Mon Tue Wed Thu Fri Sat

Date: __/__/20__

Computer Organization And Assembly Language

(Assignment-02)

Name: ALI RAZA

Roll #: BCS F19 M 513

Section-I

a) Abstraction:-

Abstraction is a technique for establishing a simpler way for a person to intract with a system. removing the details that are un-necessary for the person to interact effectively with that system.

Real life example:

ATM machine is a real life example of abstraction. All are performing operations on the ATM machine like cash withdrawal, money transfer, retrive ministatement etc. But we can't Know internal details about ATM.

b) Finite Word length.

Finite word length is a hardware concept. A Word is the natural unit of data used

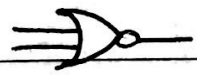
Mon Tue Wed Thu Fri Sat

Date: / /20

by a particular processor design.

Word length is an important characteristic of a specific processor design or computer Architecture, which is often described as n-bit architecture where n is usually equal to 8, 16, 32 or 64.

C) Elementry Boolear operations:

Gate	Symbol	Operation
AND		A.B
OR		A+B
NOT		A′
NAND		(A.B)′
NOR		(A+B)′
XOR		A⊕B

D) Universal gate:

Universal logic gates are the gates that are capable of

implementing any Boolean function without requiring any other type of gate.

They are called so because:

↳ They can realize all binary operations.

↳ All basic logic gates can be derived from them.

Gates:

NAND and NOR gates are universal gates.

E) Boolean function:-

An expression formed by binary variables, logical operators, parenthesis and an equal to sign. The value of a Boolean function can either be 0 or 1.

Example:

$$F(x,y) = xy' + x'$$

Mon Tue Wed Thu Fri Sat

Date: __/__/20

Truth table:

x	y	x'	y'	$x.y'$	$(x.y)+x'$
0	0	1	1	0	1
0	1	1	0	0	1
0	0	0	1	0	0
0	1	0	0	0	0

F) De Morgan's law:

De Morgan's law states that:

$(A+B)' = A' . B'$

$(A.B)' = A' + B'$

Table:

A	B	$\bar{A}$	$\bar{B}$	$A+B$	$\overline{A+B}$	$\bar{A}.\bar{B}$	$A.B$	$\overline{A.B}$	$\bar{A}+\bar{B}$
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
0	1	0	0	1	0	0	1	0	0

Mon Tue Wed Thu Fri Sat

Date: __/__/20

G)

$$f(x,y,z) = \Sigma(2,5,6,7)$$

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

H) **HDL:-**

Hardware description language (HDL) is a specialized computer language used to describe the structure and behaviour of electronic circuits, and mostly, digital logic circuits.

Applications:

There are two most common applications of HDL processing:

1. Hardware simulation:-

We let our HDL programs run inside a h/w simulator to simulate and debug the design.

2. Hardware synthesis:-

The HDL programs can be compiled into h/w implementation using synthesizer and h/w compilation tools.

Example:-

We may need to write hdl programs for combinational, sequential circuits. For example RAM, ALU, Registers etc and test them on simulator before implementation.

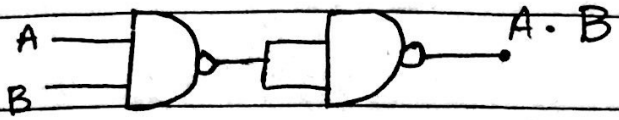
Mon Tue Wed Thu Fri Sat

Date: __/__/20__

i)

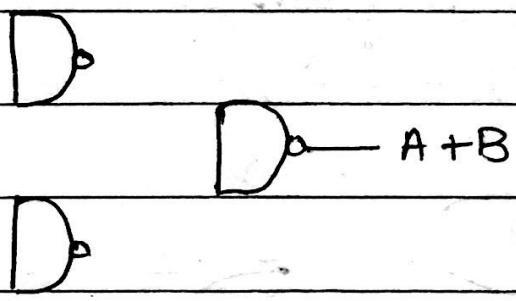
↳ AND using NAND.

We need two NAND gates to make an AND gate.



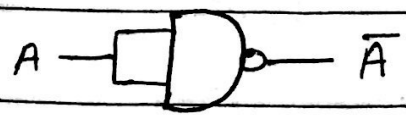
↳ OR using NAND.

We need Three NAND gates to make an OR gate



↳ NOT using NAND

We need a single NAND gate to make NOT gate.



Mon Tue Wed Thu Fri Sat

Date: __/__/20

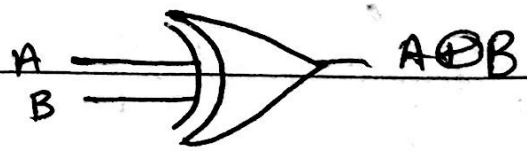
j)

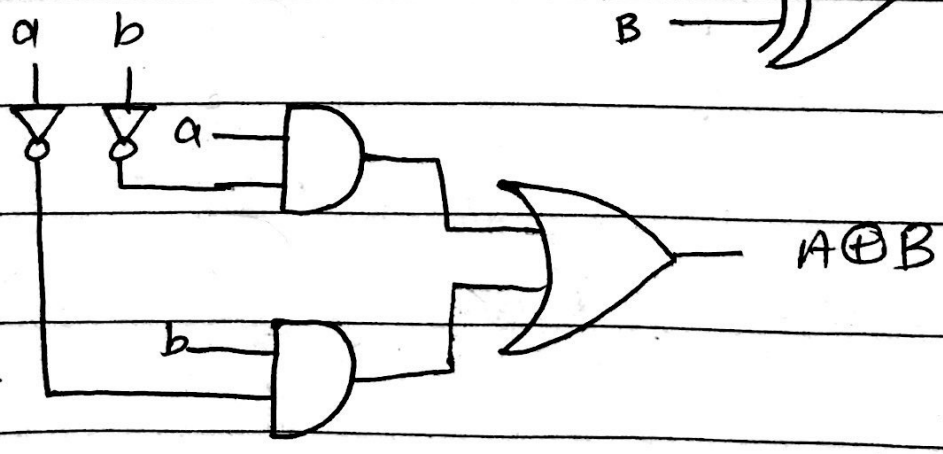
XOR gate's output is 1 iff one of the input is 1.

A ⊕ B

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

circuit diagram:-





Mon Tue Wed Thu Fri Sat

Date: __/__/20

K) Script Based Testing:

A test script is a series of commands to the simulator to test a given problem/system. In script based testing, there may or not generated output file, seperately. Similarly, the compair file can ~~~~ added or not.

L) Decoder:

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.

A BCD-to-7segment decoder is actually a 4x16 decoder, in which the outputs lines from d10 to d15 are not used, as these 6 combinations are don't care

terms.

M) Limitations of 3x8 decoder:

Limitations of a decoder are:

- Output 0 have two definitions. It may mean that all inputs are 0 or that d0 is 1.
- Secondly, if more than one input lines are higher, the encoder generates unpredictable results.

N) Multiplexers:

A multiplexer (also known as data selector) is a combinational circuit that selects binary info. from one or many (2^n) input lines and transmit it to a single output line. The selection of a particular input line is controlled

by a set of n selection lines.

Elementry gates requird to build DMUX:-

To build a DMUX, for example (1X2) DMUX, we may need 2 **AND** gates for each output line an a **NOT** gate for selection line.

Q) Application of MUX & DMUX:-

The common use of multiplexing / demuliplexing logic is to transmit multipu channel on a single shared communication line.

Suppose we have two channels in 2x1 MUX from a cable provider. We want to transmit both these channels via a single communication line. The selection line of MUX is

connected to an oscillator that produces a repetitive train of alternating 0 and 1 signals, so transmitting one bit from each channel in each oscillation.

On destination side, one can use the sel input of DMUX to select the channel to watch.

p) Buss' indexing:-

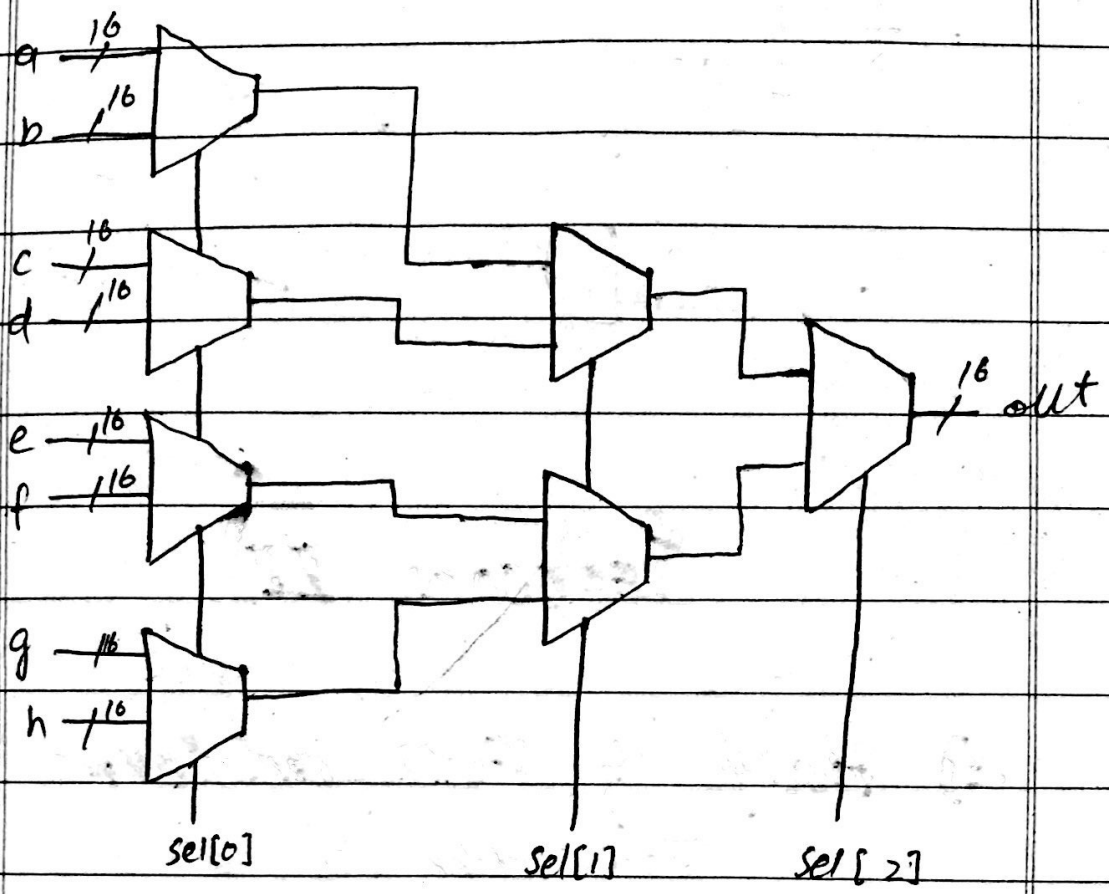
Bunch of bits maniplated together a single entity forms a Bus. Buses are indexed right to left. For example:

If we have a 16-bit bus sav b[16], then b[0] is least significant bit and b[15] is Most significant bit. Bues can be composed of sub buses, such as a 16-bit bus can be made by 2 8-bit buses.

Mon Tue Wed Thu Fri Sat

Date: __/__/20__

Q) Max 8 way 16:-



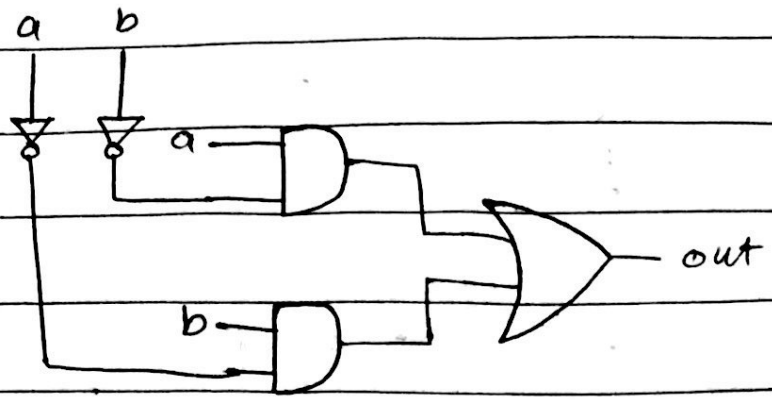
R) Chip in own implementation:-

We **can not** use a chip in its own implementation.

Mon Tue Wed Thu Fri Sat

Date: __/__/20

c)



SECTION II

a) Half & Full adder:

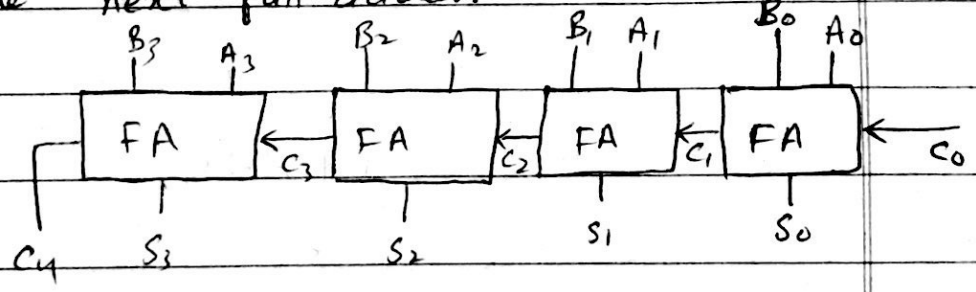
Half adder is a combination circuit that take two inputs and generate two outputs, one is sum and other is carry out.

While half adder is a combinational circuit that take three inputs (usually one is carry in) and generate two

outputs, one is sum and other is carry out.

b) Binary Adder:-

A digital circuit that produces sum of digital binary is called binary adder. It is constructed using full-adder circuits connected in series, with the output carry from one full-adder connected to the input carry of the next full-adder.



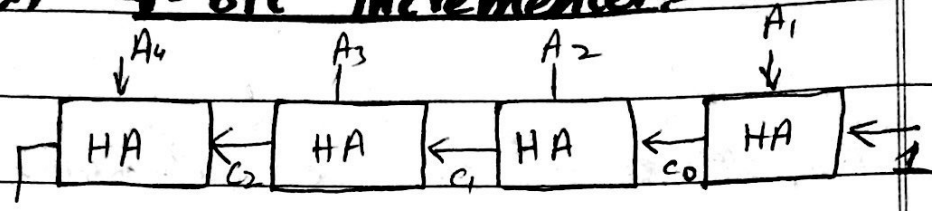
c) Disadvantage of ripple carry adder.

Ripple carry adder does not allow the full adder to use

simultaneously.

Secondly, each full adder has to wait untill the carry bit become available from its previous adder. This increase the proportion time.

a) 4-bit incrementer:-



A is a 4 bit number and this incrementer is designed using half adders [HA]. We will ignor carry out if any.

c) CPU components:-

Main components are:

ALU:-

ALU performs all Arithmetic and logical operations.

Mon Tue Wed Thu Fri Sat

Date: __/__/20__

CU:

It let computer's logic unit, memory, I/O devices know how to program to instructions received from the program.

Registers:

Fast speed memory to store data currently used in processing.

F) 'f' in hack ALU:-

'f' represents selection between OR and AND operation. If f=1 then OR operation is performed.

G)

x = -766 y = 120

x = 1111101000000010

y = 0000000011111000

Mon Tue Wed Thu Fri Sat

Date: __/__/20__

(1)

zx = 0000000000000000

nx = 1111111111111111

zy = 0000000000000000

ny = 1111111111111111

x+y ⇒ 1 1111111111111110

no = 0000000000000001

zr = 0 ng = 0

(x|y)

nx = 0000000101111101

ny = 1111111111001111

0000000101000101

x & y = ↗

no = 1111111010111010

zr = 0 , ng = 1

Mon Tue Wed Thu Fri Sat

Date: __/__/20__

H)

$x = 100 \qquad y = -100$

$x = 0000000001100100$

$y = 1111111110011100$

(0)

$zx = 0000000000000000$

$zy = 0000000000000000$

$x+y \Rightarrow 0000000000000000$

$zr = 1 \qquad ng = 0$

(x-1)

$x = 0000000001100100$

$y = 1111111110011100$

$zy = 0000000000000000$

$ny = 1111111111111111$

$x+y = 0000000001100011$

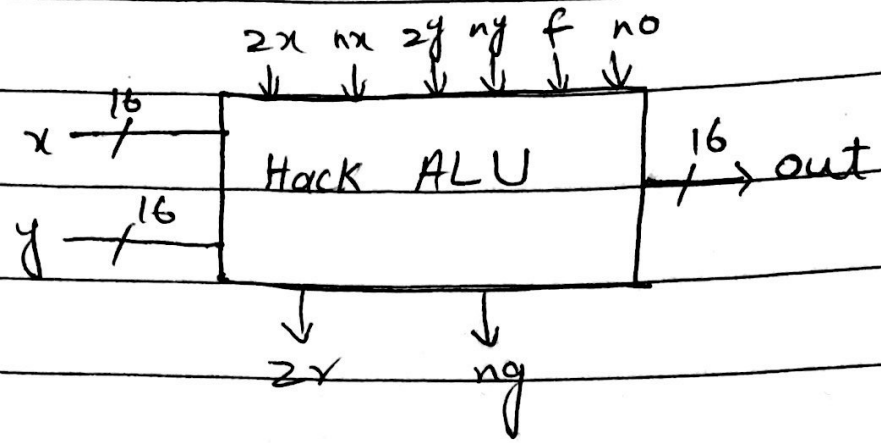
$zr = 0 \qquad ng = 0$

Mon Tue Wed Thu Fri Sat

Date: __/__/20

(i)

ALU circuit:



j) Combinational & sequential:-

↳ Combinational circuits are not time dependent but sequential circuits are time dependent.

↳ Combinational circuits can't store the previous state but sequential circuits can store.

K) Clock in sequential circuit:-

We need clock because in sequential circuits the output of time t is dependent on

input at time t-1 and optionally depends on current input. So in order to build chips that remember what was stored in previous clock cycle, a clock is needed.

4) Latches:

Latches are chips or gates that flip between 2 stable states 0 or 1. It is designed using cross coupled NAND gates.

It's limitation is that its state changes for the duration the clock remains at logic 1 level. Due to this unreliable operation latch can't be applied directly to some other latch that is common clock triggered.

Mon Tue Wed Thu Fri Sat

Date: 1/20

M) D flip flop:-

D flip flop consist of a NAND latch configured in master slave configuration. At beginning of clock cycle D flip flop contain unknown value stored in previous time unit. If we give it some input, after one cycle, input value stored in output line and appeared.

N) D vs SR flip flop:-

In SR flip flop there are 2 inputs and both inputs can't be simultaneously one because it is an invalid state.

On other hand, in D flip flop this drawback is eliminated. D flip flop has only one input and one output, and it'll be same.

Being single input, the design process is comparitively easier.

0) Registers' work:

Register is a very small, fast memory placed inside a CPU, that stores data, instruction or memory addresses. It consist of a load, input and output. Register only loads data when load is 1. and retains data until new data is loaded. So if no data needs to be changed load is set to 0.

1) 1-bit register:-

In order to read what data is stored in register load is set to 0 and output will

Mon Tue Wed Thu Fri Sat

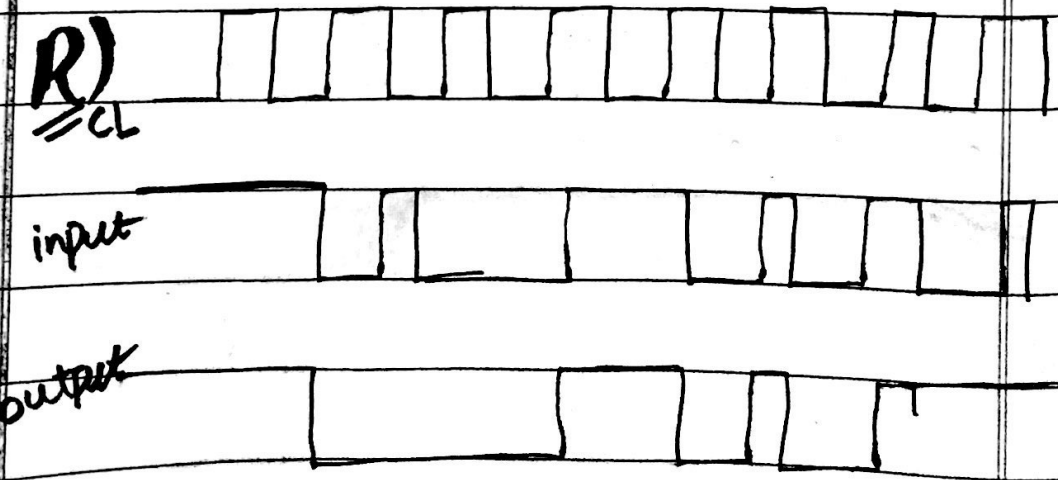
Date: 1/20

show what is stored in 0 registrater. To write data set load to 1 and give data to be written in input and it will be written in register.

Q) Mux in 1-bit register:

We use Mux in a 1-bit register and we use load as seleetion line so that if load is 1 then input is loaded into register and if load is 0 current data is retained in the register.

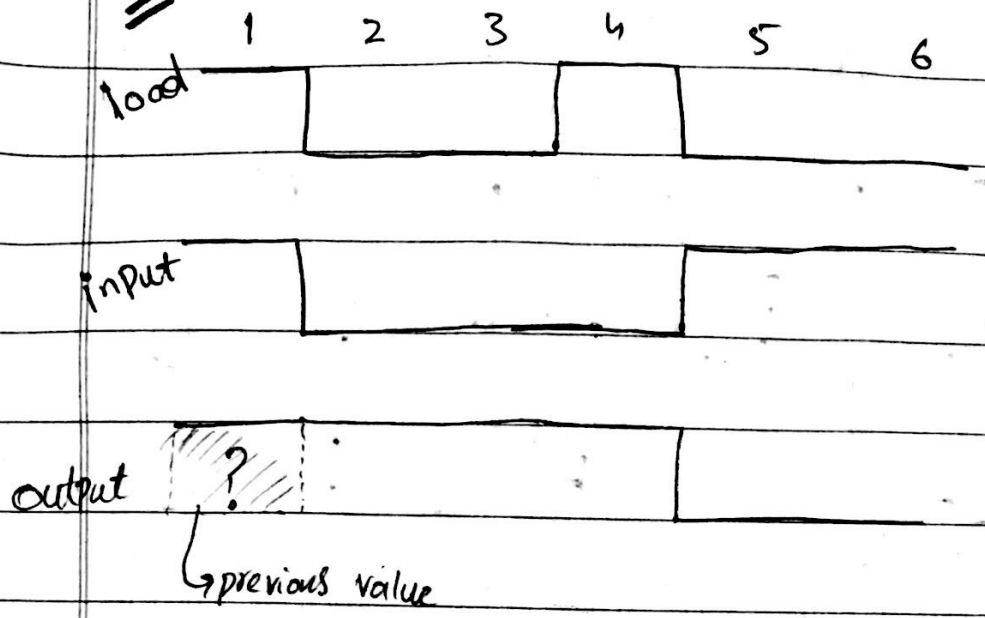
R)



Mon Tue Wed Thu Fri Sat

Date: __/__/20__

5)

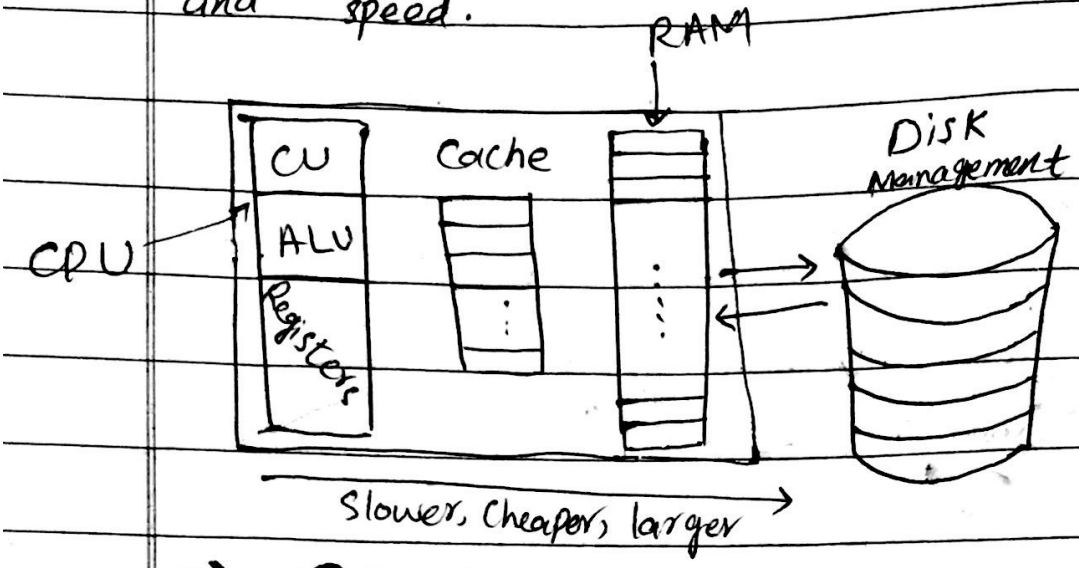


SECTION III

a) Advantage of memory hierarchy:-

The advantage of memory heirarchy is that the most frequently used instructions are stored in a smaller, high speed memory, that makes processing faster and less frequently used data is stored is slower, larger size memory, thus maintaining

a balance between cost, size and speed.



b) RAM:

The computer's main memory is called as RAM. It is called random access because irrespective of its size each word gets accessed at almost the same time.

DRAM:

Dynamic RAM consist of memory cells with each memory cell composed of transistor and capacitor it has to be refreshed

continuasly to store information.

SRAM.

Static RAM does not have capacitors and each memory cell has multiple transistors. It does not ~~have~~ ~~capacitor~~ need to be refreshed periodically.

c) Big endian vs Little endian:-

In Big Endian MSB is stored at lowest address byte. It is used in MIPS, internet. In little endian, ~~the~~ LSB is stored at lowest address byte. Used in X86, ARM etc.

Big endian ⇒ 0x1 0x2 0x3
34 63 72

Little endian ⇒ 72 63 34

Mon Tue Wed Thu Fri Sat

Date: __/__/20

D)

$64M = 64 \times 2^{20}$

Using formula to find address lines:

$$\log_2(64 \times 2^{20})$$

$$= \log_2(67108864) = 26$$

So, 26 address lines would be required.

E)

To make a RAM of 16K registers each 64-bit wide, we need to make each register 64-bit wide instead of 16-bits. So that it will be able to store the input that would also be of 64-bits. The output will also be of 64-bits. The address lines for this RAM would be 14.

Mon Tue Wed Thu Fri Sat

Date: __/__/20__

F) Counters:-

↳ Counters are used for counting number of occurance of event.

↳ Keep or calculate time between events.

↳ And to generate Baud rate

Example:-

BCD Counter

Ring Counter

Johnson Counter

G) program Counter:-

A counter is a special register that stores address of next instruction to be executed.

Control operations:-

Reset : Fetch First Instruction

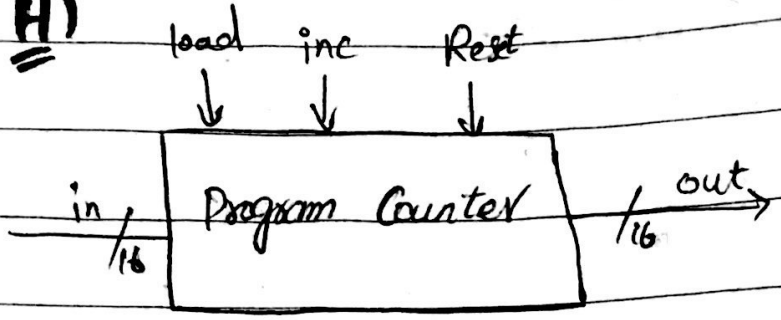
Next : Fetch Next Instruction

Goto : Fetch Instruction on specific location.

Mon Tue Wed Thu Fri Sat

Date: / /20

H)



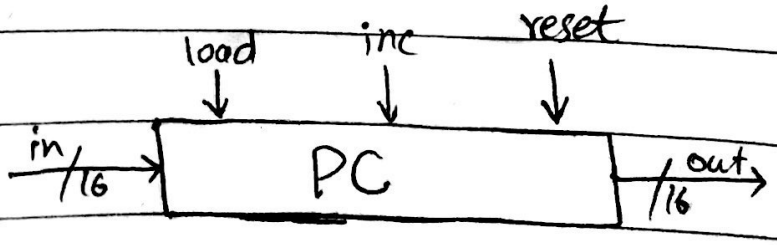
I)

We can make PC loadable by using a MUX. if load is 1, the input is loaded in PC.

Similarly by using MUX we can reset pc. If reset is 1 then 0 is sent to output.

J)

The input and output have to be 64-bit. Increment 16 gould be replaced with Mux64. The register should be of 64-bit.



Mon Tue Wed Thu Fri Sat

Date: __/__/20__

K) Machine & Assembly language:-

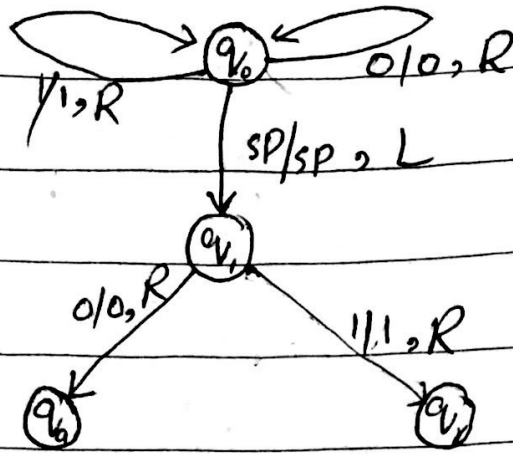
Machine language is directly understonable by the computer. It contains set of instruction in binary language. All commonds are give in binary language.

On the other hand, Assembly Language is a symbolic language that uses symbols to represent machine language instructions making it is easier for us to understand, but requires a program called Assembler to convert assembly language instructions to machine language.

Mon Tue Wed Thu Fri Sat

Date: __/__/20

L)



Q: $\{q_0, q_1, q_a, q_r\}$

Σ: Input alphabets $\{0,1\}$

X: Tape alphabets $\{0, 1, \text{space}\}$

Transition function:-

$$\delta : Q, x \rightarrow Q, x, L/R$$

Starte state $= q_0$

Accept state $= q_a$

Reject state $= q_r$

M) Von Neumann Architecture:-

Von Neumann Architecture described the architecture for digital computer with these components:-

↳ A processing unit that contains ALU

and register.

↳ A CU that contains IR and PC.

↳ A memory unit that stores data and instructions. An I/O machanism.

↳ External storage.

According to stored program concept, we can put a program inside the memory along with the data on which this program will operate.

N) Mnemonics:-

Mnemonics are eglish like words used to represent machine language instructions, to be understood easily by humans. An assembler is a program That converts instructions written in assembly language to machine language so that the computer can understand it.

Mon Tue Wed Thu Fri Sat

Date: __/__/20

Q) ISA:

ISA is a set of instructions, registers, memory space and other features that are visible to assembly language programmers.

It is an interface between hardware and low-level software.

Dimensions:-

- Class of ISA
- Types and sizes of operands.
- Operations
- Memory Addressing Models and Addressing Mode
- Control Flow Instructions
- Encoding of ISA

P)

In load store machine both operands are loaded into

register and then manipulated. The final answer is stored back in memory.

In register-memory machine one operand is in register and other is in memory.

Q) Instruction pointer and Instruction Register :-

Instruction pointer stores address of next instruction to be executed by the CPU.

Instruction Register is used to contain and later decode the instruction to be executed by the CPU.

R) Flow control instruction in ESA :-

Flow control instructions change the flow of control

i.e, instead of executing next instruction, the program branches to the address specified in branching instructions.

Examples:-

Arithmetic : Add, subtract, Divide, multiply

Logic : And, OR, NOT

Flow control: Branch, jump

Procedure calls, return, trap

5)

Yes there can be a processor having 24-bit memory address storing 16-bit data, because there is no such relation between number of address lines in memory system and size of data. This is a linear memory model. Intel 8080 CPU uses this technique.

It has 16-bit address bus. 8-bits are stored at each location when it can store even 64-bit at each location.

7) Direct and Indirect addressing mode:-

In direct addressing mode, one operand is in the register and other is in memory, whose effective is a part of memory.

i.e add R4, M[256] // R4 ← M[256]+R4

In indirect mode one operant is in register and other is in memory, whose address is placed in a register which is specified in the instruction.

i.e, add R2, @R6 // R2 ← M[R6]+R2

Mon Tue Wed Thu Fri Sat

Date: __/__/20

U) I/O interfacing :-

Like processor needs to read/write memory locations, it also needs to read/write different I/O devices. This is I/O interfacing. I/O interface acts as a communication channel between processing unit and external I/O devices.

Intefacing is done in 2-ways:

↳ Memory Mapped

↳ Isolated

~~Inter~~ Isolated interfacing:-

I/O devices are given a seperate addressing region (seperate from memory). These seperate address spaces in x86-x64 are called ports. In isolated I/O there are different read/write instructions for memory and I/O devices.

(The End)