\text{INTA}$) will be issued by the $\mu p$ in response to INTR.
• It is now the responsibility of the interrupting device to issue a RESTART or CALL instruction so that the 8085A can jump to the proper interrupt service subroutine.
• The INTR is enabled by executing an EI instruction and is disabled by executing a DI instruction.
• **Disabled means INTR will not be acknowledged.**
• It is also disabled by RESET and immediately after an interrupt is acknowledged.

**$\text{INTA}$ at pin no.11:**
- $\text{INTA}$ is an Interrupt acknowledge control signal output.
- This is an active LOW control signal output.
- When the $\mu p$ acknowledges any interrupt than instead of $\overline{RD}$ signal it issues $\text{INTA}$ signal to tell the external world that processor is now processing an interrupt acknowledge machine cycle.
- Basically, it replaces $\overline{RD}$ control signal output during $\text{INTA}$ machine cycle.
- $\text{INTA}$ is normally HIGH and becomes active LOW during T2 timing slot of the $\mu p$ and goes HIGH again during T3 state of the $\mu p$ just like $\overline{RD}$ signal.
- During this period $\overline{RD}$ signal is HIGH.
- It can be used to activate an 8259A interrupt controller chip or some other interrupt port.

**RST5.5, RST6.5 & RST7.5:**
- These are 8085A’s maskable vectored interrupt inputs.
- They operate exactly like INTR except for the following:
  1. The RESTART instruction is automatically inserted by internal logic. It does not have to be provided from outside. These instructions are:

<table>
<thead>
<tr>
<th>RST 5.5</th>
<th>identical to</th>
<th>CALL 0020 H</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST 6.5</td>
<td>identical to</td>
<td>CALL 0034 H</td>
</tr>
<tr>
<td>RST 7.5</td>
<td>identical to</td>
<td>CALL 0030 H</td>
</tr>
</tbody>
</table>

Restart instructions are special 1-byte unconditional CALL instructions. The address for any RST can be calculated multiplying...
the RST number with 8 and converting it into hexa. E.g. for RST5.5 the address is 5.5x8 = 44D = 002C H

2. RST7.5 is an edge (Low to High) sensitive interrupt unlike RST6.5, RST5.5 and INTR which are level (High) triggered.

3. These three interrupts can be individually masked or unmasked using SIM instructions.

4. They have higher priority than INTR. Among them RST7.5 has the highest priority and RST5.5 has the lowest priority.

- Like the INTR, whenever any of these interrupts is recognized it disables all the interrupts.
- These interrupts can be enabled/disabled using EI/DI instruction.

**Status Signals S1 (33) and S0 (29):**

- These two status signals along with IO/M signal output identify the type of the machine cycle being executed by the 8085A.
- These signals are issued (or become valid) at the beginning of the machine cycle and remain stable throughout the machine cycle.
- The falling edge of ALE may be used to catch the state of these lines, if required.
- The seven types of machine cycles are:
  1. Opcode Fetch Machine Cycle (OFMC)
  2. Memory Read Machine Cycle (MRMC)
  3. Memory Write Machine Cycle (MWRMC)
  4. I/O Read Machine Cycle (IORMC)
  5. I/O Write Machine Cycle (IOWRMC)
  6. Interrupt Acknowledge Machine Cycle (INTAMC)
  7. Bus Idle Machine Cycle. (BIMC)

- The truth table indicating the status of different signals to identify a machine cycle is shown below:
These control signals are normally HIGH and become active LOW during T<sub>2</sub> state and goes back to HIGH during T<sub>3</sub> state.

In between T<sub>2</sub> & T<sub>3</sub> states any no of WAIT states T<sub>WAIT</sub> can be inserted.

**HOLD at pin no 39 and HLDA at pin no 38:**

- HOLD (Hold) is a control signal input and HLDA (Hold acknowledge) is a control signal output.
- These two signals are used for hand shaken control during DMA operation (Direct Memory Access).
- The use of these control signals are depicted in the function diagram of fig.1.15.
- These two signals are used where there is more than one CPU like devices sharing the same system bus.
- The device asking for DMA makes the HOLD signal input HIGH.
- The processor which continuously monitors the HOLD signal input during each machine cycle recognizes that an external device is requesting for a DMA (i.e., the control of address bus, data bus, RD, WR and IO/M) completes its current machine cycle in, thereafter, tri states the address bus data bus and RD, WR and IO/M control signals and then enters into a HOLD state THOLD.
Also, while entering the HOLD state the µp issues the HOLD acknowledge signal HLDA high at pin no. 38.

The external device asking for DMA monitors the HLDA control signal continuously and knows that the µp has gone into HOLD state when HLDA is found HIGH.

Therefore, data bus, address bus, RD, WR and IO/M control signals which are tri-stated with respect to the µp are in the exclusive control of the external device asked for DMA.

The DMA operation between the external device and memory continues and data transfer takes place between external device and processor memory.

Internal processing not requiring the use of the system buses may continue. Note that all instructions are in program memory and, therefore, to read the instructions from the memory system bus is required which is not available to µp and therefore, µp is forced to stop the execution at the next instruction.

The µp while in HOLD state continues to monitor the HOLD control signal input.

As long as it is HIGH it remains in HOLD state.

The external device performing the DMA operation, after completing its operation makes HOLD signal LOW to tell the µp that the DMA is over.
• The processor which is continuously monitoring HOLD signal in HOLD state recognizes the above fact and comes out of HOLD state and continues the operation from where it has gone into hold state.

• HLDA goes LOW after HOLD goes LOW; the 8085A takes the control of bus one half a clock cycle after HLDA returns to ‘0’.

• The system configuration indicating address lines, data lines and all control lines including “Power ON” reset circuit, clock generator is shown in fig.1.16.

![Fig.1.16 System Configuration of Microprocessor](https://nptel.ac.in/courses/108107029/10)

![Fig.1.16 System Configuration of Microprocessor](https://nptel.ac.in/courses/108107029/11)

![Fig.1.16 System Configuration of Microprocessor](https://nptel.ac.in/courses/108107029/12)
**Internal Architecture of Intel 8085A**

- The functional block diagram of 8085A is shown in fig.1.17.

![Internal Architecture of 8085A Microprocessor](image)

- It consists of five essential blocks.
  - Arithmetic Logic Section
• Register Section
• The Interrupt Control Section
• Serial I/O Section
• The Timing And Control Unit

- There is an internal bi-directional data bus of 8-bit wide.
- This bus is used to transfer data and instructions among various internal registers.
- All the internal registers which transfer data to the internal bus are tri-state registers.
- Higher order address bus (A15-A8) and time-multiplexed lower order address data bus (AD7-AD0) are the external buses and used to interface peripherals and memory chips to CPU.
- Address buffer and address/data buffers isolate the internal data bus from the external address bus and address/data bus and drive the external address bus and address/data bus.
- The CPU can send the address of desired memory locations and I/O chip through these buffers.
- The 8-bit internal data bus is also connected to the address/data buffers.
- The bi-directional arrows indicate a tri-state connection that allows the address/data buffer to send or receive data from the 8-bit internal data bus.
- In the output mode the information on the data bus is loaded into the 8-bit data latch that drives the address/data bus output buffer.
- The output buffers are floated during input or non transfer operations.
- During the input mode, data from the external bus is transferred over the internal data bus to internal register.
- The figure shown does not include the control signals driving internal registers.

**Arithmetic & Logic Section:**
- This section consists of:
  a) Accumulator (A)
  b) Temporary Register (TR)
  c) Flag Register (FR)
  d) Arithmetic Logic Unit (ALU)

**Accumulator:**
- Arithmetic and/or logic operations on one or two operations are the basic data transformations implemented in a µp one of these two operands is always in the accumulator.
- Accumulator is an 8-bit register accessible to the user is connected to the 8-bit internal data bus.
- The bi-directional arrow between the accumulator and the bus indicates a tri-state connection that allows the accumulator to send or receive data.
• In addition, it has a two state 8-bit output.
• The content of the accumulator is always available at this two state output as one of the operands for the ALU.
• The contents of the accumulator can be manipulated through instructions.
• Its content can be incremented and decremented.
• The content of the memory location can be transferred to the accumulator and vice-versa.
• The result of arithmetic/ logical operations carried out by ALU is also stored back in the accumulator.
• In other words, it accumulates the result of the operation, hence, the name accumulator.

粽 Temporary registrar (TR):
• This is an 8-bit register not accessible to the user.
• It is used by the processor for internal operations.
• The second operand as and when necessary is loaded in to this register by the microprocessor before the desisted operation takes place in the ALU.
• The temporary register has 8-bits two state output.
• The second operand is always available at this output

粽 Arithmetic Logical Unit (ALU):
• ALU is a combination logic block which performs the desired operation on the two operands.
• The contents of the accumulator and the temporary register are the inputs to the ALU.
• This is governed by the control signals generated by the timing and control unit.
• The various arithmetic and logical operations that can be performed by ALU are:
  a) Binary addition, subtraction, increment and decrement,
  b) Logical AND, OR and EX-OR,
  c) Complement,
  d) Rotate left of right.
• The result of the operation is, in general, stored back in accumulator.
• In subtraction operation, the content of the temporary register is subtracted from the content of the accumulator and is stored back in the accumulator.
• In many applications it is appropriate to represent data in binary coded decimal (BCD) form.
• The result on any operation on BCD should also be in BCD form.
• The ALU contains additional logic to adjust result of addition operations where the operands are interpreted as BCD data.

❖ Flags register:
• The ALU influences a number of flip flops called flags which store information related to the results of arithmetic and logical operations.
• Taken together this flags constitute a flag register.
• Flag register is an 8-bit register accessible to the user through instruction. Each bit in the flag register has a specific function.
• Only 5 bits out of 8 bits are used as shown below.

<table>
<thead>
<tr>
<th>D_7</th>
<th>D_6</th>
<th>D_5</th>
<th>D_4</th>
<th>D_3</th>
<th>D_2</th>
<th>D_1</th>
<th>D_0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>Z</td>
<td>X</td>
<td>AC</td>
<td>X</td>
<td>P</td>
<td>X</td>
<td>CY</td>
</tr>
</tbody>
</table>

• The three crossed bit are redundant bits and not used.
• They can be either ‘0’ or ‘1’ but normally they are forced to be zero.
• The other five bits are affected as a result of execution of an instruction.
• All instructions do not affect these flags e.g. data transfer operation do not affect these flags.
• The meaning and the effect of these flags are as follows.

• **CY (Carry) Flag bit:**
  ✓ This particular bit is SET (=1) if there is a carry from the MSB position during an addition operation or if there is a borrow during the subtraction operation otherwise the flag is reset (=0).
  ✓ The processor, by design, does the subtraction operation also by taking 2’s complement of one operand and adding it to another operand.

• **P (Parity) Flag bit:**
  ✓ The parity flag test for the number of ‘1’s in the accumulator.
  ✓ If the accumulator holds on an even number of ‘1’s, it is said that even parity exists and the parity flag is set to ‘1’.
  ✓ However, if the accumulator holds an odd number of ‘1’ it is called odd parity and the parity flag is reset to ‘0’.
  ✓ In other words, if the module-2 sum of the bit is ‘0’, this flag is set otherwise the flag is reset.

• **AC (Auxiliary Carry) Flag bit:**
  ✓ This bit is set if there is a carry from b3 bit to b4 bit of accumulator during addition operation otherwise it is reset.
  ✓ The AC flag is useful for BCD arithmetic and is used in a particular instruction known as DAA (Decimal Adjust Accumulator).

• **Z (Zero) Flag bit:**
✓ Zero flag bit is SET if the result of an operation is zero, otherwise it is RESET.

- **Sign Flag bit:**
  ✓ The sign flag is set to the condition of the most significant bit of the accumulator following the execution of arithmetic or logical operation.
  ✓ These instructions use the MSB of the data (result) to represent the sign of the number contained in the accumulator.
  ✓ A set sign flag represents a negative number, whereas a reset flag means a positive number.

❖ **Example 1:**
  - Let us consider the execution of the instruction ADD B.
  - ADD is the mnemonic for addition.
  - The first operand is known to exist in the accumulator (Reg. A).
  - Register B contains the second operand.
  - The meaning of the instruction is add the contents of the B register to the contents of A register and store the result back in the accumulator (A).
  - Symbolically we can write,
    \[(A) \leftarrow (A) + (B)\]
  - Let us suppose the register contents are \((A) = 9BH\), \((B) = A5H\) before the execution of the instruction.
  - It means,
    \[
    \begin{align*}
    (A) &= 9BH \rightarrow (1001 1011)_2 \\
    (B) &= A5H \rightarrow (1010 0101)_2 \\
    \text{ADD B = } (A+B) &\rightarrow (0100 0000)_2
    \end{align*}
    \]
  - As a result of addition, there is a carry from b3 to b4 position and therefore AC is set.
  - Also there is a carry from the MSB out and, therefore, CY flag is also set.
  - Soon after the execution of ADD B instruction the accumulator contains \((A) = (0100 0000)2 = 40H\) and is non-zero.
  - Therefore Z flag is reset to zero. Also, result contains only one ‘1’, an odd number.
  - Therefore, parity bit is also be reset to zero.
  - Since the MSB of the result is zero, therefore the sign (S) bit is also reset.
  - Thus the flag register, soon after the execution of the instruction, contains \((0001 0001)2 = 11H\).

❖ **Example 2:**
  - Let us consider the execution of another instruction SUB B.
• SUB is the mnemonic for subtraction.
• Accumulator consists of first operand.
• Register B contains the second operand.
• The meaning of the instruction is subtract the contents of the B register from the contents of A register and store the result back in the accumulator (A).
• Symbolically we can write,
\[ (A) \leftarrow (A) - (B) \]
• Let us suppose the register contents are (A) = A5H, (B) = 9BH before the execution of the instruction. It means,
• Before execution A = A5H and B = 9BH
\[
\begin{align*}
(A) &= 1010 0101 \\
(B) &= 1001 1011 \quad \text{2's complement} \rightarrow 0110 0101 \\
\text{Carry 1} &\quad (0000 1010)_2
\end{align*}
\]
• Since result is non zero, therefore, Z bit is ‘0’.
• Sign bit is also ‘0’ because MSB of the result is ‘0’.
• AC is also ‘0’ because in addition (2’s complement), there is no carry from b3 to b4.
• Parity bit is ‘1’ (2 ones).
• CY bit seems to be ‘1’.
• But it is complemented and then stored.
• Therefore, CY bit is stored as ‘0’.
• It also indicates that (A) is having larger number than register (B), otherwise smaller one.
• Thus the flag register, after the execution of the instruction, contains (0000 0100)_2 = 04H.
• Let us consider (A) is having 9BH and (B) is A5H before execution.
\[
\begin{align*}
(A) &= 1001 1011 \\
(B) &= 1010 0101 \quad \text{2's complement} \rightarrow 0101 1011 \\
\text{Carry 0} &\quad (1110 0110)_2
\end{align*}
\]
• Therefore, in this case, the flag bits will be S=1, Z=0, AC=1, P=1, CY=1 (complement of ‘1’ obtained in addition).
• Thus the flag register, after the execution of the instruction, contains (1001 0101)_2 = 95H.
• Let us consider execution of another instruction DCR C. DCR is the mnemonic for decrement register. C register is the operand.
• This instruction means decrement the content of the C register by ‘1’ and
  store it back in the C register.
• The MACRO RTL implemented is
  \[ (C) ←← C - 1 \]
• Let us suppose \((C)\) contains D2H before the execution of the instruction.
• After the execution of instruction, \((C)\) shall contains D1H and, therefore, is
  not zero.
• Therefore the flag register will contain \((1000\ 010\ 0)\)2 or 84H.
• On the other hand, if \((C)\) contains 01H just before the execution of the
  instruction DCR C.
• Just after the execution of the instruction, \((C)\) shall contain 00H.
• Since the result of the operation is ‘0’ the zero flag shall now be SET to ‘1’.
• Other flag will be affected in the normal way.
• These flag bit are utilized in many instructions for branching operations.
• During the execution of a program normally one of these bits are tested for
  TRUE & FALSE condition.
• Depending upon the condition the program branches to different paths.
• This is shown in fig.1.18

![Fig.1.18 Branching Operation Depending on Condition](image)

**REGISTER SECTION:**
• There are six 8-bit general purpose registers designated as B, C, D, E, H
  and L. All these registers are accessible to the user.
• It means their contents can be read without destroying it or some new
  data can be written into it through instructions.
• These registers constitute a register array like a small on-chip RAM with
  addressable memory location.
• Internal control signals select the register for a read or write operation.
• This means that the CPU can either load a register from the 8-bit internal
  data bus or output the register content to the internal 8-bit data bus.
• Data can also be transferred or exchanged among registers.
• In an instruction, these six registers along with the accumulator (A) is identified by a 3-bit code designated either SSS or DDD.
• Whenever SSS is used, it corresponds to source register and whenever DDD is used, it corresponds to destination register.
• The address codes used for these internal registers are as follows.

<table>
<thead>
<tr>
<th>SSS or DDD</th>
<th>Address Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>(B)</td>
</tr>
<tr>
<td>001</td>
<td>(C)</td>
</tr>
<tr>
<td>010</td>
<td>(D)</td>
</tr>
<tr>
<td>011</td>
<td>(E)</td>
</tr>
<tr>
<td>100</td>
<td>(H)</td>
</tr>
<tr>
<td>101</td>
<td>(L)</td>
</tr>
<tr>
<td>110</td>
<td>(M)</td>
</tr>
<tr>
<td>111</td>
<td>(A)</td>
</tr>
</tbody>
</table>

• **Note:** In the above codes (110) is assigned to memory pointer or M register. Whenever it is used for SSS or DDD it means a specific register pair (H,L) together forms a 16 bit register known as memory address register (MAR) or M-pointer. In other words, whenever M is used in an instruction, it is assumed the 16-bit address of memory location, being referred, is available in (H,L) register pair.
• **As an example** consider the instruction MOV r1, r2
• This is an ALP statement. MOV is the mnemonic for move operation.
• r1 and r2 are the operand registers.
• In this statement r2 is the source register and r1 is the destination register.
• The meaning of the instruction is ‘move the contents of r2 register into r1 register.
• Symbolically this basic operation can be described by a macro RTL statement:

\[(r1) \leftarrow (r2)\]

• This is a single byte instruction and this single byte is the operation code.
• The arrangement of the operation code (op-code) of is shown below:

\[
\begin{array}{cccccc}
D & D & D & S & S & S \\
\end{array}
\]

• E.g., the op-code for MOV A, H is \((01 111 100)_{2} = 7CH\).
• When the instruction 7CH is executed the content of (H) register is transferred to (A) register.
• Note that the content of (H) register is not destroyed.
• However, the original content of register (A) is lost.
• Let us consider another instruction MOV D, M.
• This is also a single byte instruction.
• Memory pointed by (H,L) register pair is the source operand and (D) is the destination register.
• The meaning of this instruction is move the content of the memory location whose address is available in (H,L) register pair into the (D) register.
• The opcode of this instruction is (01 010 110)2 = 56H.
• Whenever this instruction is executed, the content of the memory location pointed by (H,L) register pair is loaded into the (D) register.
• The content of the memory location is not destroyed.
• However, the content of the memory location Y1Y0 H whose address is X3X2X1X0 H available in (H,L) pair goes into the D register.
• The original content of D is lost. This is illustrated in fig.1.19.

![Diagram](image)

- The six general purpose registers B, C, D, E, H, L can also be combined together as register pairs for 16-bit operation only the following pairs are possible:
  i. (B,C) pair with (C) lower order 8-bits and (B) higher order 8-bits.
  ii. (D,E) pair with (E) lower order 8-bits and (D) higher order 8-bits.
  iii. (H,L) pair with (L) lower order 8-bits and (H) higher order 8-bits.
• There is another register called stack pointer (SP) which is 16-bit register itself.
Whenever an instruction refers to the register pair (B,C), (D,E), (H,L) or (SPH,SPL), an 8-bit code RP is used in the operation code to identify the register pairs.

\[
\begin{array}{c|c}
0 0 & (B,C) \\
0 1 & (D,E) \\
1 0 & (H,L) \\
1 1 & (SPH,SPL)
\end{array}
\]

**PROGRAM COUNTER:**

- This is a 16-bit register accessible to the user.
- It is a special purpose register and it always contains the address of the next instruction to be fetched from the program memory and executed by the CPU in a program sequence.
- Thus the program counter keeps the track of the program execution in which instructions are to be executed next.
- Whenever necessary in the program execution, the address information available in PC is sent out to the address lines during T\(_1\) timing slot of a machine cycle.
- The higher order 8-bits of program counter (PCH) are sent out through A\(_{15}\)–A\(_8\) address lines & the lower order 8-bits of program counter (PCL) are sent out through AD\(_7\)–AD\(_0\) lines during T\(_1\) states.
- Since the BDB contains the lower order 8-bit address information during T\(_1\) state only, an ALE pulse is also issued by the processor.
- The above statement can be symbolically stated through macro RTL shown in the figure.

\[
\begin{align*}
T_1: & \quad A_{15}-A_8 \leftarrow (PCH), \ AD_7-AD_0 \leftarrow (PCL), \ ALE = \ \_\_\_\_ \\
T_2: & \quad (PC) \leftarrow (PC) + 1
\end{align*}
\]

- Whenever the address information sent from the program counter to the address bus (external world) during T\(_1\) state, then the (PC) shall be incremented by 1 during the subsequent T\(_2\) state so that program counter points to the next sequential byte.
- If may be the data required if the previous instruction is of two bytes or three bytes long or it may be the next instruction to be fetched and executed.
- If instructions are sequentially arranged in memory, this will guarantee that they will also be executed sequentially.
- Sometimes, program execution requires that non-sequential instructions executed e.g. JMP or CALL type instructions.
• These instructions require the program counter to be loaded with an entirely new value.
• An 8-bit microprocessor with a 16-bit program counter requires two data moves to completely modify the contents of the PC.
• **Note:** If the address information for PC has not been sent out during T state to the external world, then the PC will not be incremented using T2 state.
• When the microprocessor is RESET, the CPU initializes the PC to 0000 H.
• Therefore, the first instruction of the program should be at 0000 H in the memory address space of the CPU.

**STACK POINTER REGISTER:**

• The stack is a storage area of the processor.
• **It consists of number of sequential and RWM locations in which microprocessor saves the internal register contents during subroutine calls and interrupts so that they will not be changed or destroyed by a subroutine.**
• 8085A µµ can address directly 64K memory locations.
• This is known as directly addressable memory space starting from the address 0000H to FFFFH.
• This entire memory area is usually divided by the user into program area, data area and stack area.
• It is for the user to see that program area and data area do not overlap with that of stack memory area.
• The size of the stack memory area depends upon the application.
• For example, the user for a particular process control operation may decide to reserve memory space starting from 2600H to 2700H as the stack memory space. This is shown in fig.1.20.
• The stack pointer is a 16-bit register accessible to the user.
• It is required to refer any memory location of the stack.
• It contains the address of the top of stack into which last data is put or written.
• Writing data into a stack is called a PUSH operation and reading data from a stack is called a POP operation.
• In the figure shown 2700H is known as the bottom of the stack.
• There is an instruction in the instruction set to initialize the stack pointer register to the bottom of the stack.
• This instruction is
  
  LXI SP, BADR.
  
  • LXI is the mnemonic for Load immediate.
  • BADR is a symbolic name given to the 16-bit address which is to be loaded into the stack pointer.
  • The meaning of the instruction is to load the 16-bit of data immediately available in the instruction itself into the stack pointer.
  • In this example, BADR equals 2700 H.
  • When this instruction is executed the situation is shown in figure.
  • The stack pointer now points to the bottom of the stack.
  • Now, let us suppose that while calling a subroutine it becomes necessary to save the contents of (B,C) register pair and (D,E) register pair as they are to be used in the subroutine.
  • The process of saving the content of a register is known as push operation.
  • The push operation is performed at the beginning of a subroutine to save register contents and the instruction for pushing the contents of the internal register is PUSH e.g. PUSH B.
  • The meaning of the instruction is to push the contents of (B,C) register pair on to the stack so that it can be saved there till it is restored.
  • PUSH B operation affects the stack and stack pointer as follows:
    • Since the stack pointer always holds the address of the last byte of data pushed onto the stack, therefore, when PUSH B instruction is executed, the stack pointer is decremented by 1 and the contents of the (B) register are copied onto the stack at that address.
    • The stack pointer is decremented again, and the contents of the (C) register are copied to that address.
• Just after the execution of the PUSH B instruction, the situation is shown in fig 1.21.

![Fig.1.21](image)

• Similarly, to store the contents of (D,E) register pair PUSH D instruction is used.

• The meaning of this instruction is push the contents of the (D,E) pair onto the stack to save them there as shown in figure 1.22 just after the execution.

![Fig.1.22](image)

• Since the contents of (B,C) & (D,E) register pairs are stored at the top of the stack, these registers are now available for further computation in the subroutine.

• At a later stage of execution of the program after utilizing B, C, D, E registers, there may be a need to restore the original contents to the respective registers.

• E.g. at the end of the subroutine, the data is restored to the proper register.
• The restoration of the contents is a READ operation from the stack and is known as POP operation.
• A POP register instruction copies the stored data from the stack back into the indicated register pair.
• Just before the execution of POP instruction, let us say the situation is as shown in fig 1.23:

![Fig.1.23]

• Note that registers (B), (C), (D) and (E) have some different contents because these registers are used in the subroutine.
• To restore the contents of (B,C) register pair, POP B instruction is used.
• Whenever this instruction is executed, the contents from the top of the stack are read and written into the (B,C) register pair.
• To restore the contents (D,E) register pair POP D instruction is used.
• The question is in which sequence these instructions are to be executed so that the contents are restored properly.
• The obvious sequence in POP D first & then POP B i.e., the data must be popped off in the reverse order from which it was pushed.
• This type of stack is called Last-in-First-out (LIFO) memory.
• Just after the execution of POP D & POP B instructions, the situation is as shown in figure 1.24:
• When POP D instruction is executed, the data from the top of the stack is copied to register (E), data pointer is incremented by 1, then the next byte of the saved data is copied from the stack to the register (D), and SP is further incremented by 1.
• This is similar to earlier status (before PUSH operation) but now some data has been stored in the stack area but these are irrelevant anyway.
• They will be destroyed during the next PUSH operation on the stack.
• From the above discussion, following points emerge:
  o The stack pointer always points to the top of the stack up to which it is full with relevant data.
  o Storing or saving the data from the registers on stack is known as PUSH operation.
  o The restoring or reading data from the stack onto certain internal registers is known as POP operation.
  o The stack operates on Last-in-first-out (LIFO) basis.
  o The stack pointer can be initialized to the bottom of the stack but bottom of the stack cannot be utilized to store any useful data.
  o It is for the user to see that the program area does not overlap with stack area.

W-Z:
• (W) and (Z) are two 8-bit temporary registers not accessible to the user.
• They are exclusively used for the internal operation by the microprocessor.
• These registers are used either to store 8-bit of information in each (W) and (Z) registers or a 16-bit data in (W,Z) register pair with lower order 8-bits in (Z) and higher-order 8-bits in (W) register.
• When a 3-byte instruction containing 2-byte address is to be executed by the \( \mu P \), the first byte is the (op-code byte) which is fetched and then decoded by the decoder.

• Then two memories read machine cycles are executed one by one to read the two-byte address, one in each machine cycle and placed in \((W, Z)\) register pair.

• During instruction execution, in next machine cycle, the address in \((W, Z)\) register pair is transferred to the address latch to address memory or I/O for data transfer.

**Increment-Decrement Address Latch:**
- It is another 16-bit internal register latch available in the register section for internal operations and is not accessible to the user.
- The address latch serves two functions.
  - It selects an address to be sent out from the program counter, from the stack pointer, or from one of the 16-bit register pairs.
  - It latches this address onto the address lines for the required time.
- The 16-bit addresses from 8085A allow the microprocessor up to \(2^{16}\) memory locations through A15-A8 and AD7-AD0 lines.
- An increment/decrement register allows the contents of any of the 16-bit registers to be incremented or decremented.

**Instruction Register & Instruction Decoder:**
- The first word of an instruction is the operation code, i.e., binary code for that instruction.
- Therefore, in the first machine cycle of any instruction \( \mu P \) fetches the instruction from the memory.
- The op-code representing the instruction to be executed is fetched from the (program) memory location pointed to by (PC) and loaded into the instruction register (IR).
- The IR passes this op-code to the instruction decoder which interprets this op-code appropriately in order to decide what operation needs to be done for executing this instruction.
- The instruction decoder tells the control unit the type of instruction to be executed; the number of machine cycles necessary to execute the instruction etc.
- In response, the control unit generates all the necessary control signals which go into the different internal block of the microprocessor.
- These different control signals are generated by what is known as Micro-programming technique.
• Micro-programming means the microprocessor instruction decoding operated like a small version of a µp itself.
• As the µp goes through the fetch and execute cycles, the microprogramming logic goes through a series of fetch and executes cycles.
• E.g. if the instruction is ADI 04H, then the first binary code read by the µp is C6H into the (IR).
• After decoding this, the decoder will recognize that another memory read cycle is required to read 04H to be added to the number in the accumulator.
• The decoder will direct the control circuit to send out another memory read pulse and transfers the data coming on the data bus into the temporary register (Temp), so that it can be added to the accumulator.
• When the addition is completed the control circuit directs the result back to the accumulator.
• The program counter is then incremented to point the next memory address and send out another memory read pulse to read the µp code of next instruction from memory.

❖ Interrupt Control Section:

• Sometimes it is necessary to interrupt the execution of the main program to answer a request from an I/O device.
• For instance, an I/O device may send an interrupt signal to interrupt control unit to indicate that data is ready for input.
• The µp temporarily stops what it is doing, inputs the data and then returns to what it was doing.
• To enable the processor to service the device requesting service through interrupt, processor accepts and issues control signals through interrupt control section.

❖ Serial I/O Control:

• Sometimes, I/O devices work with serial data rather than parallel.
• In this case, the serial data stream from an input device must be converted to 8-bit parallel data before the computer can use it.
• Likewise the 8-bit data out of a processor must be converted to serial form before a serial output device can use it.
• The SID (Serial Input Data) input is where serial data enters the 8085A.
• The SOD (Serial Output Data) output is where the serial data leaves the 8085A.
• Two instructions known as SIM & RIM allow the user to perform the serial parallel conversion needed for serial I/O device.

❖ Timing and Control section:
The timing and control section supervise the complete operation of the \( \mu P \).

The on-chip clock oscillator which produces the internal clock is a part of this section.

The timing and control section also has a state generator circuit to generate 10 different states namely \( T_1, T_2, T_3, T_4, T_5, T_6, \) TRESET, \( T_{HALT}, T_{WAIT} \) and \( T_{HOLD} \).

State generator is a multi-mode counter.

The next state of the state generator from the present state is decided by many of the control signals input like READY, HOLD, Interrupt control signals - TRAP, RST7.5, RST6.5, RST5.5 and INTR.

In each state this section generates many control signals for executing the instruction fetched.

The operation of the \( \mu P \) is cyclic in natural. During the normal operation from the word GO, \( \mu P \) sequentially fetches and executes one instruction after another until a HALT instruction is executed.

The fetching and execution of a single instruction constitutes an instruction cycle.

The instruction cycle consists of one or more read or write operation to memory or an I/O device.

Each memory and I/O reference requires a mechanic cycle.

In other words every time a byte of data is move from CPU to I/O or memory or from memory or I/O to CPU, a machine cycle is required.

There are seven different kinds of machine cycles in the 8085 A:

- Opcode Fetch Machine Cycle (OFMC)
- Memory Read Machine Cycle (MRMC)
- Memory Write Machine Cycle (MWRMC)
- I/O Read Machine Cycle (IORDMC)
- I/O Write Machine Cycle (IOWRMC)
- Interrupt Acknowledge Machine Cycle (INTAMC)
- Bus Idle Machine Cycle (BIMC)

Three status signals \( IO/M, S1 \) and \( S0 \) generated at the beginning of each machine cycle and \( RD, WR \) and \( INTA \) generated during T2 state of the machine cycle identify each type of the machine cycle.

The status signals remain valid for the entire duration of the cycle.

The instruction fetch portion of an instruction cycle requires a machine cycle for each byte of the instruction to be fetched.

Since an instruction consists of 1 to 3 bytes (1, 2 or 3), the instruction fetch is one to three machine cycles in duration.

The first machine cycle of an instruction cycle is always an_OPCODE FETCH machine cycle which is always single byte long and the 8-bits
obtained during an OPCODE FETCH are always interpreted as an OPCODE of an instruction.

- Note that to fetch an instruction, i.e., to transfer an entire instruction from memory to the \( \mu p \) necessitates an OPCODE FETCH machine cycle.
- However, one or two memory read machine cycles are also needed to complete the fetch for 2nd and 3rd bytes of the instruction respectively.
- The number of machine cycles required to execute the instruction depends on the particular instruction.
- Some of the instructions require no additional machine cycles after the instruction fetch is complete, other requires additional machine cycles to write or read data to or from memory or I/O devices.
- The total number of machine cycles required varies from one to five.
- Around 50% of the instructions require only one machine cycle for fetching and executing the instruction.
- No instruction requires more than five machine cycles.
- Machine cycles like the memory read or memory write may occur more than once in a single instruction cycle.

<table>
<thead>
<tr>
<th>MC-1</th>
<th>MC-2</th>
<th>MC-3</th>
<th>MC-4</th>
<th>MC-5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The shaded area may be required for executing the instruction.
- The timing and control unit of \( \mu p \) automatically generates the proper machine cycles required for an instruction cycle from information provided by the op-code.
- Each machine cycle contains a number of 320ns clock periods when cryptal used is 6.25 MHz.
- One clock period, i.e. the period between two negative going transitions of that clock is called T state.
- The various T-states are \( T_1, T_2, T_3, T_4, T_5 \) and \( T_6 \). Most of the machine cycles have three T-states each \( (T_1, T_2, T_3) \).
- Only OPCODE FETCH machine cycle has either 4 or 6 states depending on the instruction.
- The first 3rd states of the machine cycle are identical to a MRMC, the additional T states in OFMC are the T-states required by the 8085A to decode the op-code and decide what actions are needed in succeeding machine cycles.
- The combined MCS along with T-states are shown in fig 1.25.
Thus one complete transition from state $T_1$ through the state diagram and back to $T_1$ constitutes a complete machine cycle.

The partial state transition diagram is shown in fig.1.26 assuming READY=1 i.e., no wait state.

The shaded portion shows that these states may be needed in some instructions.

Instruction cycles for various 8085A instructions require from 4 to 18 states.

<table>
<thead>
<tr>
<th>MC-1</th>
<th>MC-2</th>
<th>MC-3</th>
<th>MC-4</th>
<th>MC-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$T_2$</td>
<td>$T_3$</td>
<td>$T_4$</td>
<td>$T_5$</td>
</tr>
</tbody>
</table>

MC-i (i=2, 3, 4, 5)

Fig 1.25

Fig. 1.26 Partial State Transition Diagram indicating $T_1$-$T_6$ States
• The total number of states actually required to execute an instruction will depend on the READY & HOLD signal inputs.
• For example, consider the 3 byte instruction
  
  STA ADDR.

• STA stands for store accumulator direct.
• The meaning of the instruction is transfer the content of the accumulator to an external memory location whose address is specified in the instruction i.e., ADDR.
• Since this location can be anywhere in the 64k memory space that the 8085A can directly address, 16-bits are required for the address.
• Thus the instruction contains 3 bytes- a 1 byte op-code and 2-byte address.
• The instruction is stored in the memory as follows:

  | OP CODE | Byte -1 |
  | LOWER ADDR | Byte - 2 |
  | HIGHER ADDR | Byte - 3 |

• Three machine cycles (MC) are required to fetch this instruction.
• In MC-1, i.e., Op-code fetch machine cycle, the op-code is transferred from memory to the instruction register during T1-T3 states and then during T4 state it is interpreted.
• At this point, the CPU knows that it must do more machine cycles - two MRMCs to fetch the complete instruction.
• In MC-2 the lower address is transferred from the memory to the temporary register (Z).
• In MC-3 the third byte, i.e. the higher byte address is transferred from the memory to the temporary register (W).
• When the entire instruction is in the \( \mu P \), it is executed.
• Execution means a data transfer from the \( \mu P \) to memory.
• The content of the accumulator is transferred to the memory location, whose address was previously transferred to the \( \mu P \) by the proceeding two memory read machine cycles.
• The address of the memory location to be written is generated as follows:
  • The high order address byte in temp register (W) is transferred to the address latch and the low order address byte in temp register (Z) is transferred to address/data latch.
  • The content of the (A) is then placed on the data bus.
• This data transfer is affected by a MWRMC.
• Thus 3-byte STA instruction has four machine cycles in its instruction cycles.
The actions taken by the processor in different machine cycles are shown in fig.1.27.

Thus STA ADDR instruction has a total of 13 states.

If the 8085A is operating at 325.5ns time, the STA instruction cycle is executed in 4.23 µsec.

This time period is the instruction execution time, although it actually includes both the instruction fetch and the execution time.

Reference

- [https://nptel.ac.in/courses/108107029/13](https://nptel.ac.in/courses/108107029/13)
- [https://nptel.ac.in/courses/108107029/14](https://nptel.ac.in/courses/108107029/14)
- [https://nptel.ac.in/courses/108107029/15](https://nptel.ac.in/courses/108107029/15)
Read / Write Memory:

- **RAM**
  - It is used primarily for information that is likely to be altered, such as writing programs or receiving data.
  - This memory is volatile, meaning that when the power is turned OFF, all the contents are destroyed.
- Two type of R/W memories
  - Static Memory
  - Dynamic Memory

**Static Memory:**
- This memory is made up of flip-flops, and it stores the bit as a voltage.
- Each memory cell requires 6 transistors; therefore, the memory chip has low density but high speed.
- This memory is more expensive and consumes more power than the dynamic memory.
- In high speed processors, SRAM known as cache memory is included on the processor chip.
- In addition high speed cache memory is also included external to the processor to improve the performance of a system.

**Dynamic Memory:**
- This memory is made up of MOS transistor gates, and it stores the bit as a charge.
- The advantages of dynamic memory are that it has high density and low power consumption and is cheaper than static memory.
- The disadvantage is that the charge (bit information) leaks; therefore, stored information needs to be read and written again every few milliseconds.
- This is called refreshing the memory, and it requires extra circuitry, adding to the cost of the system.

Memory Interfacing:

- **RWM** is a group of registers to store binary information.
- Figure 1.31(a) shows a typical RWM chip; it has 2048 registers and each register can store eight bits indicated by eight input and eight output data lines.
- The chip has 11 address lines $A_{10}$-$A_{0}$, one chip select ($CS$), and two control lines: Read ($RD$) to enable the output buffer and Write ($WR$) to enable the input buffer.
- Figure 1.31(a) also shows the internal decoder to decode the address lines.
- Figure 1.31(b) shows the logic diagram of a typical EPROM with 4096 registers.
- It has 12 address lines $A_{11}$-$A_{0}$ one chip select ($CS$), and one Read control signal.
- This chip must be programmed (written into) before it can be used as a ROM.
Figure 1.31(b) also shows a quartz window on the chip that is used to expose the chip to UV rays for erasing the program.

Once the chip is programmed the window is covered with opaque tape to avoid accidental erasing.

For interfacing the RWM and EPROM, the process is similar; the only difference is that the EPROM does not require the \( WR \) signal.

Timing Diagram:

- **Instruction Cycle**: The time taken by the processor to complete the execution of an instruction. An instruction cycle consists of one to six machine cycles.

- **Machine Cycle**: The time required to complete one operation; accessing either the memory or I/O device. A machine cycle consists of three to six T-states.

- **T-State**: Time corresponding to one clock period. It is the basic unit to calculate execution of instructions or programs in a processor.

- **Opcode Fetch Machine Cycle**

- **Execute Cycle**: determines the total time required to decode the instruction fetched and executing.

**OPCODE FETCh Machine Cycle**:

- Figure shows the 8085A instruction fetch timing diagram.

- The instruction fetch cycle requires either four or six clock periods (Tstates).

- The other machine cycles that follow OFMC will need three clock cycles.
The purpose of an OFMC is to read the contents of a memory location containing the opcode addressed by the program counter and to place it in the instruction register (IR).

In the beginning of state $T_1$, the 8085A puts a low on the IO/$\bar{I}$ line of the system bus indicating a memory operation.

The 8085A sets $S_1=1$ and $S_0=1$ on the system bus, indicating the memory fetch operation.

This status information remains available for the duration of the machine cycle.

During $T_1$ state, the 16-bit address $A_{15}-A_0$ of the memory location containing the opcode is obtained from the program counter (PC) and placed on the address and address/data latches.

The higher order 8-bits of the address appear on the address bus $A_8-A_{15}$ remains constants until the end of the state $T_3$.

During $T_4$ state the data on the address bus is unspecified.

The low order 8-bits of the address are placed on the address/data bus, $AD_7-AD_0$ at the beginning of $T_1$.

This data however remains valid only until the beginning of state $T_2$ at which time the address/data bus is floated (tri-stated) because this is time multiplexed bus and used as the data bus during $T_2$ and $T_3$ states.

Therefore address latch enable (ALE) signal issued by the $\mu P$ during $T_1$ is used to latch this lower order address in some external latch 74LS373 on its falling edge.

The 16-bit address selects a particular memory location.

During state $T_2$, at the beginning, the $\overline{RD}$ signal goes low indicating read operation and the opcode to be fetched is placed on the data bus, $AD_7-AD_0$ by the addressed memory location.

The contents of (PC) is incremented be 1 during this state as during $T_1$ state the (PC) has sent the address to address bus.

The accessed memory should be fast enough to output its data before $\overline{RD}$ goes high.

Slower memories can gain more time by pulling the READY signal of 8085A LOW.

This will introduce an integral number of $T_{\text{WAIT}}$ states between $T_2$ and $T_3$ as long as READY is low.

On the rising edge of the $\overline{RD}$ control signal in $T_3$ state, the opcode obtained from the memory is transferred to the microprocessor instruction register.

During state $T_4$, the 8085A decodes the instruction and determines whether to enter state $T_5$ or to enter $T_1$ state of the next machine cycle.

From the operation code, the $\mu P$ determines what other machine cycles, if any, must be executed to complete the instruction cycle.
State T₅ and T₆ when entered, are used for internal \( \mu P \) operations necessitated by the instruction.

The micro RTL flow for 4-states OFMC is shown below

**OFMC:** Status signals \( IO/\overline{M}=0, \ S₁=1, \ S₂=1 \)

- **T₁:** \( A_{15}-A₈ \leftarrow (PCH), \ AD₇-AD₀ \leftarrow (PCL), \ ALE = \)
- **T₂:** \( RD = 0, \ (PC) \leftarrow (PC) +1, \ AD₇-AD₀ \leftarrow M(AB) \)
- **T₃:** \( RD = 1, \uparrow , \ (IR) \leftarrow BDB \)
- **T₄:** \( \mu P \) decodes the opcode and decides whether T₅ and T₆ states are required or next machine cycle executed is T₁

During T₂ state, after the \( \overline{RD} \) signal is made LOW, the external decoding circuit decodes the address put on the address bus during T₁ state.

One of the memory location is selected and it puts 8-bit information on the data bus during T₂ and T₃ states.

Processor has no control on it.

Processor has already issued the signals and now it is the job of the external decoding circuit to make use of the signals IO/\( \overline{M} \) and \( \overline{RD} \) and address lines to allow the external memory to put the data on the data bus.

Therefore, this action is shown by shaded area.

Whatever information is available on BDB at LOW to HIGH transition of \( \overline{RD} \), that will be read and processed.

The timing waveform during 4-state OFMC is shown in fig.1.32.
Fig. 1.32 Timing Diagram During 4-state OpCode Fetch Machine Cycle

- During $T_4$–$T_6$ states, AD7-AD0 lines are tri-stated and A$_{15}$-A$_8$ lines are unspecified.
- Fig. 1.33 shows the timing diagram for a 6-state OFMC:
Fig.1.33 Timing Diagram During 6-state OpCode Fetch Machine Cycle

Note: Whenever the address information is sent from the program counter to the external world during T₁ state, then the (PC) is incremented by 1 during the subsequent T₂ state so that PC points to the next subsequent byte. However, if the address information from (PC) has not been sent out during the T₁ state to the external world, then (PC) will not be incremented during T₂ state.

Memory READ Machine Cycle:

- It requires 3 states T₁ to T₃.
- The purpose of the memory READ operation is to read the contents of a memory location addressed by a register pair and place the data in one of internal registers of the μP.
- The source of address issued during T₁ is not always the program counter but may be any one of the several other register pairs in the μP depending on the particular instruction of which the machine cycle is a part.
- The 8085A uses machine cycle MC-1 to fetch and decode the instruction.
- It then performs the memory read operation in MC-2. E.g. in LXI H, Addr.
- The IO/M signal is made LOW to indicate the external world that a memory reference is required.
- Then μP made S₀=0 and S₁=1 indicating that memory READ operation is to be performed.
During T₁, the μP places the contents of higher byte of the memory address register, such as that contents of the (PCH) or (H) register on A₁₅-A₈ and the contents of the lower byte of the memory address register such as contents of the (PCL) or (L) register on AD₇-AD₀.

- The μP sets ALE signal HIGH indicating the beginning of MC-2.
- As soon as ALE goes to LOW in the middle of T₁, the lower byte of the address is latched in an external latch.
- The same bus is now going to be used as data bus.
- During T₂ state, the RD signal goes LOW indicating a READ operation.
- If the address sent out during T₁ state is from (PC), then (PC) is incremented by 1 otherwise not.
- The external logic gets the data from the memory location addressed by the memory address register such as (H,L) pair and places the data on to bi-directional data bus AD₇-AD₀.
- During T₃ state, RD signal goes HIGH.
- This LOW to HIGH transition of signal transfers the data from the data bus to internal register such as the accumulator.
- The timing diagram during memory ready machine cycle is shown in fig.1.34.

![Timing Diagram During Memory Read Machine Cycle](image)

**Memory WRITE Machine Cycle:**
- It also requires only T₁ to T₃ states.
The purpose of memory write machine cycle is to store the contents of any of the
8085A register such as the accumulator into a memory location addressed by a
register pair such as (H,L).
The 8085A \( \mu P \) made IO/M = 0 in the beginning of \( T_1 \) state to indicate memory
reference operation.
Then it puts \( S_0 = 1 \) and \( S_1 = 0 \) indicating a memory write operation.
During \( T_1 \) state 8085A places the memory address register (MAR) higher byte
such as the contents of the (H) register on lines \( A_{15}-A_8 \) and also places the MAR
lower byte such as the contents of the (L) register on lines \( AD_7-AD_0 \).
The \( \mu P \) sets ALE signal HIGH indicating the beginning of MWRMC. As soon as
ALE goes to low, the lower byte of the address is latched in an external latch.
During \( T_2 \) state, WR goes LOW indicating memory write operation.
It also places the contents of the internal register, say accumulator, on data lines
\( AD_7-AD_0 \).
During \( T_3 \) state, \( \overline{WR} \) goes HIGH.
This LOW to HIGH transition is used to transfer the data from the data lines to
the memory location address by MAR such as (H,L) register pair.

**MWRMC:** Status signals  IO/M=0, \( S_1=0, \ S_3=1 \)

\[
\begin{align*}
T_1: & \quad A_{15}-A_8 \leftarrow (H), \quad AD_7-AD_0 \leftarrow (L), \quad ALE = \_\_\_\_
T_2: & \quad \overline{WR} = 0, \quad AD_7-AD_0 \leftarrow (\mu P \ Internal \ Reg.)
T_3: & \quad \overline{WR} = 1, \uparrow, \quad M(AB) \leftarrow AD_7-AD_0 \ or \ BDB
\end{align*}
\]

Similar to MRMC, the processor simply puts the data on the data bus and makes
required signals LOW or HIGH.
It is the job of the external decoding circuit to make use of these signals to
enable the external memory to accept the data from the data bus.
Processor has no control over it.
Therefore, this action during \( T_3 \) state is shown shaded.
The timing diagram during MWRMC is shown in fig.1.35:
I/O READ and I/O WRITE M/C cycle:

- The IORDMC and IOWRMC are identical to MRMC & MWRMC respectively except that appropriate status signals are issued at the beginning of T₁ state.
- IO/ᵜ signal goes HIGH at the beginning to indicate I/O device reference is needed in case of I/O mapped input/output device.
- In these machine cycles higher & lower address bytes are identical and equal to the 8-bit address of the I/O port while in case of MRMC or MWRMC, the address bus output is the true 16-bits address.
- These machine cycles will be discussed in detail along with I/O techniques.

Memory Interfacing Circuit:

- The process of address decoding should result in identifying a register for a given address.
- We should be able to generate a unique pulse for a given address.
- For example, in figure 1.31 (b), 12 address lines (A₁₁₋ₐ₀) are connected to the memory chip, and the remaining four address lines (A₁₅₋ₐ₁₂) of the 8085 microprocessor must be decoded.
Fig 1.36 Address Decoding using (a) NAND Gate and (b) to 8 Decoder

- Figure 1.36 shows two methods of decoding these lines
  - NAND gate
  - Using a 3 to 8 Decoder

- Output of NAND goes active and selects the chip only when all address lines $A_{15}$-$A_{12}$ are at logic 1.

- The same result can be obtained with 3 to 8 decoder, which is capable of decoding eight different input address.

- In the decoder circuit, three input lines can have eight different logic combinations from 000 to 111; each input can be identified by the corresponding output line if Enable lines are active.

- In this circuit, the enable lines $E_1$ and $E_2$ are enable by grounding and $A_{15}$ must be at logic 1 to enable $E_3$.

- Figure 1.37 shows an interfacing circuit using a 3 to 8 decoder to interface the 2732 EPROM memory chip.

- It is assumed here that the chip has already been programmed, and we will analyze the interfacing circuit in terms of the same three steps outlined.
Step 1: The 8085 address lines $A_{11}-A_0$ are connected to pins $A_{11}-A_0$ of the memory chip to address 4096 registers.

Step 2: The decoder is used to decode four address lines $A_{15}-A_{12}$. The output $O_0$ of the decoder is connected to Chip Enable (CE). The CE is asserted only when the address on $A_{15}-A_{12}$ is 0000; $A_{15}$ (low) enables the decoder and the input 000 asserts the output $O_0$.

Step 3: For this EPROM, we need one control signal: Memory Read (MEMR), active low. The MEMR is connected to OE to enable the output buffer; OE is the same as RD in figure 1.31.

**Address Decoding and Memory Addresses:**

$$\begin{align*}
A_{15}A_{14}A_{13}A_{12} & \quad A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0 \\
0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0 & \quad 0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0\hspace{1em}0 = 0000H \\
0\hspace{1em}0\hspace{1em}0\hspace{1em}0 & \quad 1\hspace{1em}1\hspace{1em}1\hspace{1em}1\hspace{1em}1\hspace{1em}1\hspace{1em}1\hspace{1em}1\hspace{1em}1\hspace{1em}1\hspace{1em}1\hspace{1em}1\hspace{1em}1 = 0FFFH \\
\downarrow & \quad \downarrow \\
\text{Chip Enable} & \quad \text{Register Select}
\end{align*}$$
Interfacing RWM

Figure 1.38 shows interfacing of the 6116 memory chip with 2048 (2K) registers.

- The memory chip requires 11 address lines to decode 2048 registers.
- The remaining address lines are connected to decoder.
- However in this circuit the decoder is enabled by IO/M signal in addition to the address lines A15 and A14, and the RD and WR signals of the 8085 are directly connected to the memory chip.
- The signals MEMR and MEMW need not be generated separately; thus this techniques saves two gates.
Fig 1.38 Interfacing RWM

I/O Devices

- There are two types of peripheral device mapping they are
  - Peripheral mapped I/O [I/O mapped I/O]
  - Memory Mapped I/O

- I/O mapped I/O
  - The µP used as 8 bit address lines to identify the input and output devices. This is known as I/O mapped I/O
  - Control lines: I/O READ and I/O WRITE

- Memory Mapped I/O:
  - The µP used as 16 bit address lines to identify an I/O device. This is known as Memory Mapped I/O
  - Control lines: MEMR and MEMW

Reference:
- https://nptel.ac.in/courses/108107029/16
- https://nptel.ac.in/courses/108107029/17
Harvard vs. Princeton Architecture

- Many years ago, in the late 1940’s, the US Government asked Harvard and Princeton universities to come up with a computer architecture to be used in computing distances of Naval artillery shell for defense applications.
- Princeton suggested computer architecture with a single memory interface. It is also known as Von Neumann architecture after the name of the chief scientist of the project in Princeton University John Von Neumann (1903 - 1957 Born in Budapest, Hungary).
- Harvard suggested a computer with two different memory interfaces, one for the data / variables and the other for program / instructions. Although Princeton architecture was accepted for simplicity and ease of implementation, Harvard architecture became popular later, due to the parallelism of instruction execution.

Princeton Architecture (Single memory interface)
- 80x86 processors and ARM7 have Princeton architecture for main memory.
- Vectors and pointers, variables, program segments and memory blocks for data and stacks have different addresses in the program in Princeton memory architecture.
- Example: An instruction "Read a data byte from memory and store it in the accumulator" is executed as follows: -
  - Cycle 1 - Read Instruction
  - Cycle 2 - Read Data out of RAM and put into Accumulator

Harvard Architecture (Separate Program and Data Memory interfaces)
- 8051-family microcontrollers have Harvard architecture.
- A processor having Harvard main memory architecture has distinct address spaces, control signals, processor instructions and data paths for the bytes for data and for program.
- It helps in handling streams of data that are required to be accessed in cases of single instruction multiple data type instructions and DSP instructions.
- Separate data buses ensure simultaneous accesses for instructions and data.
- Program segments and memory blocks for data and stacks have separate set of addresses in Harvard architecture.
- Control signals and read-write instructions are also read-write instructions are also separate for accessing the program memory and data memory.
The same instruction (as shown under Princeton Architecture) would be executed as follows:

- **Cycle 1**
  - Complete previous instruction
  - Read the "Move Data to Accumulator" instruction

- **Cycle 2**
  - Execute "Move Data to Accumulator" instruction
  - Read next instruction

Hence each instruction is effectively executed in one instruction cycle, except for the ones that modify the content of the program counter.

For example, the "jump" (or call) instructions takes 2 cycles.
• Thus, due to parallelism, Harvard architecture executes more instructions in a given time compared to Princeton Architecture.

Reference:
Pipelining in Microprocessor:

- Pipelining is widely used in modern processors to improve system performance in terms of throughput.
- Pipelined organization requires sophisticated compilation techniques
- Making execution faster:
  - Use faster circuit technology to build the processor and the main memory.
  - Arrange the hardware so that more than one operation can be performed at the same time.
  - In the latter way, the number of operations performed per second is increased even though the elapsed time needed to perform any one operation is not changed.

- Pipeline
  - It is a technique of decomposing a sequential process into sub-operation, with each sub-operation completed in dedicated segment.
  - Pipeline is commonly known as assembly line operation.
  - It is a series of stages, where some work is done at each stage in parallel.
  - The stages are connected one to the next to form a pipe instructions enter at one end, progress through the stages, and exit at the other end.
  - Analogy - 1
    - It is similar like assembly line of car manufacturing [Chasis, Engine, Fitting body, painting…]
  - Analogy- 2
    - Laundry example, Washer A takes 30 minutes, Dryer B takes 40 minutes and Folder C takes 20 minutes for one load of clothes.
Pipelining in Processor

- **Definition**: It is a speed up technique where multiple instructions are overlapped in execution on a processor.
- The processor executes the program by fetching and executing instructions. One after the other.
- Let $F_i$ and $E_i$ refer to the fetch and execute steps for instruction $I_i$

**Fetch + Execution**

![Diagram of Instruction Pipelining]

Fig 1.29 Basic idea of Instruction Pipelining
- Computer that has two separate hardware units, one for fetching and another for executing them.
- The instruction fetched by the fetch unit is deposited at an intermediate buffer B1.
- This buffer needed to enable the execution unit while fetch unit fetching the next instruction.
- Figure 1.29 (c), the computer is controlled by a clock, any instruction fetch and execute steps completed in one clock cycle.

![Diagram of the pipeline stages](image)

**Fetch + Decode + Execution + Write**

<table>
<thead>
<tr>
<th>Clock cycle</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction</td>
<td></td>
<td>F₁</td>
<td>D₁</td>
<td>E₁</td>
<td>W₁</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>F₂</td>
<td>D₂</td>
<td>E₂</td>
<td>W₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>F₃</td>
<td>D₃</td>
<td>E₃</td>
<td>W₃</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>F₄</td>
<td>D₄</td>
<td>E₄</td>
<td>W₄</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) Instruction execution divided into four steps

(b) Hardware organization

**Fig 1.30 4-stage Pipeline**

- Fetch (F) – Read the instruction from memory
- Decode (D) – Decode the instruction and fetch the source operand
- Execute (E) – Perform the operation specified by the instruction
- Write (W) – Store the result in the destination location

- In 6 stage pipeline
  - Fetch Instruction (FI) – Read the instruction from memory
  - Decode Instruction (DI) – Decode the instruction and sending out the various control lines to the other parts of the processor
  - Calculate Operands (CO) – in this stage where any calculations are performed. The main component in this stage is ALU.
- Fetch Operand (FO) & Execute Instructions (EI) – Responsible for storing and loading values to and from memory. They also responsible for input and output from the processor respectively.
- Write Result (WR) – Store the result in the destination location

- The potential increase in performance resulting from pipelining is proportional to the number of pipeline stages.
- However, this increase would be achieved only if all pipeline stages require the same time to complete and there is no interruption throughout program execution.

**Advantages**
- Pipelining makes efficient use of resources
- Quicker time of execution of large number of instructions
- The parallelism is invisible to the programmer

**Reference:**
- [https://www.slideshare.net/siddiqueibrahim37/pipelining-41608675](https://www.slideshare.net/siddiqueibrahim37/pipelining-41608675)
- [https://www.slideshare.net/SaidurRahmanKohinoor/instruction-pipeline-computer-architecture](https://www.slideshare.net/SaidurRahmanKohinoor/instruction-pipeline-computer-architecture)