Microprocessor

Date
Page

UNIT - I

Microprocessor

"A microprocessor is a multipurpose, programmable, clock-driven, register-based electronic device that read binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions and provides result as output."

Microprocessor Evolution and Types

→ Historical background (Advances in semiconductor technology)

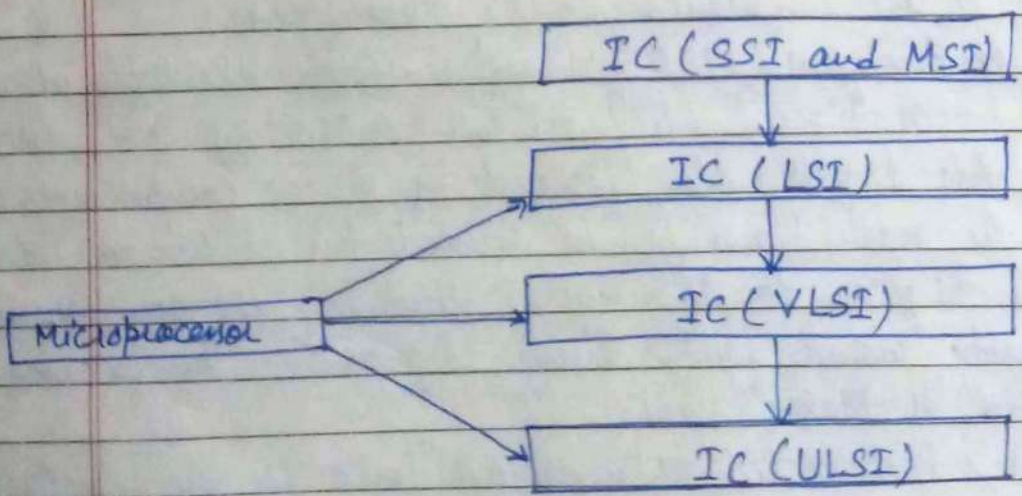
- The invention of the transistor and its commercial availability, signifying the overpowering of the transistor over its rival - the vaccum tube.
- The tremendous success of the transistor led to vigorous research activity in the field of microelectronics. In late 1950s the realization of a complete electronic circuit having a number of devices and interconnecting them on a single silicon wafer was possible and known as Integrated Circuit (IC).
- In 1965, Gordon Moore concluded that component density in an IC would double every year.

Later in the year 1975, Moore revisited the topic at IEEE International Electron Devices Meeting and observed that his 10 years old forecast of 65000 components was on the mark. However, he revised his prediction rate from 01 year to 18 months, that is, the components density would double every 18 months. This became known as Moore's law.

Moore's law acted as a driving force for the spectacular development of IC technology leading to different types of products.

- Based on the scale of integration, the IC technology can be divided into five different categories:

- The first half of the 1960s was the era of small-scale integration (SSI), with about ten transistors on a chip. The SSI technology led to the fabrication of gates and flip-flops.
- In the second half of the 1960s, counters, multiplexers, decoders and adder were fabricated using the medium-scale integration (MSI) technology having 100 - 1000 components on a chip.
- In the 1970s, large scale integration (LSI) technology with 10000 - 20000 components on a chip producing typical products like, 8-bit microprocessor, RAM, ROM.
- The 1980s was the era of VLSI with about 20000 - 50000 components on a chip. Typical products were 16-bit and 32-bit microprocessors.
- Beyond 1985 is the era of ultra large scale integration (ULSI) with more than 50000 devices on a chip. ULSI led to the fabrication of digital signal processors (DSPs), RISC processors, etc.



→ Evolution of Microprocessors

- Microprocessor is the greatest invention of the 20th century.
- Intel was the first Microprocessor unit producer and has been holding a large share of the world market for this product.
- The evolution of the microprocessor is categorized into five generations:

- First Generation (1971-1973)

- First generation microprocessors processed their instruction serially.
- Intel corporation introduced 4004, the first microprocessor in 1971. It is evolved from development effort while designing a calculator chip.
- There were three other microprocessors in the market during the same period Rockwell's International's PPS-4 (4 bit), Intel's 8008 (8-bits), National semiconductor's INP-16 (16 bit)
- They were fabricated using PMOS technology which provided low cost, slow speed and low output currents. They were not compatible with TTL.

- Second Generation (1974-1978)

- The second generation marked the beginning of very efficient 8-bit microprocessors.
- Some of the popular processors were Intel's 8085, Motorola's 6800, Zilog's Z80.
- They were manufactured using NMOS technology. This technology offered faster speed and higher density than PMOS.
- It is TTL compatible.

- Third Generation (1978-1980)

- The third generation introduced in 1978, was dominated by 16-bit processors (such as Intel's 8086 and Zilog's Z8000) with minicomputer like performance.
- These processors had the technology of 16-bit arithmetic and

pipelined instruction processing.
- They were designed using HMOS technology.
- HMOS provides some advantages over NMOS, as its speed-power-product is four times better than that of NMOS. HMOS can accomodate twice the circuit density compared to NMOS.

- Fourth Generation (1981-1995)
- The microprocessor entered into their fourth generation with designs containing more than a million transistors in a single package.
- This era marked the beginning of 32-bit microprocessors.
- Intel introduced 80386 and Motorola introduced 68020/68030.
- They were fabricated using high density / high speed complementary metal-oxide semiconductor (HCMOS), a low power version of the HMOS technology.

- Fifth Generation (1995 - Till date)
- The fifth generation microprocessors design contains more than 10 million transistors.
- In this age the emphasis is on introducing chips that carry on-chip functionalities and improvements in the speed of memory and I/O devices along with introduction of 64-bit microprocessors.
- Intel leads the show here with Pentium, Celeron and very recently dual and quad core processors working with up to 3.5 GHz speed.

→ Types of microprocessors (classification of Microprocessors)

- Microprocessors can be classified based on their specification, applications, and architecture.
- Based on the size of the data that the microprocessor can handle:
  * Microprocessors are classified as 4-bit, 8-bit, 16-bit, 32-bit and 64-bit microprocessors.
- Based on the application of the processors: ~~can handle~~
  * Microprocessors are classified as follows:
    - General purpose processors
    - Microcontrollers (system on chip)
    - special purpose processors

- General purpose processors

Processors that are used in general computer system integration and can be used by the programmer for any application are known as General purpose processors.
Common microprocessors ~~for~~ from Intel 8085 to Intel Pentium are example of general purpose prepocess.

- MicroControllers

These are microprocessors chips with built in hardware for the memory and ports. These chips can be programmed by the user for any generic control application.

- special purpose processors

These processors are designed specifically to handle special functions required for an application. Digital signal processors are examples of special purpose processors; these have special instructions to handle signal processing. Application specific Integrated circuit (ASIC) chips are also ~~ex~~ examples of this category of microprocessors.

- Based on the architecture and hardware of the processors:
  - RISC processors
  - CISC processors

- VLIW processors
- Superscalar processors

- **RISC processors**
- RISC is a processor architecture that supports limited machine language instructions.
- RISC processors can execute program faster.
- eg: IBM RS6000, SPARC

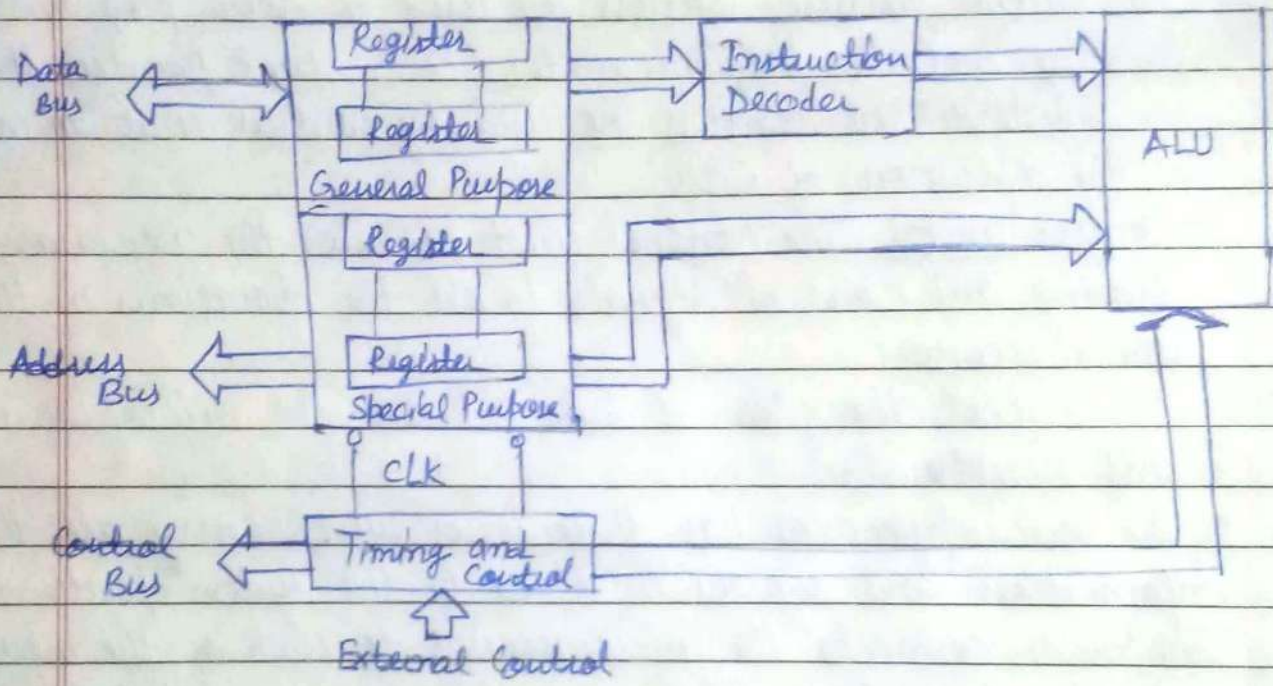
- **CISC processors**
* CISC processors have about 70 to a few hundred instructions and are easier to program.
* CISC processors are slower and more expensive than RISC.
eg: Intel x86, Pentium

- **VLIW (Very Large Instruction Word)**
VLIW processors have instructions composed of many machine operations. These instructions can be executed in parallel. This parallelism is called instruction level parallelism.
These processors have a large number of registers.
eg: Intel i860.

- **Superscalar processors**
Superscalar processors use complex hardware to achieve parallelism. It is possible to have overlapping of instruction execution to increase the speed of execution.
eg: MIPS R10000
DEC Alpha 21264

Microprocessor architecture and Operation of its components

- The architecture of a microprocessor defines suitable placement of its different functional blocks in the form of required circuitry for efficient flow of data/information and result from one block to the other.
- The most common architecture of a microprocessor is as follows:



- For clarity, the microprocessor can be divided into three segments, arithmetic / logic unit (ALU), register unit, and control unit.

- Arithmetic and Logic Unit: This segment of microprocessor performs computing functions on data. The CPU performs arithmetic operations such as addition and subtraction and logic operations such as AND, OR and exclusive OR. Results are stored either in register or in memory or sent to output devices.

- Register Unit: This segment of microprocessor consists of various registers. The registers are used primarily to store data temporarily during the executing of a program.

Some registers are general purpose registers and some are special purpose registers.

→ General purpose registers are used to store temporary data in the time of different operations in microprocessor.
eg:- 8085 microprocessor has 8 addressable 8-bit registers
A, B, C, D, E, H, L, F

→ Special purpose registers are used to hold program state; they include program counter (PC), stack pointer, status register, etc. These registers are used by control unit to control the operations of CPU.

- Control unit: The control unit provides the necessary timing and control signals to all the operations in the microprocessor.
It controls the flow of data between the microprocessor and peripherals.

• The microprocessor has three buses which carry all the information and signals involved in the system operations. These buses connects the microprocessor to each of the memory and I/O elements so that information can flow between CPU and any of these elements.

- Data bus: This is a bidirectional bus, because data can flow to or from the CPU.
- Address Bus: This is a unidirectional bus, because information flows over it in only one direction from CPU to the memory or I/O elements.
- Control Bus: This is the set of signals that is used to synchronize the activities of the separate microcomputer elements. Some of the control signals are RD, read and WR, write are sent by the CPU to other elements.

• Instruction decoder: This unit decodes the instruction present in instruction register and decoded information is used by timing and control circuit to issue necessary signals for instruction execution.

Microprocessor operations

All the operations of the microprocessors can be classified into one of three types:

1. Microprocessor initiated operations
2. Internal operations
3. Peripheral initiated operations

1. Microprocessor initiated operations: These are the operations which are initiated by microprocessor itself and peripheral devices will execute these operations.

Following operations are microprocessor initiated operations:

(i.) Memory Read: Reads data from memory
(ii.) Memory Write: Writes data into the memory
(iii.) IOR: Accepts data from input devices
(iv.) IOW: Sends data to output devices

All these operations are part of communication process between microprocessor and peripheral devices or memory. To perform these communication operations, microprocessor executes the following steps:

(a.) Identify the address of memory location or peripheral device
(b.) Provide timing or synchronization signals.
(c.) Transfer the data.

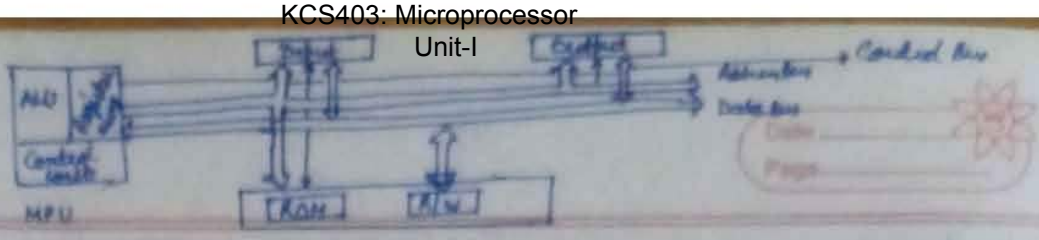
The microprocessor performs these operations through the communication lines called system bus.

As three types of information is communicated, so there are three types of buses

→ Address buses: carries the address of memory location or I/O devices that microprocessor wants to access. It is a unidirectional bus.

→ Data bus: is used to transfer data between the processor and memory and I/O devices. It is bidirectional in nature.

→ Control bus: is used to carry signals between



microprocessor and various devices connected to it. It also carries synchronization and timing signals.

2. Internal Data Operations

- These are the operations which are internally performed by microprocessor.
- The internal architecture of microprocessor determines how and what operations can be performed with the data.
- The internal operations performed by the microprocessors are classified into five groups:
  (i) Store data
  (ii) Perform arithmetic and logic operations.
  (iii) Test for conditions.
  (iv) Sequence the execution of instructions.
  (v) Store data temporarily during the execution in the defined R/W memory locations called the stack.
- These operations are performed with the help of registers, ALU, and control logic and path for communication.

3. Peripheral or Externally Initiated Operations

External devices can initiate the following operations, for which individual pins on the microprocessor chip are assigned. These operations are:

Reset: When reset pin is activated all internal operations are stopped and the program counter is reset to 0000. Program execution again begins from zero memory address.

Interrupt: The microprocessor's current operation is suspended and the microprocessor executes some emergency task what is called a service routine.
This routine "handles" the interrupt. Then microprocessor returns to its previous operations and continues.

Ready: This pin is used by external device to synchronize the speed of microprocessor with the slower peripherals. As long as the Ready pin is low, the microprocessor will be in a 'wait' state.

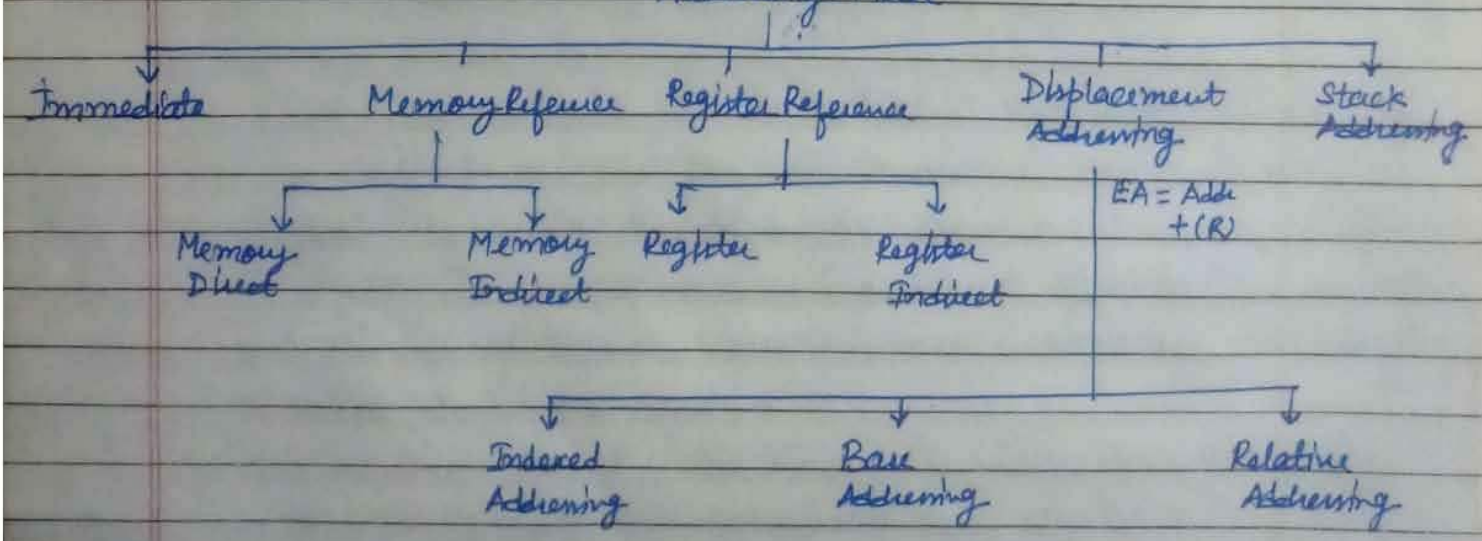
Hold: This pin is used by external devices to gain control of the buses. When the HOLD signal is activated by an external device, the microprocessor suspend current execution and stop using the buses. This would allow external devices to control the information on the buses, such as, Direct Memory Access (DMA).

Addressing Modes

- An instruction format with an addressing mode field is as shown

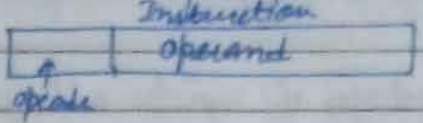
Opcode	Mode	Address

  - The Opcode (operation code) specifies the operation to be performed.
  - The Mode field is used to locate the operands needed for the operation.
  - There may or may not be an address field in the instruction.
- The operation field of an instruction specifies the operation to be performed on some data stored in computer register or memory word. The ways of choosing the operands depends upon the addressing mode of the instruction.
- The addressing mode specifies a rule for interpreting a modifying the address field of the instruction before the operand is actually referenced.
- Addressing mode techniques
  (i) gives programming versatility to the user by providing facilities such as pointers to memory, counters for loop control, indexing of data, and program relocation.
  (ii) reduces the number of bits in address field of instruction.
- Common addressing modes are shown in following tree representation



- In general not all are used in for all microprocessors.

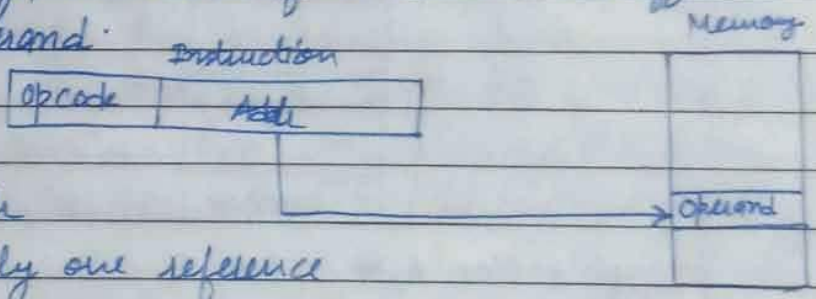
- Immediate Addressing:
  - The simplest form of addressing is immediate addressing, in which the operand value is present in the instruction.



eg: ~~ADD~~ ADI 5H

  - This addressing mode can be used to define and use constants or set initial values of variables.
  - The advantage of immediate mode addressing is that no memory reference other than the instruction fetch is required to get operand.
  - The disadvantage is that the size of the number is restricted to the size of the address field.

- Direct Addressing
  - In this addressing, the address field contains the effective address of the operand.



Effective Address = Addr

  - It requires only one reference

eg: LDA 2000H

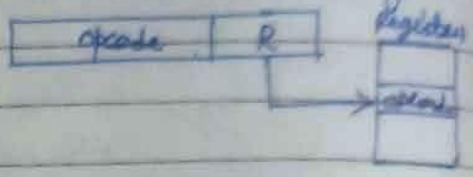
- Indirect Addressing
  - In this addressing mode, address field refer to the address of a word in memory, which in turn contains a full length address of the operand. This is known as indirect addressing.

Effective address = (Addr)

(Addr) ⇒ Content of Addr

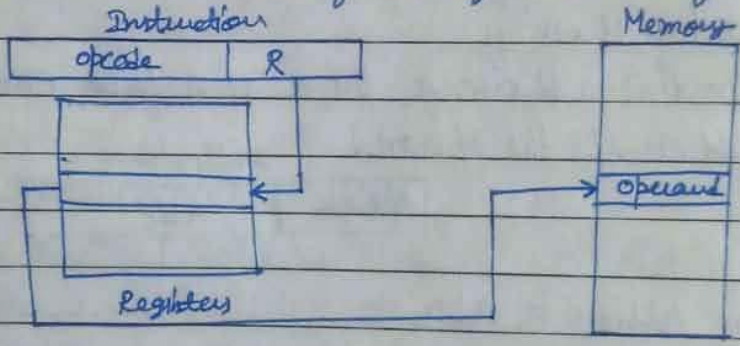
  - It requires three or more memory reference to fetch an operand.

- Register Addressing:
  - It is similar to direct addressing. The only difference is that the address field refers to a register rather than a memory address.
  - The advantages of register addressing are:
    (1.) Only a small address field is needed in instruction.
    (2.) No time consuming memory references are required.
  - The disadvantage is that the address space is very limited.



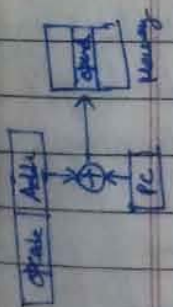
eg:- MOV A,B

- Register Indirect Addressing:
  - It is analogous to indirect addressing; the only difference is address field refers to a register.



eg:- 1. MOV A,M
2. MOV CX, [BX]

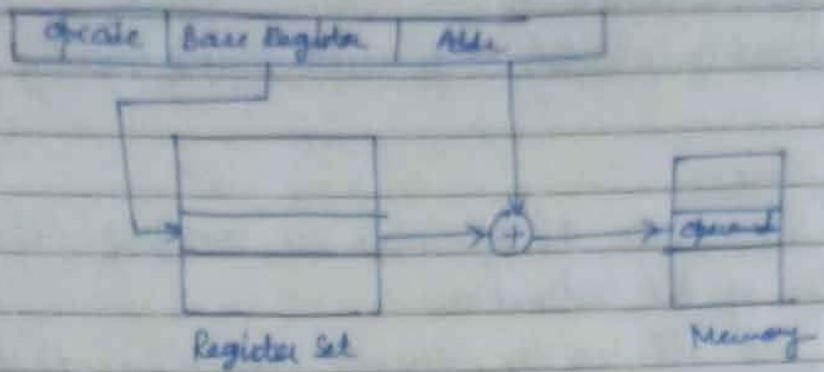
- Relative Addressing:
  - It is also known as PC relative addressing, the implicitly referenced register is the Program Counter (PC).
  - The next instruction address is added to the address field to produce effective address.



Effective address = Addr + (PC)

  - Relative addressing exploits the concept of locality.
- Base Register addressing:
  - For base register addressing, the referenced register contains main memory address, and address field contains a displacement from that address.

· The register reference may be explicit or implicit



eg: MOV AX, [BX + 08H]

EA = (BX) + 0008H

- Indexing: For indexing,
· The address field references a main memory address, and the referenced register contains a positive displacement from that address.
· This usage is just the opposite of the interpretation for base register addressing.
· The value of Addr is stored in the instruction's address field, and the register chosen is called an index register and is initialized to 0. After each operation, the index register is incremented by 1.
· Some systems automatically do this as part of the same instruction cycle. This is known as autoindexing.

[Effective Address = Addr + (R)
(R) = (R) + 1]

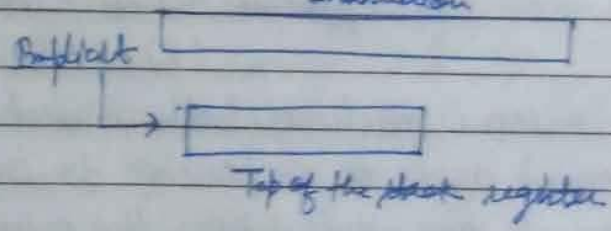
· In some machines, both indirect addressing and indexing are provided and it is possible to employ both in same instruction. There are two possibilities: the indexing is performed either before or after the indirection.
→ Postindexing: If indexing is performed after the indirection.
Effective Address = (Addr) + (R)
First, contents of the address field are used to access a memory location containing a direct address. This address is then indexed by the register value.

→ Preindexing: The indexing is performed before indirection.
Effective address = (A + (R))
An address is calculated as with simple indexing.

Implied Mode:
- In this mode the operands are specified implicitly in the definition of the instruction.
- In fact all register reference instructions that use an accumulator are implied mode instructions.

eg: The instruction "Complement accumulator".

Stack addressing:
- Stack is also referred to as a last in first out queue.
- The top two elements of the stack may be in processor registers, in which case the stack pointer references the third element of the stack.
- The stack pointer is maintained in a register. Thus, the references to stack locations in memory are in fact register indirect addresses.
- The stack mode of addressing is a form of implied addressing. The machine instructions need not include a memory reference but implicitly operate on the top of the stack.



eg: ADD

Interrupts

- Interrupt is a mechanism by which the processor is made to transfer control from its current program execution to another program of more importance or higher priority.
- Interrupts are generated by a variety of sources and I/O I/O devices to obtain services of CPU.
- Processor executes a program or routine upon interrupts this program is known as Interrupt service routine (ISR).
- The interrupts are needed for the following reasons.
  - Input/output devices are slower than memory or CPU.
  - Different devices require different amount of time from CPU.
  - Uncertainty of when device will be ready.

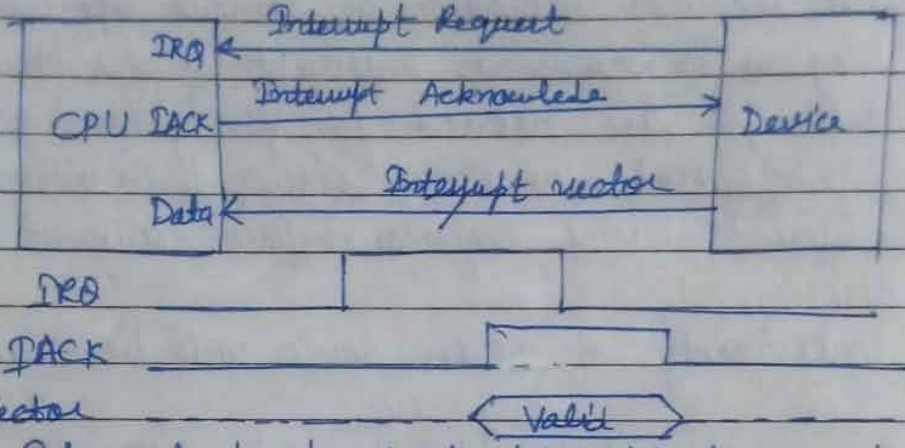
The processor handles interrupts internally. The interrupt structure has the following key features:

→ The number and types of interrupt available.
→ The address of memory location where the ISR is located for a particular interrupt signal. This address is called Interrupt vector address.
→ The masking and unmasking feature for interrupt signal.
→ The priority of interrupts when more than one interrupts are available.
→ The timing of interrupt signal.
→ The handling and storing of information about the interrupted program.

- Types of interrupts: The are following types of interrupts

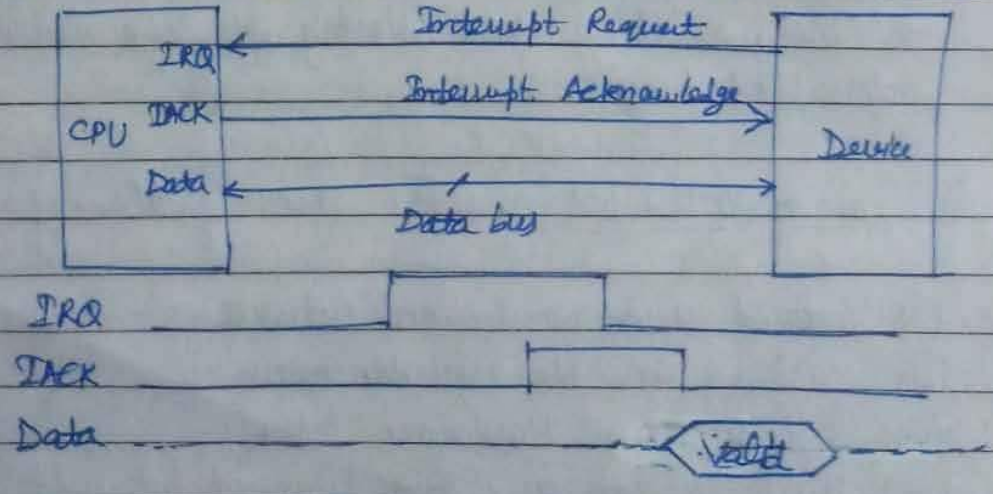
(i) Vectored and Non Vectored interrupt
(ii) Maskable and Non Maskable interrupt
(iii) Software and Hardware interrupt
(iv) level-triggered and Edge triggered interrupt.

(i) Non vectored Interrupts: Non vectored interrupts supply the CPU with information which is used to generate the address of the handler routine for the interrupt.
These interrupts are useful for CPUs that receive interrupt request from several devices via the same control line.



→ External device sends interrupt to CPU by asserting IRQ signal.
→ CPU inputs and interrupt vector from the device and call an interrupt service routine (handler).
→ The device sets IRQ low.
→ Handler routine proceeds and starts transferring data.

Vectored Interrupts: A vectored interrupt uses a handler routine (ISR) at a known address.
CPU does not need any additional information to generate this address and access its handler routine.



→ An external device sends an interrupt to the CPU by asserting its IRQ signal.
→ When CPU is ready, it asserts its interrupt acknowledge signal (IACK).
→ The device sets IRQ low, which cause CPU to set IACK low.
→ Handler routine proceeds and starts transferring data.

(ii) Maskable Interrupts: Maskable interrupts are interrupts that can be blocked.
That is, It is a hardware interrupt that can be disabled or ignored.

Non maskable Interrupts:
Non maskable interrupt is interrupt that will always be acknowledged and accepted. This interrupt will never be disabled. It is mostly used for the attention of system for non-recoverable hardware errors.
Non maskable interrupts are generally used when response time is critical and when an interrupt should not be disabled in the normal operation of the system.

(iii) Software and Hardware Interrupts:

Software interrupts
→ arises in response of an instruction.
→ is of synchronous nature.
→ is requested by executing instruction or due to arithmetic errors.

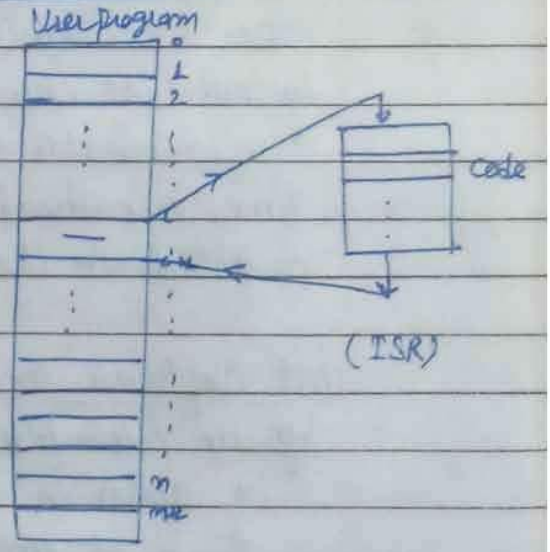
Hardware interrupt
→ is signal given to the processor.
→ is requested by the external devices and is asynchronous event.
→ is a way to avoid wasting the processor's time.

(iv.) Level triggered interrupt: The interrupts which are triggered at high or low level are called level triggered.

Edge triggered interrupt: An edge triggered interrupt is an interrupt signaled by a level transition on the interrupt line, either a falling edge (high to low) or a rising edge (low to high).

Negative edge triggering Positive edge triggering

- Interrupt request and Interrupt Handler



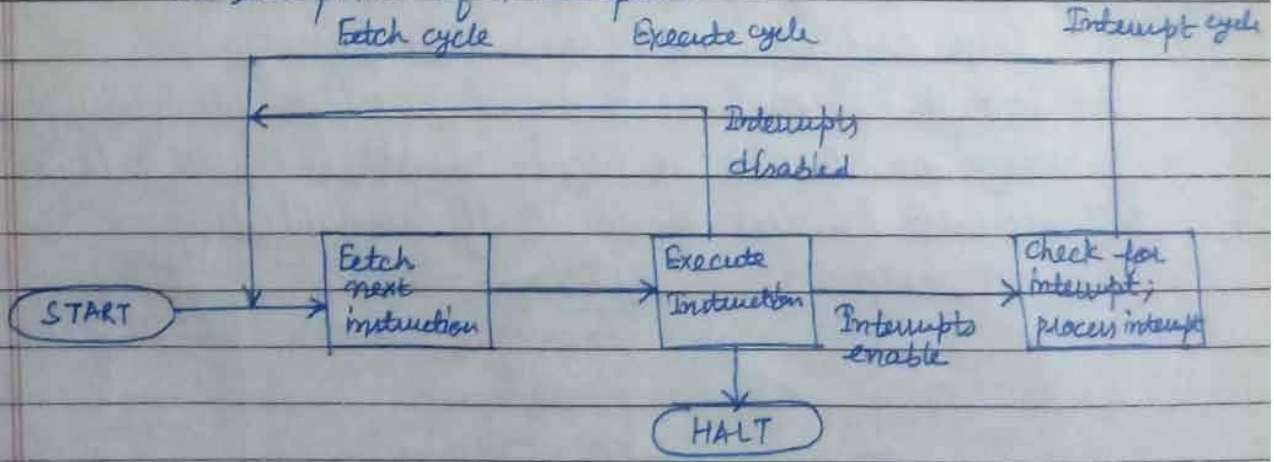
→ CPU executing some task.
→ User uses the keyboard to issue a high priority command.
→ This issues an interrupt request to the CPU.
→ CPU saves the PC content and supplementary information about current state in stack.
→ Load PC with beginning address of the code written to handle this interrupt (This code is known as Interrupt Service Routine or Interrupt Handler) and execute it.
→ After finishing the ISR, the processor returns to the point i in the interrupted program where execution was interrupted.

Interrupt cycle

- To accomodate interrupts, an interrupt cycle is added to the instruction cycle.
- In interrupt cycle, the processor checks to see if any interrupts have occured, indicated by the presence of an interrupt signal.
- If no interrupts are pending, the processor proceeds to the fetch cycle and fetches the next instruction on the current program.
- If interrupt is pending, the processor does the following:
  - → It suspends execution of current program being executed and save its context.
  - → It sets the program counter (PC) to the starting address of an interrupt handler routine.

  The processor now proceeds to fetch cycle and fetches the first instruction in interrupt handler program, which will service the interrupt.

  When the interrupt handler routine is completed, the processor can resume execution of the user program at the point of interruption.

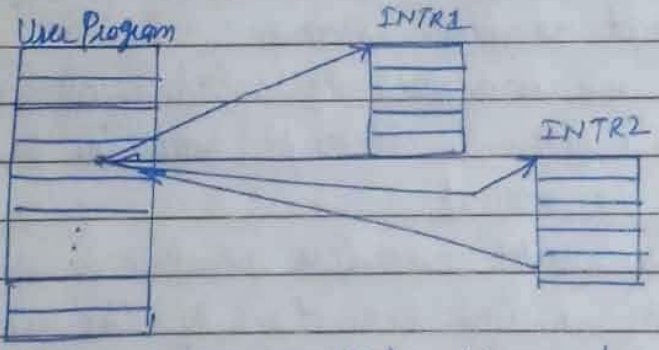


- Handling multiple interrupts

In case of occurance of more than one interrupts, there are two approaches to deal with multiple interupts.

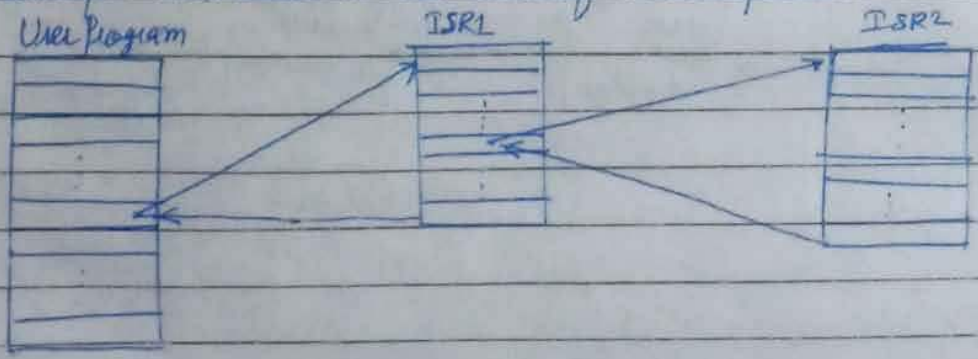
→ First approach is to disable interrupts while an interrupt is being procesed. It means that processor will ignore the interrupts during this time, it generally remains pending and will be checked by the processor after processor has enabled interrupts.

When a user program is executing and an interrupt occurs, interrupts are disabled immediately. After the interrupt handler routine completes, interrupts are enabled before resuming the user program, and processor checks to see if additional interrupts have ~~occured~~ occured.



- Disadvantage of this approach is it does not considers time critical needs (or relative priority)

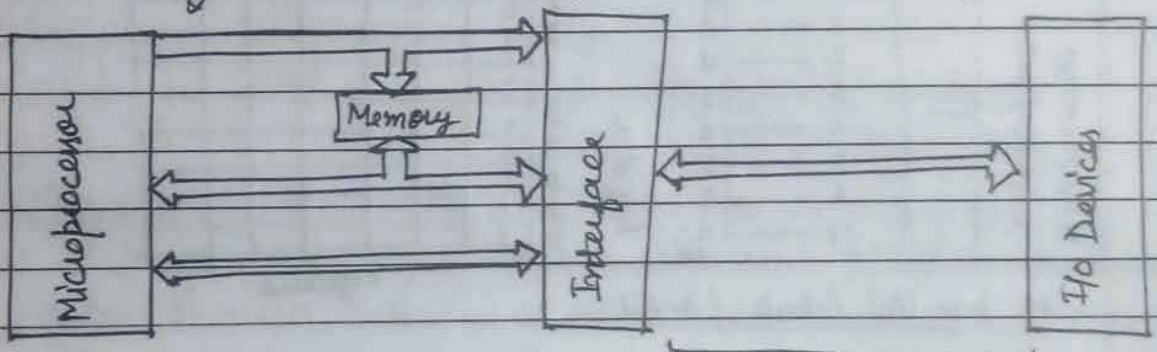
→ Second approach is to define priorities for interrupts and to allow an interrupt of higher priority to cause a lower priority interrupt handler to be itself interrupted.

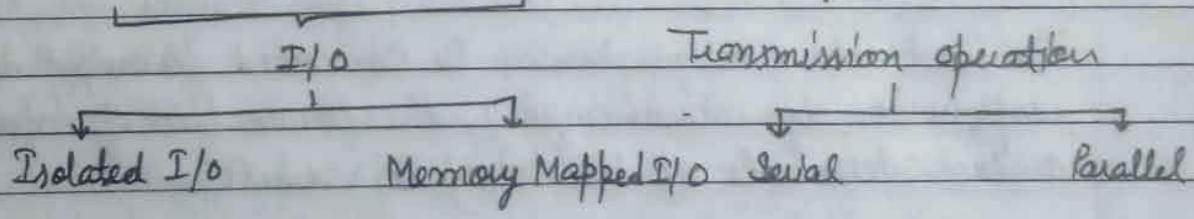


Data Transfer Schemes

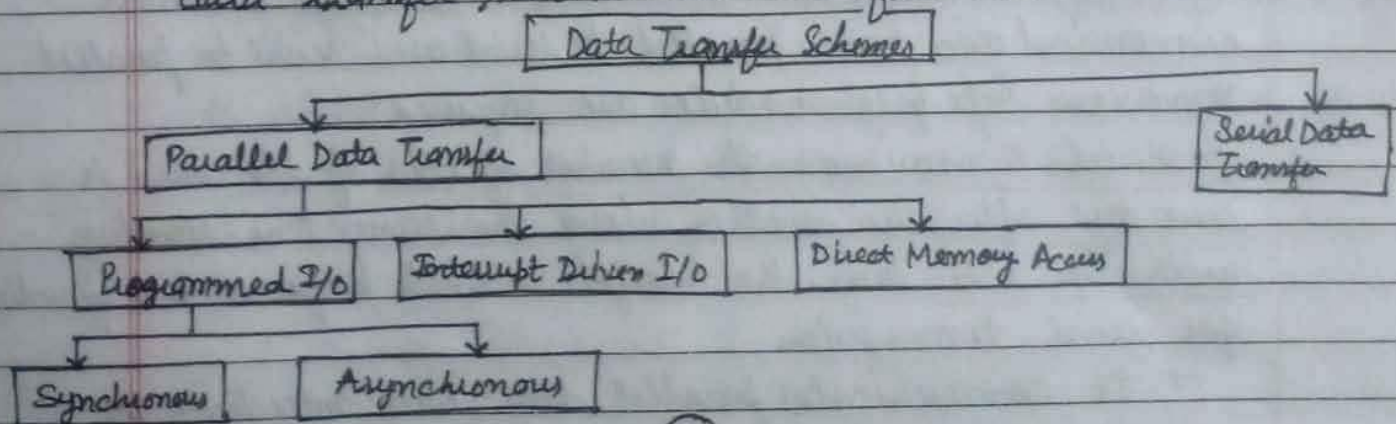
Data transfer takes place between two devices at a time:
- → CPU and Memory
- → CPU and I/O Devices
- → Memory and I/O Devices

- When data transfer takes place between Processor and memory no additional circuitry is required as memory is compatible with microprocessor.
- But the data transfer between microprocessor and I/O devices needs an interface between them because of the difference between nature of data they deal, level of power they use and the speed of data transfer.

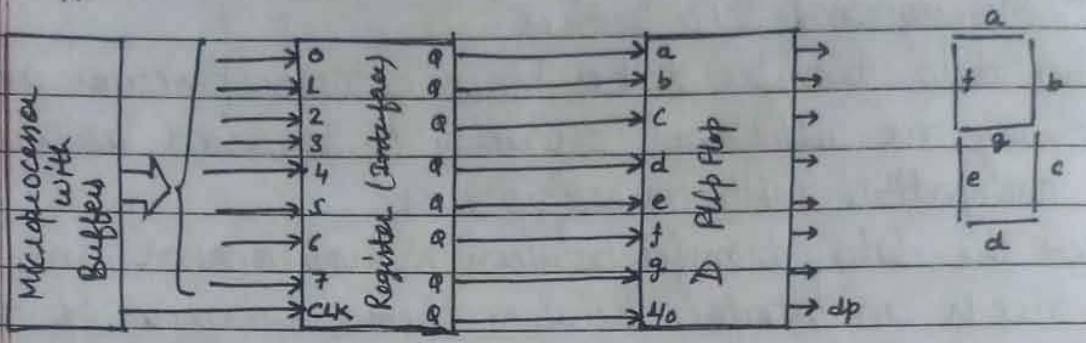




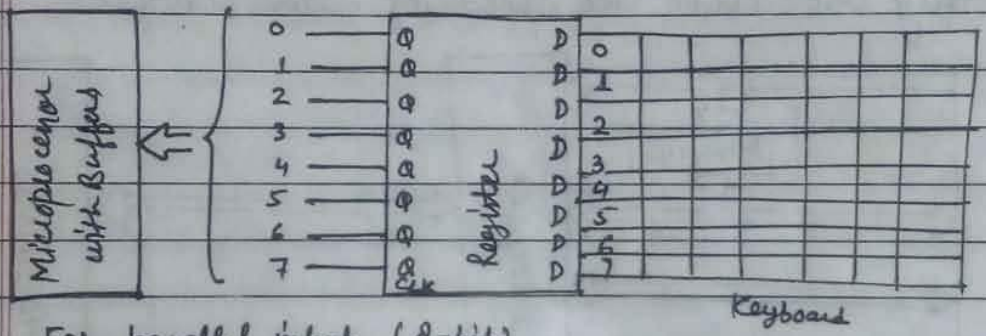
- To make the slow devices compatible with faster microprocessors Data transfer schemes have been developed. These data transfer schemes are classified as:



- Parallel Data Transfer In parallel data transfer, the bits of data flow in or out of the microprocessor simultaneously appear on data lines.



For parallel output (8-bit)

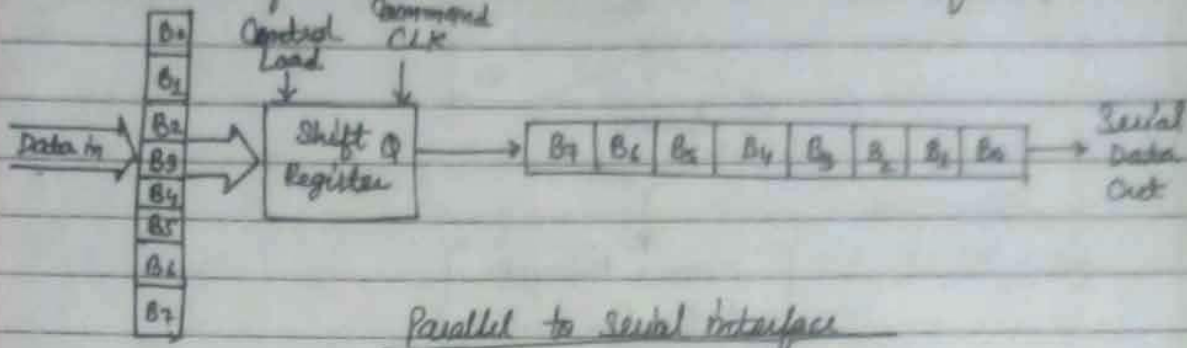


For parallel input (8-bit)

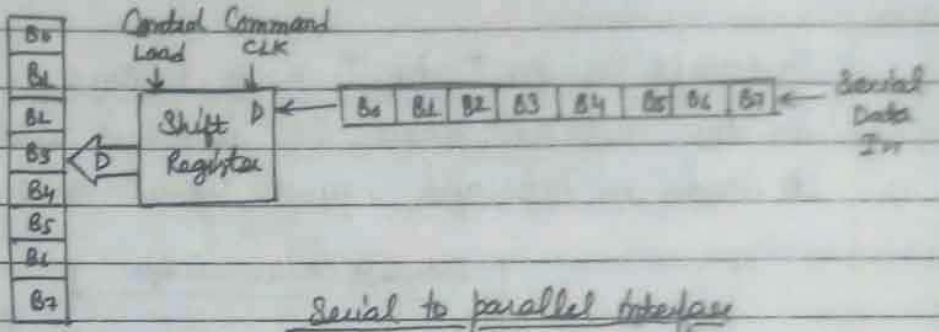
The input data configuration can be converted to output data configuration by interchanging the connection of input and outputs terminals of the interface module.

- Serial Data Transfer Parallel data transfer from or to the microprocessor and I/O devices becomes very expensive. In such situations, serial transfer is more economical and hence parallel to serial and serial to parallel conversions interface modules are required.
* In serial transmission, the number of bits of the word is sent one after the another along the same line starting with the LSB. Thus, the interface module performs two functions for serial transmission
  → It communicates parallel data with microprocessor

→ Converts parallel to serial and vice-versa for I/O devices



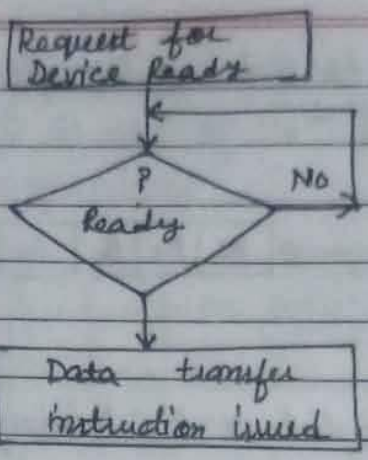
Parallel to Serial Interface



Serial to parallel Interface

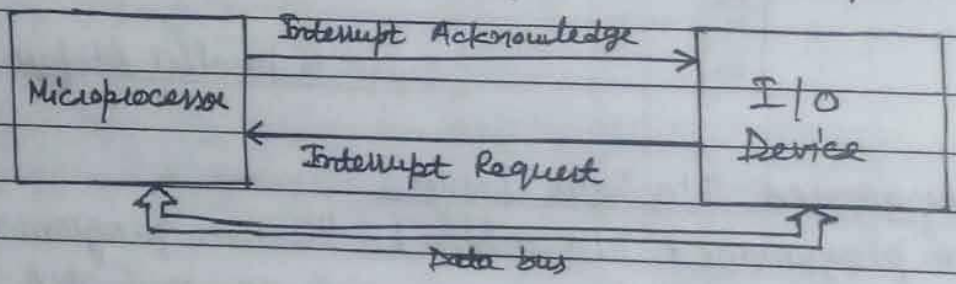
- Programmed I/O Data Transfer

- In programmed I/O method, the user program controls the data transfer between microprocessor and I/O Devices.
- The data transfer can be synchronous or asynchronous depending upon the type and speed of I/O devices.
  → When the speed of I/O devices matches the speed of microprocessor; the synchronous type of data transfer can be achieved. In this type of ~~the~~ data transfer microprocessor does not have to wait for the availability of data.
  → When the speed of I/O devices is slower than the speed of microprocessor, asynchronous type of data transfer method is used. In this method, the microprocessor has to check the status of I/O devices for availability of data to be transferred to the microprocessor or whether the I/O device is free to accept the data from the microprocessor.



- Interrupt Dsiven Data Transfer Scheme

- In this method when the device is ready for data transfer it sends an interrupt request signal to microprocessor.



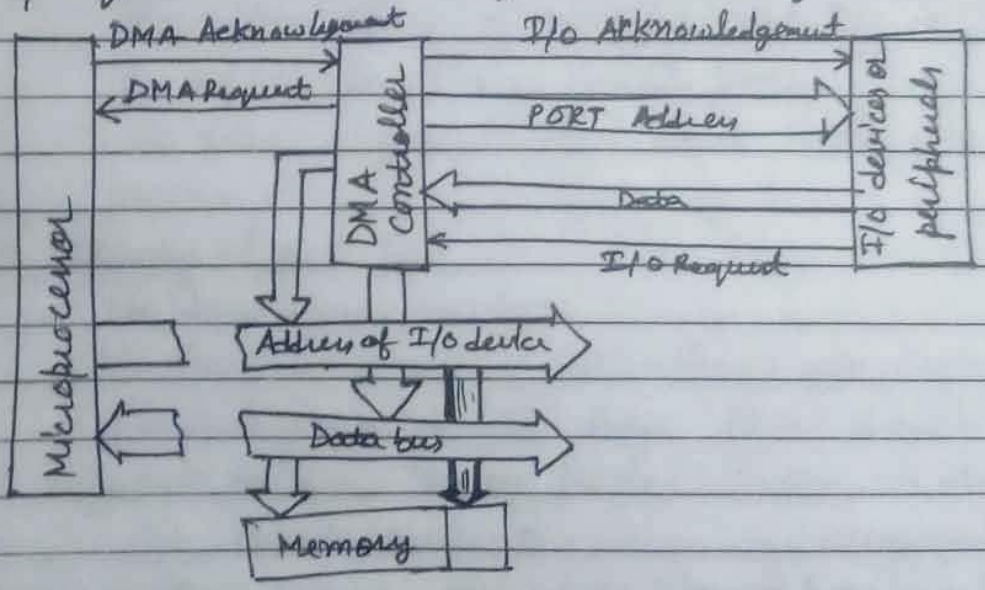
- Microprocessor sends back an interrupt acknowledgement signal to the I/o device.
- Microprocessor suspends its job after completing its current instruction and saves the current address and status, and executes the Interrupt Service Routine (ISR).
- Then transfer of data is performed.
- After completing data transfer processor status is restored and processor returns back to the original program.

- Direct Memory Access (DMA)

- For voluminous data transfer between I/O Devices and external memory going through microprocessor register (ex. accumulator) is a time consuming and uneconomical.
- For such a huge data transfer it is suitable to bypass microprocessor.

• It is done by sending a request to the microprocessor by the I/O device for Direct Memory Access (DMA). On receiving such request, the microprocessor relinquishes the address and data buses and informs the requesting I/O device by sending DMA acknowledge signal. The I/O device withdraws the DMA request as soon as the data transfer between I/O and the external memory is over.

• Direct Memory Access involves data transfer between I/O device and memory by the help of DMA controller without involving microprocessor. This DMA controller deals with the control functions during DMA operations. The address registers of DMA controller generate address to transfer data blocks and the count register counts and hold number of data block transfered. Also, it specifies the direction of data transfer.



• The DMA Controller operates under three modes:

→ Burst mode: DMA Controller switch over the control to microprocessor only on completion of entire data transfer.

→ Cycle Stealing mode: DMA controller relinquish the control to microprocessor on transfer of every byte, thereby microprocessor gets control and become able to process highly ___ ations.

prioritized instruction. DMA need to make a request for bus grant for each byte.

→ Transparent Mode: DMA controller can transfer data blocks only when microprocessor ~~are~~ is executing such instruction that does not requires system bus utilization.

Input / Output Devices

- Microprocessor needs to communicate with the various memory and input - output devices.
- There are three ways in which system bus can be allotted to them:
  (i) Separate set of address, control and data bus to I/O and memory.
  (ii) Common bus (data and address) for I/O and memory but separate control lines.
  (iii) Common bus (data, address and control) for I/O and memory.

- Isolated I/O

- We have common bus (data and address) for I/O and memory but separate read and write control lines for I/O.
- The address space of memory and I/O is isolated. The address for I/O is called ports.
- We have different read/write instruction for I/O and memory.

- Memory Mapped I/O

- We have common bus.
- We have same set of instructions for memory and I/O.
- I/O and memory both have same address space.

- Difference between Memory Mapped I/O and I/O mapped I/O

Memory mapped I/O	I/O Mapped I/O
• Memory and I/O share entire address range of processor	• Processor provides separate address range for memory and I/O.
• Processor provides more address lines for accessing memory.	• less address lines for accessing I/O
• More decoding is required.	• Less decoding is required.
• Memory control signals ($\overline{MEMW}$, $\overline{MEMR}$) are to control Read and write I/O operations.	• I/O control signals ($\overline{IOR}$, $\overline{IOW}$) are used to control Read and write I/O operations.

Instruction and Data flow

- A processor receives instructions and data in binary form;
- A group of 8 bits is called a byte,
- Data and instructions for processor are specified in byte form.
- The microprocessor handles a number of bits at time which is called the word length. Each word may be of length 8, 16, 32, ... bits.
- Each word is either instruction word (Opcode) or Data word. There two kinds of words are processed during an instruction cycle.
- To understand instruction ~~and~~ execution and data flow it is necessary to understand opcode.

"An opcode is machine language instruction which denotes the microprocessor about what operation should be perfed on specific data."

- The microprocessor first reads opcode from the instruction and then performs the operation specified over given data.
- In the beginning of instruction cycle first opcode fetch and decode is performed.

There are following steps in fetch and decode:

→ The content of Program Counter (PC) are transferred to a special register called Memory Address Register (MAR).

→ The contents of MAR are transferred to the memory through the address bus.

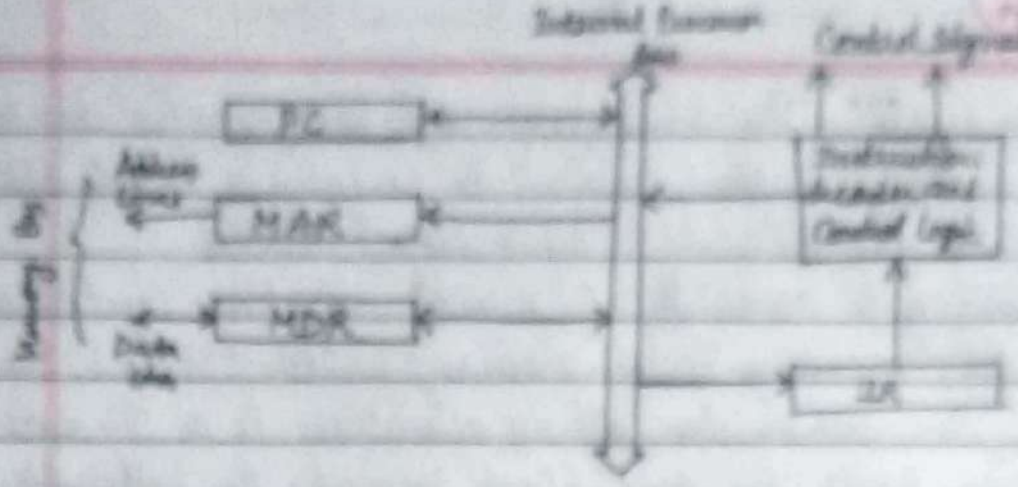
→ Control unit sends certain control signals to the memory

→ The decoder circuitry of memory activates and memory understands what is to be done. Then the memory sends the opcode to the microprocessor through the data bus. The opcode first comes in memory data register (MDR).

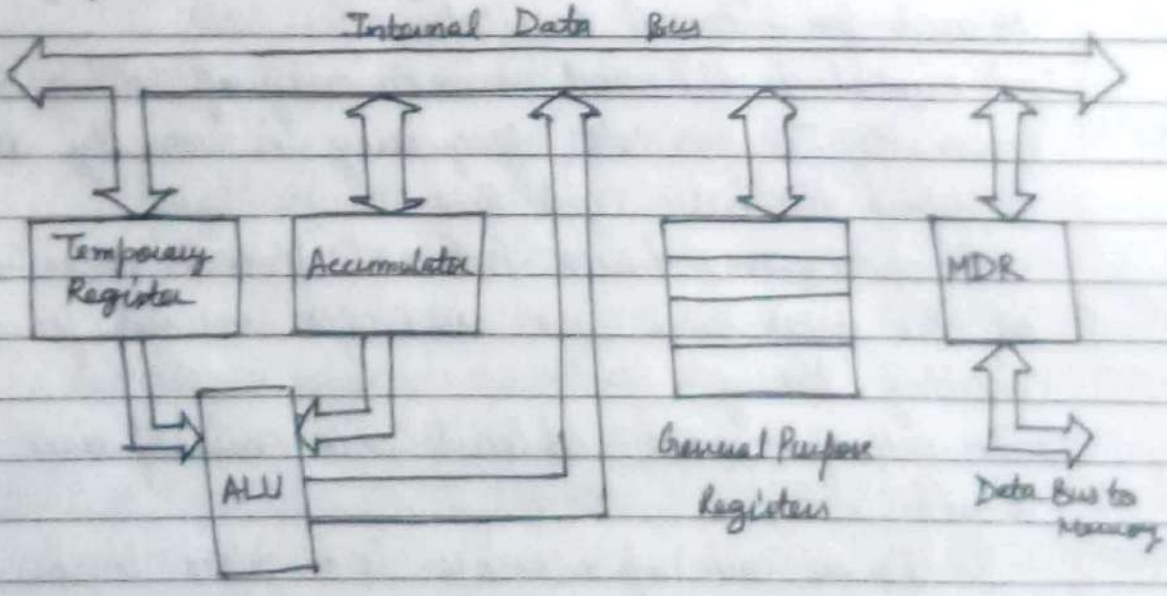
→ Opcode is then placed in instruction register (IR).

→ The instruction is decoded by instruction decoder and is executed.

→ Finally, the content of the PC is incremented.



- The execution of an instruction requires the flow of data word. The flow of data word could be from memory or input device.
The data word flows to the processor through the data bus and is placed in accumulator or any general purpose register depending on instruction.
- After execution of an instruction / program that data is placed in either register or memory location or sent to output device.
- This process of placing the address and reading / writing the data is considered a single cycle and is known as read cycle or write cycle.



Timer and Timing Diagram

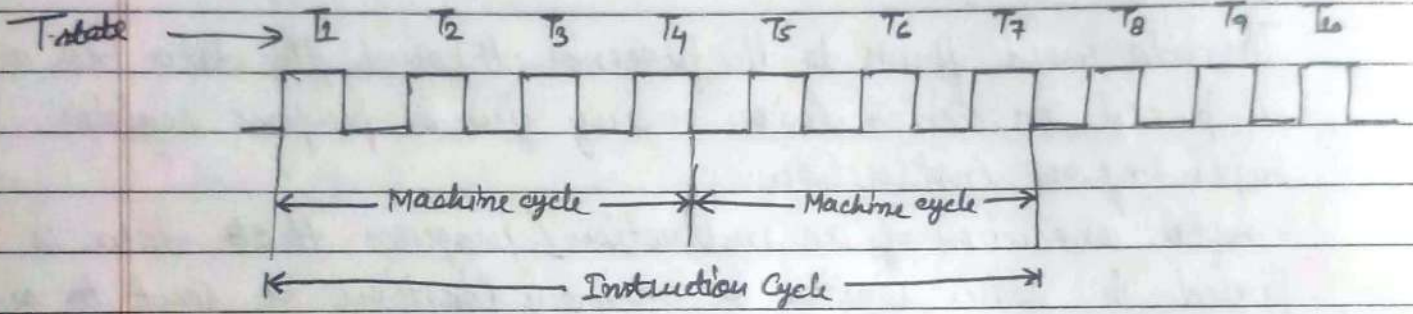
Instruction cycle: The time required to execute an instruction

Machine cycle: The time required to perform a single operation or to access the memory / input output devices.

Instruction cycle consists of several machine cycles (such as opcode fetch, memory read, memory write, I/O read, I/O write)

T- state: One subdivision of an operation is a T-state.

A T-state lasts for one clock period. An instruction's execution length is usually measured in a number of T-states. (clock cycles).

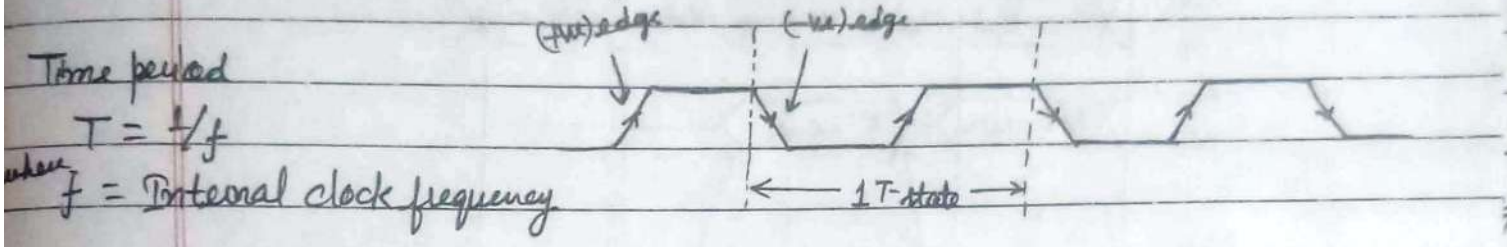


Clock and Timers

- Microprocessor must have a timing mechanism that defines the instruction cycle. This is done by an oscillator.
- Oscillator may be internal or external. Usually RC oscillator is used for internal oscillation.
- A crystal is the most common way of setting the frequency externally. The oscillator frequency is usually divided internally to define the basic cycle time.
- Microprocessors have internal timers under the control of the users and are used for various functions requiring counting / timing.
- In microprocessors, atleast one ~~microprocessor~~ counter is available.
- In larger microprocessors 4 or more timers are available; some of them are 8-bit timers and some are 16-bits timers.
- The microprocessors also include a watchdog timer for the purpose of resetting the processor.

Timing diagram

- It is a graphical representation. It represents the execution time taken by each instruction in a graphical format. The execution time is represented in T-states.



- Let us discuss timing diagrams of basic machine cycles of 8085 microprocessor:

The basic machine cycles of 8085 microprocessor are:

→ Op-code Fetch cycle (4T or 6T)
→ Memory Read cycle (3T)
→ Memory Write cycle (3T)
→ I/O Read Cycle (3T)
→ I/O Write cycle (3T)
→ Interrupt acknowledge (3
→ Bus idle

→ Opcode Fetch cycle (4T or 6T)

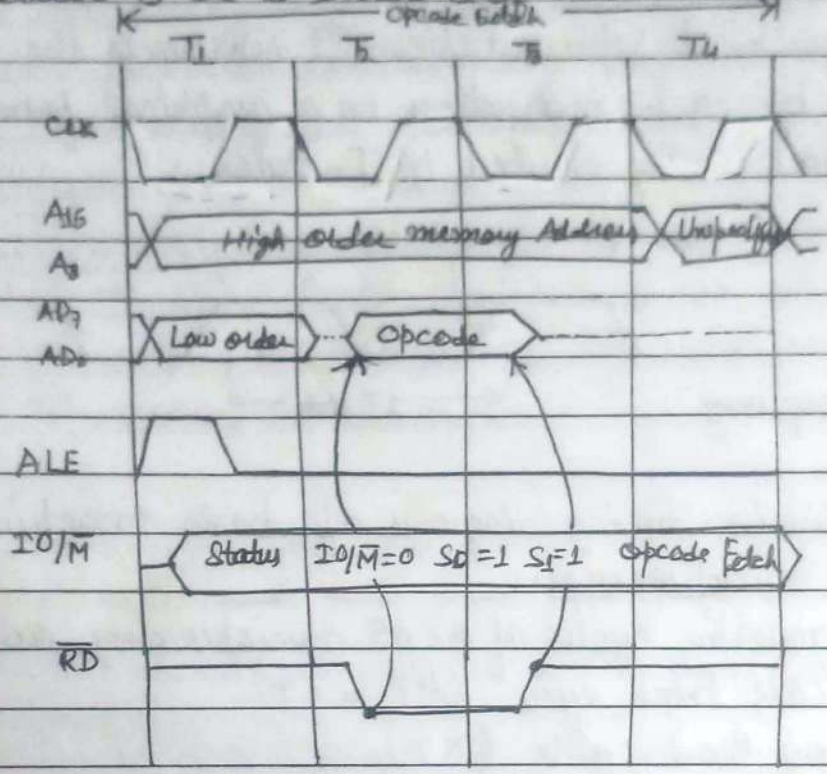
- It is the first machine cycle of every instruction.
- Length of this machine cycle is not fixed.
- Following are the steps in opcode fetch cycle.

(*) Step 1 8085 microprocessor places the contents of program counter on address bus, activate ALE and send the status signals $IO/\overline{M}$, S_1, S_0 (0 1 1) respectively.

Step 2 * Low order address disappears from $AD_0 - AD_7$ lines. Processor activates the $\overline{RD}$ signals to enable the addressed memory location which places its contents on the data bus ($AD_0 - AD_7$).

Step 3 * The processor loads the content of data bus on its instruction register and deactivates the $\overline{RD}$ signals to disables the memory device.

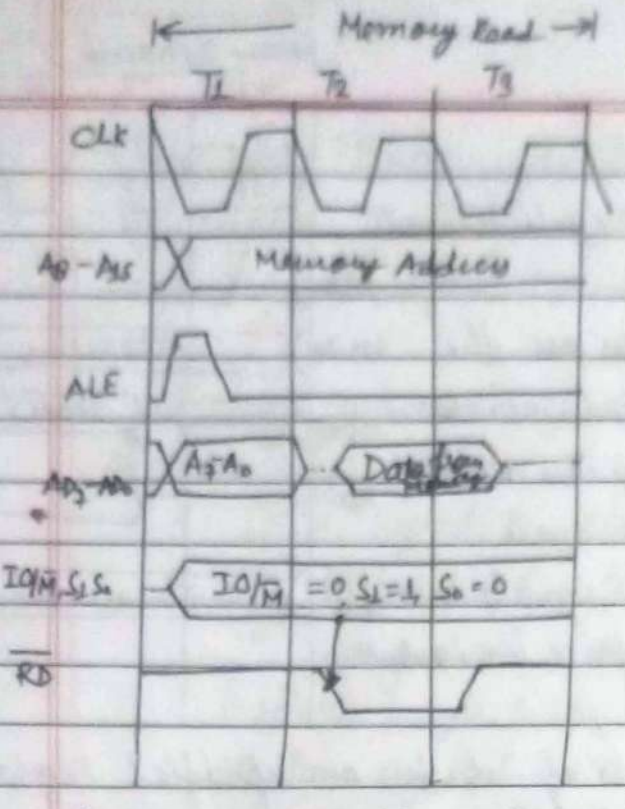
Step4 * Processor decode the opcode and decides whether to enter T_5 or to enter T_1 of new machine cycle.

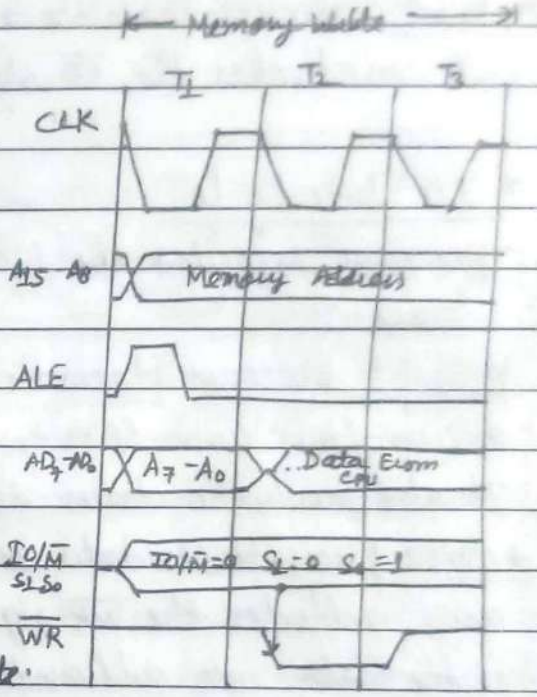


→ Memory Read Cycle (3T)

- The microprocessor executes the memory read cycle to read the data from RAM or ROM memory.
- Steps involving in memory read cycle are as follows:

* Step1 processor places the address on address lines from SP, BP or PC and activate ALE to latch low order of address. Also, processor sends the status signals (010) for this machine cycle.
* Step 2 processor activates the $\overline{RD}$ signals to enable the addressed memory location which places its contents on the data bus.
* Step 3 The processor loads the contents of data bus on specified register and deactivates the $\overline{RD}$ signal to disables the memory devices.





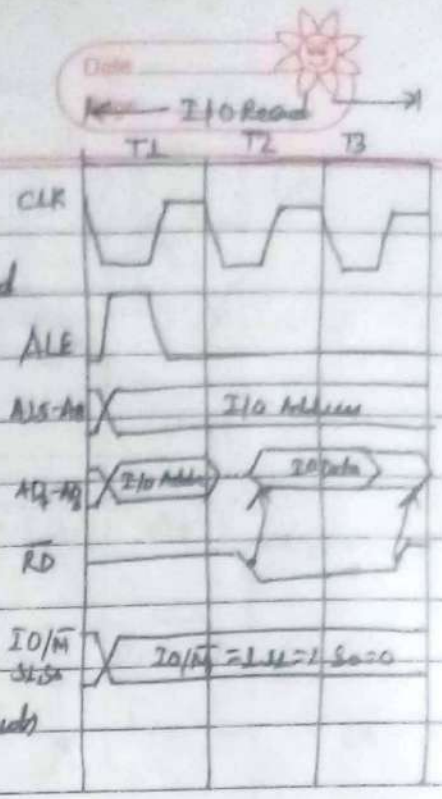
→ Memory Write cycle (3T)

- The microprocessor executes the memory write cycle to store the data into RAM or stack memory.
- Following are the steps involved in this machine cycle:

*__Step 1__ processor places the address on address bus from SP or Rp and activates ALE to latch low order of address and sends the status signals (001) for memory write.

*__Step 2__ processor places the data on data bus and activates $\overline{WR}$ signal to write data into addressed memory location.

*__Step 3__ processor deactivates the $\overline{WR}$ signals which disables the memory device and terminates the write operation.



→ IO Read cycle

· Processor executes the I/o read cycle to read the data from input device.

* Step1 processor places the address on the address lines from SP, Rp or PC and activates ALE to latch low order address. Also sends status signals I/O for machine cycle.

* Step2 processor activates the $\overline{RD}$ signals to enable addressed input device to place its contents on the data bus.

* Step3 processor loads the contents of data bus on specified Register and deactivates the $\overline{RD}$ signal to disable input device.

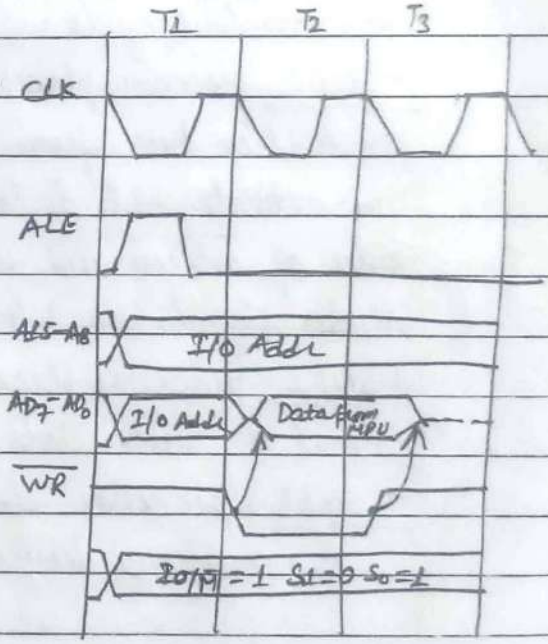
→ IO Write Cycle

· processor executes the I/O write cycle to store the data into output device.

* Step1 processor places the address on address lines from SP or Rp, activates ALE. It also sends the status signals IO/M.

* Step2 process places data on data bus and activates the $\overline{WR}$ signals to write data into addressed output device.

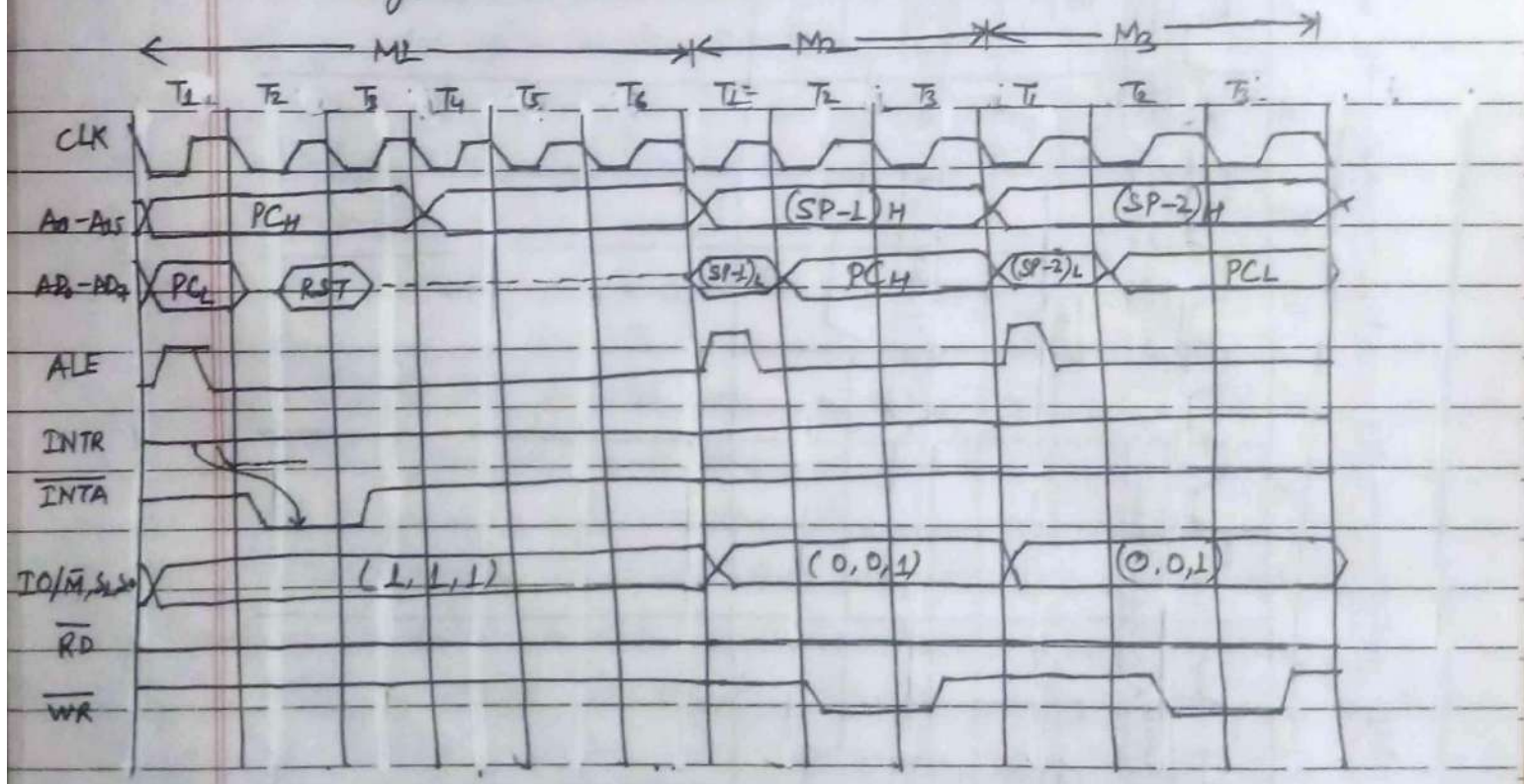
* Step3 processor deactivates the $\overline{WR}$ signal which disables the output device and terminates the operation.



→ Interrupt Acknowledge Machine cycle

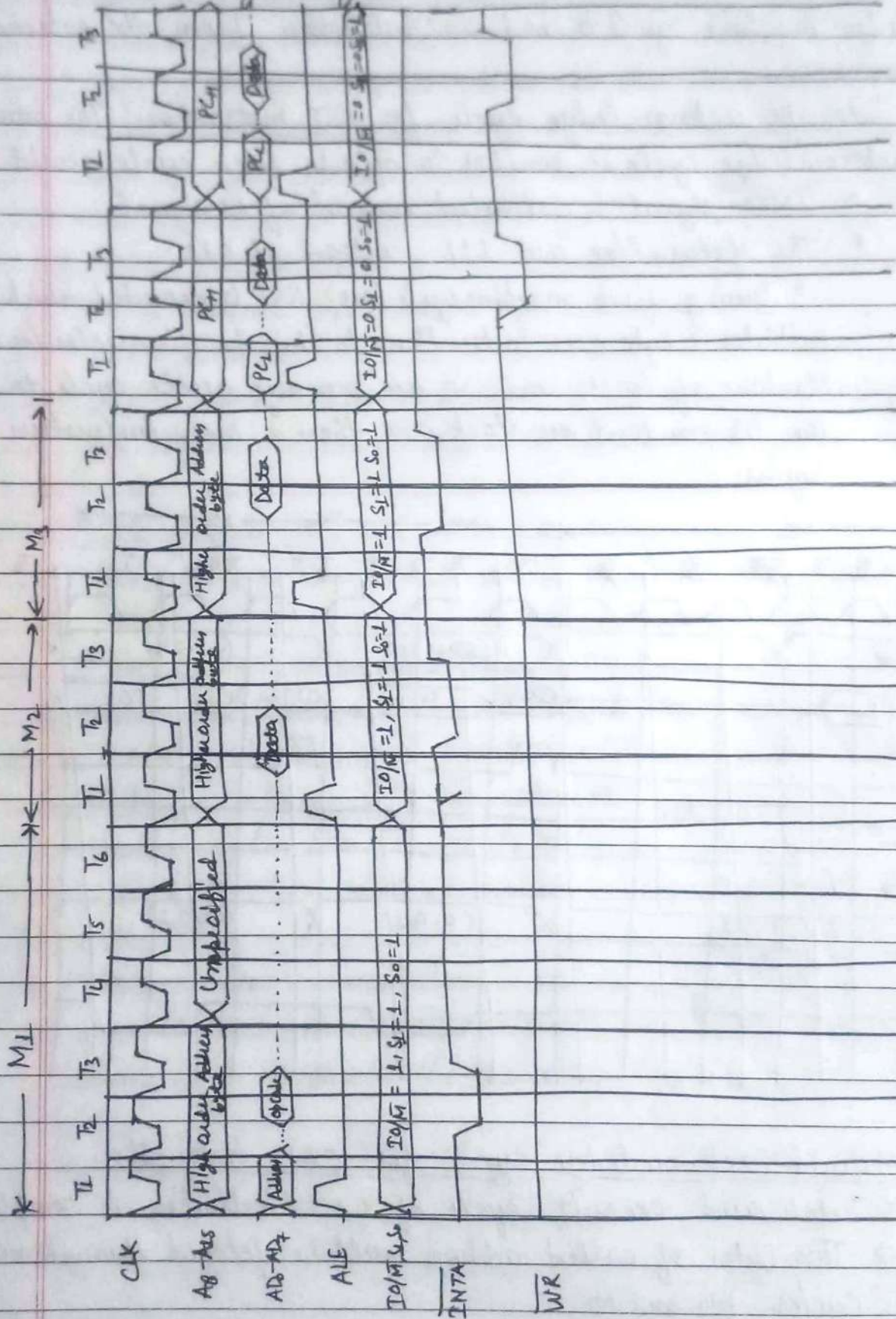
- In response to INTR signal, processor executes interrupt acknowledge machine cycle to read an instruction from the external device.
- Interrupt acknowledge cycle for RST instruction: The interrupt acknowledge cycle is similar to opcode fetch cycle except
  * $\overline{INTA}$ signal is activated instead of $\overline{RD}$ signal
  * The status line are 111 instead of 011.

  → During first machine cycle (M1) RST is decoded, which initiates 1 byte CALL instruction to the specific vector location. Machine cycle M2 and M3 are memory write cycle to store the PC content on stack and then a new instruction cycle begins.



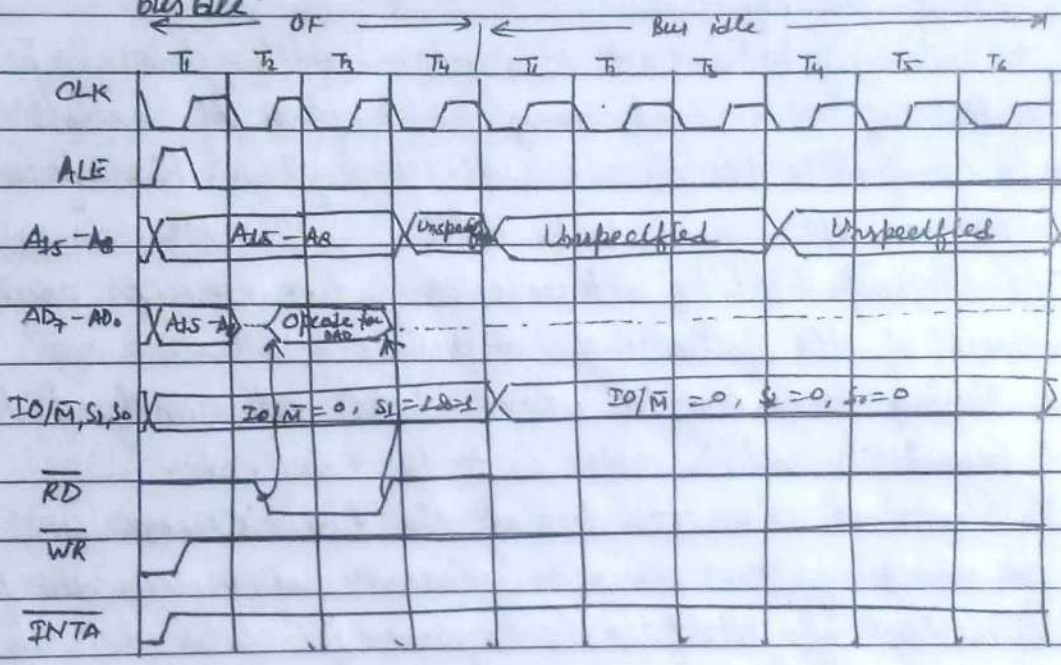
- Interrupt acknowledge cycle for CALL instruction:
  ⇒ Fetch and execute cycle of CALL instruction is completed.
  ⇒ Two bytes of called address will be fetched through machine cycles M2 and M3
  ⇒ The machine cycles M4 and M5 are memory write cycles

that stores the contents of PC on stack memory and then new instruction cycle begins.



→ Bus Idle Machine cycle:

- There are few situations where the machine cycles are neither read and write.
- DAD instruction is example of such situation. This instruction requires 10 T states.
  → Four T states for fetch cycle and other six T states are bus idle.



Date
Page

Interfacing and Its need

- A microprocessor needs to be interfaced to memory, data storage and input/output devices for communicating with the environment.
- A microprocessor designer has to face task of
  ⇒ selecting microprocessors, memories, and I/O devices from a large variety.
  ⇒ reducing the cost of system without loss in performance.

The output of this task may be in form of collection of incompatible devices to built a system; therefore suitable interface circuits need to be built.

- The incompatibility between any two devices could arise because of the following differences:
  1. Timing according to which data transfer is to take place.
  2. Format in which data is to be tranfered.
  3. Electrical characteristics of the two devices.
- For resolving the above mentioned differences the following circuitry or interface is required:
  → For resolving timing differences, the processor must be capable of waiting for the interfaced device for as long as desired or else an external buffer would be required.
  → For resolving data transfer format differences the microprocessor needs external chip to perform conversion of data
  → Differences in electrical characteristics such as voltage and current can be resolved using level translators and current drivers as interfaced elements.

Interfacing Devices

→ Any microprocessor based system design involves interfacing of the processor with one or more peripheral devices for purpose of communication with environment.

→ Most of the general and special purpose peripheral devices are of single chip circuits.

→ Interfacing devices may be classified in to two categories.

* General purpose peripherals.
* Special purpose peripherals or dedicated function peripherals.

General purpose peripherals may be used for interfacing a variety of I/O devices to the microprocessor.

⇒ For example

- An I/O port
- Programmable Peripheral Interface (PPI)
- Programmable Interrupt Controller (PIC)
- Programmable DMA Controller
- Programmable Communication Interface
- Programmable Interval Timer (PIT)

Special Purpose Peripherals are devices that may be used for interfacing a microprocessor to a specific type of I/O device. These peripherals are much more complex and relatively more expensive than the general purpose peripherals.

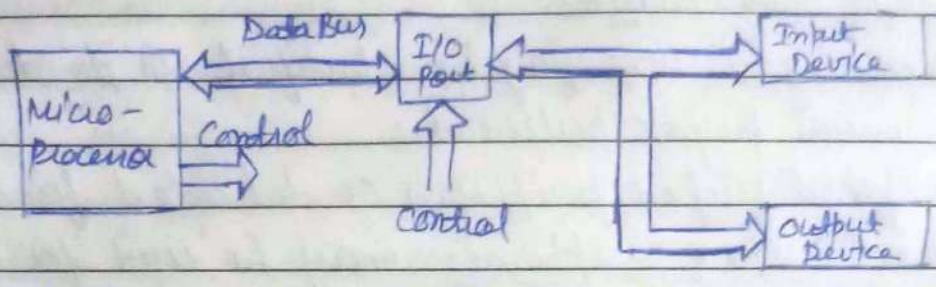
⇒ For example

- Programmable CRT Controller
- Programmable Floppy disk controller
- Programmable Hard disk controller
- Programmable Keyboard and Display Interface.

General purpose peripherals

→ Input/Output Ports:

- An input/output port is a place for the unloading and loading of data.



⇒ Data transfer from input device begins with the device unloading data into the port. Subsequently processor loads it into one of its register from the port.

⇒ For data output, the microprocessor unloads data into ports and output device loads it into its internal register.

- The use of I/O ports reduce the chip count and external decoding logic in a microprocessor based system.
- I/O port consists several lines for data transfer and a few control lines.

- Programmable I/O ports
  - → It is the one in which the attributes associated with port can be programmed using I/O instructions like IN and OUT.
  - → A port may be programmed for one or more of the following attributes:
    - Simple input or output mode
    - Strobed input or output mode
    - Control mode

→ Programmable Peripheral Interface

· Interfacing peripheral devices to microprocessor is a very common activity. Several single chip devices are available.
· The 8255 Programmable Peripheral Interface (PPI) is one such chip using which data or control signals can be transferred between microprocessor and peripheral devices.
· The 8255 provides three 8-bit ports that can be programmed by the microprocessor to obtain a configuration which best suits a designers needs.

→ Programmable Interrupt Controller

· Several I/O devices are to be interfaced to microprocessor using interrupt driven transfer, one may either use the polled or the vectored interrupt scheme or a combination of both.
· Programmable Interrupt controller are designed to perform these two tasks.
· 8259 is a single chip, PIC, it receives an interrupt from an I/O device, checks its priority and mask and interrupts the microprocessor. After recognizing the interrupt the microprocessor acknowledges this interrupt from the PIC. In response to acknowledge signal, the PIC sends three byte CALL instruction to microprocessor.
· One 8259 can receive up to 8 interrupts and can handle.

→ Programmable DMA Controller

· During DMA transfer various buses are controlled by the I/O devices that has been granted the DMA request.
· I/O devices are normally not designed with all the circuitry to perform DMA transfer. Some chips are used to perform all the chores necessary for a succesful DMA operation. Intel's 8257 is such a programmable DMA Controller.

⇒ 8257 interfaces upto four I/o devices to the microprocessor for data transfer using DMA.
⇒ It has four separate channels, each consists of two 16-bit registers; out of these two registers one is used for address and other one is for count of bytes.
⇒ Three different operations may be performed
• Read • Write • Verify

→ Communication Interface

• Many application require data transmission in serial form For doing this there are few chips are available,
• Intel 8251 is an example of such a chip.
• It is designed for synchronous / asynchronous serial data transfer.
• It receives parallel data from microprocessor and transmits serial data after conversion of parallel data to serial data. Also receives serial data from outside and transmits parallel data to microprocessor after conversion of serial data into parallel data.

→ Programmable Interval Timer

• The most common problem in any ~~mi~~ microprocessor based system is ~~of~~ generation of accurate time delay. It is solved by using PIT.
• The 8254 is such a programmable interval timer from Intel. It has three independent 16-bit counters. Each counter can be programmed to operate in any one of different modes.

Special Purpose Peripheral Interfacing Device

→ Programmable Keyboard / Display Interface

- These are used for interfacing the keyboard and the display to microprocessor.
  - ⇒ The keyboard section ~~converts~~ of the device debounces the keyboard entries and provides data to microprocessor in desired format.
  - ⇒ The display section converts the data output of the microprocessor into the form desired by display device in use.
- 8279 is general purpose programmable keyboard and display I/O interface device designed for use with Intel microprocessor.
  - ⇒ The keyboard portion can provide a scanned interface to a 64 contact key matrix
  - ⇒ Display portion provide a scanned display interface for LED and other popular technologies.

Both numeric and alphanumeric segment display may be used. The 8279 has 16×8 display RAM.

→ Programmable CRT Controller

- It is a device to interface CRT raster scan displays with the microprocessor system.
- Its primary function to refresh display by buffering the information from the main memory and keeping track of the display position of the screen
- 8275 is programmable CRT controller.
- It allows a simple interface to almost any raster scan CRT display with a minimum of external Hardware and software.
- The number of display characters per row and number of character rows per frame are software programmable.

→ Floppy Disk Controller
- It is used for disk drive selection, head loading, the issue of read/write commands, data separation, serial to parallel and parallel to serial conversion of data.
- Floppy disk controllers include Intel's 82078, 8272.

→ Programmable Hard disk controller
- Modern computer platforms are expected to provide a specially designed peripheral processor such as hard disk controller which can be programmed by system software to carry out data transfers between a disk and computer's primary memory.
- Programming interface for hard disk controllers continues to evolve, but a stable set of standard capabilities exists.

→ IDE Controller (Integrated Device Electronics)
- It is most commonly used for Hard drives and CD or DVD drives. It can also be used for tape drives and zip drives.
- It is a primary interface which any IDE device will be plugged into.
- It handles the flow of data to and from these devices.
- It consists of two channel and both of which can handle up to 2 devices each.
- It is also possible to add more IDE controllers in form of PCI cards.

→ SATA Controller
- It is an interface for connecting bus adapters to mass storage devices.
- SATA offers advantages are
  ⇒ Reduced cable size and cost
  ⇒ faster data transfer rates
  ⇒ more efficient transfer.