Final – 19-20

(1) (a). (20-21) 1 (b)
(b). note part (moor's Law)
(c). (20-21) 1 (c)

(2) (a) (20-21) 2 (a)
(b). note 8080 ∴ 16 bit address bus?

$2^{16} = 65,536$ bytes = 64 KB

(c). Asynchronous read/write.

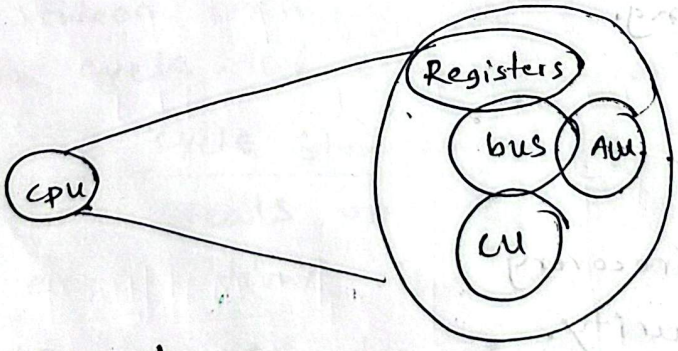
(3) (a). (20-21) 3 (a)
(b). (20-21) 3 (b)

(4) (a). (20-21) 4 (a)
(b). (20-21) 4 (b)
(c). (20-21) 4 (c)

(5) (a). Explain the performance of a processor
(20-21) 1 (a)
(b). (20-21) 5 (b)
(c). (20-21) 5 (c)

(6) (a). (20-21) 6 (a)
(b). (20-21) 6 (b)
(c). (20-21) 6 (c)

(7) (a). By showing the major components of a CPU, briefly describe how a CPU works.



Major component of CPU:

- Arithmetic Logic unit (ALU):
  performs all arithmetic and Logical operations.

- **Control Unit (CU):**
  Controls and co-ordinates all operations. It fetches instructions, decodes them, and tells other components what to do.
- **Registers:**
  Small, very fast memory inside the CPU. They temporarily store data, instructions and addresses.
- **Buses:**
  paths that transfer data and signals.
  (Data, Address, control).

How a CPU Works:

A CPU works in a cycle called the fetch-Decode-execute cycle:

i) **Fetch:**
- the program counter (PC) gives the address of the next instruction.
- the instruction is feteched from memory and stored in the instruction register (IR).

ii) **Decode:**
the control unit (CU) interprets the feteched instruction and decides what action are needed.

iii) **Execute:**
- the ALU performs the required operation
- OR the CU signals other components to move data, access memory, or jump to a new instruction.

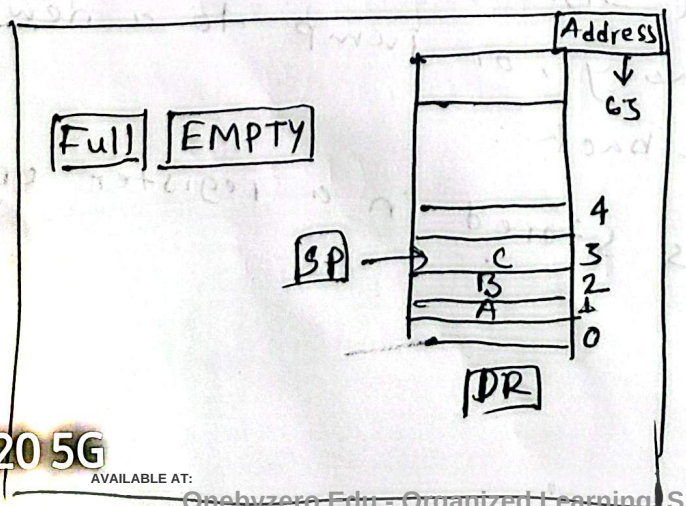
iv) **Store/Write-back:**
The result is stored in a register or sent to memory.

(b) distinguish between RISC and CISC processor.

RISC	CISC
- small and simple instruction set.	- Large and complex instruction set.
- fixed length instruction	- Variable length instruction
- usually 1 cycle per instruction	- multiple cycles per instruction
- more general purpose register	- fewer registers.
- Simple and limited addressing modes	- complex and many addressing modes.
- Larger code size	- Smaller code size
- Simple hardware, easier pipelining	- Complex hardware, harder pipelining
Ex: ARM, RISC-V, MIPS	Ex: X86, Intel 80386.

(c) How stack is organized in CPU? With proper block diagram, describe following types of stack
i) Register and
ii) Memory stack

A stack is a region used to store temporary information such as return addresses, parameters, local variables and saved registers. it follows LIFO → Last In First out



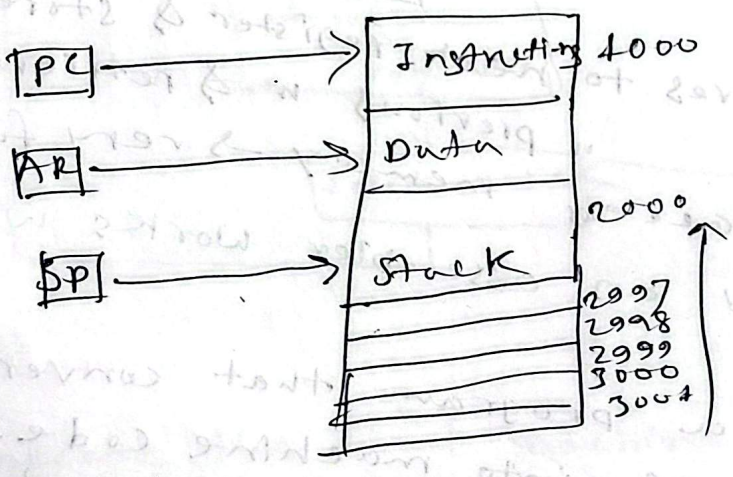
• Stack pointer (SP).
  - A special register in the CPU
  - Always points to the top of the stack
  - changes whenever data is pushed or popped

• push:

• pop

Memory stack
Register stacks

- Stack is stored in main memory (RAM)
- Controlled by the stack pointer (SP) and sometimes a stack Base (SB)
- Larger size compare to Register stack
- Slower than register stack because memory access is slower



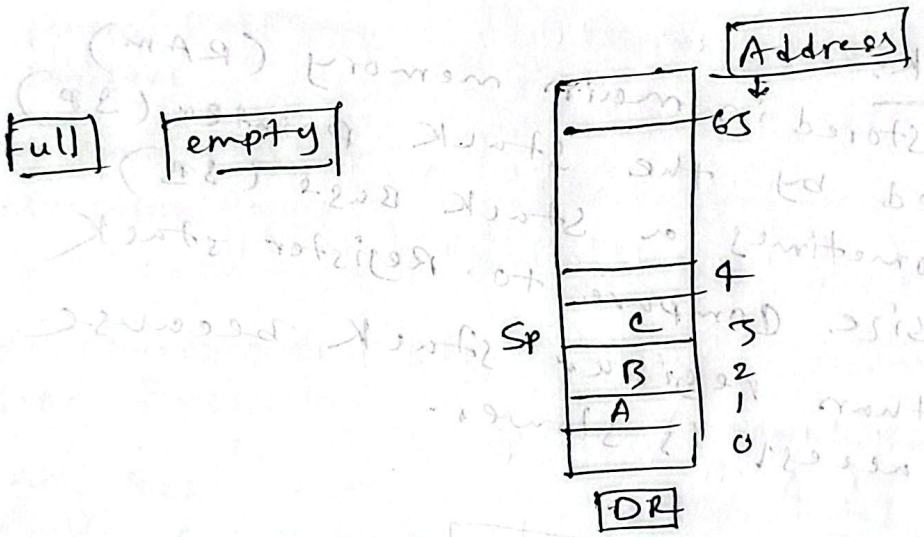
How it works:

push: SP is decreamented, data is writ
to that memory location.

POP: Data is read from SP, SP is increme
- used in function calls, interrupt handling, and context switching.

Register stack:

- The stack is implemented using a set of CPU registers.
- Very fast because registers are inside the CPU
- limited in size
- When the stack becomes full, a stack overflow occurs.



push: SP moves to next register & store data.
pop: SP " " previous " & retrieves it
no need to access memory → very fast.

(8) (a) Describe how an assembler works with its two phases.

An assembler is a program that converts assembly language into machine code.
assemblers generally work in two phases:

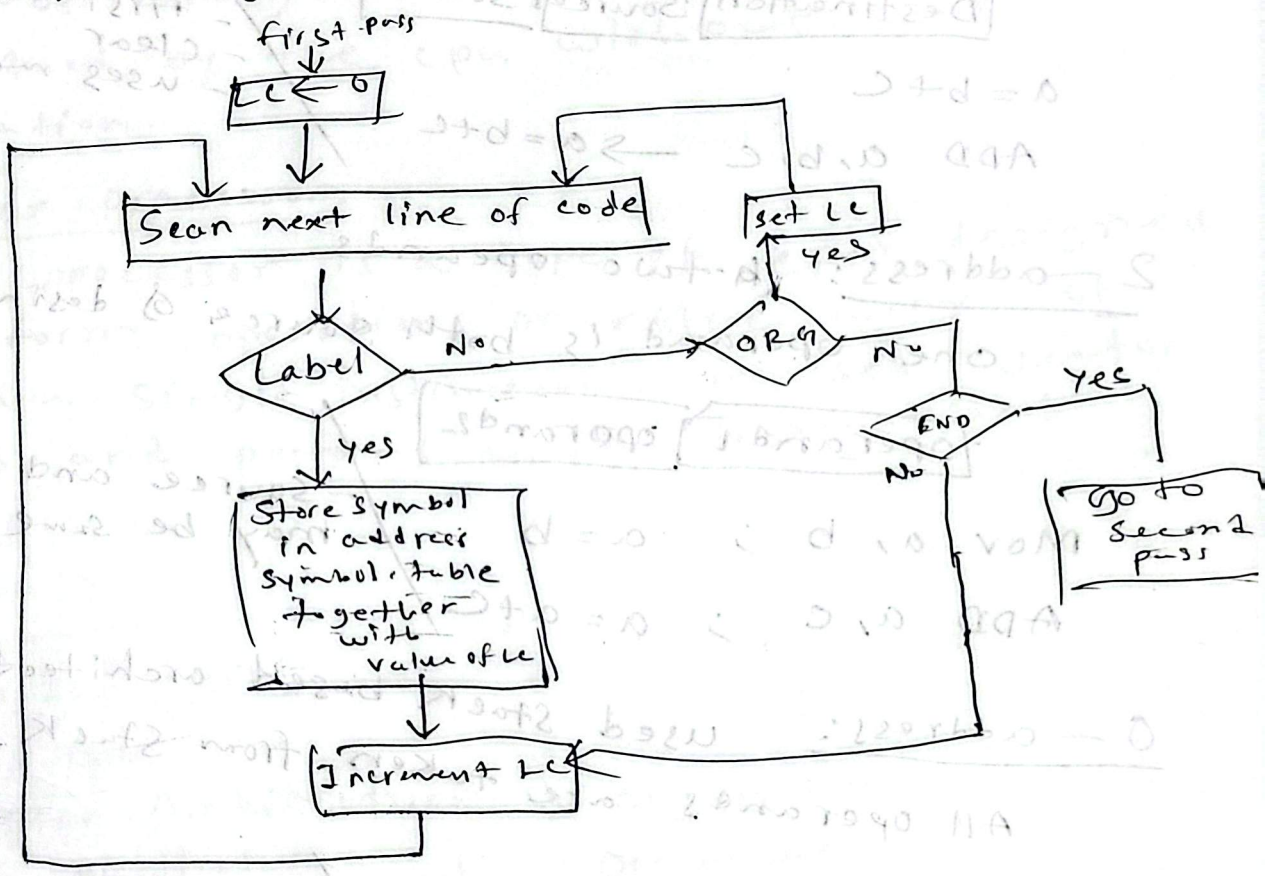
Assembly code → [Assembler] → machine code

first pass (pass-1)

purposes:
- scan the program and create symbol information.

operations:

- Scans the entire source program once.
- Builds the symbol table
- Calculate memory addresses
- identifies directives
- Does not generate machine code



2nd pass:

purpose:

to generate the final machine code using the information collected in pass-1.

Operation:

- read the ~~increment~~ intermediate file created in pass-1
- replaces symbolic operands with actual address from symbol table.
- generates machine opcodes
- handles literals, constants, and directives
- produces final object code

8(b) Distinguish between 0-addresses, 2-addresses 3-address instruction with the example a = b+c

3-address: 3 operands
- 2 source "
- One destination "

Destination	Source1	Source2

a = b+c

ADD a, b, c → a = b+c

- first
- clear
- uses more bits

2-address: two operands.

one operand is both source & destination

operand1	operand2

Mov a, b ; a = b

ADD a, c ; a = a+c

- source and destination may be same

0-address: used stack based architecture.

All operands are taken from stack top.

```
push b
push c
ADD          (b+c)
pop  a
```

Stack based operations

(c) write short note: i) op-code ii) machine language iii) Vector processor

op-code: An op-code is the part of an instruction that specifies the operation the CPU must perform. It tells the processor what to do, (ADD, Sub)

the op-code is essential for defining the action of the instruction.

ii) Machine language:
lowest-level programming language, consisting only of binary code (0 or 1). It is directly understood by the CPU without any translation.

iii) Vector processor:
Vector processor is a type of CPU designed to perform operations on entire arrays of data in a single instruction. It uses vector registers and performs vector instructions.

final 18-19

1. (a) 20-21 (2(b))
(b) Moor's law
(c). Compter Architecture & organization:

Architecture	organization
- Architecture describe what the computer does.	- The organization describes how it does it.
- deals with functional behaviour.	- deals with structural relationship.
- deals with high-level deging issue	- low-level design issue.
- indicate hardware	- indicate hardwar's performance
- Architecture fixed first	- decided after architecture
- low level	- high level

(2) (a). 20-21 5(6)

(b). Mismatch between processor & Main memory.

Reason of mismatch:

- cpu speed increase every generation
- Memory speed " Very slowly
- Memory bottleneck — cpu waits for memory.

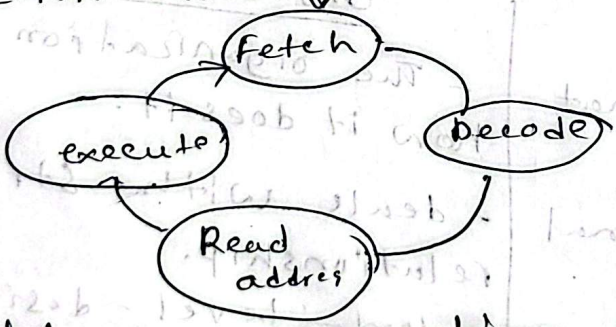
Solve processor - memory mismatch:

① cache memory
② increase memory bandwidth.
③ pipelined memory / SDRAM (next request accept before 1st complete)
④ Multiple-level caches (reduce average access time)
⑤ pre-fetching (fetch before ~~this~~ request)
⑥ Write buffer & Victim cache (Hold data temporarily. when cpu writes)
⑦ Virtual memory + page Replacement optimization
⑧ block transfer (DMA).

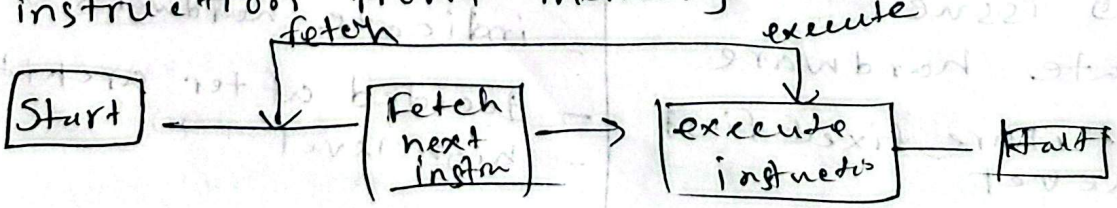
(c). fetch cycle vs instruction cycle

Instruction cycle = fetch + execution

Instruction cycle:



fetch: the instruction fetches the next instruction from memory



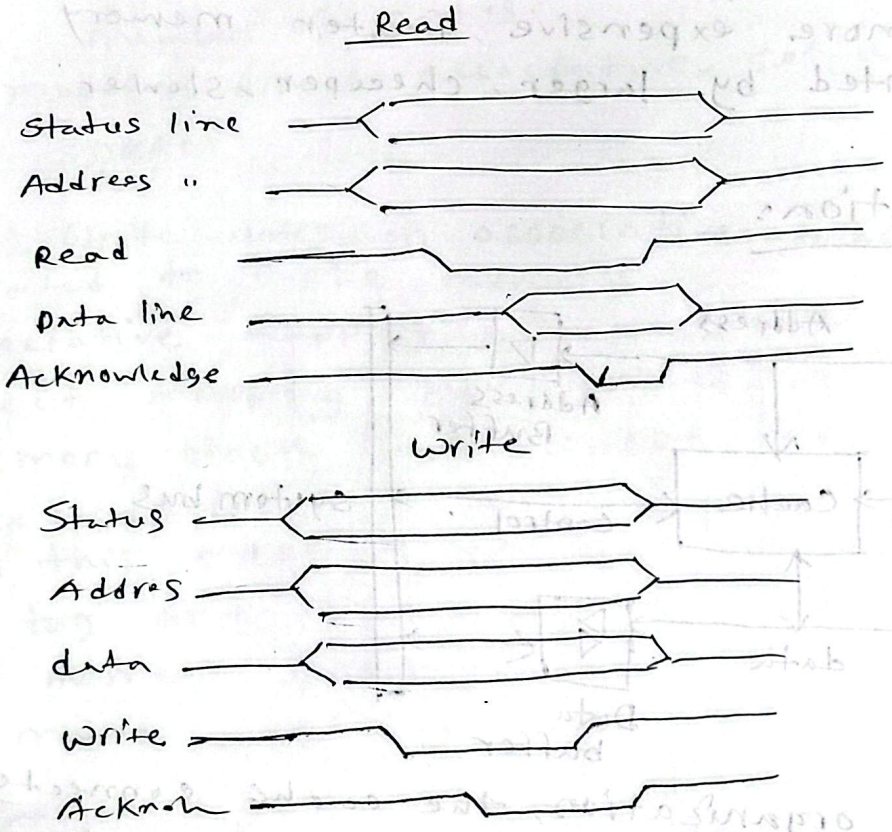
execute:

Cpu decodes the fetched instruction and performs the required operation.

3(a): 20-21 2(a)

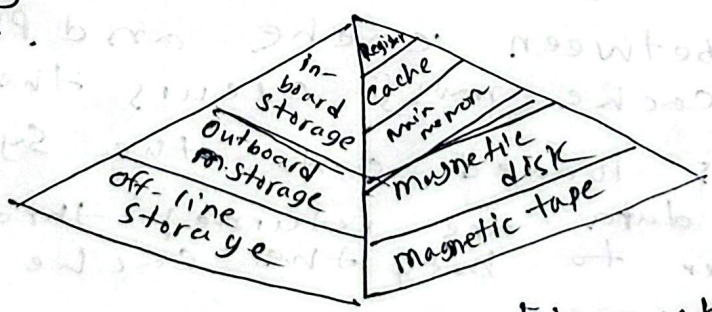
(b). traditional vs high performance bus

(c). Asynchronous bus operations timing.



④ (a) Discuss about the memory hierarchy and typical cache organization.

Memory hierarchy: In the computer system design memory hierarchy is an enhancement to organize the memory such that it can minimize the access time.



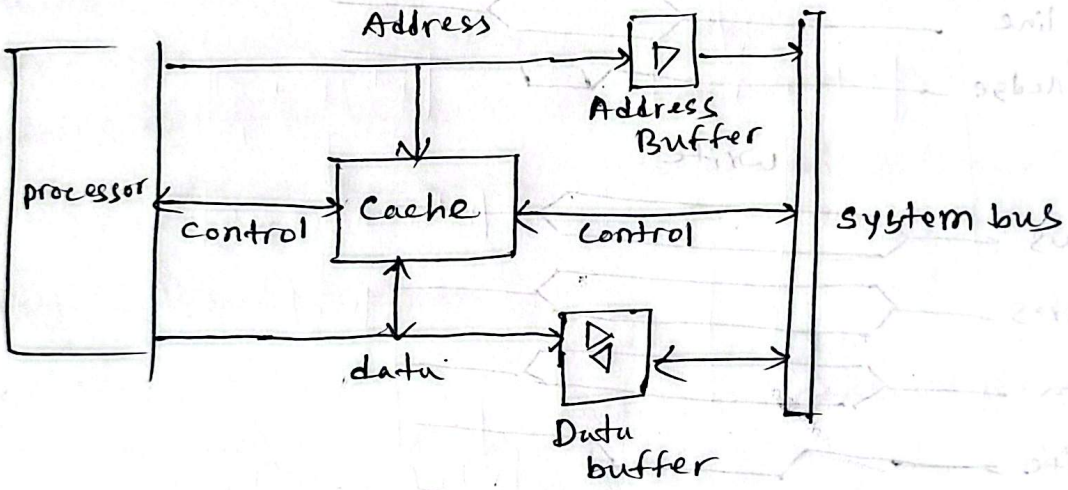
the memory hierarchy

As one goes down the hierarchy, the following occur:

- Decreasing cost per bit
- Increasing capacity
- Increasing access time
- Decreasing frequency of access of the memory by the processor. ~~the sm~~

the smaller more expensive, faster memory are supplemented by larger, cheaper, slower memories.

Cache organization:



In this cache organization, the cache connects to the processor via data, control and address line also attached to data and address buffer which attached to a system bus from which main memory is reached. When a cache hit occurs the data and address buffers are disabled and communication is only between cache and processor. When a cache miss occurs the desired address is loaded onto the system bus and the data are returned through the data buffer to both the cache and the processor.

elements of cache design:

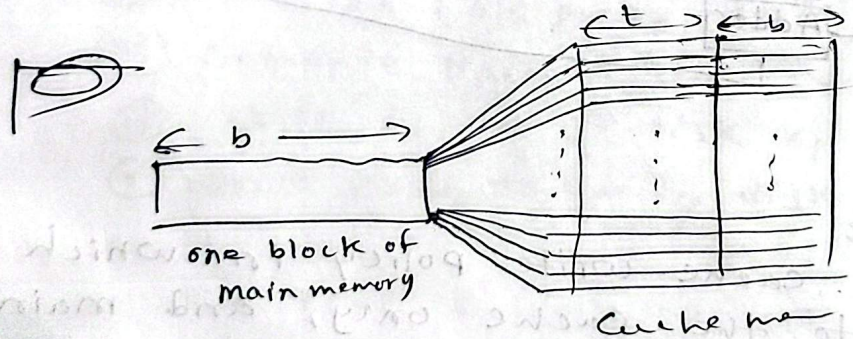
Cache address: (logical, physical)
Cache size.
Mapping function (Direct, associative, set associative)
Replacement Algorithm (LRU, FIFO, LFU)
Write policy (write through, write back)
line size
number of caches.

(b) mapping direct, associative, set associative

(c). SDRAM

(5) (a). Write notes on associative mapping function related to cache memory.

associative mapping overcomes the disadvantage of direct mapping by permitting each main memory block to be loaded into any line of the cache.
In this mapping memory address is interpreted as tag and word. tag uniquely identifies block of memory. Every lines tag is examined to a match cache searching gets expensive.



address length = (s+w) bits
number of addressable unit = 2^{s+w} words or bytes.
block size = line size = 2^w words or bytes
number of blocks in main mem = $\frac{2^{s+w}}{2^w} = 2^s$
" " line in cache = undetermined
Size of tag = s bits.

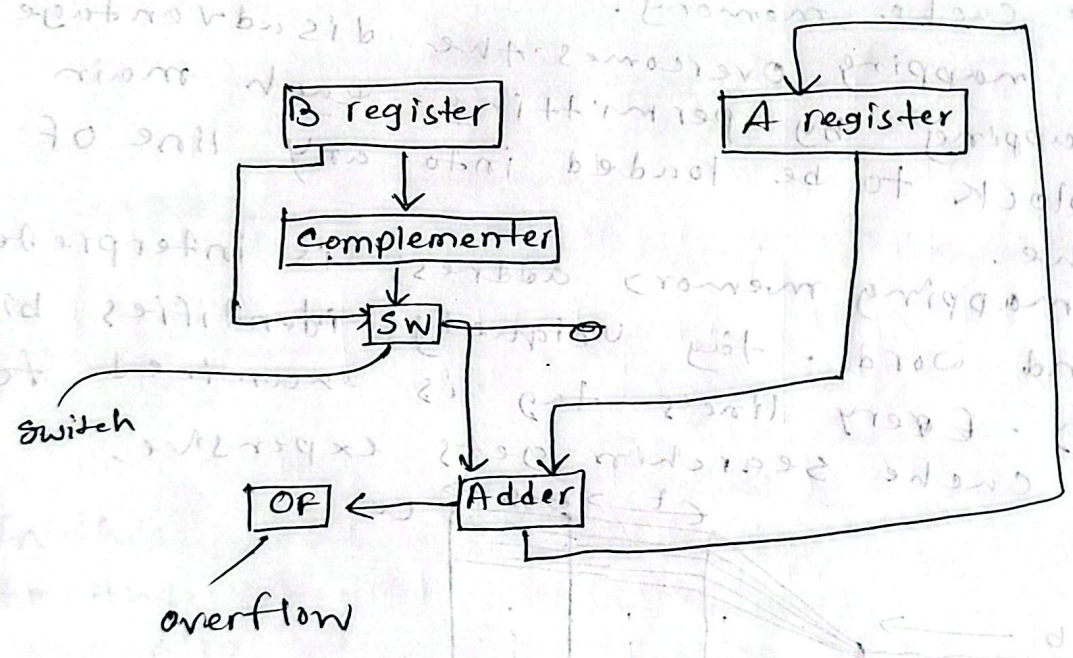
Address structure:

Tag 22 bit	Word 2 bit

22 bit tag stored with each 32 bit block of data.

- Compare tag field with tag entry in cache to check for hit.
- least significant 2 bits of address identify which 16 bit word is required from 32 bit data block.

(b) Hardware addition and substraction



(6) (a) Write back:

Write-back is a cache write policy in which data is written to the cache only, and main memory is updated later only when the block is replaced.

How written block works:

(1) When the processor writes data
   - only the cache block is updated.
   - A dirty bit-1 is set to indicate, the block has been modified.

(2) The modified block is written to main memory only when it is removed from the cache.

adv:

- lower memory traffic → faster performance.
- multiple writes to the same location cause only one memory write at the time of block replacements.
- good for mutiprocessor systems with write buffers.

Disadv:

- Requires maintaining a dirty bit.
- Main memory does not always hold the latest data until cache replacement.
- complecates cache coherence.

6(b) RAID 4

parity defination:

~~The parity on a small write to x.~~

The parity block on x_4 is the XOR of the data blocks.

$$P = x_0 \oplus x_1 \oplus x_2 \oplus x_3$$

updating parity on a small write to x_1

$$P_{new} = P_{old} \oplus x_{1,old} \oplus x_{1,new}$$

(i) Read $x_{1,old}$ (old data)
(ii) Read P_{old} (old purity from x_4)
(iii) compute $P_{new} = P_{old} \oplus x_{1,old} \oplus x_{1\,new}$.
(iv) write $x_{1,new}$ to disk x_1
(v) Write P_{new} to parity disk x_4.

This is the usual small-write procedure and causes the RAID-4 write penalty: 2 read + 2 write = 4 I/O operations for each small-write.

Reconstructing data if a disks fails:

If one disk fails, you can recover its block by XORing the remaining data blocks with purity

$$x_1 = P \oplus x_0 \oplus x_2 \oplus x_3$$

more generally, for any failed data disk x_i:

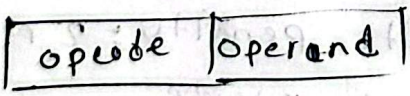
$$x_j = P \oplus_{i \neq j} x_j$$

→ that's all: parity is XOR of data; update parity using
$P_{new} = P_{old} \oplus D_{old} \oplus D_{new}$;
reconstruct missing data by XORing parity with the other data blocks.

7(a). Distinguish between RAID 4 and RAID 5

Comparison	RAID 4	RAID 5
Storage pattern	Block-Stripping	Block-Stripping
parity	Single disk parity	Distributed parity
Minimum no of disk	2	3
cost of set up	Affordable	Affordable
Modern relevance	obsolete	Not obsolete but rarely used
performance and speed	Decent	Better
Fault tolerance	Single disk failure	Single disk failure
Best Application	High data transfer task	High data transfer task

(b). What is the purpose of using Addressing mode techniques in computer?
Addressing modes are used to specify how to operand is accessed by the CPU. they make instructions flexiable, efficient and powerful.



main purposes:
① Reduce instruction size
② increase programming flexibility
③ support complex data structure
④ enable efficient memory use.

⑤ improve program redeablity.
⑥ optimize execution Speed.

(c) classify computer instruction? explain logical and bit manipulation instruction.

① Data transfer instruction (mov, LOAD, STORE, push, POP)
② Arithmetic instruction (ADD, SUB, MUL)
③ Logical and bit manipulation instruction (AND, OR, XOR) (SHL, SHR, ROL, ROR)
④ Control transfer / Branch instr (JMP, CALL)
⑤ input / output instr. (IN, OUT)
⑥ processor control instruction (WAIT, INT)

(c) Interrupt-Driven I/O Technique:

Interrupt-driven I/O technique in which the CPU does not contineously wait for an device instead, the cpu continues executing other instructions, and the I/O device sends an interrupt signal when it is ready for data transfer.

How it works:

- cpu issues an I/O command to a device.
- cpu continues executing other instruction instead of waiting.
- when the device is ready it sends an interrupt signal.
- Cpu stops its current task and jump to the (ISR)
- ISR transfer the required data between memory and the device.

- After completing the service, CPU return to interrupt program.

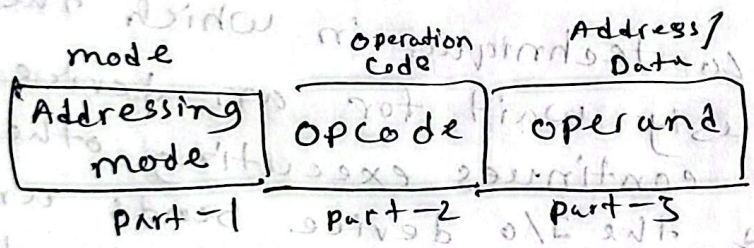
Adv:

- No busy waiting
- faster and more efficient than programmed I/O.
- Good for device with unpredictable time delay.

disadv:

- Requires interrupt handling hardware
- frequent interrupt can slow down the CPU
- ISR adds overhead.

a) Instruction ? element?



opcode: this field specifies the operation to be performed by the CPU.

Operand: these fields contain the data or references.

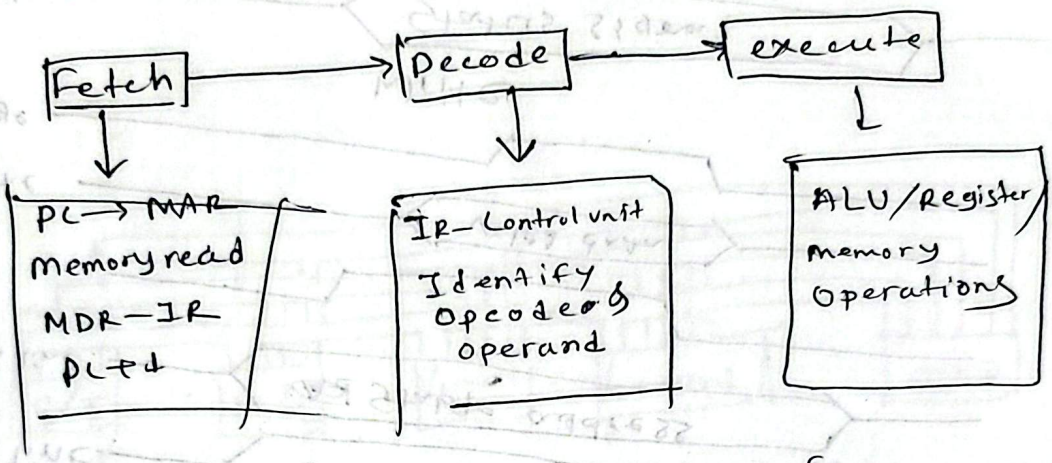
Addressing mode: this specifies how to interpret or locate the Operand, such as direct, indirect.

[An ~~ist~~ instruction tells the cpu what to do and which data to use]

Data processing: involves operations that modify data.

Data movement: involves operations that transfer data between memory, register, and I/O device.

Instruction Cycle:



Fetch cycle: CPU retriere instruction from memory using PC

Step:
- the address in PC is transfered to MAR
- Control unit sends a read signal to memory
- MDR

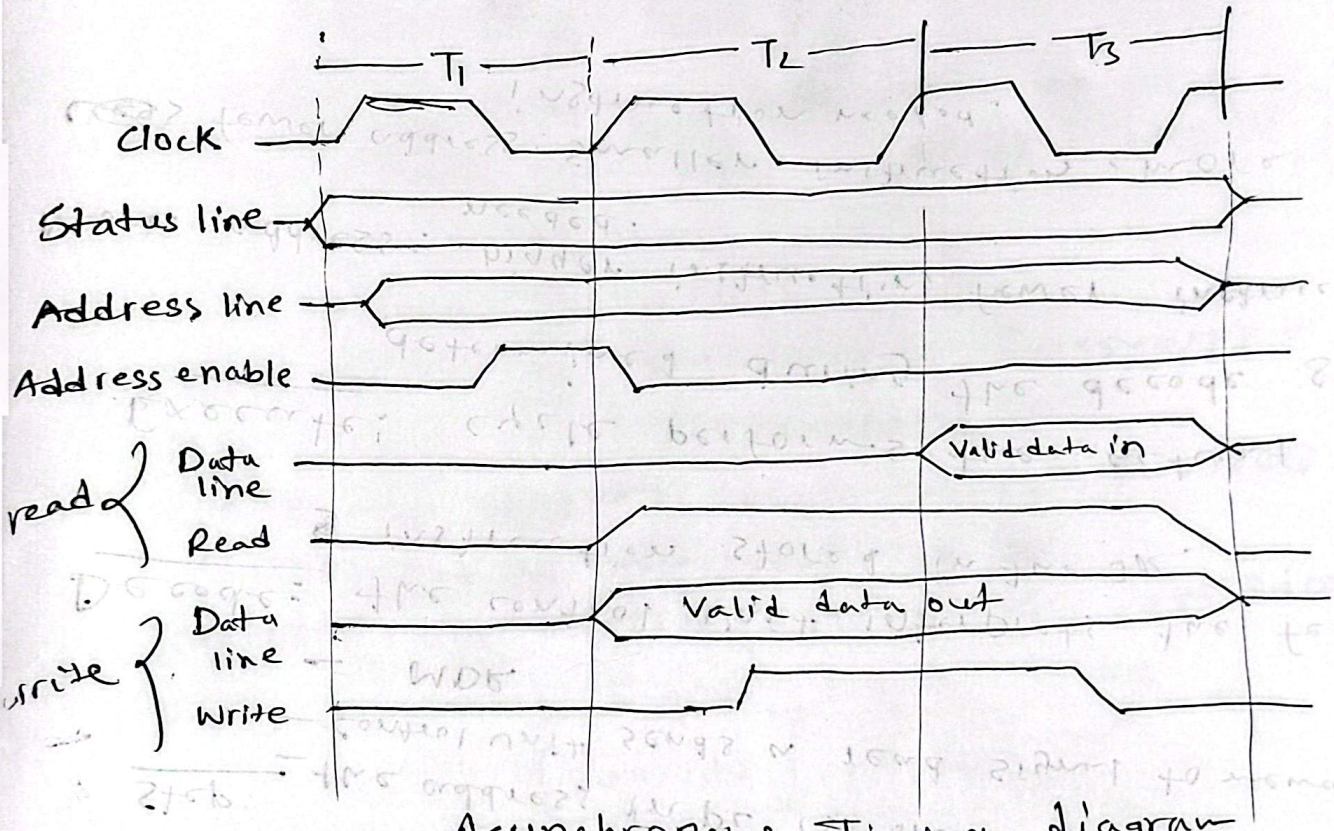
Decode: the control unit interprets the fetched instruction stored in the IR.

Execute: cycle performs the actual operation determined during the decode stage.

More address: bigger instruction, fewer instruction needed.

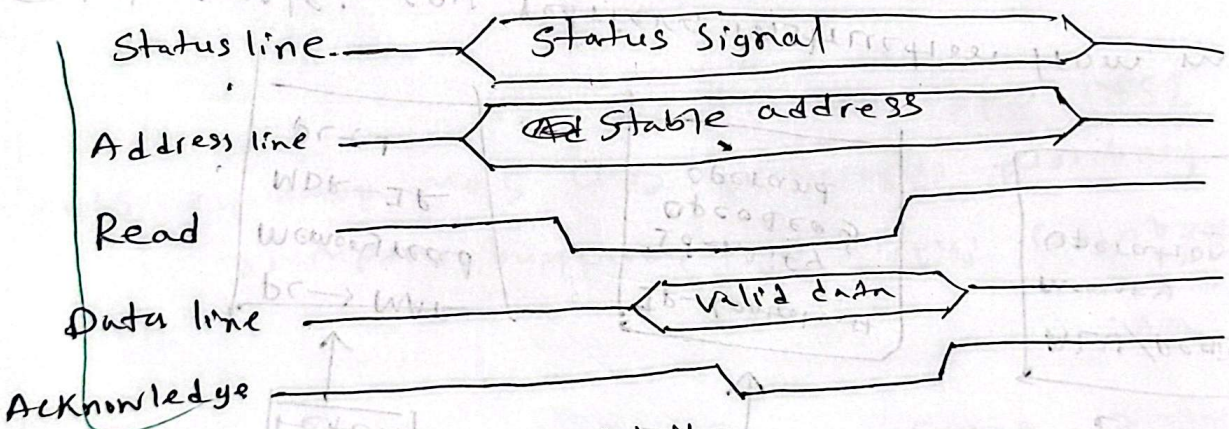
fewer address: Smaller instruction, more instruction needed.

Synchronous timing diagram

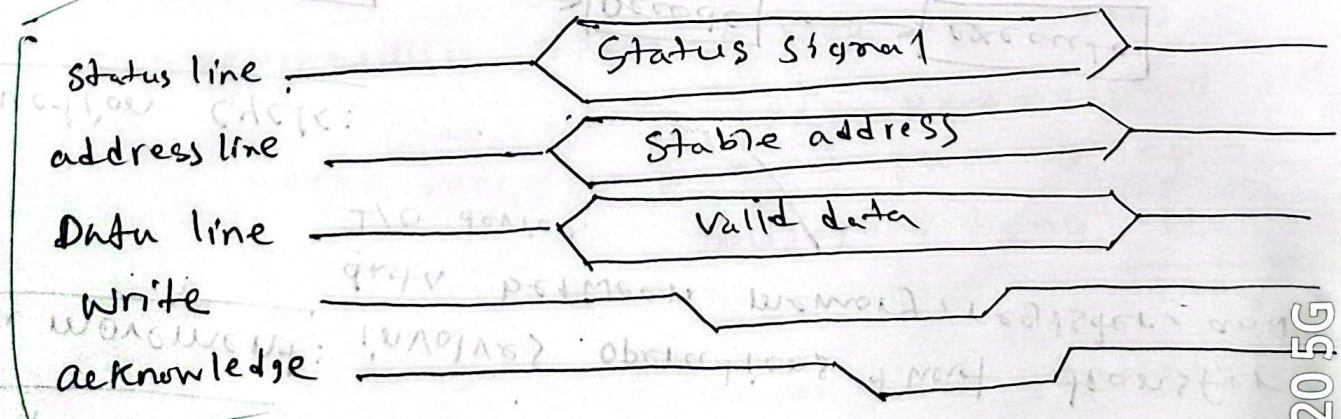


Asynchronous Timing diagram

read



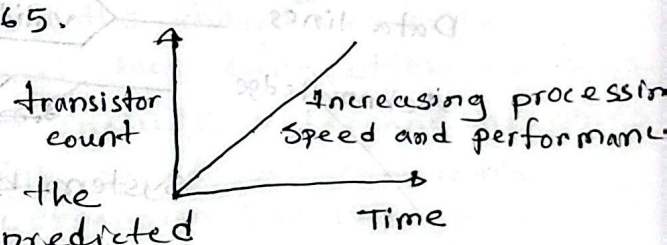
Write



Moore's Law

স্মৃতি (18-24) নামার মাইক্রোপ্রসেসর এবং প্রোগ্রামিং সংক্রান্ত বিষয় বই

Moore's Law principle states that since the number of transistor on a silicon chip roughly doubles every two years. the performance and capability of computer will continue to increase while the price of computer decrease. It is a prediction mode by American engineer Gordon moore in 1965.



Explained:

Moore's law was one of the best technological prediction of the last 50 years. Gordon E moore predicted that component on integrated circuit would increase two years. His postulation become known as moore law and was confirmed true in 1975.

The majority of this growth in chip density is due to four primary factors: diesize, line dimenson, technical brillirance, and technology innoration.

According to moor's observation, one of the major attraction Integrated electronics to low cost. This benifit grows as technology progress a single semiconductor substrate can produce more complex circuit function.

CPU: it fetches instructions and data from memory, decodes them, executes the required operations, and stores the results back in memory.

Instruction Register (IR): Responsible for temporarily holding the current instruction being executed. When the cpu fetches an instruction from memory, it is loaded into the IR.

program counter (PC): The primary function of the program counter is to keep track of the memory address of the next instruction to be fetched and executed by the cpu.

Discuss SDRAM:

SDRAM which stands for synchronous Dynamic Random access memory is a type of volatile computer memory that has been widely used in computing systems for several decades. It is commonly found in computers, servers and other electronic devices. SDRAM has played a crucial role in the advancement of computing technologies by providing first and efficient memory acces.

- Synchronous operation: SDRAM is synchronous meaning it 'synchronizes it's operation with the computer's bus speed. this synchronization allows for faster and efficient data transfer between memory and the processor.
- Data organization: SDRAM stores data in individual cells, each consisting of a capacitor and a transistor. these cells are organized into rows and columns, forming on array structure.
- Speed and bandwidth: SDRAM offers faster access times and higher bandwidth compared to tridational SDRAM
- Types of SDRAM: there are several generation of SDRAM have been developed and some notable SDR SDRAM, DDR SDRAM, DDR2, DDR3, DDR4 and latest DDR5.
- advantages and limitations:

SDRAM offers several advantages including faster data transfer rates, increased system performance and compatibility with a wide range of computer architecture.

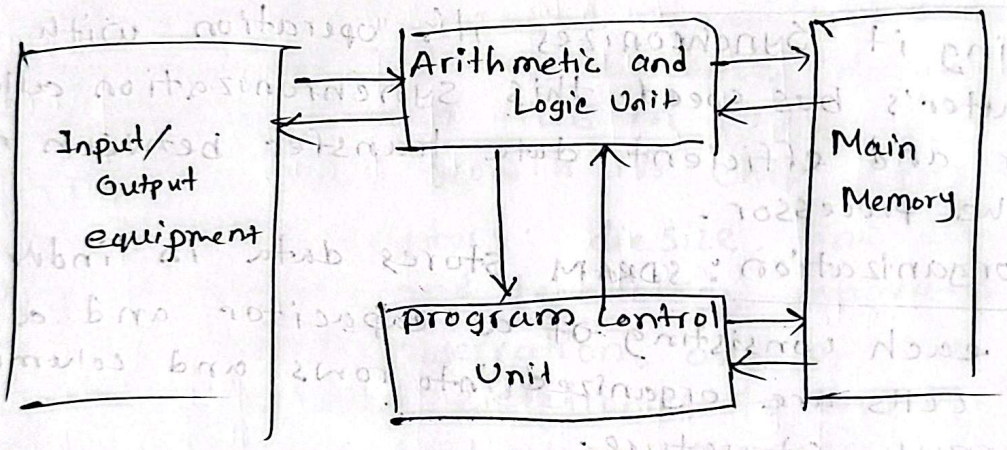
Ex: 8080 has 16 bit address bus, then how much memory space it will provide?

Can address a maximum of 64 Kilobytes of memory, Each memory address represents a Unique location in the memory space, and with 16 bits,

$2^{16} = 65536$ different address - 65536 bytes or 64 Kilobytes.

Slide-2:

Structure of von Neuman machine



Structure of IAS – detail