Microprocessor and Computer Architecture

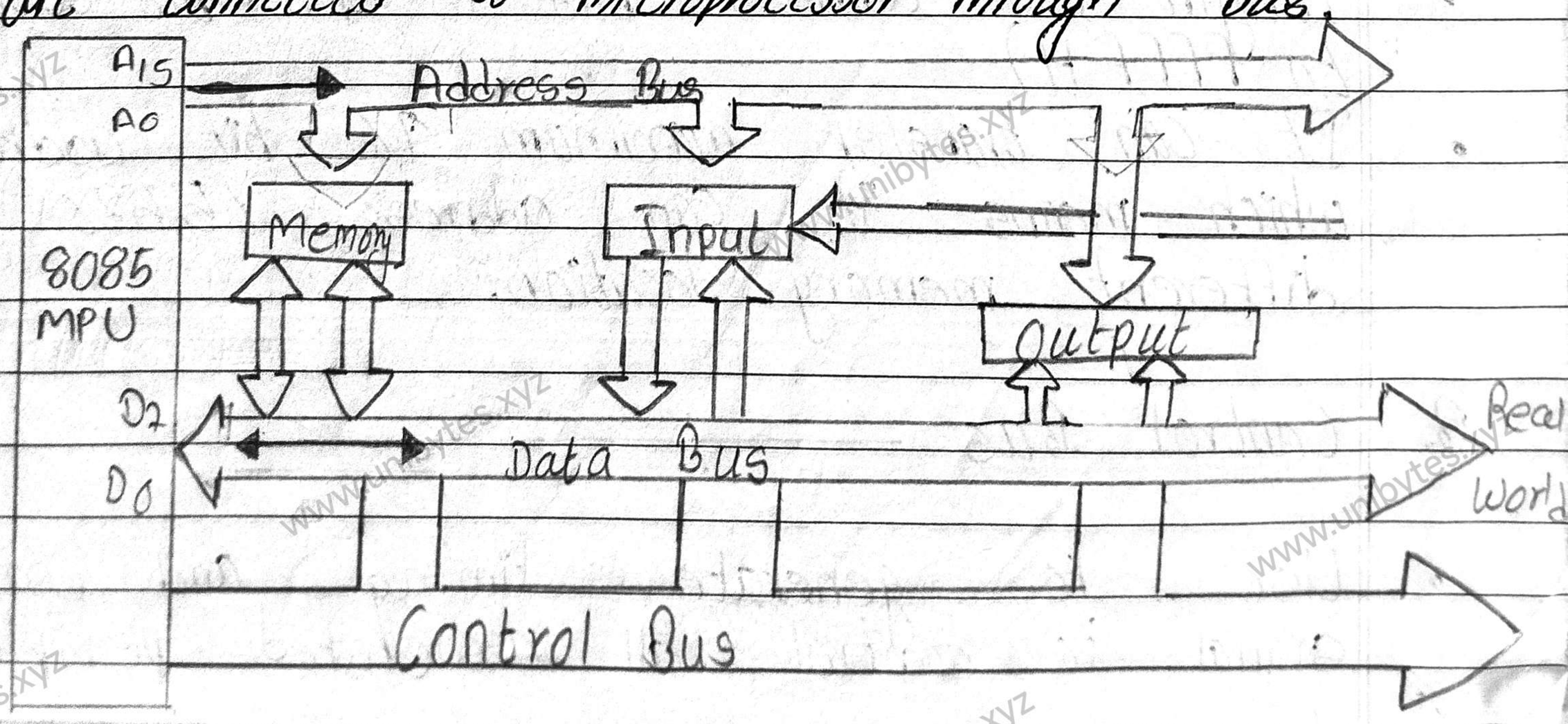
Microprocessor

- It is brain of computer
- It is a single chip which is capable of processing data.
- It controls all the components in computer e.g Monitor, Keyboard, USB etc.
- It fetch, decode and execute the instruction

A microprocessor is a programmable electronic chips that have computing and decision making capacity similar to central processing unit of computer.

Bus Organization of 8085 Imp

↳ Bus is a group of conducting wires which carries information all the peripherals are connected to microprocessor through bus.



1. Data Bus

- Conducting wire carries data only.
- Bidirectional ~~to memory~~ because data flow in both direction from microprocessor to memory or i/o device.
- Length of data bus of 8085 is 8 bit (00H to FFH)
- The width of data bus is directly related to the largest number that bus can carry such as 8 bit bus can represent 2^8 (0-255).

2. Address Bus

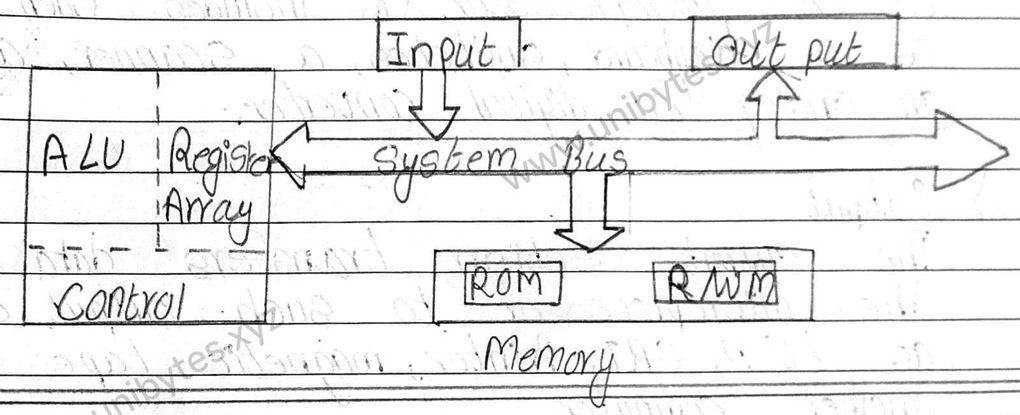
- It is a group of conducting wires which carries address / particular location only.
- It is unidirectional only i.e data flows from microprocessor to memory or from microprocessor to I/O device.
- Length of address bus is 16 bit (0000H to FFFFH)
- It can transfer maximum 16 bit address which means it can address 65536 different memory location.

3. Control Bus

- Use to generate timing and control signal to control all associated peripherals.
- Uses control signal to process data that is what to do with selected memory location. Some control signal are:

i) Memory read
ii) Memory write
iii) I/O read
iv) I/O write
v) Opcode fetch

* Microprocessor Architecture and Operation



1) ALU (Arithmetic / Logic Unit)

It perform such arithmetic operation as addition and substraction, and such logic operations as AND, OR and XOR. Results are stored either in register or in memory.

2) Register Array

It consists of various registers identified by letter such as B, C, D, E, H, L, IX and IY. These register are used to store data and addresses temporarily during the execution of program.

3) Control Unit
The control unit provides the necessary timing and control signals to all the operations in the micro computer. It controls the flow of data between the microprocessor and memory and peripherals.

4) Input
The input section transfers data and instructions in binary from the outside world to the microprocessor. It includes such devices as a keyboard, switches, a scanner, and an analog to digital converter.

5) Output
The output section transfers data from the microprocessor to such output device as LED, CRT, Printer, magnetic tape or another computer.

6) Memory
It stores such binary information as instructions and data, and provides that information to the microprocessor. To execute programs, the microprocessor reads instructions and data from memory and performs the computing operations in it's ALU section. Results are either transferred to the output section for display or stored in memory for later use.

7) System Bus
It is a communication path between the microprocessor and peripherals. The microprocessor communicates with only one peripheral at a time. The timing is provided by the control unit of the microprocessor.

Imp

	Microprocessor	Microcontroller
i)	CPU is stand alone, RAM, ROM, I/O and timer are separate.	CPU, RAM, ROM, I/O and timer all are on single chip.
ii)	Designer can decide amount of RAM, ROM and I/O ports.	Fixed amount of on-chip RAM, ROM and I/O ports.
iii)	High processing power.	Low processing power.
iv)	High power consumption.	Low power consumption.
v)	Typically 32/64 bit.	Typically 8/16 bit.
vi)	General Purpose	Single Purpose.
vii)	Less reliable	Highly reliable
viii)	E.g 8086, 8085	E.g 8051

www.unibytes.xyz

Processor Overview.

Processor	Year of Intro	No. of Transistor	Initial Clk speed	Address Bus	Data Bus	Addresable Memory
4004	1971	2,300	908 KHZ	10-Bit	4-Bit	6-10 Byte
8008	1972	3,500	200 "	14- "	8- "	16 K
8080	1974	6,000	2 MHZ	16-Bit	8-Bit	64 K
8085	1976	6,500	5 MHZ	16-Bit	8-Bit	64 K
8086	1978	29,000	5 MHZ	20-Bit	16-Bit	1 M
8088	1979	29,000	5 MHZ	20-Bit	8-Bit	1 M
80286	1982	134,000	8 MHZ	24-Bit	16-Bit	16 M
80386	1985	275,000	6 MHZ	32-Bit	32-Bit	4 G
80486	1989	1.2 M	25 "	"	32/64	4 G
Pentium	1993	3.3 M	60 "	"	"	"
Pentium Pro	1995	5.5 M	150 "	36-Bit	"	64 G
Pentium II	1997	8.8 M	233 "	"	64 Bit	"
Pentium III	1999	9.5 M	850 "	"	"	"
Pentium 4	2000	4.2 M	1.4 GHZ	"	"	"

Imp

Interrupts in 8085

Interrupts are the ~~single~~ signals generated by the external devices to request the microprocessor to perform a task. There are 5 interrupt signals, i.e TRAP, RST 7.5, RST 6.5, RST 5.5, and INTR

Interrupt are classified into following groups based on their parameter.

i) Vector Interrupt
In this type of interrupt, the interrupt address is known to the processor. In another word, we can say that the processor know which ports are interrupt and request processor time. For e.g RST 7.5, RST 6.5, RST 5.5, TRAP.

ii) Non - Vector Interrupt
In this type of interrupt, the interrupt address is not known to the processor so, the interrput address needs to be sent externally by the device to perform interrupt. For e.g INTR

iii) Maskable interrput
In this, we can disable the interrput by writing some instructions into the program. For e.g: RST 7.5, RST 6.5, RST 5.5

iv) Non- Maskable interrput
In this, we cannot disable the interrput by

ISR = Interrupt service routine

writing some instructions into the program.
For e.g : TRAP

v) Software Interrput
In this, the programmer has to add ~~execute~~ the instructions into the program to execute the interputs. There are 8 software interputs in 8085, i.e. RST0, RST1, ---- RST7.

vi) Hardware Interput.
There are 5 interput pins in 8085 used as hardware interput, i.e TRAP, RST7.5, RST6.5, RST5.5, INTA (Interrput Acknowledgment)

Note - INTA is not an interput, it is used by the microprocessor for sending acknowledgement. TRAP has the highest priority, then RST7.5 and so on.

* TRAP
It is non-maskable interput, having the highest priority among all interputs. By defaults, it is enabled until it gets acknowledged. In case of failure, it executes as ISR (interput Service Routine) and sends the data to backup memory. This interput transfers the control to the location 0024 H.

* RST 7.5
It is maskable interrput, having the second highest priority among all interputs. when this interrput is executed, the processor saves the content of the PC register into the stack and branches to 003CH address.

* RST 6.5
It is a maskable interrput, having the third highest priority among all interrput. When this interrput is executed, the processor saves the content of the PC register into the stack and branches to 0034 H address.

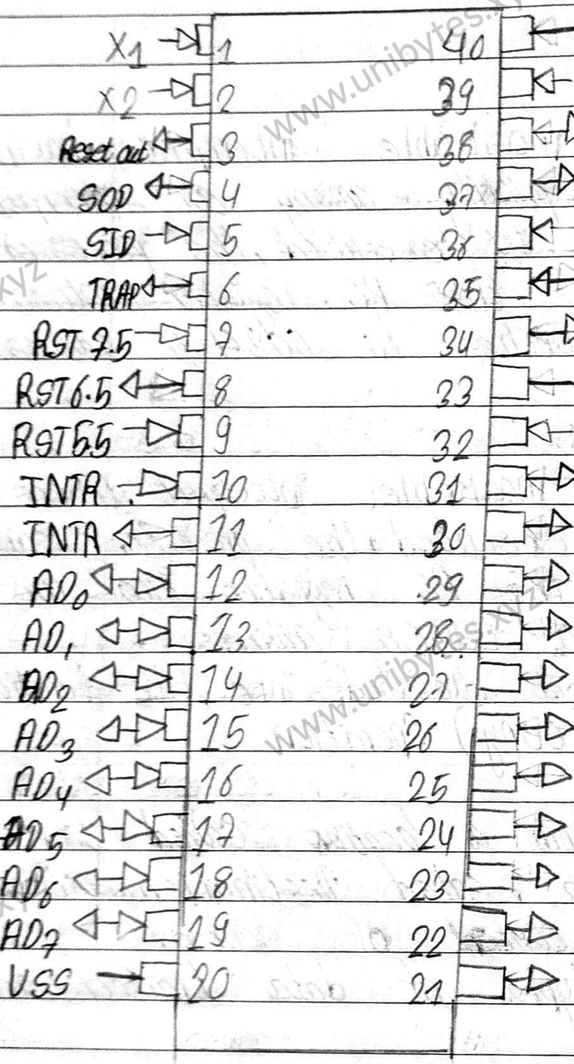
* RST 5.5
It is a maskable interrput. When this interrput is executed, the processor saves the content of the PC register into the stack and branches to 002CH address.

* 8085 instruction set classified into five functional categories
1. Data transfer (copy) Operation

- Copies data from a location called source to another location called Destination without modifying the content of source.
- The various types of data transfer are:
  - Between register
  - Specific data byte to register
  - Between memory location and a register
  - Between I/O device and accumulator

www.unibytes.xyz

* The following image depicts the pin diagram of 8085 microprocessor.



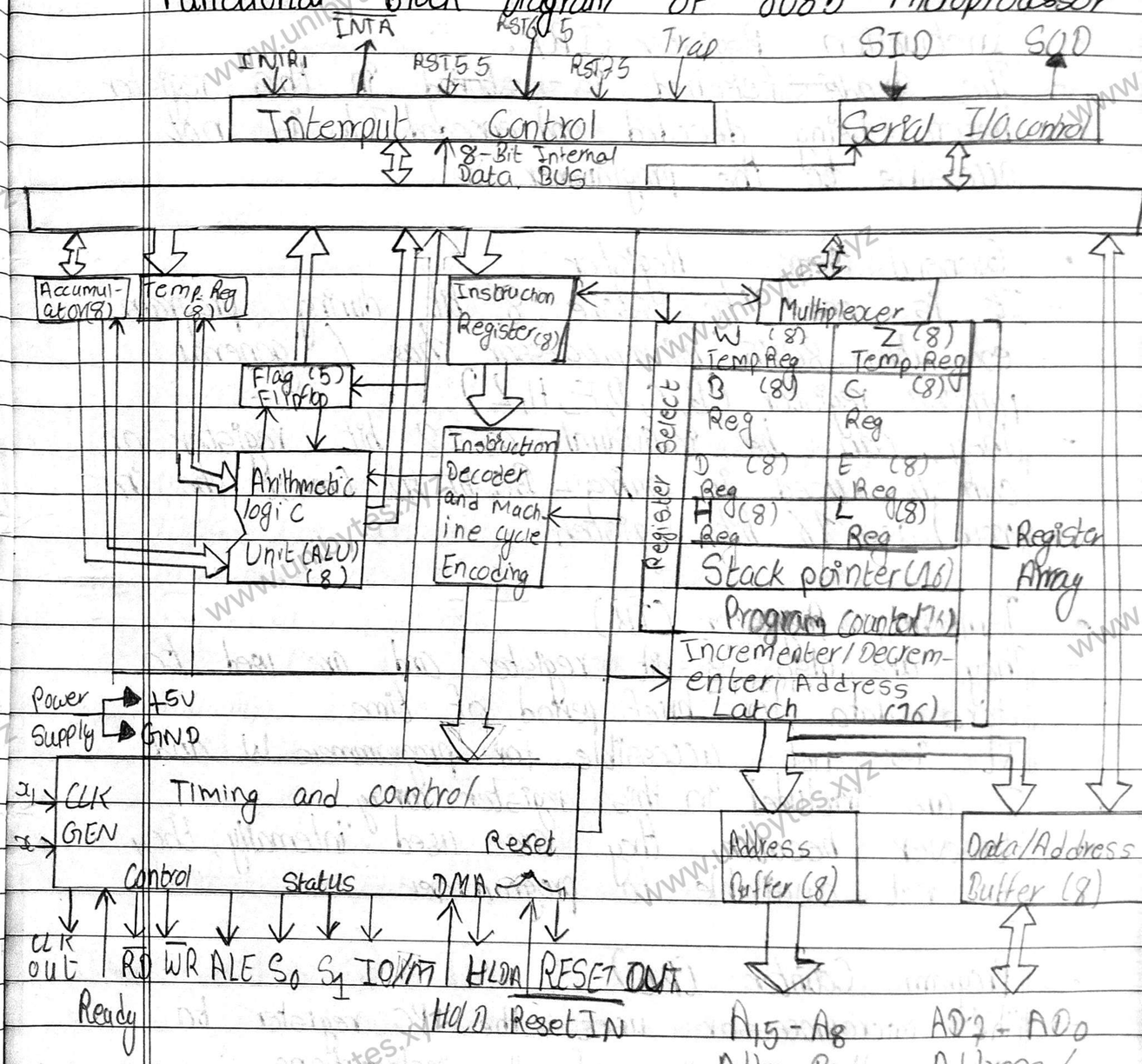
Microprocessor Architecture — VVVImp

It can run as 8 bit processor available in 40 pin IC package and was +5v for power.

- It can run at maximum frequency of 3MHz.
- It is data bus with 8 bit width and address bus 16 bit thus it can address 2^16 = 64Kb of memory.

Imp

Functional Block Diagram of 8085 Microprocessor



- Accumulator
  - This register is used to store 8-bit data and to perform arithmetic and logical operation.
  - Result is stored in Accumulator.
  - It is identified as A.

- Instruction Register (IR):
  - The opcode fetched is stored in this register before being decoded and encoded. It is not accessible to the programmer.

- General Purpose Register
  - It is used to store 8 bit during program execution, 8085 microprocessor has 6 general purpose register (B, C, D, E, H, L)
  - They can be individual as 8 bit register or can be used in pairs - BC, DE, HL (must be in row) as 16 bit register.

- Temporary Register (TR)
  - They are also 8-bit register and are used to store data for brief period of time.
  - It is not accessible for programmers. W and Z are included in this register array.
  - However because they are used internally, they are not available to programmer.

- Program Counter (PC)
  - The microprocessor uses the PC register to sequence the execution of the instructions.
  - The function of the program counter is to point to the memory address from which the next byte is to be fetched.
  - When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.
  - These are two 16-bit register use to hold memory address.
  - The size of these register is 16 bit because the memory address are 16 bit.
  - PC register used to sequence the execution instruction.
  - The function of PC is to point the memory address from which next byte is to be fetched.

- Stack Pointer (SP)
  - It is 16 bit register that points to the memory location of R/W memory called stack. It holds the address beginning of stack. It alway incremented/decremented by 2 during push & pop operation

Imp
- Flags or Flip-Flops
  - There are five flip-flops which are SET/RESET according to the result of operation and are called the flags. Types of flags are:

i) Carry (CY): Carry will be set (1) if carry/Borrow occurred in addition and substraction.

ii) Sign flag (S): It is set (1) when the MSB of the result of the last operation is 1.

iii) Zero flag (Z): It will be set if the whole result is 0.

iv) Parity flag (P): P will set if the number of 1 after remove the extra bit or carry or MSB came even then it will set.

v) Auxiliary Carry (AC): Carry from lower to upper nibble. Borrow from upper to lower nibble it will be set.

* Find the flags of

	upper	lower nibble
AB =	1010	1011
+ BC	1011	1100
MSB ← 1)	0110	0111

i) Carry = 1
ii) Sign = 1
iii) Zero = 0
iv) Parity = 0 (Number of 1's even)
v) Auxiliary Carry = 1

Branching Operation

This group of instruction alters the sequence of program execution either conditionally or unconditionally.

- JUMP
Conditional Jumps are an important aspect of the decision-making process in programming. These instructions test for a certain condition (e.g carry or zero flag) and alter the program sequence when the conditions is meet. In addition the instruction set includes an instruction called unconditional jump.

- Call, Return, and Restart : These instructions changes the sequence of program either by calling a subroutine or returning from a subroutine. The conditional call and return instructions also can test condition flags.

Logical Operation
These instructions performs various logical operations with the contents of the accumulator.

i) AND, OR, Exclusive-OR
Any 8 bit number, or the contents ANDed, ORed or Exclusive-ORed with the contents of the accumulator. The results are stored in the accumulator.

ii) Rotate - ~~Ex~~
Each bit in the accumulator can be shifted either left or right to the next operat

iii) Compare
Any 8 bit number or the contents of a register or a memory location can be compared for equality, greater than or less than with the contents of the accumulator.

iv) Complement
The contents of the accumulator can be complemented: all 0s are re-placed by 1s and all 1s are replaced by 0s.

8085 addressing Modes → (V.V. Imp)

i) ~~Immediate Addressing Modes.~~ Instruction are command to microprocessor to perform a certain task. Instruction consists of Opcode and data (Operand).
- The various formats of specifying the operand are called addressing modes.

i Types:

i) Immediate addressing modes.
- In this modes the 8/16 bit data is specified in the instruction itself as one of its operand.

Eg MVIB, 20H
20H is copied into register B.

ii) Register addressing modes.
- Data is copied from one register to another.
- In this mode, the address part of instruction specifies the memory location.
- This modes specifies the address of address rather than address itself as a part of instruction

MOV B, C

MOV B, C means data in register C is copied to register 'B'.

iii) Direct addressing modes

- In this data is directly copied from given address to the register
- Instruction of this mode specifies the effective address as part of instruction.
- E.g: ~~LDB~~ LDB 5000H means data at address 5000H is copied to register B.

iv) Register Indirect addressing modes
- Data is transferred from one register to another by using the address pointed by the register.

E.g MOV A, M means data is transferred from the memory address pointed by the register ~~to~~ B to the register K.

v) Implied addressing mode
- This mode doesn't require any OPERAND, the data is specified by OPCODE itself
- E.g CMP

MVI A 43
MOV
MOA 2050H
MOV K, B

↗ (Paxi ko page maa ~~ki~~ extra)

Timing diagram (V.Imp) - 5 marks fix

ans Timing diagram is a graphical representation. It represents the execution time taken by each instruction.

* Instruction cycle : The time required to execute an instruction.

* Machine cycle : The time required to access the memory or Input/output devices.

* T-Stages : (Timing stage)

i) Opcode fetch - 4T
ii) Memory read - 3T
iii) Memory write - 3T
iv) I/O read cycle - 3T
v) I/O write cycle - 3T

Machine Cycle	Status $IO/\overline{M}$	S_1	S_0	No. of machine cycle	Control
Opcode fetch	0	1	1	4	$\overline{RD}=0$
Memory Read	0	1	0	3	$\overline{RD}=0$
Memory write	0	0	1	3	$\overline{WR}=0$
I/O Read	1	1	0	3	$\overline{RD}=0$
I/O write	1	0	1	3	$\overline{WR}=0$

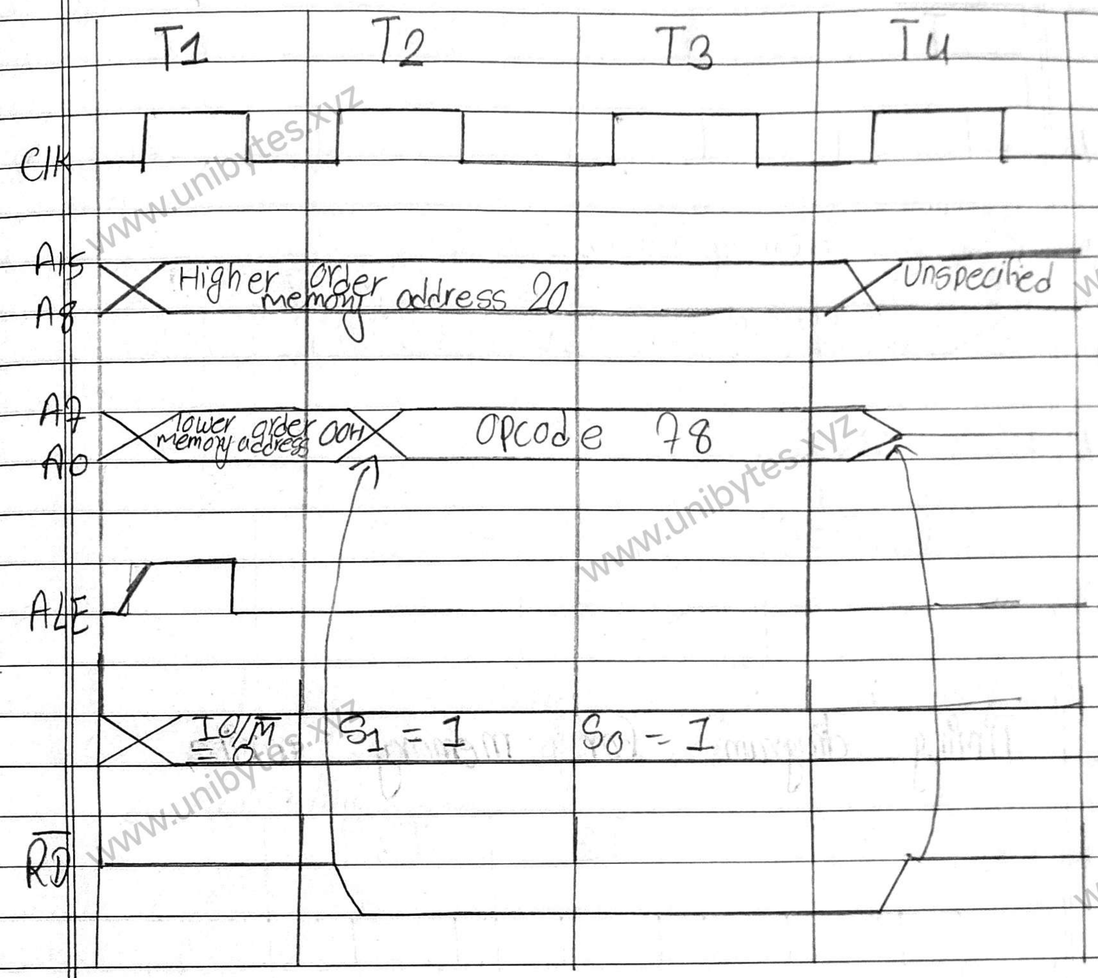
Q.5 Draw a Timing diagram for

MVI A, 32H
MVI B, 42H
MOV D, B
ADD D
STA 2050H
HLT

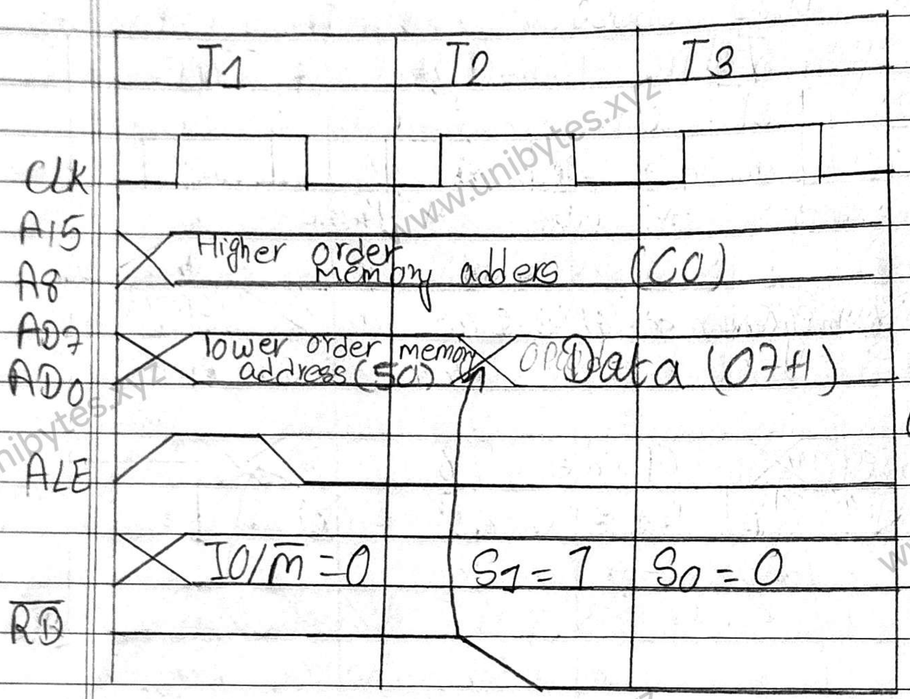
MOV A, B = 78

higher ← 2000H → lower

Timing diagram for Opcode fetch

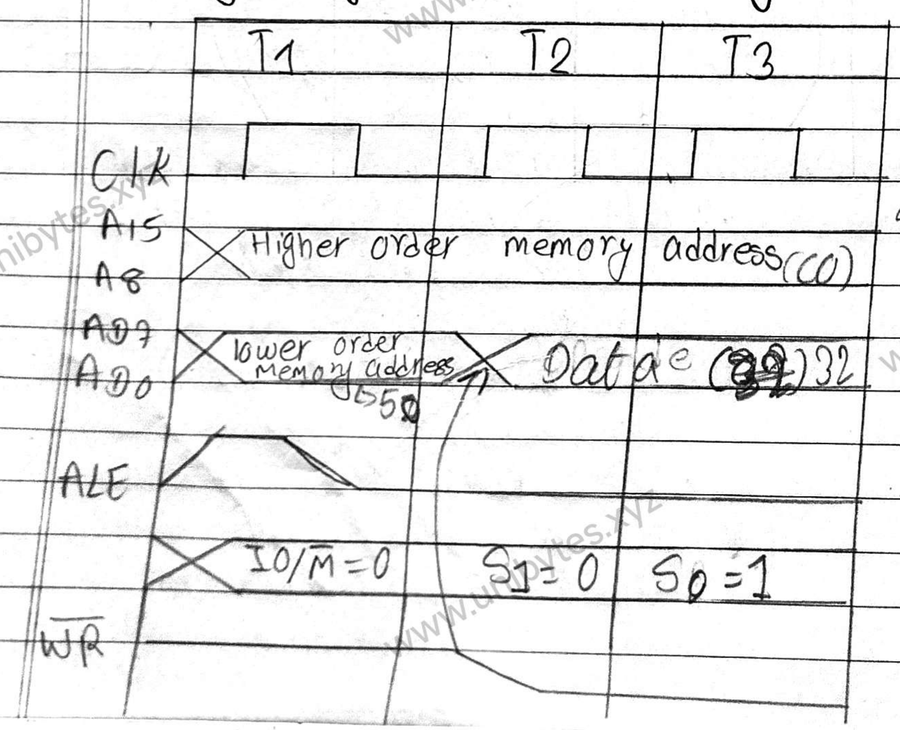


Timing diagram for memory Read



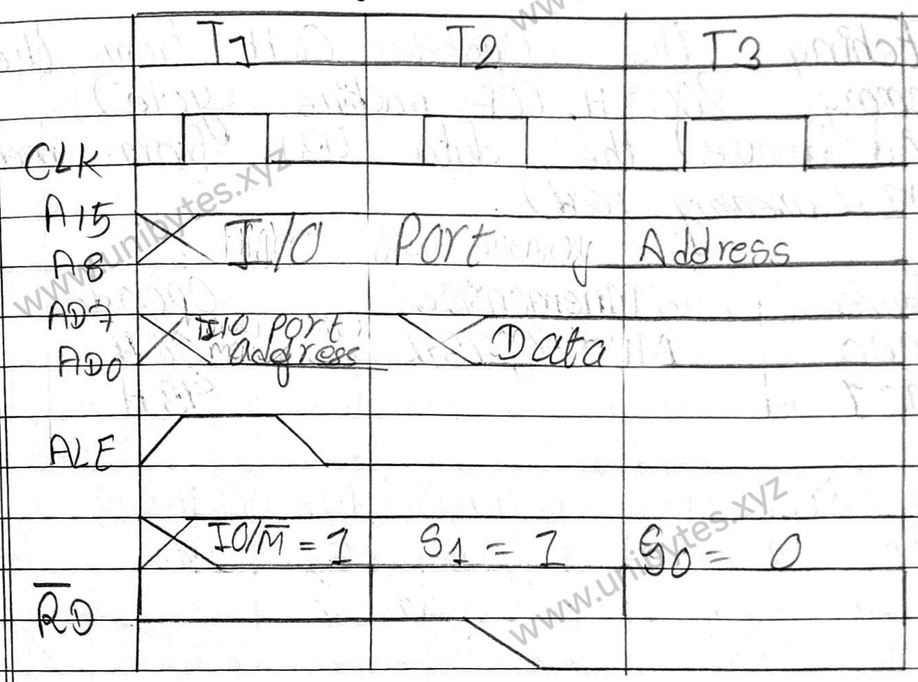
MVI B,07H
C050H ← 07H

Timing diagram for memory write



STA C050H
32, C050H
STA opcod 32

Timing diagram for I/O read

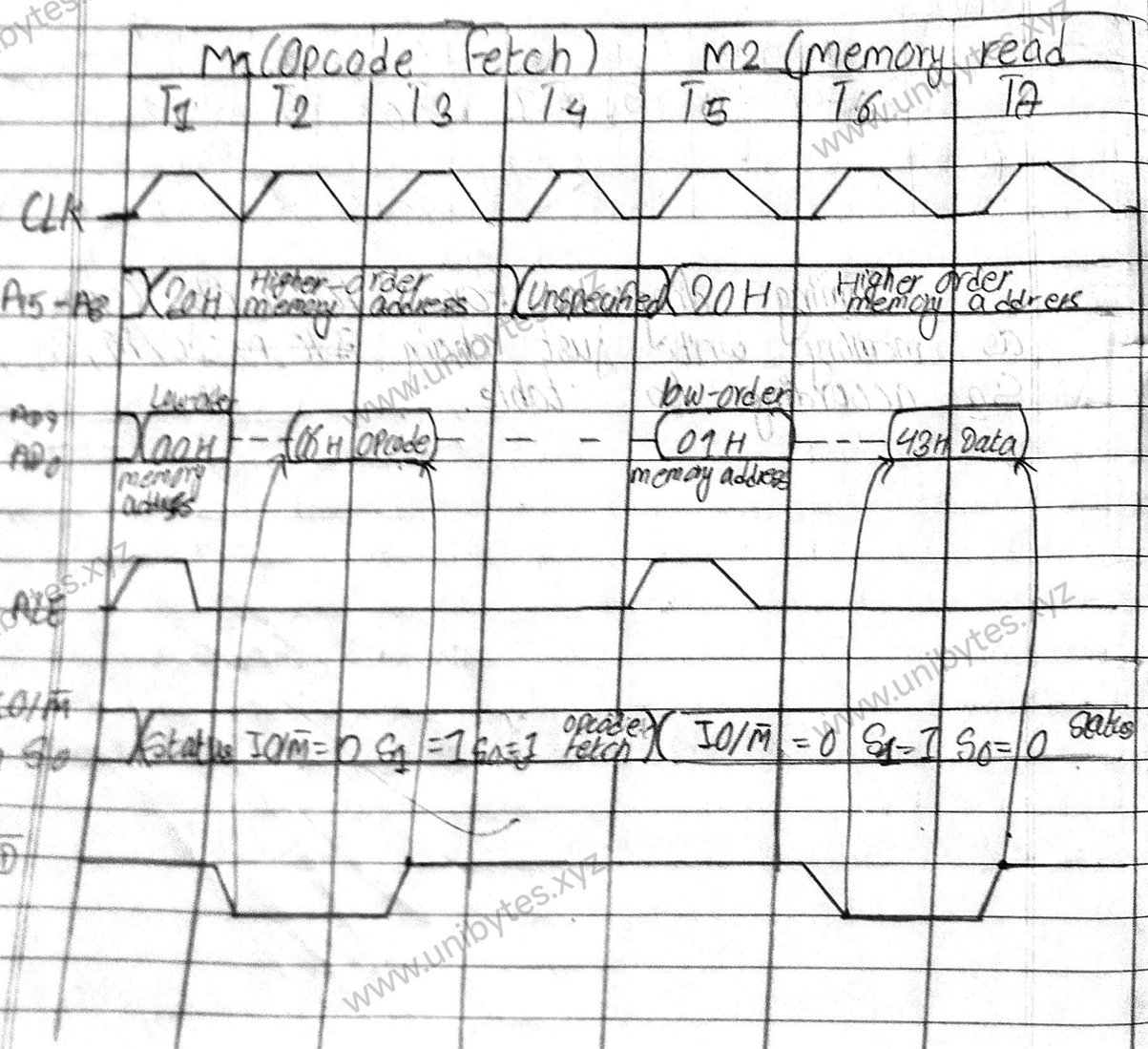


Timing diagram for I/O write (same as memory write just chang ~~I/O M~~ IO/M, S1, S2 according to table.

Timing diagram for MVI B, 43H

- Fetching the Opcode 06H from the memory 2000 H. (OF machine cycle)
- Read (move) the data 43H from memory 2001 H. (memory read)

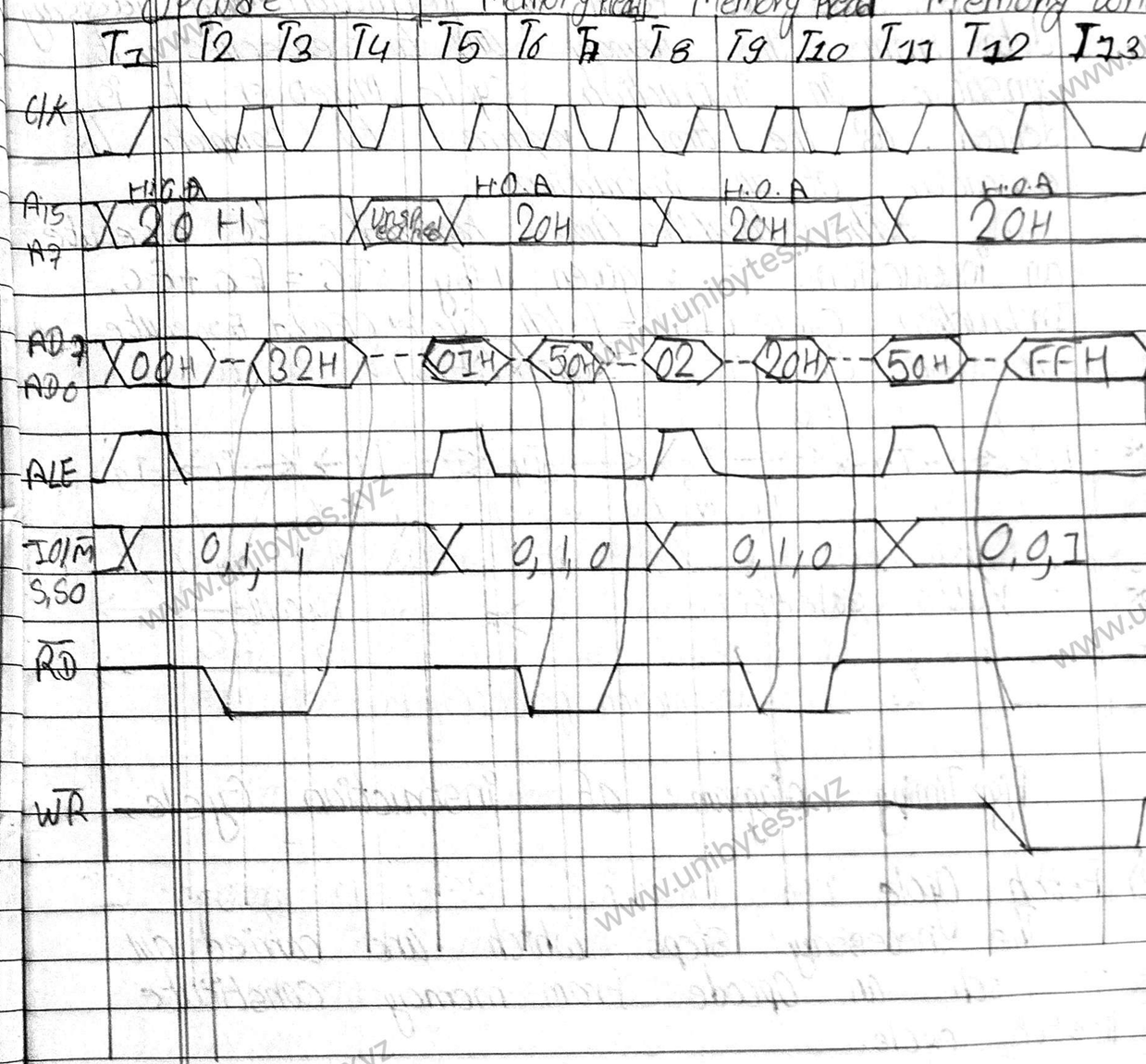
Address	Mnemonics	Opcode
2000	MVI B, 43H	06H
2001		43H



other timing diagram in powerpoint of my Phone

Timing diagram of STA 2050H

Address	Mnemonics	Opcode
2000	STA, 2050 H	32 H
2001		50 H
2002		20 H



Instruction Cycle, Machine Cycle and T-states

1) Instruction Cycle:

The necessary steps that the CPU carries out of ~~the~~ to fetch an instruction and necessary data from the memory and to execute it constitue an instruction cycle. Moreover, it is defined as the time require to complete the execution of an instruction.

The total time required to execute an instruction is given by IC = FC + EC.
Instruction cycle (IC) = Fetch Cycle (FC) + Execute cycle (EC)

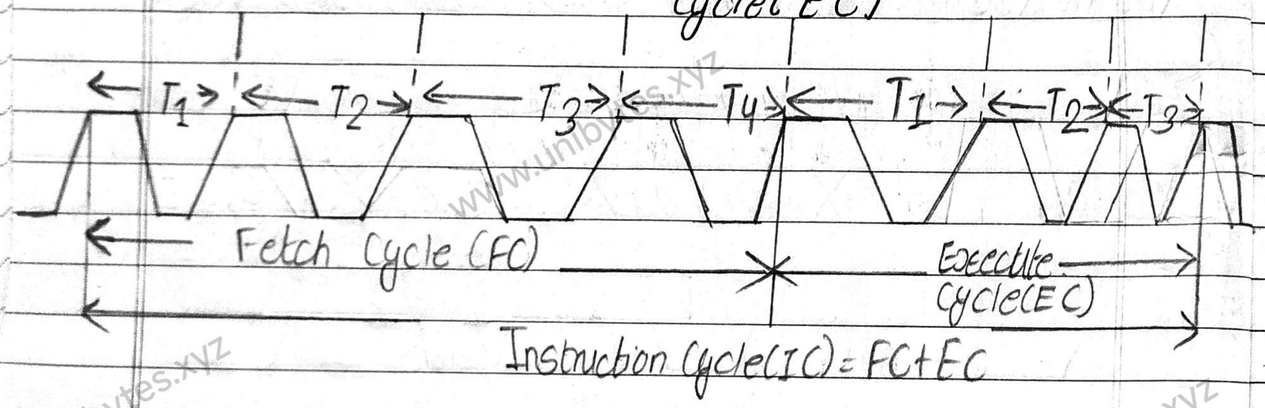


Fig: Timing diagram of Instruction Cycle

2) Fetch Cycle

The necessary steps which are carried out to fetch an Opcode from memory constitute fetch cycle.

The program counter keeps the memory address of the next instruction to be executed in the beginning of fetch cycle. The content of program counter, which is the address of the memory location where opcode is available is send to the memory. The memory places the opcode on the databus so as to transfer it to CPU. The entire process takes 3 clock cycle and then the instruction is decoded in next one clock cycle.

Diagram Book

3. Execute Cycle / operation:

The necessary steps which are carried out to get data if any from the memory and to perform the specific operation specified in an instruction constitute an execute cycle.

The opcode fetched from the memory goes to IR (instruction register) from the IR it goes to the decoder which decodes instruction. After the instruction is decoded execution begins.

Imp

1) Write a program to perform the following

a) Load the no: 1BH in D
b) Load the no: B5H in B
c) Increment the content of B by 1
d) Decrement the content of D by 1
e) Subtract the content of D from the content of B.

f) Display the result at OUT port 1.

ans:

```
MVI D, 1BH
MVI B, B5H
INR B
DCR D
MOV A, B
SUB D
OUT PORT 1
HLT
```

2) WAP to load the data byte in the register C. Mask the high-order bits ($D_7 - D_4$) and display the low order bits ($D_3 - D_0$) at outport 1. Exclusive-OR the result with 57 H and display at OUT PORT 2.

Sol:

```
MVI C, A8 H
MOV A, C
ANI 0F H
OUT PORT 1
XRI 57H
OUT PORT 2
HLT
```

3. WAP to load the byte 8EH in register D and F7H in register E. Mask the higher-order bits ($D_7 - D_4$) from both the data bytes, EX-OR the low order bit ($D_3 - D_0$) and display the answer.

Sol:

```
MVI D, 8E H
MOV A, D
ANI 0F H
MOV D, A
MVI E, F7H
MOV A, E
ANI 0F H
XRA D
OUT PORT 1
HLT
```

4) The following block of data is stored in memory location from C055 to C05A H. Transfer the entire block of data to the location C080 to C085 H in reverse order.

Data : 22, A5, B2, 99, 7F, 37

Sol:

```
LXI H, C055 H
LXI D, C085 H
MVI B, 06 H
Next MVI A, M
STAX D
INX H
DCX D
DCR B
JNZ next
HLT
```

Logic

C055	22
C056	A5
C057	B2
C058	99
C059	7F
C05A	37

C080	37
C081	7F
C082	99
C083	B2
C084	A5
C085	22

Note

HL pair data are denoted by M.

MSB ← {1 0 1 0 1} → LSB

Scratch learning - gt

* WAP to check a given number is Odd or Even. ⇒ wrong

Sol:

```
LDA C050H
RAR → // Right routed to check LSB
JC iloop
MVI A, 00H
STA, ~~C055H~~ C055H
HLT
Loop: MVI A, 01H
STA ~~C055H~~ C055H
HLT
```

exam mein nahi lekhni

* WAP to check the number is positive or negative

Logic

```
LDA C050H
RAL // Left routed to check MSB
JC Loop
MVI A, 00H
STA C051H
HLT
Loop: MVI A, 01H
STA C051H
HLT
```

-ve → 01H
+ve → 00H

* WAP an ALP to multiply two no. 05H x 08H

Sol:

```
MVI A, 00H
MVI B, 08H
MVI C, 05H
Loop: ADD B
DCR C
JNZ -loop
STA C050H
HLT
```

* WAP to find largest of two no 1'st in C001 and 2nd in C002 and result in C003H.

Sol:

```
LXI H, C001H
MOV A, M
INX H
CMP M
JNC Loop
MOV A, M
Loop:
STA C003H
HLT
```

[∴ JNC = Jump on no carry].

* Write an ALP to find the smallest no in data array. Data from location C000H to C005H.

Sol:

```
LXI H, C000H
MVI C, 06H
```

```
      MOV A, M
      DCR C
loop: INX H
      CMP M
      JC      Loop I
Loop I: MOV A, M
        DCR C
        JNZ  loop
        STA C060 H
        HLT
```

1. ALP to add two 8-bit numbers through 8085 and store the result in a suitable memory location

Sol:

```
MVI A, 23H
MVI B, 03H
ADD B
STA 1050 H
HLT
```

Same question but user can input any data ~~at~~

Sol:

```
LXI H, 1000H
MOV A, M   (A ← 05H)
INX H      ← (H ← 1001H
ADD M        (H ← 07H
STA 1002H  ← 0C H
HLT
```

logic

05H	1000
07H	1001
	1002H

Note: Loop sun thau maa call hunxa teo thau maa jump garxa program. Loop hierarchy order maa hunxa.

2) Addition of two 8 bit number with carry:

Sol:

```
LXI H, C050 H (← F1H)
MOV A, M
INX H  (C051 ← 15H)
~~ADD~~ MVI B, 00H
ADD M , (A = F1H + 15H → 06H)
DAA
JNC Loop
INR B  (B = B + 01 = 01H)
Loop STA C053 H (storing data)
     MOV A, B
     STA C052H (storing carry)
     HLT
```

To make result in decimal form

Assuming data:

F1 → 11110001
16 → 00010101
100000110

3) 1's complement of 8-bit number

Sol:

```
LDA C050H
CMA
STA C051 H
HLT
```

2's complement

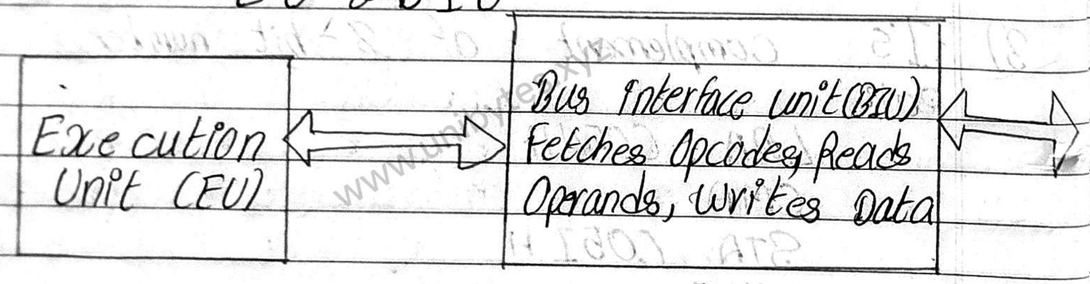
Sol:

```
LDA C050H
CMA
INR A
STA C051 H
HLT
```

EU & BIU

- The 8086 CPU logic has been partitioned into two functional units namely Bus Interface Unit (BIU) and Execution Unit (EU)
- The major reason for this separation is to increase the processing speed of the processor.
- The BIU has to interact with memory and input and output devices in fetching the instruction and data required by the EU.
- EU is responsible for execution the instruction of the programs and to carry the required processing.

EU & BIU



* Bus Interface Unit (BIU)

The BIU performs all bus operations for EU.
- Fetching instructions
- Responsible for executing all external bus cycles.
- Read operands and write result.

* Execution Unit (EU)

Execution unit contains the complete infrastructure required to execute an instruction.

Introduction to 8086

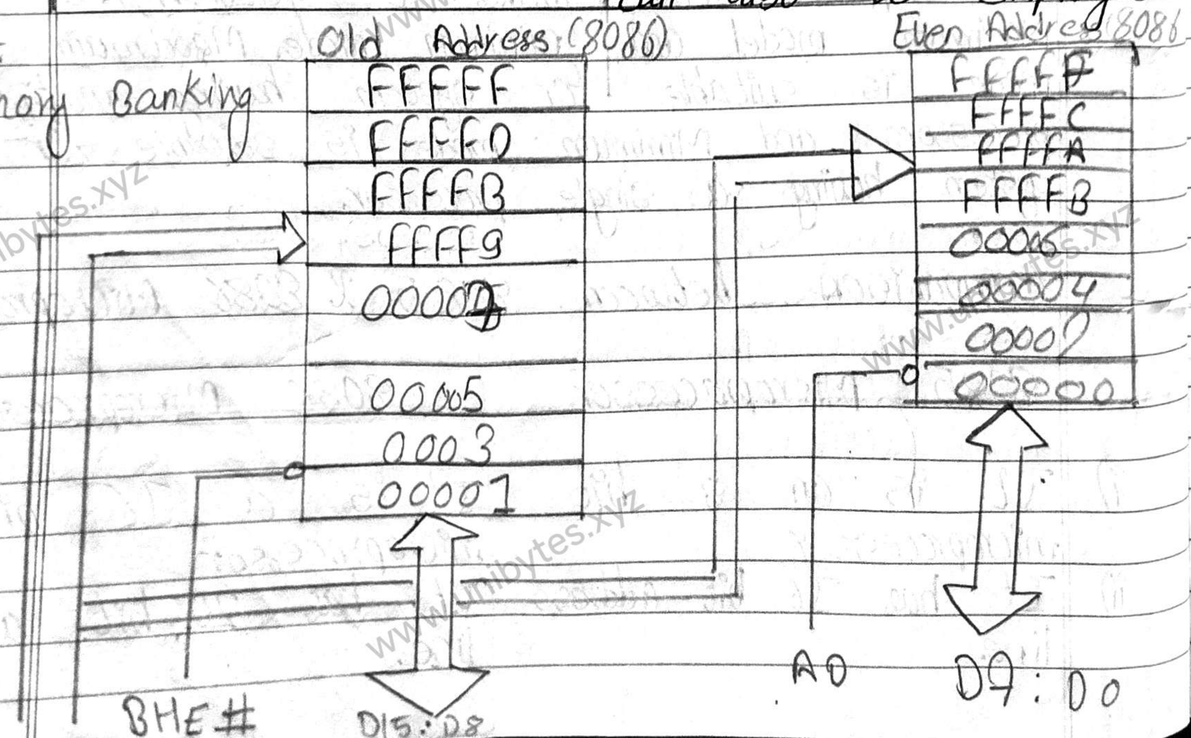
- 8086 Microprocessor is a enhanced version of 8085 microprocessor that was designed by Intel.
- It is a 16-bit Microprocessor having 20 address lines and 16 data lines that provides up to 1 MB storage.
- It consist of powerful instruction set, which provides operation like multiplication and division easily.
- It supports two modes of operation, i.e Maximum model and minimum mode. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor.

Comparison between 8085 & 8086 Microprocessor

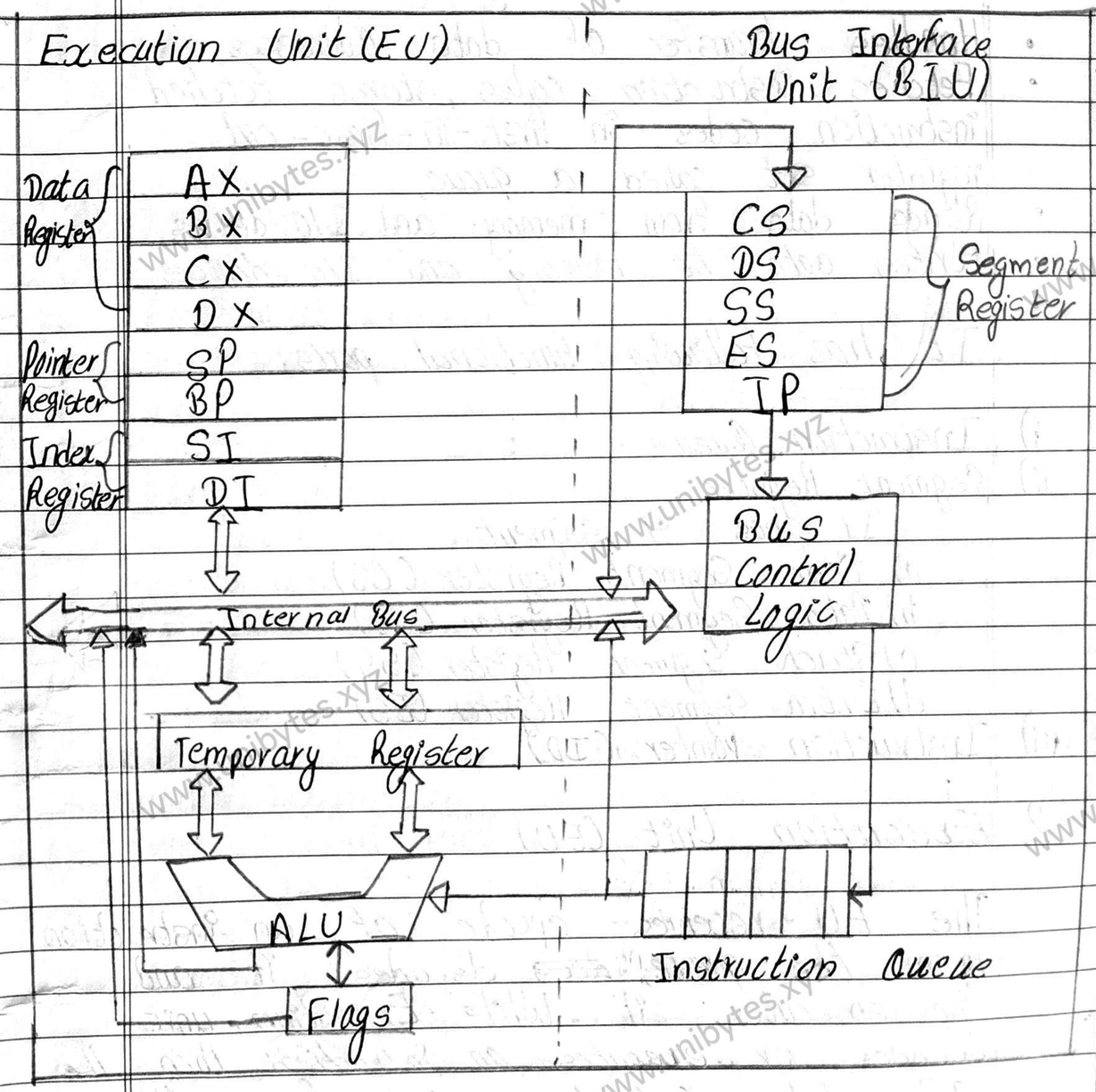
8085 Microprocessor	8086 Microprocessor
i) It is an 8 bit microprocessor	It is a 16 bit microprocessor
ii) It has 16 bit address line.	It has 20 bit address line.

iii)	It has 8-bit data bus	It has 16 bit address
iv)	The memory capacity is 64 KB.	The memory capacity is 1 MB.
v)	It has 5 flags	It has 9 flags
vi)	8085 microprocessor doesnot support memory segmentation	8086 microprocessor supports memory segmentation.
vii)	It doesnot support pipeling.	It supports pipeling.
viii)	It is accumulator based processor.	It is general purpose register based microprocessor
ix)	It has no minimum or maximum mode.	It has minimum and maximum mode.
x)	In 8085, only one processor is used.	In 8086 more than one processor is used. Additional external processor can also be employed.

Fig: Memory Banking



* Internal Architecture of 8086



Functional Units

8086 contain two independent functional units: Bus interface Unit (BIU) and Execution Unit (EU)

1 The BIU:

- Handles transfer of data addresses,
- Fetches instruction codes, stores fetched instruction codes in first-in-first-out register set called a queue,
- Reads data from memory and I/O devices
- Writes data to memory and I/O devices

It has following functional parts:

i) Instruction Queue
ii) Segment Register
It has 4 segment
a) Code Segment Register (CS)
b) Data Segment Register (DS)
c) Stack Segment Register (SS)
d) Extra Segment Register (ES)
iii) Instruction Pointer (IP)

2 Execution Unit (EU)

The EU receives opcode of an instruction from the queue, ~~does~~ decodes it and the executes it. While Execution, unit decodes or executes an instruction, then the BIU fetches instruction codes from the memory and store them in the queue.

i) General Purpose Register
ii) Index Register
iii) ALU
iv) Flag Register
v) Interupts

Timing diagram 8085

i) LXI H, 2000H
- Fetch → 4T
- lower read → 3T
- Higher read → 3T
10T

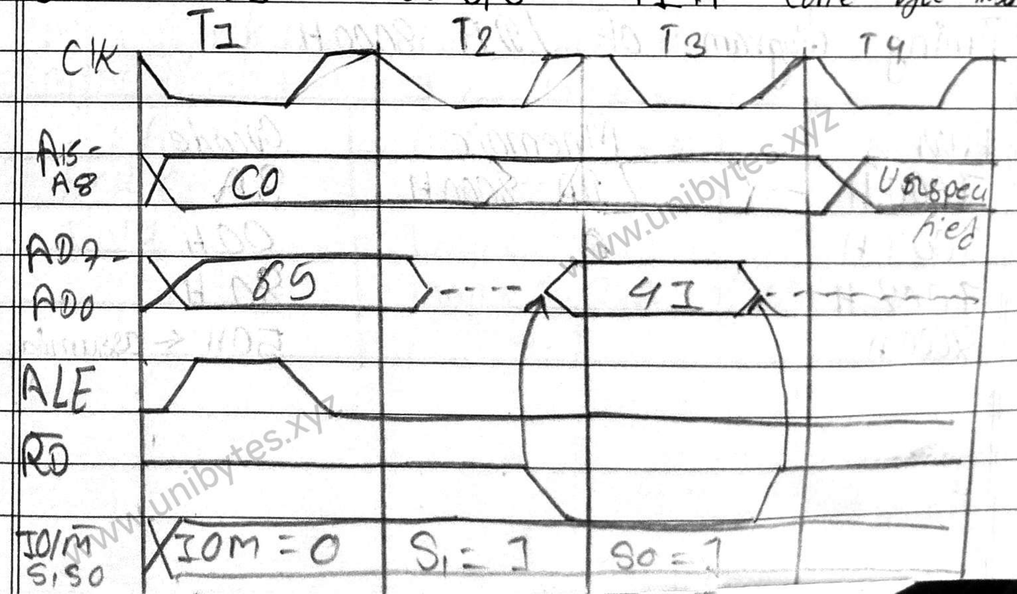
ii) LDA 2050H
- Fetch - 4T
- lower read - 3T
- Higher read - 3T
- Address read - 3T
LDA read 13T

iii) ADD B
- 4T

iv) ADD I 03H
- Fetch - 4T
- Read - 3T

* Timing Diagram of mov B, C instruction

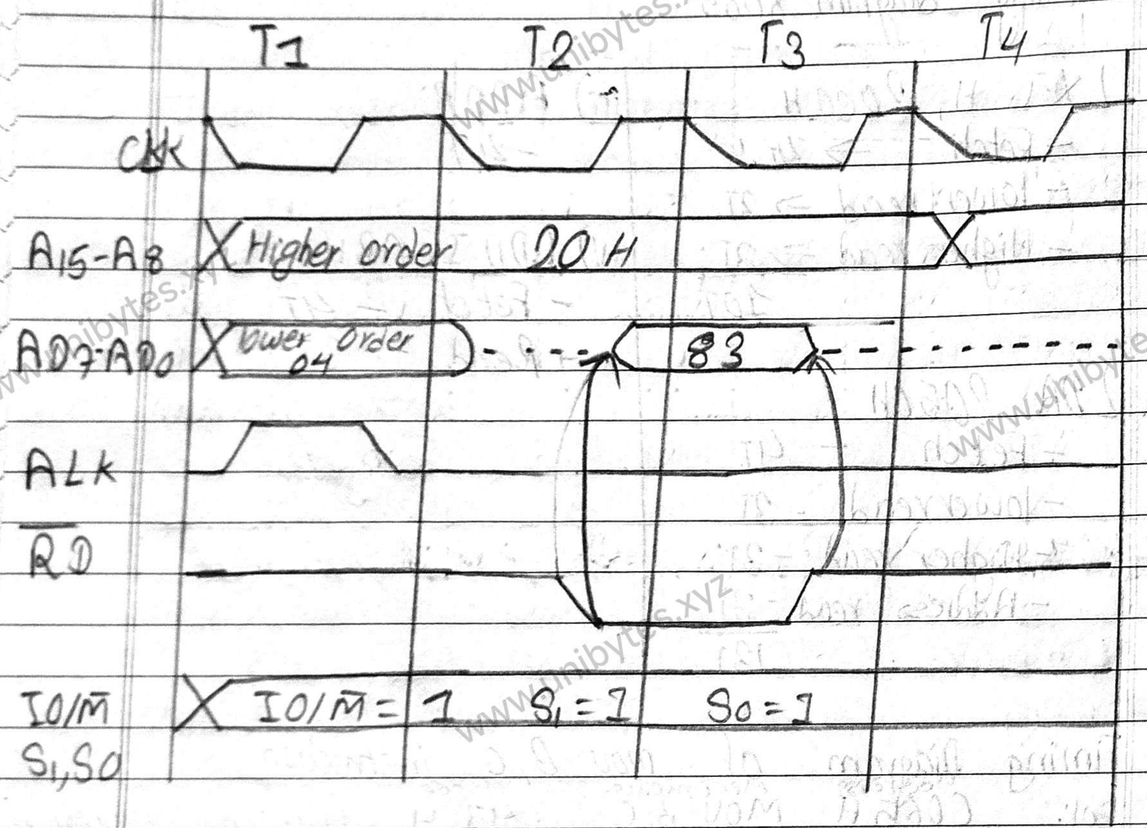
Sol: C065H (address) MOV B, C (mnemonics) 41H (opcode) One byte instruction



1 bit = 4T
2 bit = 7T
3 bit = 13T

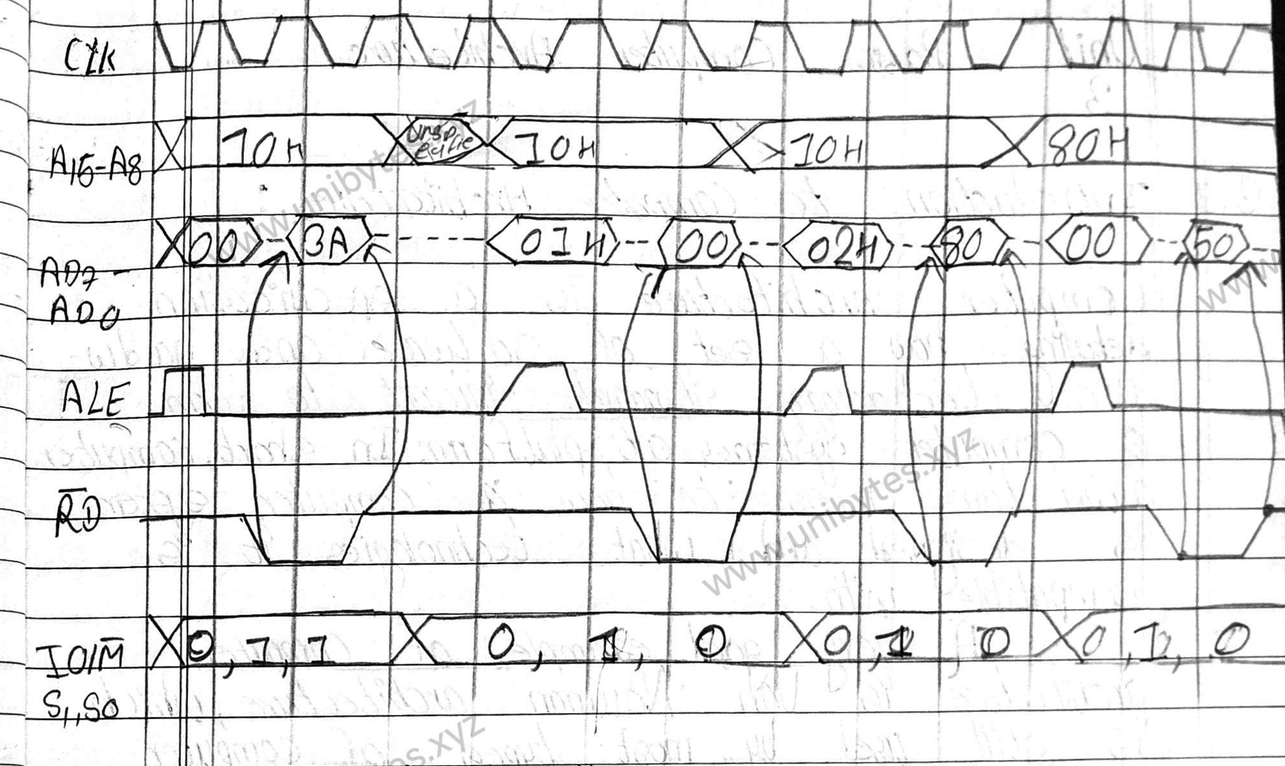
* Timing diagram of ADDE

Address	Mnemoic	Opcode	Comment
2004	ADDE	83	A = A + E



* Timing diagram of LDA 8000H

Address	Mnemoic	Opcode
1000 H	LDA 8000H	3A
1001 H		00 H
1002 H		80 H
8000 H		50H ← assuming



* Code for odd and even

```
MVI A, 0FH        LDA 2050H
MVI B, 06H        ANI 01H
MVI C, 03H        JZ loop
X : ADDB          MVI A, 00H
DCR C             JMP loop1
JNZ X             loop : MVI A, 0EH
                  loop1 : STA 2055H
                          HLT
```

Chapter 4 ~~Basic Computer Organization And Design~~

Unit 3 Basic Computer Architecture

3.1 Introduction to Computer Architecture :

Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. In short, computer architecture refers to how a computer system is designed and what technologies it is compatible with.

A very good example of computer architecture is Von Neuman architecture, which is still used by most types of computer today.

There are three categories of computer architecture :

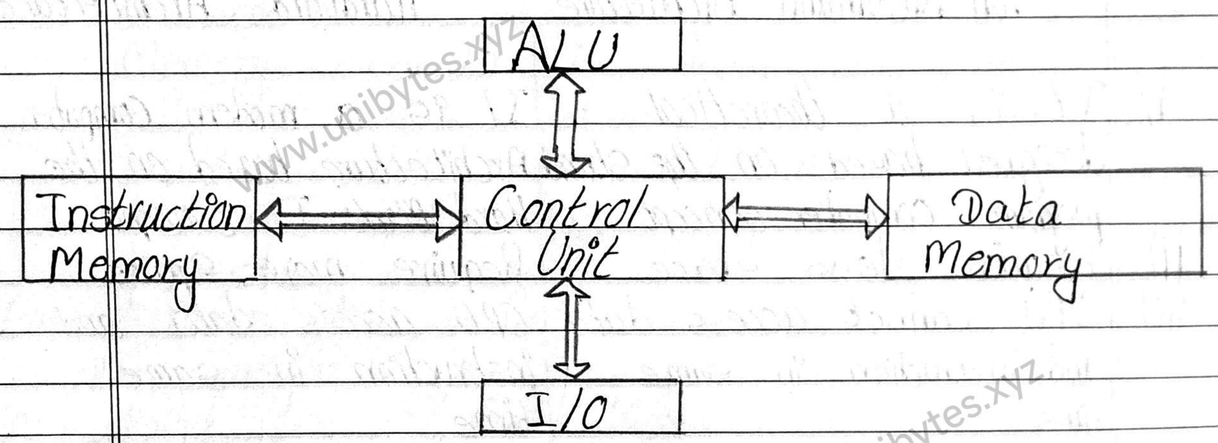
a) System Design
b) Instruction Set Architecture (ISA)
c) Microarchitecture

3.2 History of Computer Architecture

The first document of Computer Architecture was a correspondence between Charles Babbage and Ada Lovelace, that describes the analytical engine. In ancient period, computer architecture were designed and prepared on paper and then, directly built into final hardware form.

i) Harvard Architecture

- It has physically separate storage and sign pathways for data and instructions.
- It requires two memories for instruction and data.
- The idea behind the Harvard architecture is to split the memory into two parts- one for data and another for programs.
- Hardvard architecture requires separate bus for instruction and data.



ii) Von Neumann Architecture

- Based on stored program computer concept where instruction data and program data are stored in the same memory.
- This design is still used in most computer

produced todays.

- It was publish by Joh Von Neumann in 1945.
- This architecture requires only one bus for instruction and data

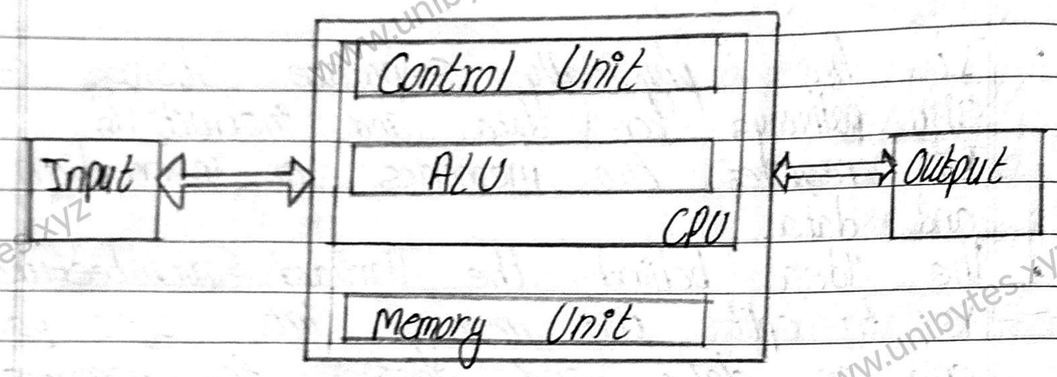


Fig: Von Neumann Model

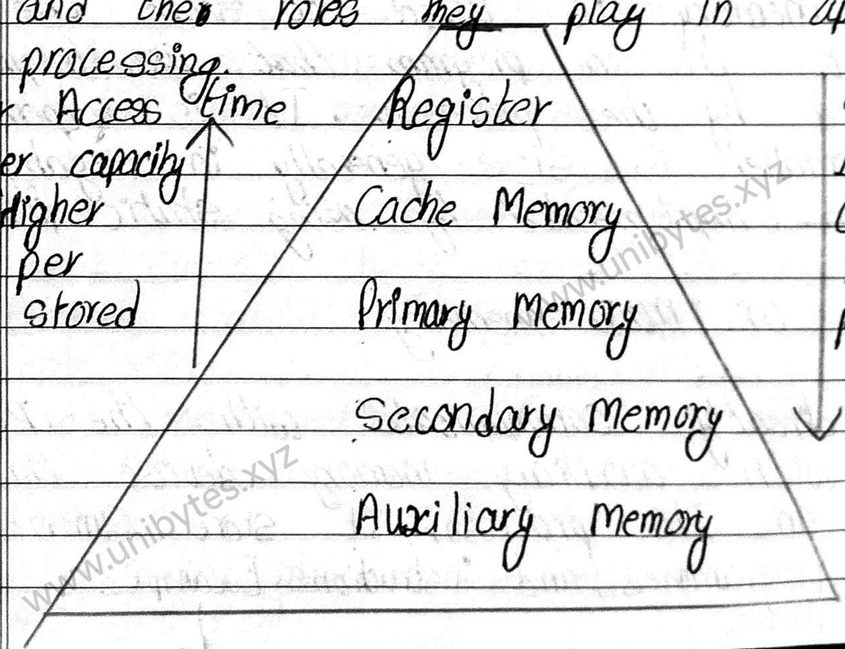
	Von Neumann Architecture	Hardward Architecture
i)	It is a theoretical designed based on the stored program computer concept.	It is a modern computer Architecture based on the Hard Mark I concept.
ii)	Require less space	Require more space
iii)	CPU cannot access data and instruction in same time	CPU access data and instruction in same time.
iv)	Comparatively slower	Comparatively faster
v)	Use one channel / bus	Use seperate busses (two busses)
~~vi)~~	~~Simple to~~	
vi)	Simple to use	Costly than Von Neuman
vii)	Mainly use in personal Computer	Mainly use in micro controller.

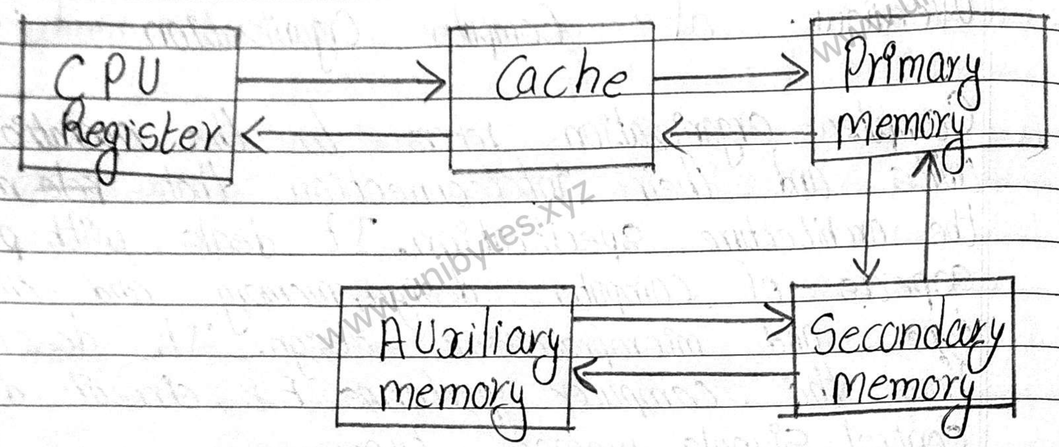
Overview of Computer Organization

Computer organization refers to the operational units and their interconnection that ~~rele~~ realize the architecture specification. It deals with physical aspects of computer design, memory and their types and microprocessor design. It describes how the computer performs. Ex, circuit design, control signals, memory types.

3.3 Memory Hierarchy

- Hierarchy refers to an organizational structure in which items are ranked in a specific manner, usually according to levels of importance.
- Memory hierarchy is about arranging different kinds of storage devices in a computer based on their size, cost and access speed and their roles they play in application processing.





i) Register

The registers are presented inside the CPU. As they are presented inside the CPU, they have least access time. Register are most expensive and smaller in size generally in kilobyte. They are implemented by using flip-flops.

ii) Cache Memory :

Cache memory is used to store the segments of a program that are frequently accessed by the processor. It is expensive and smaller is size generally in megabyte and is implemented by using static RAM.

iii) Primary or Main memory

It directly communicate with the CPU and with auxiliary memory device through an I/O processor. It store current data, programmes, and instructions. Example : RAM, ROM.

iv) Secondary Memory

Secondary memory is computer memory that is non-volatile and persisent in nature and is not directly accessed by a computer/processor. Examples : Hard Drive, Magnetic Disk, SSD, etc.

v) Auxiliary memory

An auxiliary memory is known as lowest cost, highest-capacity and slowest access storage in a computer system. They are used to store removable files.

3.4 Cache Memory

- Cache is a fast small capacity memory that should hold those information which are most likely to be accessed. ~~The~~
- The cache memory is employed in computer systems to compensate for the speed differential between main memory access time and processor logic.
- The cache is used for storing segments of ~~prams~~ programs currently being execute in the CPU and temporary data frequently needed in the present calculation.
- The transformation of data from main memory to the cache memory is termed as mapping memory.
- Cache is placed between CPU and main memory

- The basic operation of cache is, when the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from fast memory. If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the world.

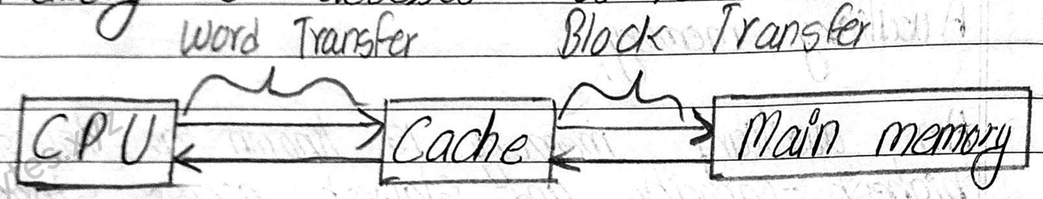
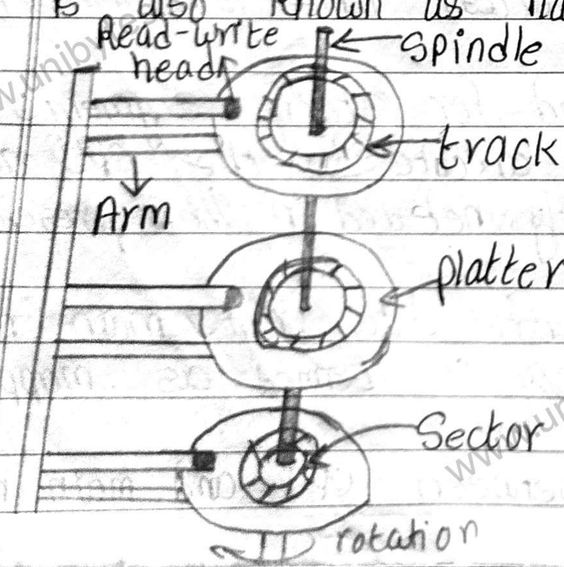


Fig: Cache memory

3.5 ORGANIZATION of Hard Disk

A hard disk (HDD) is a non-volatile computer storage device containing magnetic disk or platters rotating at high speeds. It is a secondary storage device used to store data permanently. Non-volatile means data is retained when the computer is turned off. A hard disk drive is also known as hard drive.



A hard drive consist of following:

i) Magnetic platters

Each magnetic platter holds a certain amount of information, so a drive with a lot of storage and retrieved from the platters it is done so in concentric circles, called tracks, which are further broken down into segments called sectors.

ii) Arm

The arm is the piece sticking out over the platters. The arm will contain read and write heads which are used to read and store the magnetic information ~~ab~~ onto the platters. Each platter will have its own arm which is used to read and write data of it.

iii) Motor

The motor is used to spin the platters from 4,500 to 15,000 rotation per minute (RPM). The faster RPM of driver, the better performance you will achieve from it.

[Remaning phone]

3.6 Instruction Code

- An instruction code is a group of bits that instruct the computer to perform a specific operation.
- The most basic part of an instruction code is its operation part which define add, substract, multiply, shift and complement.
- The computer reads each instruction from memory and places it in a control register.
- AC means accumulator, the operation is performed with the memory operand and the content of AC

3.7 Stored Program Organization

- To organize a computer there will processor register and an instruction code format with two parts.
- The first part specifies the operation to be performed and the second specifies an address.
- The operand is read from memory and used as the data to be operated together with the data stored in the processor register.
- The bits from 0 to 11 specifies memory address and bits from 12 to 15 specifies opcode (operation code)
- Every computer has its own unique instruction set.
- Instructions are stored in one section of memory and data in another.

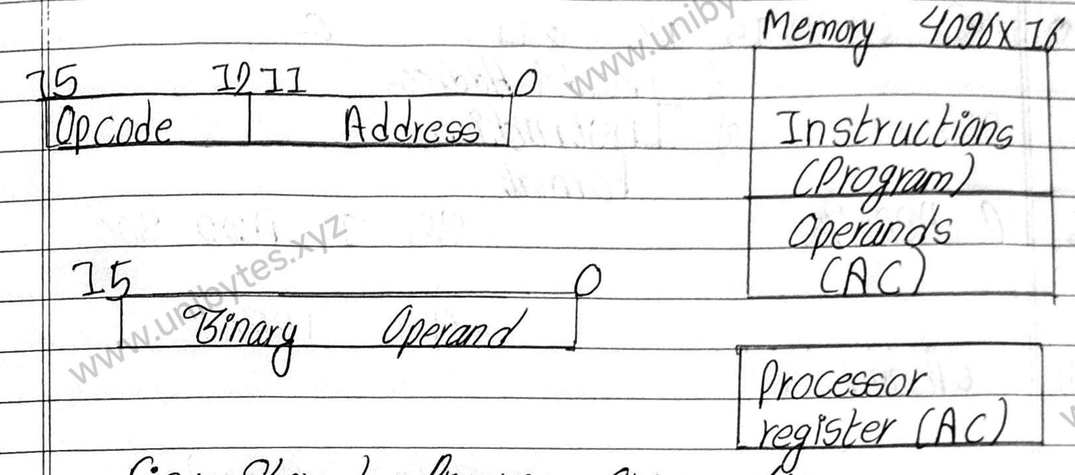


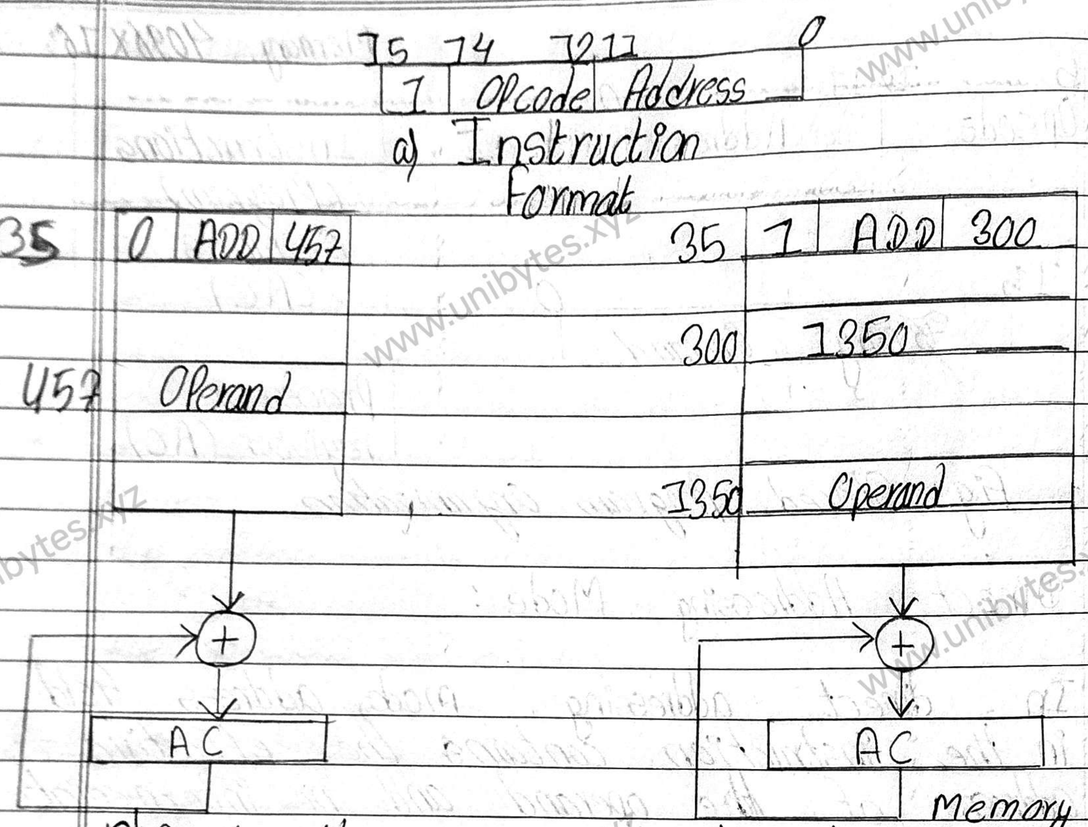
fig: Stored Program organization

3.8 Direct Addressing Mode:

In direct addressing mode, address field in the instruction contains the effective address of the operand and no intermediate memory access is required.

3.9 Indirect Addressing Mode

- In indirect addressing mode, address field in the instruction contains the memory location or register where effective address of operand is present. It required two memory access.
- It consits of a 3-bit operation code, a 12 bit address and an indirect address mode bit designated by 'I' where, the mode bit is '0' for the direct address and 'I' for the indirect address.



a) Instruction format

c) Direct address

b) Indirect address

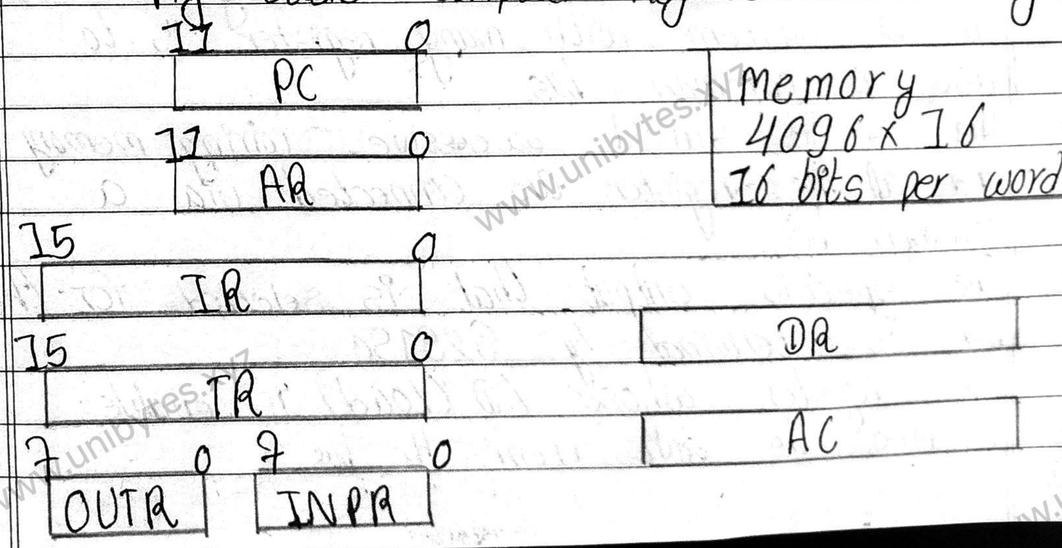
3.10 ~~The~~ Computer Register

Register are the fatest and smallest type of memory elements available to a processor. Register are normally measured by the number of bits they can hold, for example, an "8-bit register", "32-bit register" or a "64-bit register" (or even with more bits). A processor often contains several kinds of registers, which can be classified according to their content or instruction that operate on them.

Table: List of Register for Basic Computer

Register Symbol	Bits	Register Name	Function
DR	16	Data register	Holds memory operand
AR	12	Address Register	Holds address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction register	Holds instruction code
PC	12	Program Counter	Holds address of instr
TR	16	Temporary register	Holds temporary data
INPR	8	Input register	Holds input character
OUTR	8	Output register	Holds output charact

Fig: Basic Computer Register and Memory



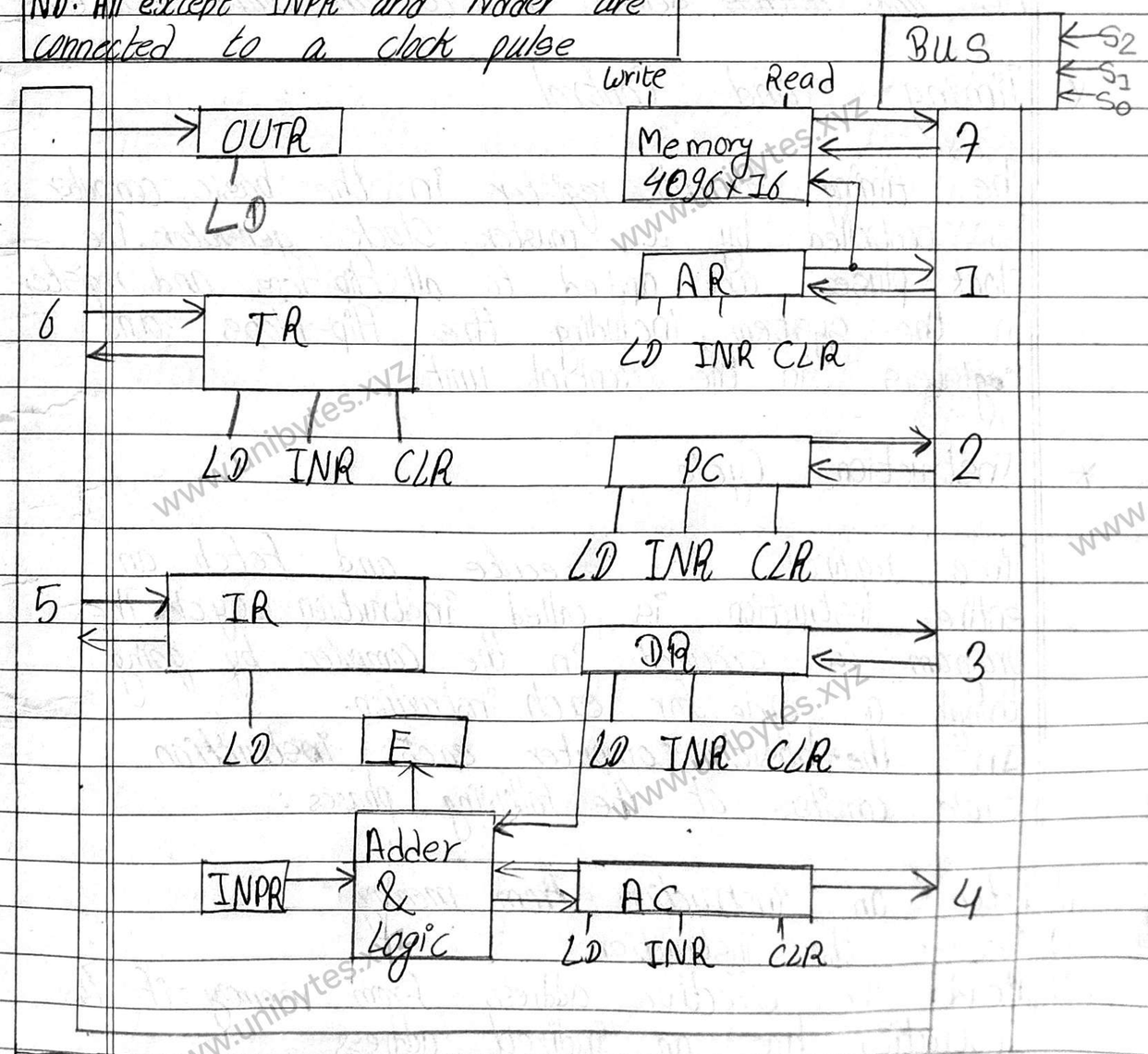
- The memory unit has a capacity of 4096 words, and each word contains 16 bits.
- The Data Register (DR) contains 16 bits which hold the address for the memory location
- The Program Counter (PC) also contains 12 bits which hold the address of the next instruction to be read from memory after the current instruction is executed.
- The Accumulator (AC) register is a general purpose processing register.
- The instruction read from memory is placed in the instruction register (IR).
- The Temporary Register (TR) is used for holding the temporary data during the processing.
- The input Register (INR) holds the input characters given by the user.
- The Output Register (OUR) holds the output after processing the input data.

3.11 The Common Bus

- An efficient scheme for transferring information in a system with many register is to use a common bus.
- ~~The~~ To avoid excessive wiring, memory and all the register are connected via a common bus.
- The specific output that is selected for the bus is determined by $S_2S_1S_0$.
- The register whose LD (Load) is enable receives the data from the bus.

- Registers can be incremented by setting the INR control input and can be cleared by setting the CLR control input.
- The Accumulator's input must come via the Adder & Logic Circuit. This allows the Accumulator and Data Register to swap data simultaneously.
- The address of any memory location being accessed must be loaded in the Address Register.

Nb: All except INPR and Adder are connected to a clock pulse



LD= Load Data
INR = INCREMENT
CLR- Clear / E - extra AC-bit

The input register INPR and output register OUTR have 8 bits each and communicate with the eight least significant bus. INPR is connected to provide information to the bus but OUTR can only receive information from bus. This is because INPR receives a character from an input device which is then transferred to AC. OUTR receives a character from AC and ~~receive~~ deliver it to an output device.

* Timing and Control

The timing for all register in the basic computer is controlled by a master clock generator. The clock pluses are applied to all flip-flops and register in the system, including the flip-flops and registers in the control unit.

* Instruction Cycle

Time required to execute and fetch an entire instruction is called instruction cycle. The program is executed in the computer by going through a cycle for each instruction.
In the basic computer each instruction cycle consits of the following phases:

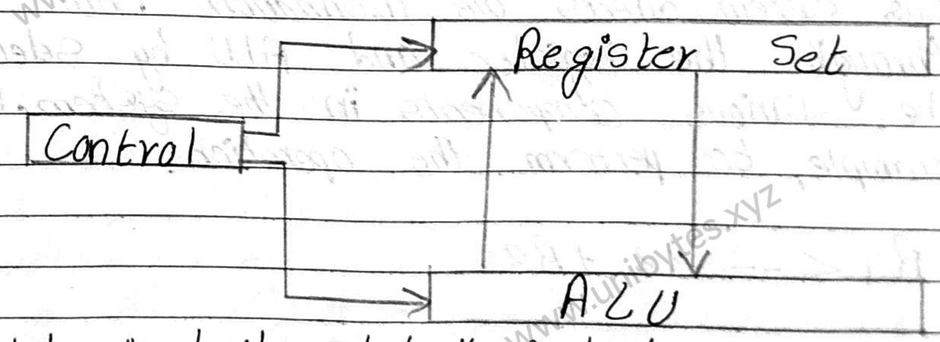
i) Fetch an instruction from memory
ii) Decode the instruction
iii) Read the effective address from memory if the instruction has an indirect address.
iv) Execute the instruction.

Chapter 5 Control Processing Unit

- The Part of the computer that performs the bulk of data processing operation is called central processing unit (CPU) which consits of ALU, control unit and register array.

The CPU is made up of 3 major parts

- The register set stores intermediate data used during the execution of the instructions.
- The arithmetic logic unit (ALU) performs the required micro operations for executing the instructions.
- The control unit supervise the transfer of information among the register and instructions the ALU as to which operation to perform.



Control unit decide what the instructions means and direct the necessary data to be moved from memory to ALU.

5.1 General Register Organization

When a large number of registers are included in the CPU, the most efficient

to connect them through a common bus system. The register communicate with each other not only for direct data transfer, but also while performing various micro operations.

- Register organization show how registers are selected and how data flow between register and ALU.

- A decoder is used to select a particular register.
- The A and B buses form the two inputs of an ALU. The operation select lines decide the micro operation to be performed by ALU.
- The result of the micro operation is available at the output bus.

The control unit that operates the CPU bus system directs the information flow through the register and ALU by selecting the various components in the system. For example, to perform the operation.

$R1 \leftarrow R2 + R3$

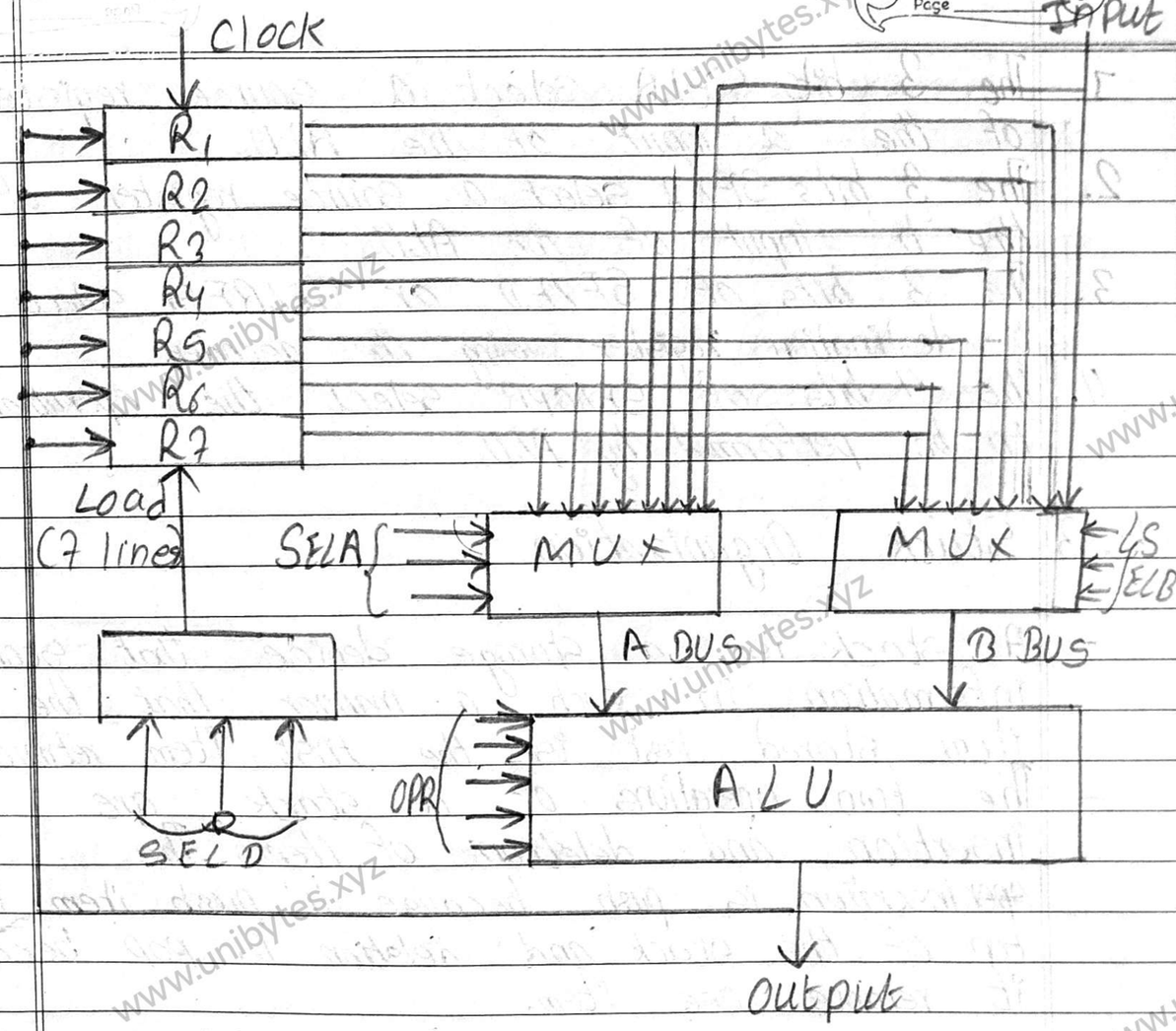


Fig: General Register Organization

5.2 Control Word

There are 13 binary selection inputs in the unit, and their combined values specifies a control word

3	3	3	4
SELA	SELB	SELREG or SELD	SELOPR

1. The 3 bits SELA select a source register of the 'a' input of the ALU.
2. The 3 bits SELB select a source register of the 'b' input of the ALU.
3. The 3 bits of SELED or SELREG select a destination register using the decoder.
4. The 4 bits of SELOPR select the operation to be performed by ALU.

5.3 Stack Organization

- A stack is a storage device that store information in such a manner that the item stored last is the first item retrived.
- The two operations of a stack are insertion and deletion of items. The ~~oper~~ insertion is push because it push item in top of the stack and deletion is pop because it remove one item.

Register stack

A stack can be placed in a portion of a large memory or it can be organized as a collection of a finite number of memory words or register.

There are 3 items place on the stack: A, B, C, in that ~~other~~ order. C is on the top of the stack so that the content of SP is now 3.

If the stack is not full.

SP <- SP + 1 (increment stack pointer)

M[SP] <- DR write item on top of the stack

If the stack is not zero.

DR <-- M[SP] Read item from top of stack

SP <-- SP - 1 Decrement stack pointer

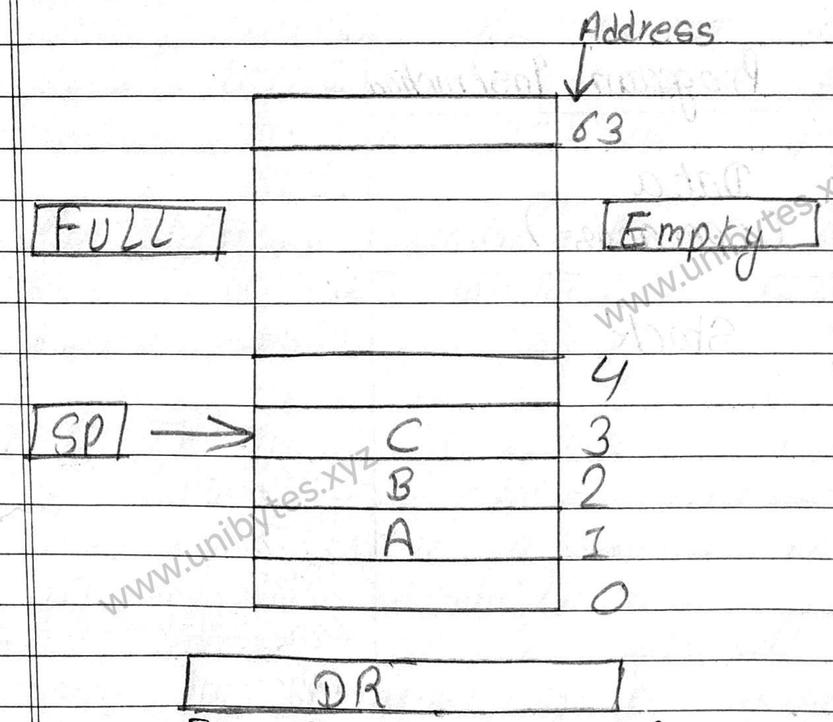
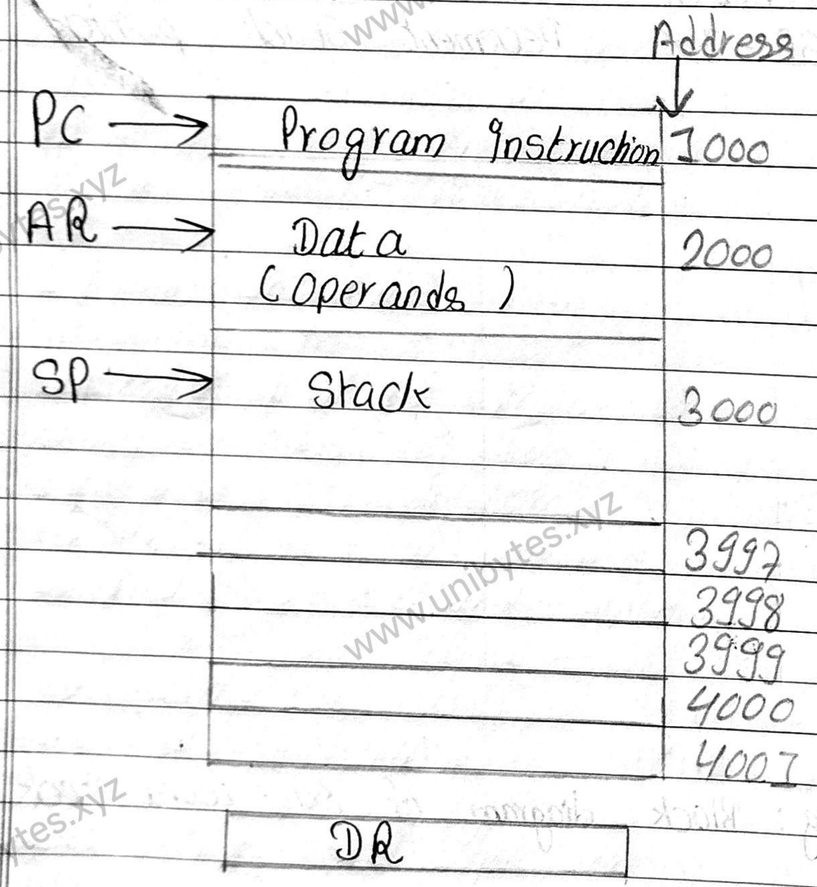


Fig: Block diagram of 64 - word stack

Memory Stack

The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer.

- The program Counter PC points at the address of the next instruction in the program. The address register AR points at an array of data.
- PC is used during the fetch phase to read an instruction.



Computer memory with program, data & stack segments.

5.4 Program Interrupt

- Program interrupt refers to the transfer of program control from a currently running program to another service program as a result of an external or internal generated request
- The concept of program intterput is used to handle a variety of problems that arise out of normal program sequences.

The interrupt producre is, in principle, quite similar to a subroutine call except for three variations:

i) The intterputs is usually initiated by an internal or external signals rather than from the execution of an instruction (except for software interrupt),
ii) The address of the intterput service program is determined by the hardware rather than from the address field of an instruction,
iii) An intterput procedure usually store all the information necessary to define the state of the CPU rather than storing only the program counter.

Types of Interrupts
i) External Interrupts
ii) Internal interrupts
iii) Software interrupts

i) External Interrupts

External interrupts come from input/output devices, from a timing device, from a circuit monitoring the power supply, or from any other external source.
Examples that cause external interrupts are I/O device requesting transfer of data, I/O device finished transfer of data, elapsed time of an event, or power failure.

ii) Internal Interrupts

Internal interrupts arise from illegal or erroneous use of an instruction or data. Internal interrupts are also called traps. They are synchronous with the program. Ex: attempted to divided by zero.
Internal VS External Interrupts

The difference between internal and external interrupts is that the internal interrupt is initiated by some exceptional condition caused by the program itself rather than by an external event.
If the program is rerun, the internal interrupts will occur in the same place each time. External interrupts depend on external conditions that are independent of the program being executed at the time.

iii) Software Interrupts

A software interrupt is initiated by executing an instruction.
It is a special call instruction that behaves like an interrupt rather than a subroutine call. The most common use of software interrupt is associated with a supervisor call instruction.

5.5 Instruction Format

An instruction format defines layout of bits of an instruction. An instruction in computer comprises of groups called fields.
The most common fields are:

- Operation field which specifies the operation to be performed like addition.
- Address field which contain the location of operand, i.e. register or memory location.
- Mode field which specifies how operand is to be founded.

Generally, CPU organization are of three types on the basis of number of address fields:

- Single Accumulator Organization
- General Register Organization
- Stack Organization

i) Single
ADD X

ii) General
ADD R_1, R_2, R_3
ADD R_1, R_2

iii) Stack
PUSH X
ADD

Instruction Format

$X = (A+B) * (C+D)$

1. Three - Address Instruction

ADD	R_1, A, B	$R_1 \leftarrow M[A] + M[B]$
ADD	R_2, C, D	$R_2 \leftarrow M[C] + M[D]$
MUL	X, R_1, R_2	$M[X] \leftarrow R_1 * R_2$

2. Two - Address Instruction

MOV	R_1, A	$R_1 \leftarrow M[A]$
ADD	R_1, B	$R_1 \leftarrow R_1 + M[B]$
MOV	R_2, C	$R_2 \leftarrow M[C]$
ADD	R_2, D	$R_2 \leftarrow R_2 + M[D]$
MUL	R_1, R_2	$R_1 \leftarrow R_1 * R_2$
MOV	X, R_1	$M[X] \leftarrow R_1$

3. One - Address Instruction

LOAD	A	$AC \leftarrow M[A]$
ADD	B	$AC \leftarrow AC + M[B]$
STORE	T	$M[T] \leftarrow AC$
LOAD	C	$AC \leftarrow M[C]$
ADD	D	$AC \leftarrow AC + M[D]$
MUL	T	$AC \leftarrow AC * M[T]$
STORE	X	$M[X] \leftarrow AC$

exam mai, lekna pro ← Here, T is the temporary memory location required for storing the intermediate result

4) Zero - Address Instruction

PUSH	A	$TOS \leftarrow A$
PUSH	B	$TOS \leftarrow B$
ADD		$TOS \leftarrow (A+B)$
PUSH	C	$TOS \leftarrow C$
PUSH	D	$TOS \leftarrow D$
ADD		$TOS \leftarrow (C+D)$
MUL		$TOS \leftarrow (C+D) * (A+B)$
POP	X	$M[X] \leftarrow TOS$

5.6 Addressing Modes

The operation code specifies the operation to be perform. The mode field is used to locate the operands needed for the operation.

www.unibytes.xyz

1) Implied Mode

It is also known as "Implict" or "Inherent" addressing mode is the addressing mode in which, no operand (register or memory location or data) is specified in the instruction. For e.g : CMA, RLC

2) Immediate Mode

In this mode the operand is specified in the instruction itself. It has operand field rather than an address field. The operand field contain actual operand to be used and address field specifies either memory word or a processor register. MVI B, 50H

3) Register Mode

In this mode the operands in the register that reside within the CPU. The particular register is ~~used~~ selected from a register field in the instruction.

4) Register Indirect Mode

In this mode the instruction specifies a register in the CPU whose contents give the address of ~~the~~ the operand in memory. In other words, the selected register contains the address of the operand rather than operand itself.

Ex MOV A, B

~~MOV A, M~~ LDAX B

5) Auto Increment or Auto Decrement Mode

This is similar to the register indirect mode except that the register is increment or decrement (or before) its value is used to access memory.

6) Direct Address Mode

Eg LDA 4000H

Here, the operand resides in memory and its address is given directly by the address field of the instruction. In this mode the effective address is equal to address part of the instruction.

7) Indirect Address Mode

In this mode the address field of the instruction gives the address where the effective address is stored in memory. Effective address = address part of instruction + content of CPU register

8) Relative Address Mode

In this mode the content of the program counter is added to the address part of the instruction in order to obtained the effective address. EA = PC + A.

9) Indexed Addressing Mode

In this mode the content of an index

register is added to the address part of the instruction to obtain the effective address.

10) Base Register Addressing Mode

Base register is added to the address part of the instruction to obtain the effective address.

5.7 Data Transfer and Manipulation

Computer provides an extensive set of instruction to give the user the flexibility to carry out various computional tasks. The instruction set of computer differ from one another. The actual operations available in the instruction set are not very different from one another computer.

Most computer instructions can be classified into three categories:

i) Data transfer instructions
ii) Data manipulation instructions
iii) Program control instructions

Data transfer instructions cause transfer of data from one location to another without changing the binary information content.

Data manipulation instructions are those that perform arithmetic, logic, and shift operations. Program control instruction provide decision-making capabilities and change the path taken by the program where executed in the computer

i) Data Transfer Instructions

The most common transfers are between memory and processor register, between processor registers and input or output, and between the processor register themselves.

Typical Data Transfer Instruction

Name	Mnemonic
Load	LD
Store	ST
Move	MOV
Exchange	XCH
Input	IN
Output	OUT
Push	PUSH
pop	POP

ii) Data Manipulation Instructions

The data manipulation instruction in a typical computer are usually divided into three types:

Imp

i) Arithmetic Operations
ii) Logical and Bit Manipulation Instruction
iii) Shift Instruction

iii) Program Control Instruction

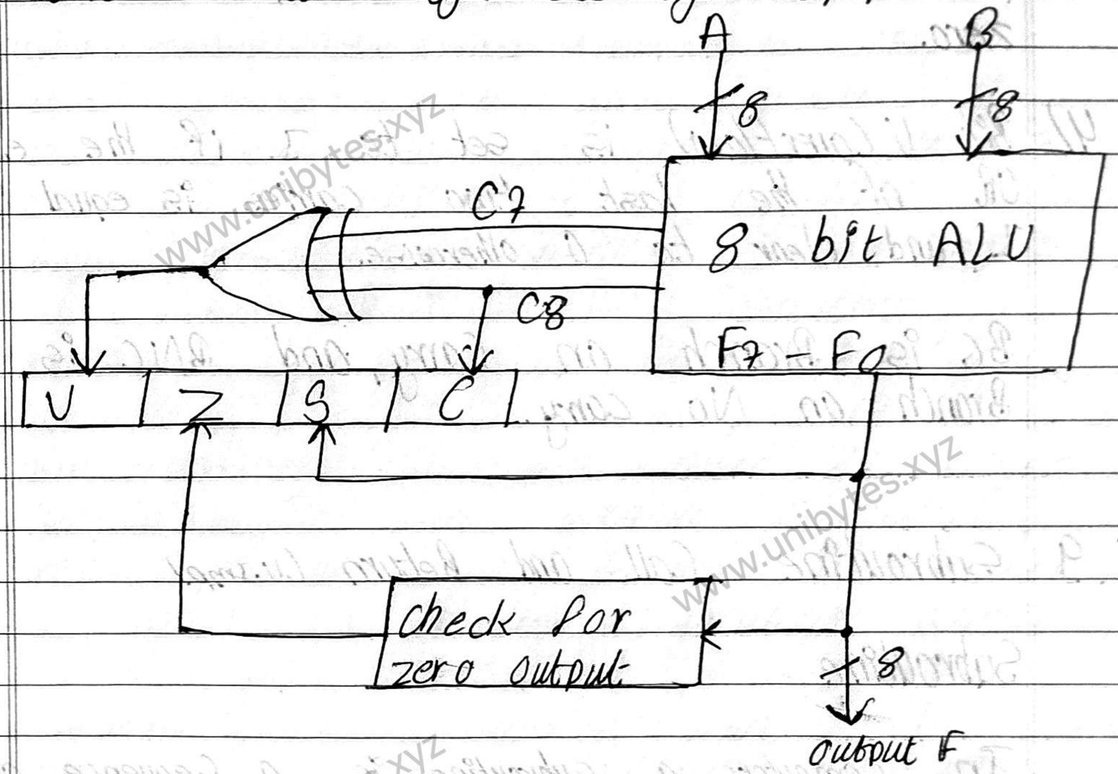
Each time an instruction is fetched from memory, the program counter is incremented so that it contains the address of the instruction next in sequence. Program control instruction specify conditions for altering the content of the program counter, while data transfer and manipulation instruction specify condition for data processing operations.

Typical Program Control Instructions

Name	Mnemonic
Branch	BR
JUMP	JMP
Skip	SKP
Call	CALL
Return	RET
Compare (by subtraction)	CMP
Test (by ANDing)	TST

5.8 Status Bit Conditions

Status bit are also called condition-code bits or flag bits. figure below shows the block diagram of an 8 bit ALU with a 4-bit status register. The four status bits are symbolized by C, S, Z and V.



The bits are set or cleared as a result of an operation perform in the ALU.

1. Bit C (Carry) is set to 1 if the end carry C8 is 1. It is cleared to 0 if the carry is 0.

2) Bit S (sign) is set to 1 if the highest order bit F, is 1. It is set to 0 if the bit is 0. (1 - negative. 0 - positive)

3) Bit Z (zero) is set to 1 if the output of ALU contain all 0's. It is cleared to 0

otherwise. In other word, z=1 if the Output is zero and z=0 if the output is not zero.

4) Bit V (overflow) is set to 1 if the exclusive OR of the last two carries is equal to 1, and clear to 0 otherwise.

BC is Branch on Carry, and BNC is Branch on No carry.

5.9 Subroutine Call and Return (V.Imp)

Subroutine

In computers, a subroutine is a sequence of program instruction that performs a specific task, packed as a unit. Then this can be used in programs wherever that particular task has to be performed. A subroutine is often coded so that it can be called several times and from several places during one execution of a program including from other ~~subtraction~~ subroutines and then branch back to the next instruction after the call, once the subroutine task is done.

The instruction that transfer program control to a subroutine is known by different names. The most common names used are call subroutine, jump to subroutine, branch to subroutine, or branch and save address.

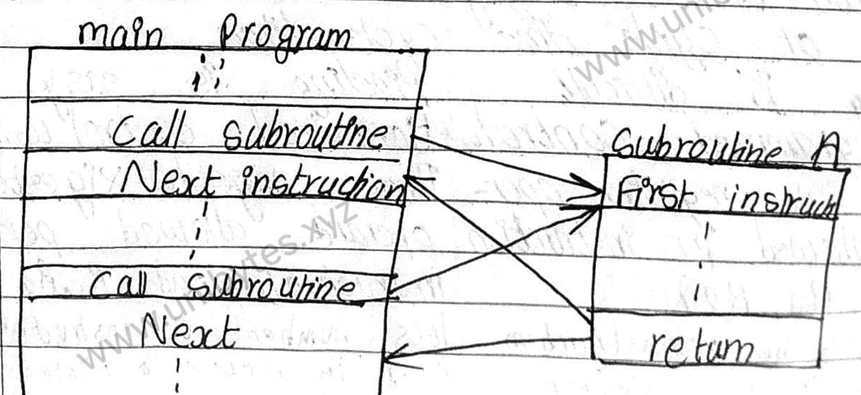
A call subroutine instruction consists of an operation code together with an address that specifies the begining of the sub routine. The instruction is executed by performing two operations:

i) The address of the next instruction available in the program counter (the return address) is stored in a temporary location so the subroutine knows where to return, and

ii) Control is transferred of the beginning of the subroutine.

The last instruction of every subroutine, commonly called return from subroutine, transfer the return address from the temporary location into the program counter.

① Subroutine Call



Example:

```
START:  Call Display
        NOP (No operation)
        JMP START

DISPLAY: MVI A, 55H
         OUT PORT 1
         RET
```

5.10 RISC & CISC - Different between.

RISC = Reduce Instruction Set Computer
CISC = Complex Instruction Set Computer

Imp

	CISC	RISC
i)	Types of instruction are complex	Type of instruction are simple
ii)	Emphasis on hardware	Emphasis on software
iii)	Multiple instruction size and formats.	Instructions of same set with new format
iv)	Less register	Uses more register
v)	More addressing mode	Fewer addressing mode.
vi)	Extensive use of microprogramming	Complexity in compiler
vii)	Instruction take a varying amount of cycle time	Instruction take one cycle time.
viii)	Pipelining is difficult	Pipeling is easy
ix)	Microprogrammed control unit	Hardwired control unit
x)	One or two register operands allowed per instruction (e.g add R1, R2)	Three ~~rigeste~~ register operand allowed per instruction (e.g add R_1, R_2, R_3
xi)	Large number of instruction	less number of instruction
xii)	Used in laptops & desktop	Used in mobile & tablets

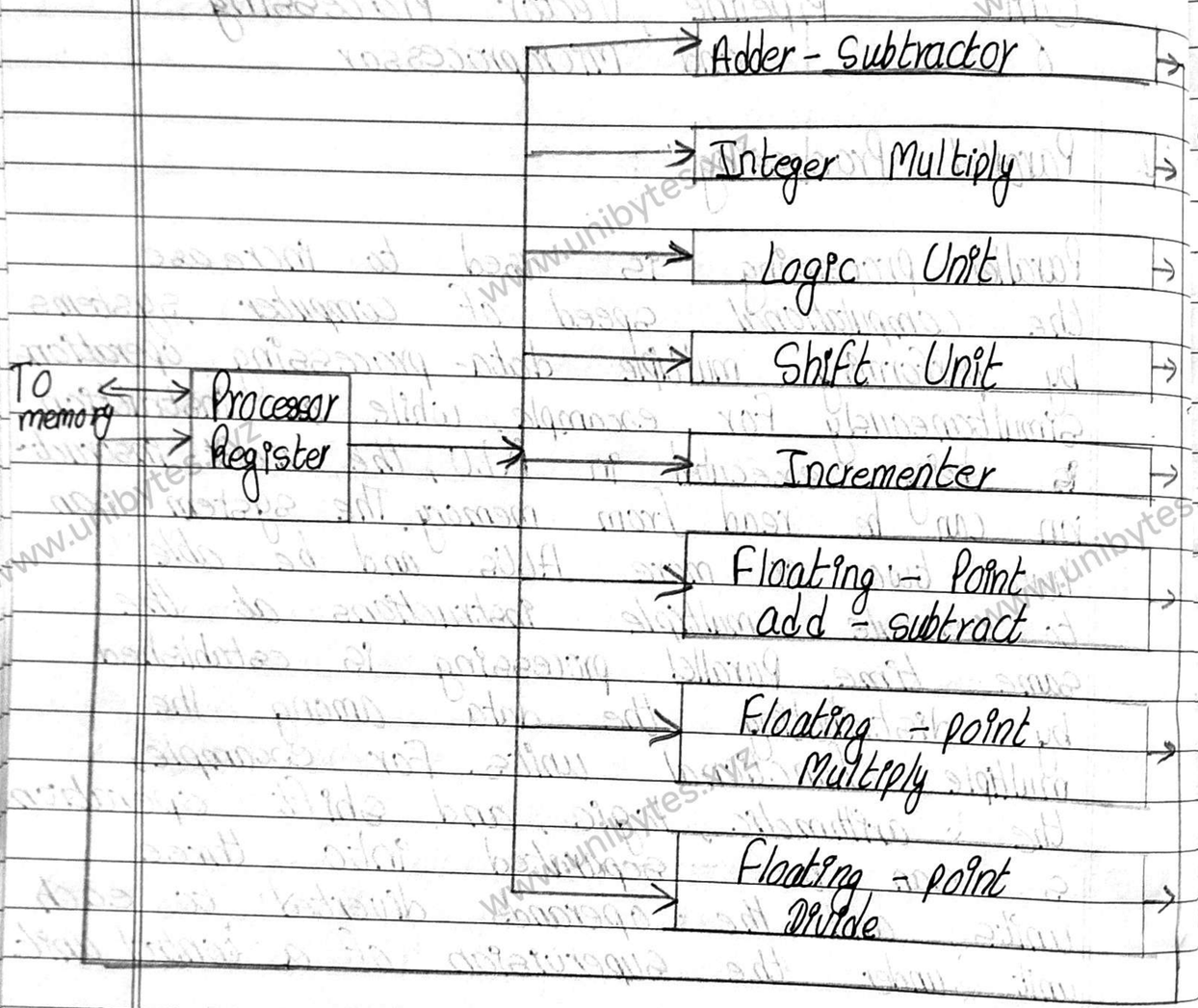
(V.V Imp) chapter

Unit 6 Pipeline, Vector Processing and Microprocessor

6.1 Parallel Processing

Parallel processing is used to increase the computational speed of computer systems by performing multiple data-processing operations simultaneously. For example, while an instruction is being executed in ALU, the next instruction can be read from memory. The system can have two or more ALUs and be able to execute multiple instructions at the same time. Parallel processing is established by distributing the data among the multiple functional units. For example, the arithmetic, logic, and shift operations can be separated into three units and the operands diverted to each unit under the supervision of a control unit.

The main advantage of parallel processing is that it provides better utilization of system resources by increasing resource multiplicity which overall system throughput.



6.2 Pipelining (v.v.Imp)

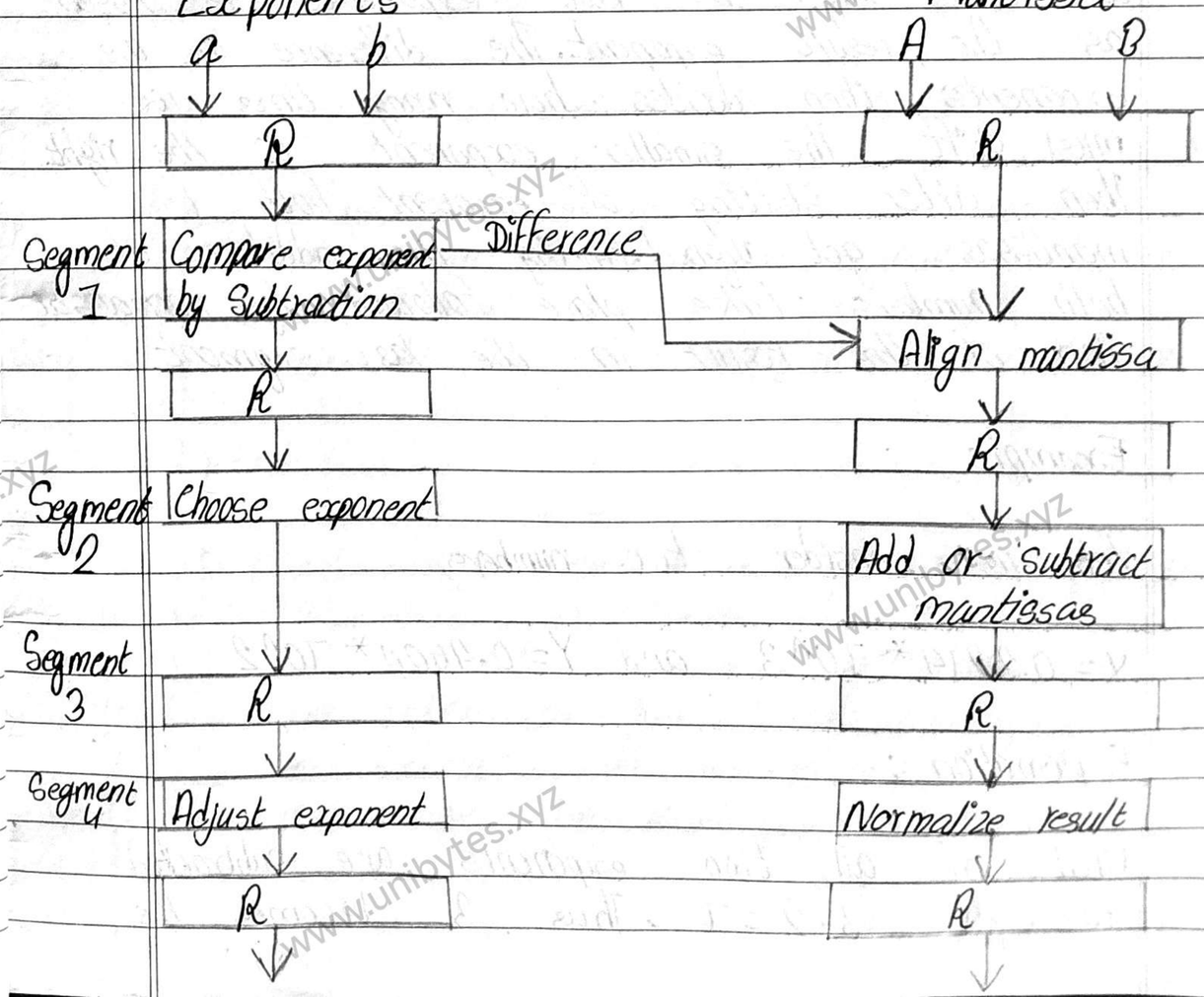
- Pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased.
- A pipeline can be visualized as a collection of processing segments through which binary information flows.
- It is characteristic of pipeline that several computations can be in progress in distinct

The result obtained from computation in each segment is transferred to the next segment in pipeline. The register provide isolation between each segment so that each can operate on distinct data simultaneously.

segment at the same time.

6.3 Arithmetic Pipeline (v.imp)

An arithmetic pipeline divides an arithmetic problem into various sub problems for execution in various pipeline segments. It is used for floating point operations, multiplication and various other computations. The process or flowchart arithmetic pipeline for floating point addition is shown in the diagram. They are usually found in very high speed computers.



Floating point addition using arithmetic pipeline:

The following sub operations are performed in this case:

i) Compare the exponents
ii) Align the mantissas
iii) Add or subtract the mantissas
iv) Normalise the result

First of all two exponents are compared and the larger of two exponents is chosen as the result exponent. The difference in the exponents then decides how many times we must shift the smaller exponent to the right. Then after shifting of exponent, both the mantissas get aligned. Finally the addition of both numbers take place followed by normalisation of the result in the last segment.

Example

Let us consider two numbers,

X = 0.3214 * 10^3 and Y = 0.4500 * 10^2

Explanation :

First of all two exponents are subtracted to give 3-2=1. Thus 3 becomes the exponent of result and the smaller exponent is shifted 1 times to the right to give.

Y = 0.0450 * 10^3

Finally the two numbers are added to produce

Z = 0.3664 * 10^3

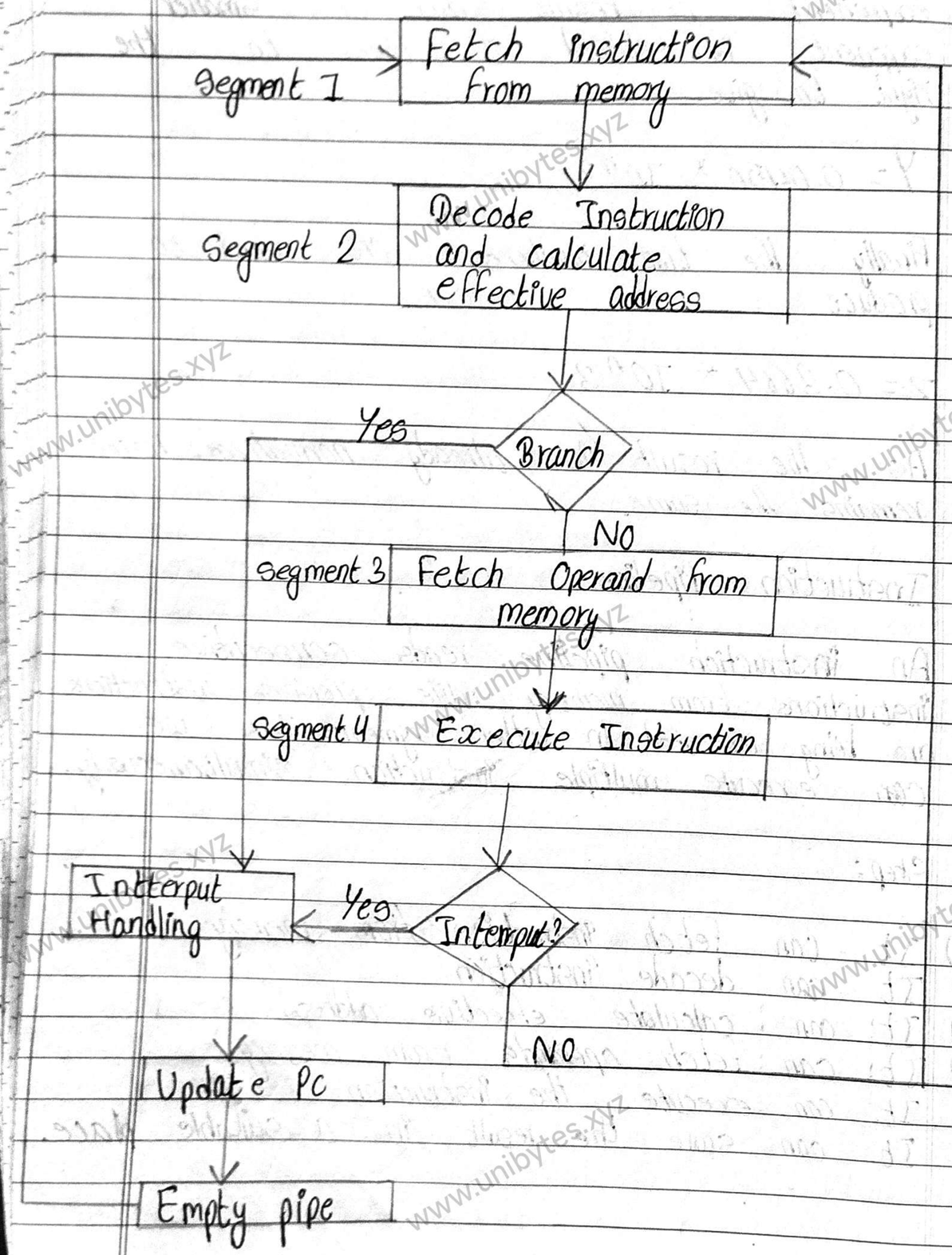
As the result is already normalized. result remains the same.

6.3 Instruction Pipeline

- An instruction pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments. Thus we can execute multiple instruction simultaneously.

Step:

i) It can fetch instruction from memory
ii) It can decode instruction
iii) It can calculate effective address.
iv) It can fetch operands from memory
v) It can execute the instruction
vi) It can save the result in a suitable place.



i) Segment 1

- The instruction fetch segment can be executed using first-in-first-out (FIFO) buffer

ii) Segment 2

- The instruction fetched from memory is decoded in the second segment. The effective address is computed in an independent arithmetic circuit.

iii) Segment 3

- An operand from memory is fetched in the third segment

iv) Segment 4

The instruction are finally implemented in the final segment of the pipeline organization.

Example

Stage	1	2	3	4	5	6	7	8	9	10	11	12	13
1	FI	DA	FO	EX									
2		FI	DA	FO	EX								
3			FI	DA	FO	EX							
4				FI	-	-	FI	DA	FO	EX			
5								FI	DA	FO	EX		
6									FI	DA	FO	EX	
7										FI	DA	FO	EX

Imp

6.4 Flynn's Classification of Parallel Processing.

- One classification introduced by M.J. Flynn consider the organization of computer system by the number of instructions and data items that are manipulated simultaneously.
- Multiprocessing can be defined using Flynn's classification, it is based on multiplicity of instruction stream and data stream in a computer system.
- An instruction stream is a sequence of instruction executed by computer.
- A data stream in a sequence of data which includes input data or temporary results.
- They are classified into 4 types.

i) SISD (single instruction single Data)
ii) SIMD (single instruction multiple Data)
iii) MISD (Multiple instruction single Data)
iv) MIMD (Multiple instruction Multiple Data)

Imp

6.5 Pipeline Conflicts (Hazards)

A pipeline hazards occur when the instruction pipeline deviates at some phase, some operational conditions that do not permit the continued execution.

Three major difficulties :

i) Resource Conflicts

Caused by access to memory by two segments at the same time. Resolve by using seperate instruction and data memories.

ii) Data dependency

Conflict arise when an instruction depends on the result of a previous instruction, but this result is not ~~valid~~ yet available.

iii) Branch difficulties

Arise from branch and other instructions that change the value of PC.

6.6 Data Dependency (phone)

Imp

6.7 Vector Processing

Vector processing is a computer method that can process numerous data components at once. It operates on every element of entire vector in one operation. Normal computational system are not enough for special processing requirements. Such as, in special processing system like AI, weather forecasting system, the normal system are not sufficient.

www.unibytes.xyz

- In many science and engineering applications, the problem can be formulated in terms of vectors and matrices that lend themselves to vector processing.

Application Areas of Vector processing
- Long-range weather forecasting
- Petroleum explorations
- Medical diagnosis
- Image processing

6.8 Vector Operations

A vector is an ordered set of one-dimensional array of data items. A vector V of length n is represented as a row vector by $V = [V_1, V_2, V_3 \ldots V_n]$.

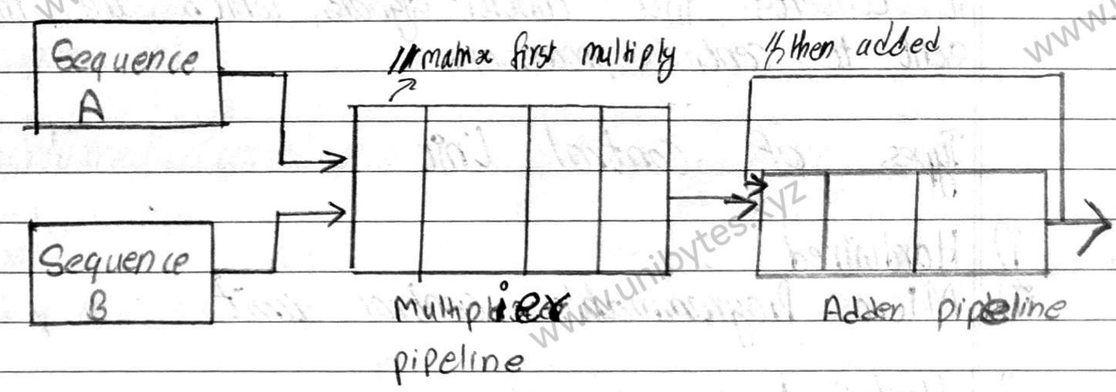
A conventional sequential computer is capable of processing operands one at a time. ~~The~~ A computer capable of vector processing eliminates the overhead associated with the time it takes to fetch and execute the instruction in the program loop

Conventional Computer	vector
Initialize I = 0	C[I:100) = A(I:100) + B(I:100)
20 Read A(I)	
Read B(I)	
Store C(I) = A(I) + B(I)	
Increment I = I + 1	
If I ≤ 100 goto 20	

6.9 Inner Product

In general, the inner product consits of the sum of k product term of the form.

$$C = A_1B_1 + A_2B_2 + A_3B_3 + \ldots$$



6.10 Multiprocessor system

A multiprocessor is a computer system with two or more central system with two or more central processing units (CPUs) share full access to a common RAM. The key objective of using it is to boost the system's execution speed. It helps to improve computer speeds, performance.

Characteristics:
i) Consists of more than one CPU
ii) Fast processing
iii) Reliability
iv) Cost ~~t~~ Effective
v) Simulaneous processing of programs.

Chapter 4 Microprogrammed Control

The control unit is a part of the computer's central processing unit (CPU), which directs the operation of the processor. A control unit works by receiving input information which it converts into control signals, which are then sent to central processor.

Types of Control Unit

i) Hardwired
ii) Micro Programmable control unit

i) Hardwired control unit

Control logic is implemented with Gates, flip-flops, decoders and other digital circuits. The instruction decoder constitutes a set of many decoders that decode different field of the instruction opcode.

Advantage:
Can be optimized to produce a faster mode of operation.

Disadvantage:
Updation in control logic is difficult.

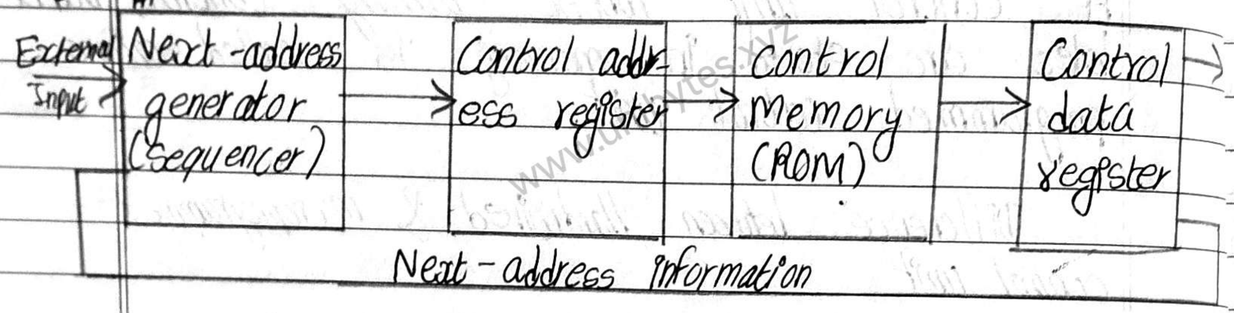
Micro Programmed Control Unit

A control unit whose binary control variables are stored in memory is called micro-programmed control unit.

Difference between Hardwired & microprogrammed control unit - Imp

Hardwired Control Unit	Microprogrammed Control Unit
i) Speed is fast during execution	Speed is low during execution
ii) Implementing hardwired structure is costly	Implementing microprogrammed is not costly.
iii) It works well for simple instructions.	It work well for complex instructions also.
iv) RISC style instruction	CISC style instruction.
v) It is a circuitry approach	It is implemented by programming
vi) It has small size of instruction set.	It has large ~~inst~~ size of instruction ~~stage~~ set
vii) less chips area is required.	More chips area is required
viii) Occurence of error is more	viii) Occurence of error is less.
ix) Control memory is absent	Control memory is present.
x) Complex decoding and sequencing logic.	Easier decoding and sequencing logic.

Micro-programmed Control Organization



- The control memory address register specifies the address of the micro-instruction.
- The control memory is assumed to be a ROM, within which all control information is permanently stored.
- The control Address register holds the microinstruction fetched from the memory.
- The micro-instruction contains a control word that specifies one or more micro-operation for the data processor.
- While the micro-operations are being executed, the next address is computed in the ~~then~~ next address generator circuit and the transferred into the control address register to read the text of next microinstruction.
- The next address generator is often reffered to as a micro-program sequences, as it determines the address sequence that is read from control memory.

Micro-operation

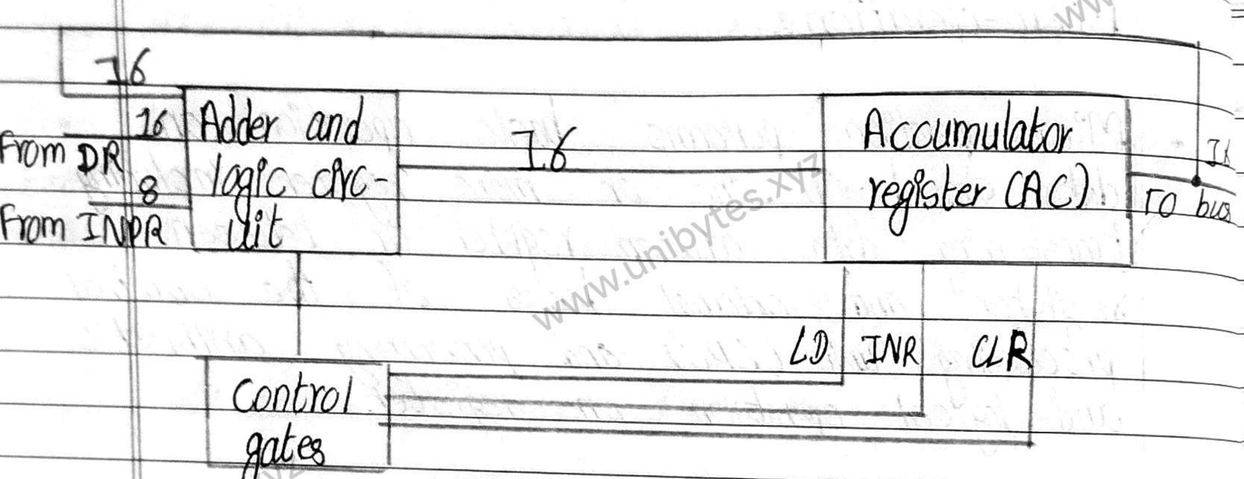
- Micro-operation performs basic operation on data stored in one or more register including transferring data between register or between register and external buses of the central processing unit (CPU) and performing arithmetic and logical operation on register.

4.2 Accumulator

An accumulator is a register for short-term, intermediate storage of arithmetic and logic data in a computers CPU.

Design of Accumulator Logic:

- One set of 16 inputs comes from the output of AC. Another set of 16 inputs come from the Data Register (DR).
- A third set of eight inputs comes from the input register INPR.
- The output of the address and logic circuit provide the data inputs for the register. In addition, it is necessary to include logic gates for controlling the LD INR and CLR in the register and for controlling the operation of the adder and logic circuit.

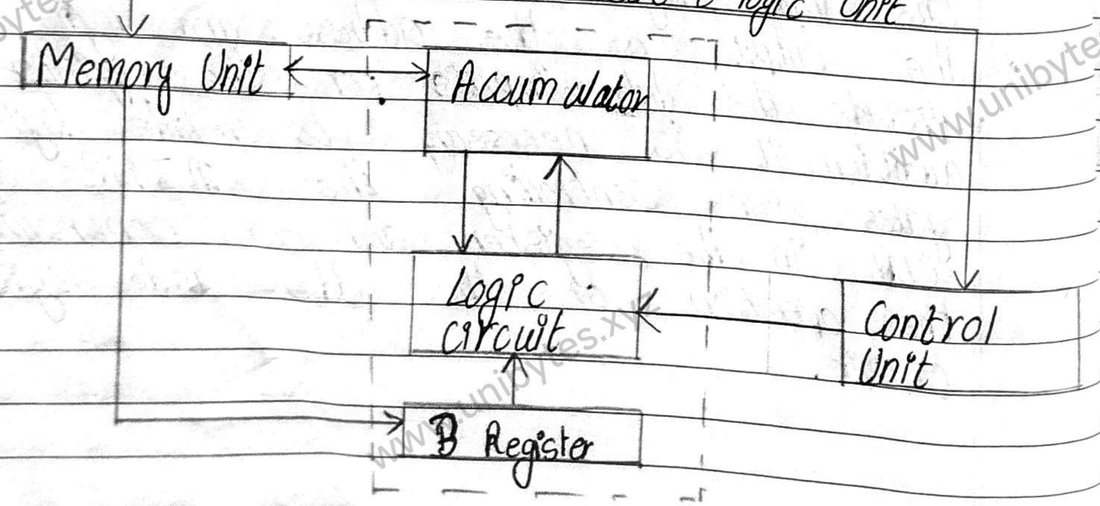


Circuits associated with AC

4.3 ALU organization

- Various circuits are required to process data or perform arithmetic operation which are connected to microprocessor ALU.
- Accumulator and data Buffer stores data temporarily. These data are processed as per the control instruction to solve problem such as addition, multiplication etc.

organization of arithmetic & logic Unit

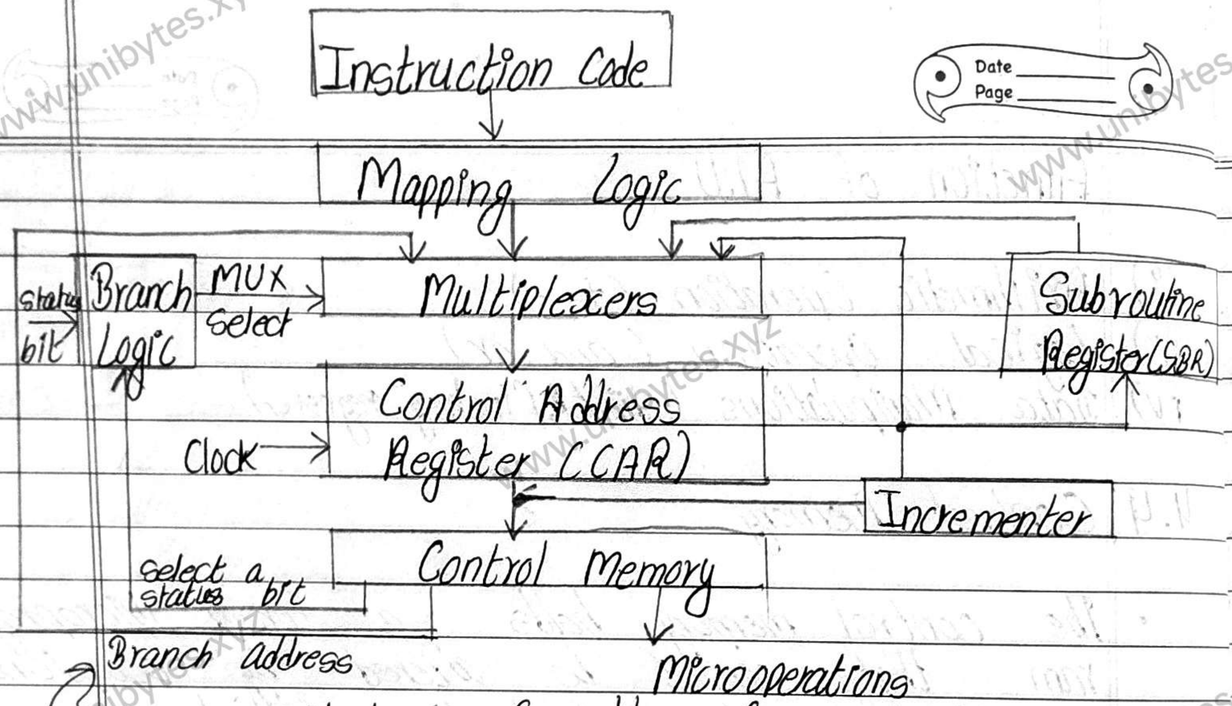


Function of ALU

i) Arithmetic Operation (+, -)
ii) Logical Operation (and, or)
iv) Data Manipulations (shift, flushing register)

4.4 Control Memory

- The control memory holds a fixed microprogram that cannot be altered by occasional user. Each machine instruction initiates a series of microinstruction in control memory.
- When the control signals are generated by hardware using conventional logic design technique, the control unit is said as hardwired.
- The principle of microprogramming is systematic method for controlling the microoperation sequences in a digital computer.
- A control unit whose binary control variables are stored in memory is called microprogrammed control unit.
- A memory that is part of control unit is referred to as a control memory.
- To execute any instruction the microinstruction stored in its control memory must be executed.



Selection of address for control memory

yo address sequencing ko ho ↳ yo ~~micro program sequence ko diagram ho~~

In the above diagram, we can see a a instruction code and it transfer into mapping logic which convert the instruction code into address. The ~~address~~ converted address is transform into multiplexers which get input from incremented ~~addr~~ address register, subroutine register, and branch addreches. Then the status bit select the branch logic which can be conditional or unconditional both from which the MUX select. CAR holds the microinstruction. The control memory hold the fix microprogram. After the CAR the address is incremented by the incrementer and subroutine register hold the incremented address which is incremented by one. The subroutine help the to ~~system~~ cover the system.

3.5 Address Sequencing / instruction

- Microinstruction are stored in control memory in groups, with each group specifying a routine.
- An initial address is loaded into control address register (CAR) when power is turned on in the computer. It is the address of the first microinstruction that activates the instruction fetch routine.
- The control memory will find the effective address of operand.
- Next, micro-operation is generated, which will be used to execute the instruction fetched from memory.

The control memory required the capabilities of address sequencing, which is described below:

- On the basis of the status bit conditions, the address sequencing selects the conditional branch or unconditional branch.
- Addressing sequence is able to increment the ~~CA~~ CAR.
- It provide the facility for subroutines calls and returns.
- A mapping process is provided by the addressing sequence from the instructions bits to a control memory address.

3.6 Conditional Branching

Conditional branching is obtained by using parts of microinstruction to select a specific status bit in order to determine its condition. The status bit, together with the field in the microinstruction that specifies a branch address, control the branch logic. The branch logic hardware may be implemented by multiplexer. If the selected status bit in the 1 state, the output of the multiplexer is 1; otherwise 0. The 1 output transfer the branch address from the micro-instruction into the control address register. 0 output increment the address register.

3.7 Mapping of Instruction

Each instruction has its own microprogram routine stored in a given location of control memory. The transformation from the instruction code bits to an address in control memory where the routine is located is called mapping process. The mapping process converts the 4-bit opcode to a 7-bit address for control memory. The mapping process take place in the bit ~~code~~ of opcode. For mapping we add One '0' at MSB and two '0' at LSB.

Mapping Logic

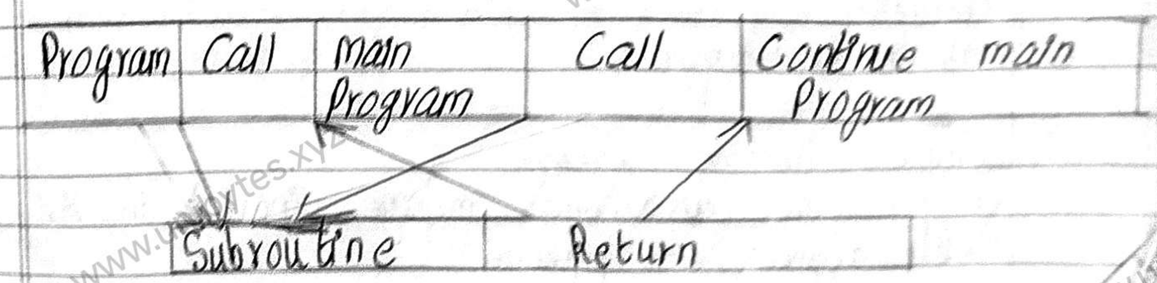
- CAR (0,1,6) <- 0
- CAR (2,5) <- IR (1 4)

Computer Instruction		1	0	1	1		
Mapping bits :	0	x	x	x	x	0	0
Microinstruction address :	0	1	0	1	1	0	0

Mapping from instruction code to microinstruction address.

3.8 Subroutines

Subroutines are programs that are used by other routines to accomplish a particular task. A subroutine can be called from any point within the main body of the microprogram. For example, the sequence of micro-operations needed to generate the effective address of the operand for instruction is common to all memory reference instruction. This sequence could be a subroutine that is called from within many other routines to execute the effective address computation.



Handling Branch of instruction

- Branch hazard arises from branch and other instructions that change the value of program counter (PC).

- One of the major problems in operating an instruction pipeline is the occurence of branch instructions.
  - A uncondition branch always alters the sequential program flow by loading the program counter with the target address.

In a conditional branch, the control selects the target instructions if the condition is satisified of the next sequential instruction if the condition is not satisfied.

A variety of approaches have been used to deal with branch hazard and they are described below.

i) Prefetch branch target
gesma target instruction lai paile jo fetch garna. and ~~branch~~ both are store until the branch execute

ii) Multiple streaming

iii) Branch Transfer Buffer
It is an associative memory included in the fetch segment of pipeline.

It store address of all the previously execut instruction branch and the target instruction for that branch.

It also store the next few instructions after the branch target instruction.

iv) Loop buffer.

High speed register file mainted by the instruction fetch segment of pipeline and containing the most recently fetched instructions in sequence. If a branch is to be taken, the hardware first checks ~~is~~ whether the branch target is with in buffer. If so, the ~~no~~ next instruction is fetch from buffer.

v) Branch prediction

It uses some additional logic to guess the outcome of a conditional branch instructions before it executed. It predict the result ~~of the branch condi is true to~~ for conditional branch instruction. Popular approach are: ~~no~~ predicit never taken, predict always taken, predict by opcode.

vi) Delayed branch

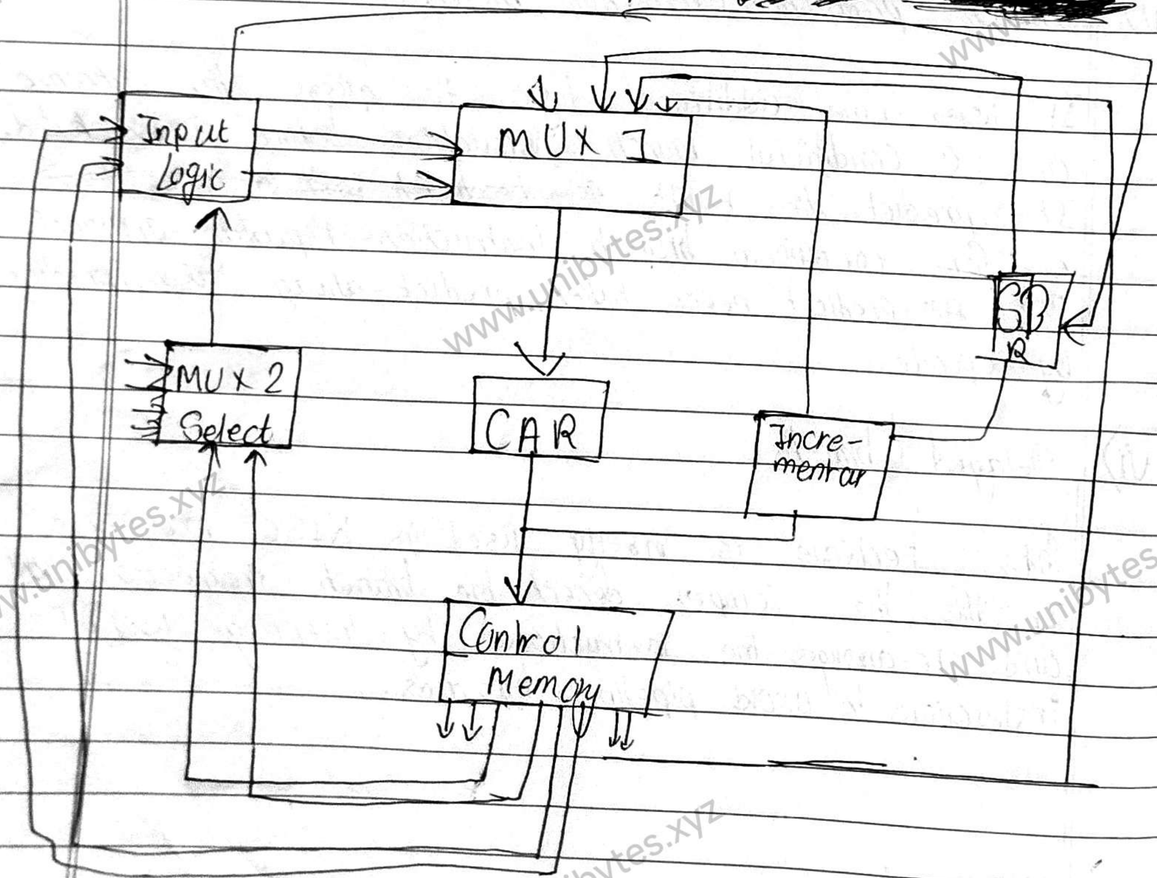
This technique is mostly used in RISC processors. In this the compiler detect the branch instruction and and re-arranges the instructions by inserting useful instructions to avoid pipeline hazards.

www.unibytes.xyz

* Microprogram Sequencer – (Imp)

The basic components of a microprogrammed control unit are the control memory and the circuits that select the next address. The address selection part is called microprogram sequencer.

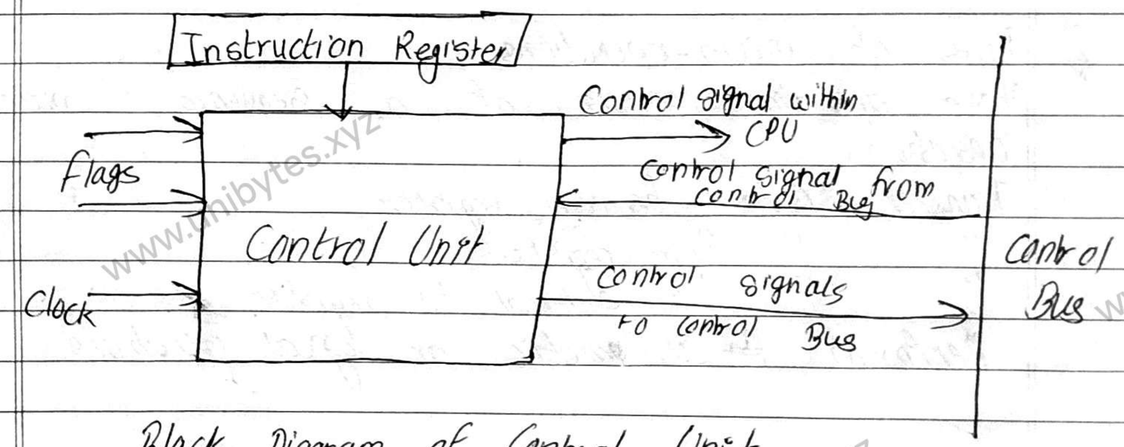
The purposed of the microprogram sequencer ~~to~~ is to present an address to the control memory so that a microinstruction may be read and executed.



* Design / Structure of Control Unit (Imp)

Control Unit generate the ~~control signals~~ timing and control signals for the operations of the computer. It is the responsibility of the control unit to tell the computer's memory, arithmetic / logic unit, and I/O devices how to respond to the instructions that have been sent to the processor.

Types of control Unit
- i) Hardwired CU
- ii) Micro programmable CU



Block Diagram of Control Unit

Basic requirement of Control Unit

The functional requirement of the control unit are those functions that the control unit must perform and these are the basis ~~of~~ for the design and implementation of CU.

Phone page 67.pdf.

A three step process that leads to the characterization of Control Unit.

1) Define the basic elements of the processor.
2) Describe the micro-operations that the processor performs.
3) Determine the functions that the Control unit must perform to cause micro instruction to perform

1) Basic element of processor

i) ALU
ii) Register

2) Types of micro-operations
These operation consists of a sequence of micro operation.

i) Transfer data between register
ii) " " from registers
iii) " " " external to register
iv) Perform ~~the~~ arithmetic or logical operations

3) Functions of CU.

i) Sequencing: The CU causes the CPU to step through a series of micro operations in proper sequence base the program being executed
ii) Execution: The CU causes each micro-operation to be perform.

BSR → Branch to Subroutine

~~Cl~~ Classification of instruction set

i) Data Transfer instruction. MOV, MVI
ii) Arithmetic Instruction: ADD, SUB, INR, DCR
iii) Logical " : AND, OR, RAR, RAL, CMP
iv) Branching " : JUM, Call and Return
v) Control " : HLT, NOP, EI (enable interput), DI (Disable interp
NOP (Nop operation perform)

Program control & stack instruction

The control instruction control the operation of microprocessor.

Instruction	Opcode
LDA	3A
STA	32
MVI A	~~03~~ 3E
MVI B	06
LXI H	16
ADD A	87
ADD B	80