MP UNIT 1 - microprocessor unit 1 notes

Microprocessor (Dr. A.P.J. Abdul Kalam Technical University)

UNIT-1

Evolution of Microprocessors

4 bit Microprocessor (1st Generation) :- The first μp was introduced in 1971 by intel. It was named intel 4004 as it was a 4 bit μp.

*) It could perform simple arithmetic (like addition and subtraction) & logical (like AND, OR) operations.

8 bit Microprocessor :- (2nd Generation from 1974 to 1978)

*) The have 8 bit data bus.

*) They have 5 times greater speed than 4 bit processors and 5 times less area and execution time.

*) exp ⇒ 8080 and 8085 by intel, Motorola 6800 & 6801 Zilog-80.

16 bit Microprocessor :- (3rd Generation, 1978 to 1982)

*) Third generation μp's uses HCMOS technology and implement RISC based architecture.

*) exp ⇒ 8086 and 80286

32 bit Microprocessor :- (4th Generation, 1982 to 1993)

*) uses HCMOS technology

*) can process multiple instructions per clock cycle.

*) exp ⇒ intel 80386, 80486, Motorola 68020

64 bit Microprocessor :- (5th Generation, From 1993 onwards)

*) They have more than 10 millions (1 crore) transistors on it.

*) 64 bit data bus

Name	Year	Transistors	Data bus	Clock speed	Memory
8080	1974	6000	8 bit	2 MHz	64 KB
8085	1976	6500	8 bit	5 MHz	64 KB
8086	1978	29K	16 bit	5 MHz	1 MB
80286	1982	134K	16 bit	6 MHz	16 MB
80386	1985	274K	32 bit	16 MHz	4 GB
80486	1989	12M	32 bit	25 MHz	4 GB
Pentium	1993	31M	32/64 bit	60 MHz	4 GB
Pentium II	1997	75 M	64 bit	233 MHz	64 GB
Pentium III	1999	95 M	64 bit	450 MHz	64 GB
Pentium IV	2000	420M	64 bit	1.5 GHz	64 GB

Microprocessor Architecture and operation of its components

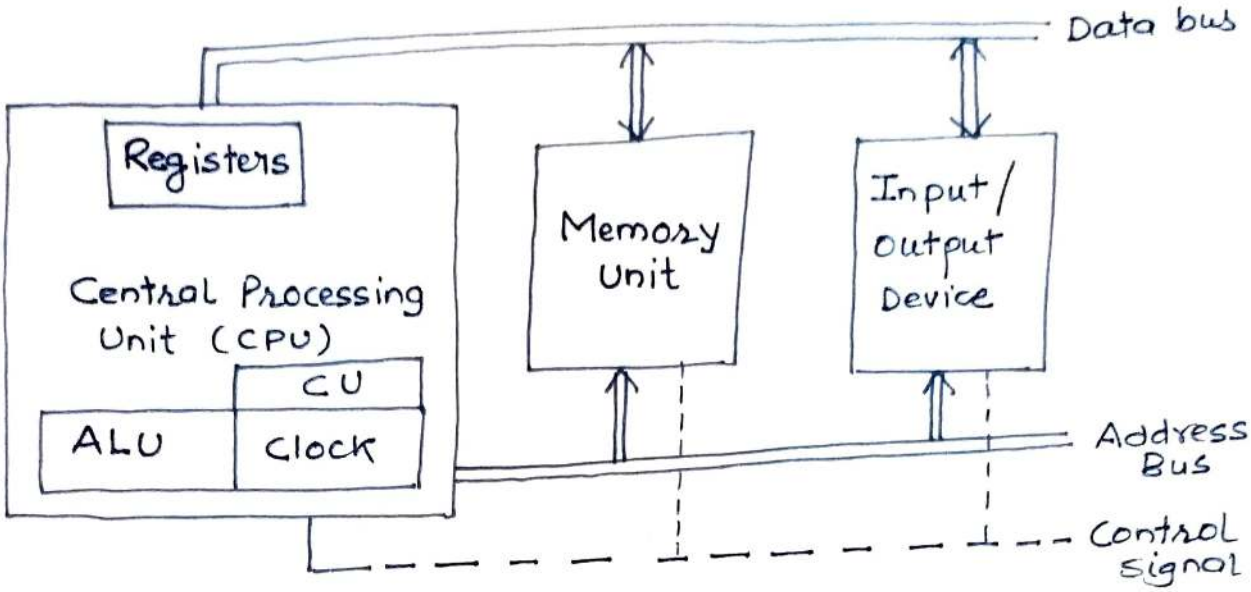


Fig: General architecture of microprocessor

Arithmetic and Logic Unit (ALU) -: This unit performs mathematical computations such as addition, subtraction, division and boolean functions (logical operations like AND, OR etc). The ALU also executes comparisions and logical testing.

*) The CPU or processor transmits signal to the ALU, which interprets the instruction and perform the calculation.

Registers -: Registers are used to hold or store binary data temporarily. These registers can be used to store general data, instruction codes, intermediate result of any logical or mathematical operation.

Some registers are also used as part of a instruction such as Accumulator (A). most common registers of microprocessor are Program counter (PC), stack pointer (SP), intruction register.

Control Unit -: Control unit consists of different components such as Instruction decoder, clock circuit, control logic circuits. All these components work together to receive and transmit signals from different components of microprocessor.

for example *) The Instruction decoder interprets the instruction and take action according to the instruction.

*) The clock sends signal that synchronize and ensure timely execution of instruction and processes.

Buses -: Microprocessor have a system of buses to move different type of data.

*) The data bus transfers data between the CPU and RAM.
*) The control bus sends necessary information to coordinate and control multiple tasks.
*) The address bus holds the address of RAM/ROM locations.

Cache Memory -: Some advance microprocessors have memory cache, which retains the last data used by the CPU.

*) Memory caches speed up the computing process, because the CPU does not have to go to the slower RAM to retrieve data.

Addressing Modes

The operation field of an instruction specifies the operation to be performed. The way any operand is selected during the program execution is dependent on the addressing mode of the instruction. The format of instruction is given below

opcode	operand

operation code specifies the type of operation

operand can be 1, 2 or 3 which specifies where the operation has to be performed.

The purpose of using addressing mode is as follows

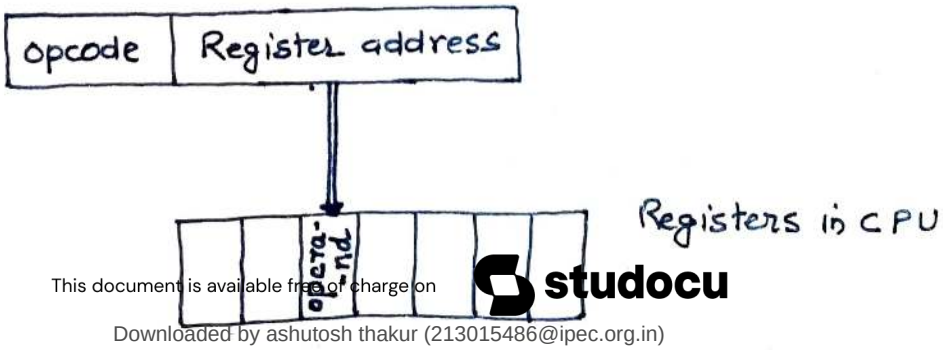
i) To give the programming versatality to the user.
ii) To reduce the number of bits in addressing field of instruction.

Type of Addressing Modes -:

1) **Immediate mode -:** In this mode operand is specified in the instruction itself.

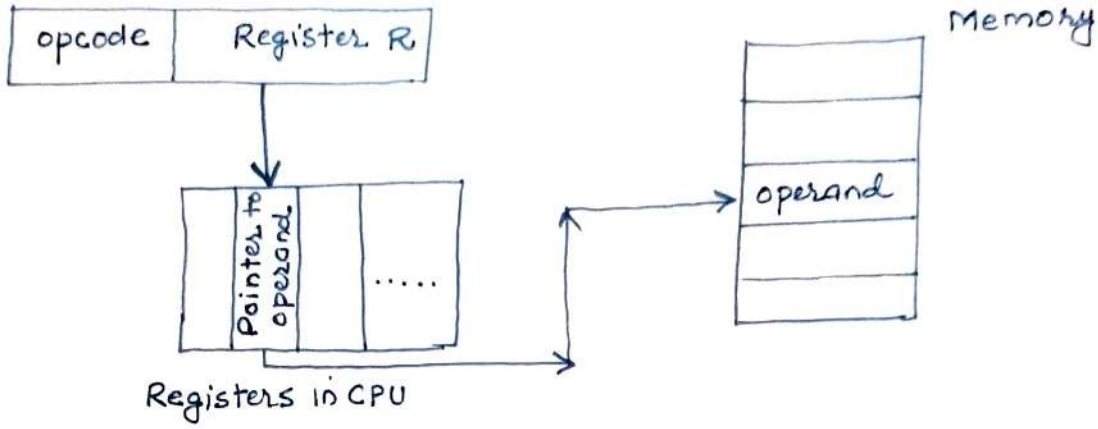
Exp= ADD 7 ; which says Add 7 to the contents of Accumulator 7 is the operand here.

2) **Register mode -:** In this mode the operand is stored in the register

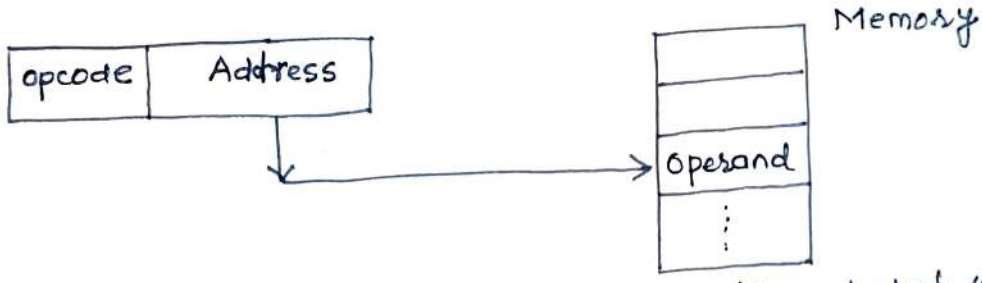


3) Register indirect mode -: In this mode, the instruction specifies the register whose content give us the address of operand stored in memory.

*) The register contains the address rather than operand itself.



4) Direct addressing mode -: In this mode, effective address of operand is present in the instruction itself.

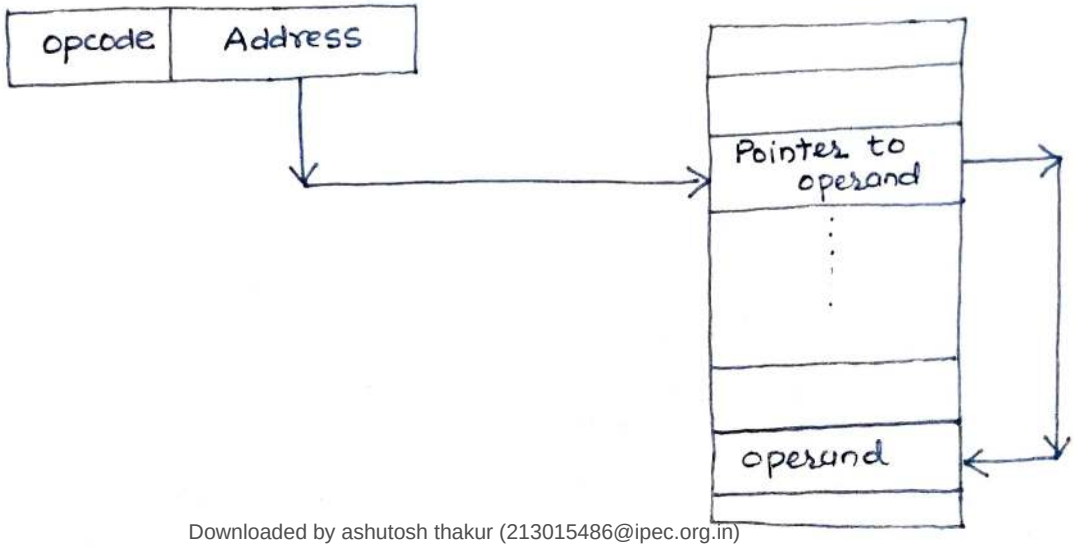


Exp⇒ ADD B, 4000H ; add the content of B with content of 4000H.

Here 4000H is the effective address where operand is present.

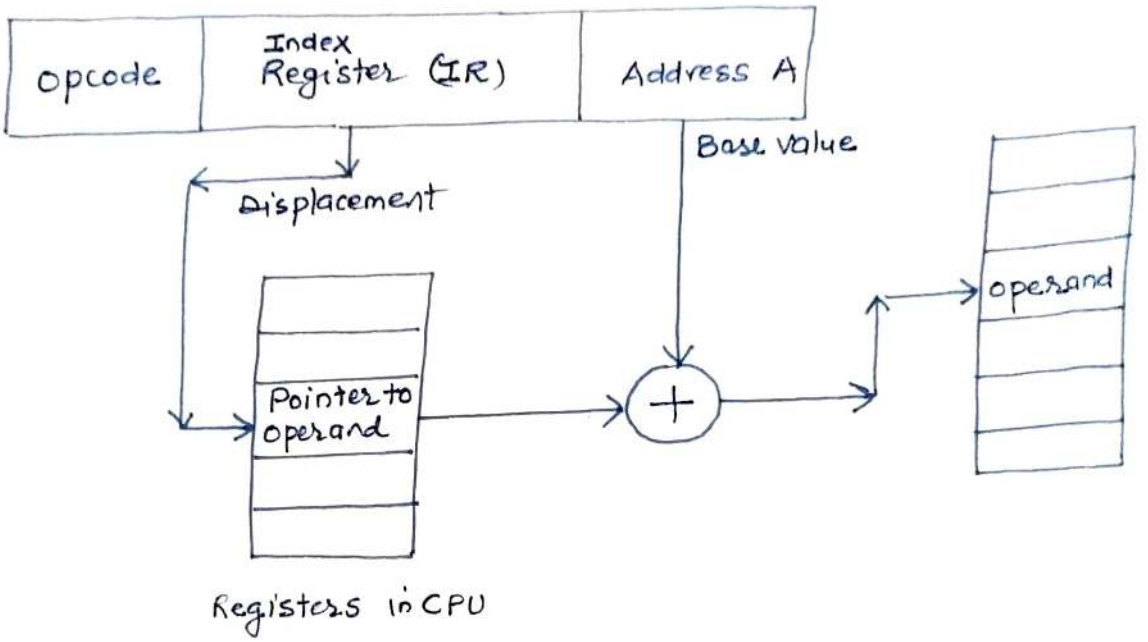
5) Indirect addressing mode -: In this mode, the instruction will give the address where the effective address is stored in the memory.

*) This mode slows down the execution.



6) Displacement or Index addressing mode :- In this mode the content of indexed register is added to the address part of the instruction, to obtain the effective address of operand.

Effective address = Address given in instruction + indexed register value



Exp ⇒ ADD R4 , 100 (R1)

↳ Index register

→ Base value

EA = 100 + (R1)

*) The Displacement can be defined by Indexed register, Program counter or any normal register.

⇒ if the displacement is defined by program counter

EA = A + (PC)

Exp ⇒ ADD R4 , 100 (PC)

EA = 100 + (PC)

⇒ if the displacement is by normal registers only.

EA = (R1) + (R2)

Exp ⇒ ADD R4 , (R1 + R2)

Here EA = (R1) + (R2)

Instruction Cycle

An instruction cycle is also known as fetch-decode-execute cycle. this process is repeated continuously by CPU from boot up (starting / turn ON) to shut down of computer.

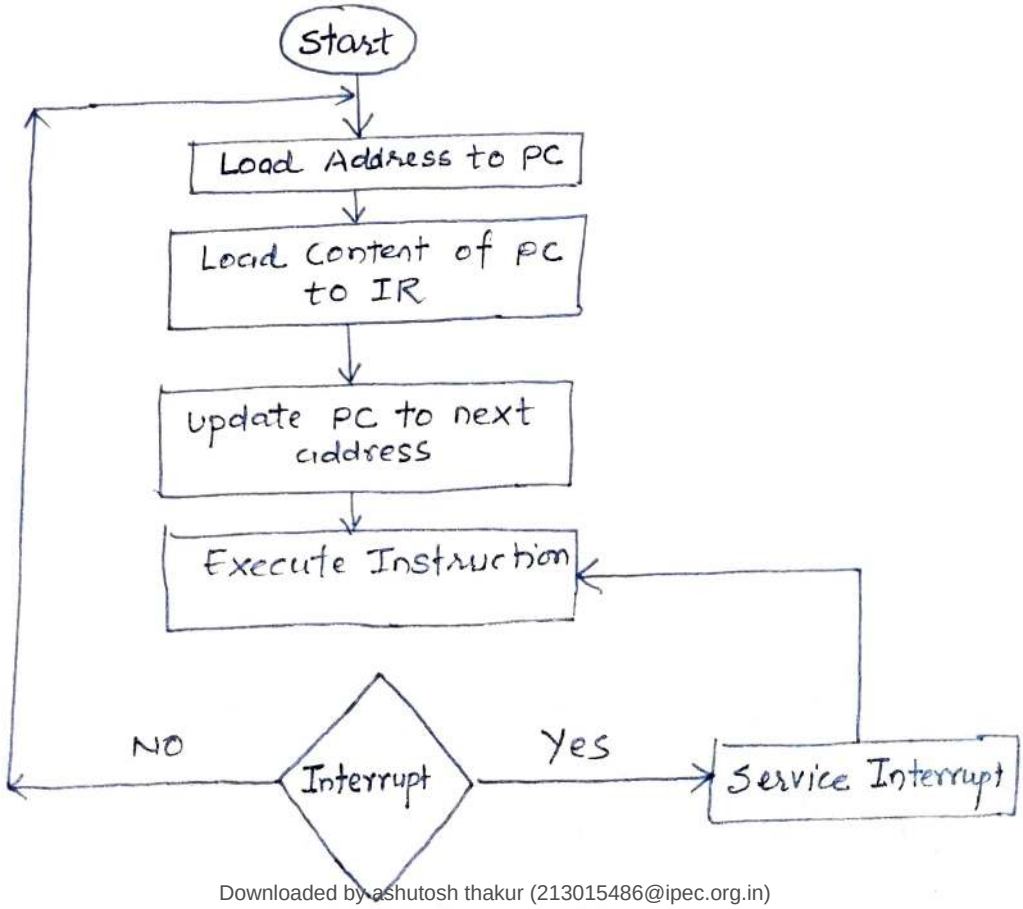
following are the steps that occur during the instruction cycle.

1) Fetch the instruction -
*) The instruction is fetched from the memory.
*) the address of the memory location will be given by the program counter (PC).
*) After fetching the instruction from memory it is placed in IR (Instruction Register).
*) After fetching operation PC is automatically incremented by 1.

2) Decode the instruction -: the instruction placed in IR is decoded by the decoder.

3) Read the effective address -: if there is an indirect address, the effective address is read from the memory.

4) Execution of instruction -: The control unit passes the signal to the functional unit of CPU to execute the operation given by instruction. the result generated is stored in the main memory or sent to an output device.



Data Transfer Schemes

We can connect several input/output and memory peripherals to a microprocessor. All these device may differ in the speed of operation and data transfer.

Usually, when memory is connected ~~to~~ with the microprocessor, there is not a major difference in the processing speed.

But the problem arises when external peripherals are connected as input/output. A slow i/o device won't be able to transfer data at a satisfactory rate. This might lead to severe data losses, or the device might get damaged. to avoid this problem, a number of data transfer schemes have been introduced

Data Transfer Schemes

Parallel Data Transfer

1) Programmed I/O Data Transfer
   i) Synchronous
   ii) Asynchronous
2) Interrupt Driven Data Transfer
3) Device or DMA Data Transfer
   i) Burst or Block Transfer DMA
   ii) Cycle steal or single ~~cycle~~ byte transfer DMA
   iii) Transparent or hidden DMA

Serial Data Transfer

1. Synchronous Data Transfer
2. Asynchronous Data Transfer

Programmed I/O Data Transfer

*) Data are transferred from an I/O device to CPU/memory and vice-versa.
*) this type of transfer is used when small amount of data has to be transferred.
*) IN and OUT instructions are used in synchronous transfer.
*) READY signal is used in Asynchronous transfer. this mode is also known as handshaking mode.

Interrupt Driven data Transfer -:

*) The problem with programmed I/o transfer is that CPU has to wait long time for the I/o device to be ready for transmission or reception of data.

*) In the interrupt driven mode, the i/o device raises (HIGH) a special signal called Interrupt, when he becomes ready to transfer data.

DMA Transfer -:

*) This type of transfer is used when there is a huge amount of data have to be transferred.

*) The CPU releases the control of buses to external device for data transfer.

*) A peripheral device called DMA controller takes over the buses and manage the transfer directly between the peripheral and the memory.

*) This is the fastest transfer scheme among all the schemes.

Timing Diagram

The timing diagram of microprocessor represents following terms -

i) No. of clock cycles used
ii) Duration, delay and content of address bus
iii) Type of operation (read / write / status signal)

With the help of timing diagram, one can understand the working of each instruction and its execution.

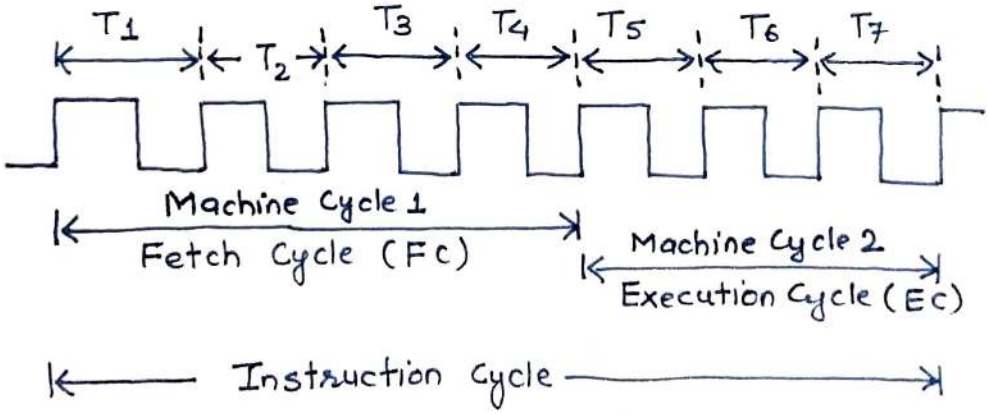
Important terms related to timing diagram

1. Instruction cycle -: It is defined as the no. of steps (clock cycles) required by the CPU to complete the entire process (Fetching and execution of instruction).

2. Machine Cycle -: It is the time required by the processor to complete the operation of
   i) accessing the memory
   or
   ii) accessing the I/O device

   In machine cycle various operations are performed like
   *) opcode (operation code) fetch from memory
   *) Memory read or write
   *) I/O read or write

3. T-states -: Each clock cycle is called as T-state.



Instruction cycle = opcode fetch cycle (Fc) + Execution Cycle (Ec)
↓ Machine Cycle 1 ↓ Machine cycle 2

process during opcode fetch -:

*) During T1 state -: PC generates [redacted] address, where code is stored.

*) During T2 state -: opcode is ready to be read by the processor.

*) During T3 state -: opcode is stored in the instruction register.

*) During T4 state -: processor will decode the opcode and provide necessary actions.

Interrupts

Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.

The microprocessor responds to that interrupt with an ISR (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt.

- Interrupt
  - Hardware Interrupt
    - Maskable Interrupt
    - Non maskable Interrupt
  - Software Interrupt

Hardware Interrupts -: A hardware interrupt is normally created by the external device such as mouse, keyboard, pendrive etc.

*) Whenever we click a mouse or tap on a touch screen, we send an interrupt signal to the processor.

*) Each input and output device has a unique interrupt service queue setting or priority so that multiple devices do not create conflict.

i) **Maskable Interrupts** -: these interrupts can be enabled/disabled by using programming instructions.

Exp- INTR

ii) **Non Maskable Interrupts** -: these interrupts have high priority than maskable interrupts. the processor can not ignore the non-maskable interrupt in any circumstances.

Exp- timeout signal from timer circuit.
power down signals

Software interrupts -: Software interrupts are used to handle errors and exceptions that occur while a program is running.

*) these interrupts allows the program to handle the error before continuing.

*) these interrupts are also used to break an infinite loop, which could cause a program to be unresponsive.

Memory Address	Mnemonics		
2000	LXI	SP, 2400(H)	
2003	LXI	B, 0000(H)	
2006	PUSH	B	
2007	POP	PSW	
2008	CALL	DELAY	; DELAY starting at 2064(H).
200E	OUT	01(H)	
2010	HLT		

Save to STACK

2064	DELAY :	PUSH	H
2065		PUSH	B
2066		LXI	B, 80FF(H)
2069	L1 :	DCX	B
206A		MOV	A, B
206B		ORA	C
206C		JNZ	L1
206F		RET	

*) When CALL instruction executed, the program flow will transfer from main program to subroutine (DELAY). this is accomplished by placing the starting address of subroutine into program counter.

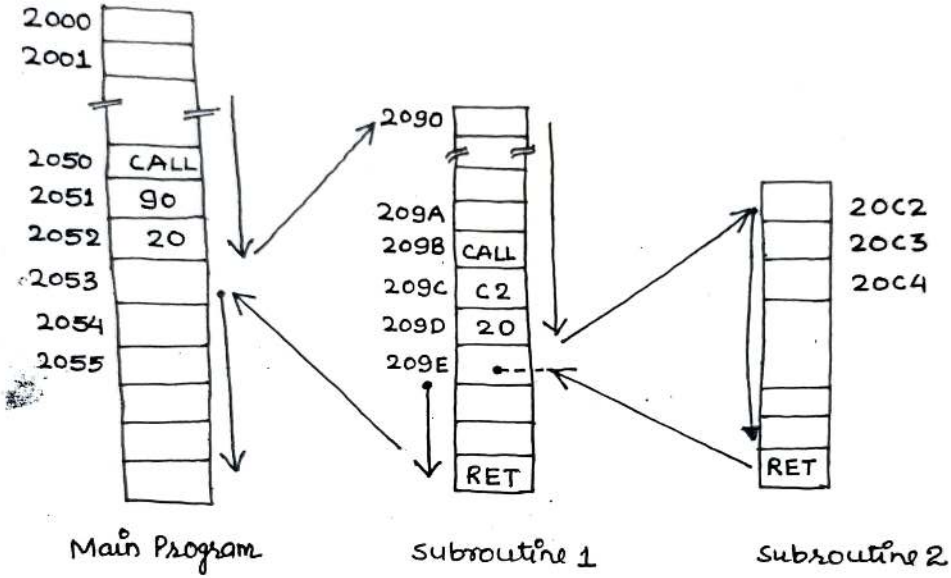
*) Before changing the content of PC, the address of next to CALL instruction (200E) must be saved, so that after execution of subroutine, main program can execute.

*) to save the content of PC (200E) internal PUSH operation is performed by the microprocessor. after storing the content of PC, new content is loaded into program counter.

*) When RET is executed the content of STACK is retrieved and loaded into program counter.

Advanced Subroutine Concepts :- From till now we have seen that we can call a subroutine multiple times. Other type of subroutines are also used which are discussed here.

Nesting -: The programming technique of a subroutine calling another subroutine is called nesting. when a subroutine calls another subroutine, all return address are stored in STACK.

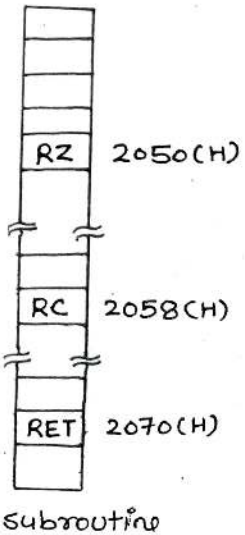


Multiple Ending Subroutine :-

In this technique when a subroutine is executed, there are 2 conditional returns (RZ and RC) and one unconditional Return (RET).

*) if the zero flag is set i.e Z = 1 then subroutine will return from 2050(H).

*) if the carry flag is set i.e C = 1 then subroutine will return from 2058(H).



RISC and CISC Architecture

1. RISC – It stands for "Reduced Instruction Set Computer." It is a type of processor architecture that uses a small set of simple instructions of uniform length. these instructions are normally executed in one clock cycle. (one instruction in one clock cycle).

2. CISC – It stands for "complex Instruction Set computer" These processors have hundreds of instructions (large instruction set) of variable sizes. these instructions can interact with internal registers as well as internal memory to access data. each instruction execution requires one or more than one clock cycles.

Difference between RISC and CISC Architecture

RISC	CISC
*) very few instructions (< 100)	*) large set of instructions (>100)
*) Main focus is on software i.e compiler has a complex functioning	*) Main focus is on hardware so compiler is simple.
*) Fixed size instructions	*) Variable size instructions
*) Can perform only register to register arithmetic operations.	*) can perform Reg to Reg, Reg to Mem and Mem. to Reg. operations
*) each instruction gets executed in one clock cycle.	*) each instruction require one or more than one clock cycles.
*) It does not support Array operations	*) It supports array operations.
*) Program code is large so large memory required to save the program code.	*) program codes are short, which require lesser memory to save the program code.
*) RISC processors have large set of registers.	*) It has small set (or very few) of registers.
*) It has simple hardware.	*) Hardware is complex.
*) Simple hardware leads to less energy consumption	*) Complex hardware require more energy.
*) Heavy use of RAM	*) Efficient use of RAM.
*) Few (3-4) addressing modes	*) large (10-24) addressing modes.
exp ⇒ intel X86 processors, AMD processors used in Laptop & desktops.	exp ⇒ Used in smart watches, phones, printers, tablets, home automation systems etc. (ARM processors) (AVR processors)

Logic Devices for Interfacing

1) Tri-state Devices :-

*) Tri-state devices has three (3) logic states -
i) logic '0' ii) logic '1' iii) High Impedance (z)

*) Other than I/P and O/P line, these devices has 3rd line called "Enable".
When Enable is active, the device works as normal device
When Enable line is deactivated (disabled), the device goes into high impedance (z) state.

*) In High impedance state no current is drawn from the system.

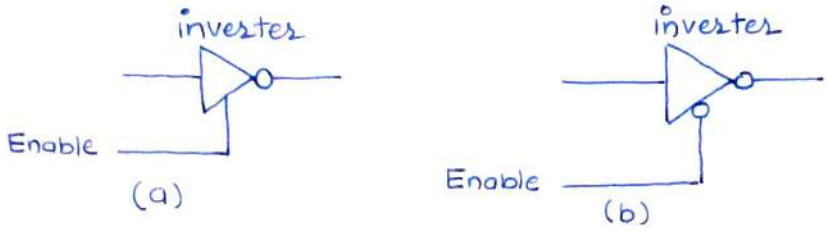
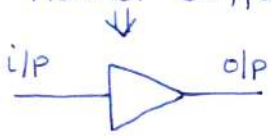


Fig: Tri-state inverter with a) Active high Enable line
b) Active low Enable line

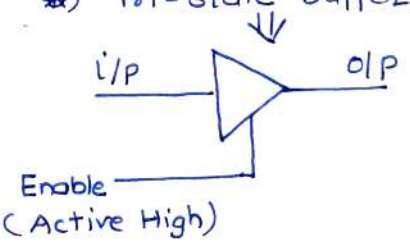
2) Buffers -:

*) Buffers are the logic circuit that amplifies the current or power.

*) Normal buffer



*) Tri-state buffer



*) Buffers are used to increase the driving capability of a logic circuit. (Data & Address bus)
*) They are also known as a 'driver'.

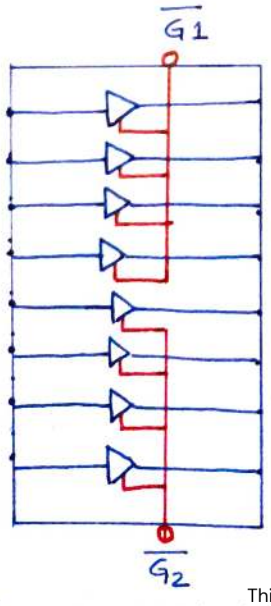


Fig: 74LS244 Address bus driver

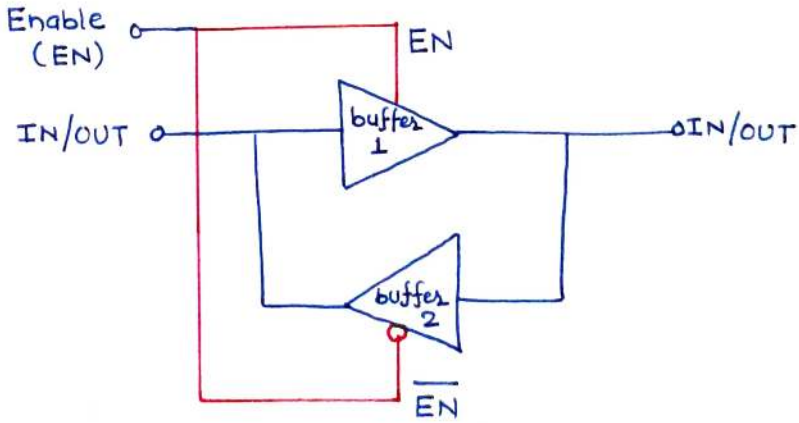


Fig: Bi-directional Tri-state buffer 74LS245 used in data bus

3) **Decoder** -: *) Decoder has n- input lines and 2^n output lines.

*) Decoder activates only one o/p lines based on the logic combination of input lines.

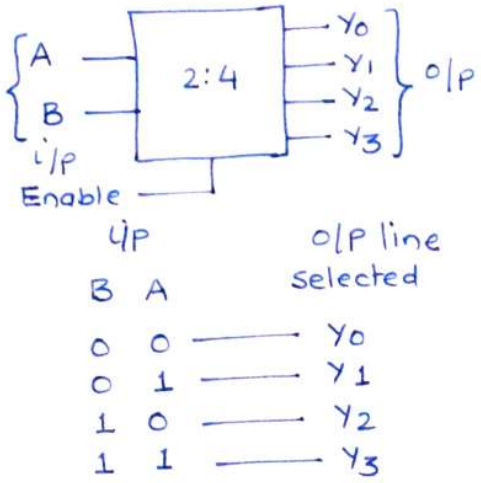


Fig: 2:4 Decoder with enable signal

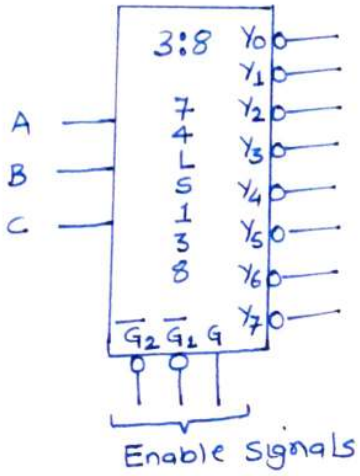


Fig: 74LS138 decoder (3:8)

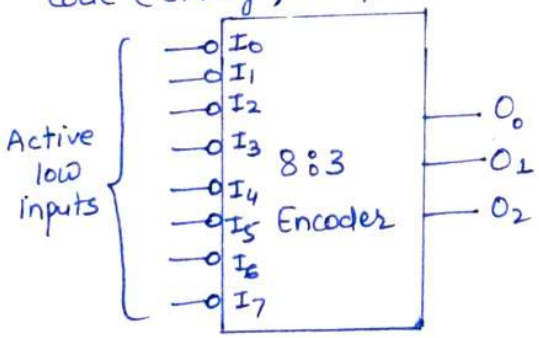
*) To active this decoder Enable signals should be

$\overline{G_2} = 0$, $\overline{G_1} = 0$, $G = 1$

*) Decoders are widely used in interfacing of I/O peripherals (to select a particular peripheral) and Memory.

4) **Encoder** -: *) Encoders are logic Circuits that provides the appropriate code (Binary or BCD) as output for each input signal.

*) Encoders are normally used with keyboard. for each key, an appropriate code (binary) is placed on the data bus.



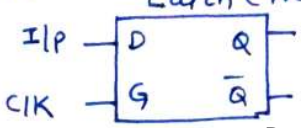
When $I_0 = 0$, binary code 000 will be generated at o/p

$I_7 = 0$, binary code 111 will be generated at the o/p.

5) **D-Flip-Flops** -: (Latch & clocked)

*) A latch is used to interface normally output devices. when µp sends data to o/p device, data are available on the data bus for a short period of time (few µsec.). therefore a latch is used to hold the data.

Latch (Transparent D-flip flop)



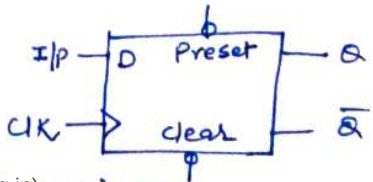


Fig: D Flip Flop

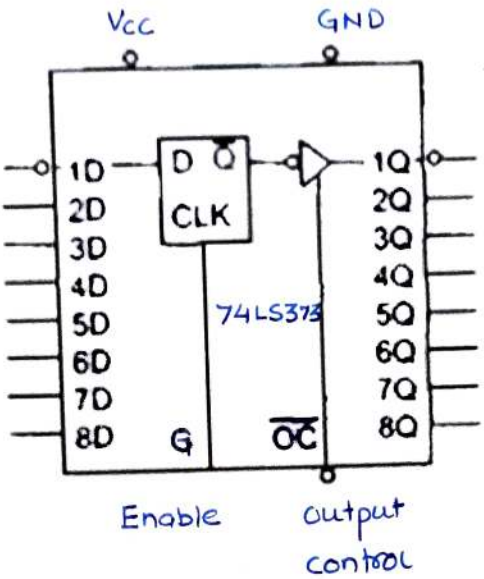


Fig: 74LS373 D Latch used in Latching of 8 bit data