

mid - 24 - 22

Q1

This representation is wasteful, because its use more physical components (vacuum) than necessary to represent a given value.

→ Electronic Numerical Integrator & computer, ENIAC used vacuum tubes mainly as switches. If we try to represent integers directly in a binary-like system by using the ON/OFF states of vacuum tubes.

→ Each vacuum tube can be ON (1) or OFF (0)

→ If we have 20 vacuum tubes, the total no. of distinct ON/OFF patterns is

$$2^{20} = 1,048,576$$

At first glance, that seems efficient.

⊙ But why "wasteful"?

① ENIAC was decimal-oriented :-

→ ENIAC decimal digits were represented

Using 10 state ring counters (each digit
0-9)

→ this is a 'One-Hot' representation for a decimal

→ Using ON/OFF binary states directly
would require many tubes per digit

ex: representing a single decimal

digit in binary needs 4 vacuum tubes

(since $2^4 = 16 \geq 10$)

② Vacuum tube are expensive and unreliable

→ ENIAC has 18,000 vacuum tubes, which
were prone to failure

→ Using them for raw binary representation
wastes tubes compared to using

them efficiently for decimal counters

and arithmetic

③ Direct ON/OFF patterns don't match decimal arithmetic hardware :-

→ ENIAC circuit were optimized for decimal addition, subtraction, multiply etc.

→ A pure binary representation would require extra circuitry for decimal conversion, making the design more complex.

④ The wasteful scenario :

If multiple tubes could be ON/OFF, it's effectively treating each tube as a binary bit. For 20 tubes, we get 2^{20} possible states.

So, using the ON/OFF states of tubes directly as a binary number is "wasteful" because many tubes are needed per digit and the machine wasn't optimized for that. It's more efficient to use to represent decimal digit with

ring counter.

Range of Integer using 20 vacuum tubes.

If we ignore ENIAC's decimal Design & just think in pure binary :-

→ per vacuum tube → per digit

20 vacuum tube → 20 digit

∴ Maximum Integer representable.

$$(2^n - 1) = 2^{20} - 1 = 1048576 - 1$$

$$= 1048575$$

→ minimum Integer : $n=0$

$$\Rightarrow (2^n - 1) = 2^0 - 1 = 0$$

∴ the range 0 to 1048575

Q2

Given data,

mid : 21-22

L2: access time

We know, To compute effectiveness access time

(EAT) for two-level cache-system,

$$EAT = h_1 \times t_1 + (1 - h_1) \times \left[h_2 \times (t_1 + t_2) + (1 - h_2) \times (t_1 + t_2 + t_m) \right]$$

miss in both caches

if data found in L2 if data is found in L2

where, h_1 = hit ratio in L1 cache

t_1 = access time of L1

h_2 = hit ratio of L2 cache.

t_2 = access time of L2

t_m = main memory access time

Given, $t_1 = 5 \text{ nsec}$ (L1 access time)

$t_2 = 80 \text{ nsec}$ (L2 access time)

$t_m = 250 \text{ nsec}$ (main memory access time)

$h_1 = 0.5\% = 0.005$ (L1 hit ratio)

$h_2 = 75\% = 0.75$ (L2 hit ratio)

$$\therefore EAT = 5 \times 0.05 + (1 - 0.05) \left[\cancel{0.75 \times 85} + \cancel{100 \times 0.75} \right]$$

$$\left[0.75 \times 85 + (1 - 0.75) \times (5 + 80 + 250) \right]$$

$$= 4.75 + 0.05 \left(\cancel{100} + 85 \times 0.75 \right)$$

$$= 4.75 + 0.05 (15.099 + 85.75)$$

$$= 4.75 + 0.05 (147.72) = 147.72$$

Now,

$$EAT = \cancel{5} \times 0.05 + (1 - 0.05) \left[\cancel{100 \times 0.75} \right]$$

$$0.75 \times (5 + 80) + (1 - 0.75) (5 + 80 + 250)$$

$$= 4.75 + 0.05 (147.5)$$

$$= 4.75 + 7.375$$

$$= 12.125$$

$$= 12.125$$

$$\underline{\underline{\text{Answer}}}$$

$$= 12.125$$

$$= 12.125$$

$$= 12.125$$

Note that, when there is a miss in ~~hit~~ L1, we access L2. If there is a hit L2, we have already spent t_1 and then t_2 . Similarly if there is a miss in L2, we then access main memory spending $(t_1 + t_2 + t_m)$

④ Address Bus:

The address bus ~~tell us~~ width tell us how many unique memory locations the CPU can directly access. Each unique address corresponds to a memory location (typically 1 Bytes)

For Intel 8080 microprocessor: $\rightarrow$

$\rightarrow$ Address Bus width = 16 bits

$\rightarrow$ Number of unique address = 2^{16}

$\therefore 2^{16} = 65,536$ unique addresses

the 8086 can access 65,536 bytes of memory space.

Since, $1 \text{ KB} = 1024 \text{ bytes}$.

Now
$$\frac{65,536}{1024 \times 8} = \frac{64 \text{ KB}}{8 \text{ KB}}$$

So, the 8086 provide 64 KB of memory space.

3) mid-21-22

For 32 bit $\rightarrow 32$ location
 max memory = 2

$$\therefore \frac{2^{32}}{1024 \times 1024 \times 1024} = 9 \text{ GB}$$

↓ ↓ ↓
 KB MB GB

interrupts = A, B and C

priority $A > B > C$

ISR execution time

A $\rightarrow$ 10 unit, B $\rightarrow$ 30 unit, C $\rightarrow$ 20 unit

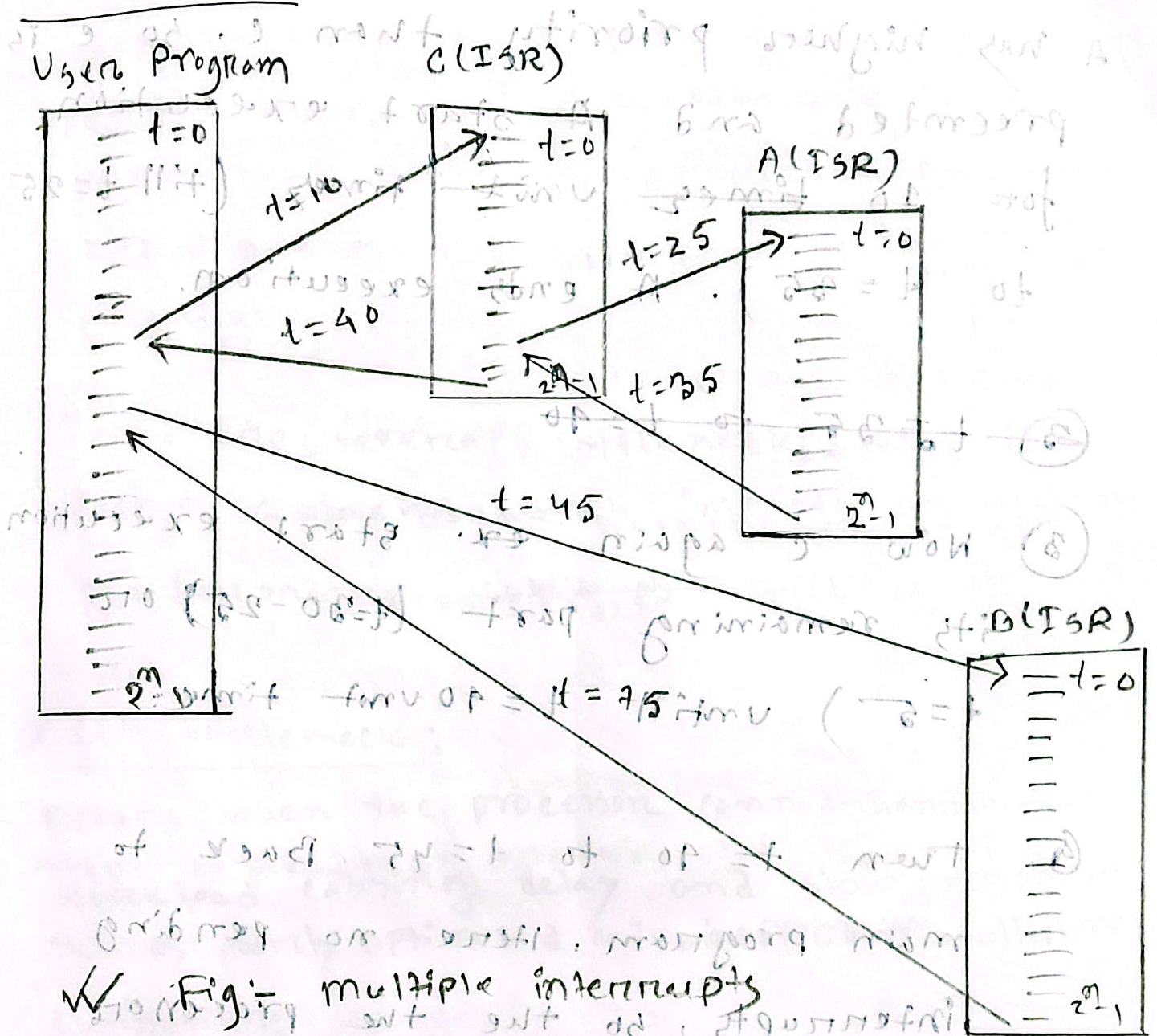
Interrupts sequence (time):

C occurs at $t = 10$

A " " $t = 25$

B " " $t = 45$

Graphically:



✓ Fig:- multiple interrupts

Explain:

- ① $t=0$ to $t=10$ main user program, where CPU runs the normal user main code, so here is no interrupts, and processor is idle.
- ② $t=10$ to $t=25$ when $t=10$ it starts execution. C needs 20 times unit for execution (scheduled time $t=10$ to $t=30$).

But at $t=25$ A interrupts because A has higher priority than C. So C is preempted and A starts execution. For 10 times unit time (till $t=25$ to $t=35$). A ends execution.

~~③ $t=35$ to $t=40$~~

③ Now C again starts execution its remaining part ($t=30-25$) or $t=5$) until $t=40$ unit time.

④ Then $t=40$ to $t=45$. Back to main program. Here no pending interrupts, so the processor remain idle.

⑤ When $t=45$ then B starts execution with no higher-priority tasks left.

that's why CPU runs B's full until $t=75$ unit.

⑥ Now, $t=75$. So, Back to main program again B complete, CPU return to normal execution.

⑤ A bottleneck refers to a limitation or constraints that slow down the overall performance of a system. It is the weakest point that restricts efficiency of a system and prevents the system to achieve its full potential.

There are several bottlenecks we face to design a system which impacts on system performance ~~such as~~ such as.

① CPU bottleneck:

Occurs when the processor cannot handle the workload, causing delay and slow response. Such as poorly optimized code, inefficient multithread.

Mitigation:

- parallel/multi core processing to distribute tasks
- optimize algorithm and code
- upgrade CPU or add hardware accelerators

② Memory Bottleneck:

Inefficient or slow RAM leads to frequent disk swapping and high latency.

Mitigation: occurs when the system's

memory resources are inefficient to meet the demand of data processing and storage leading to increase access time and decrease overall system performance.

Mitigation:

→ Optimize data storage/retrieval to reduce memory usage.

→ Increase RAM size or use faster memory.

Employ caching for frequently accessed data.

③ Network Bottleneck:

Occurs when the network bandwidth is inefficient to handle the data transmission requirement, resulting in communication delays, packet losses and degraded network performance.

Mitigation:

→ Traffic shaping and quality of service (QoS)

→ upgrade routers, switch, cable to higher speed

→ use data compression and efficient protocols.

④ Storage bottleneck

Occurs when storage infrastructure is unable to handle the data storage and retrieval demands efficiently, leading to increased latency, slow data access time, and data loss.

This device cannot meet read/write demand, causing I/O wait time.

Mitigation:

storage tiering: keep hot data on faster media.

→ Duplication & compression ~~an~~ to
reduce I/O

→ upgrade to SSD: frequent and is
common cause

Inefficient resource Allocation (cpu, memory,
bandwidth, storage)

Inefficient code or algorithm creating
unnecessary load

Hardware limitation from outdated

under-space component

consequences:

→ Reduced throughput: system handle fewer
operation per second

increased latency

→ system instability

storage utilization

performance of processor:

refers to how fast it can execute task or instruction. It depends on several factors that can impact the performance of computer system, including processor speed, memory, storage, I/O devices and software optimization.

Execution time:

→ How long to CPU take complete a program or task.

$$\text{CPU time} = \text{Instruction count} \times \text{CPI} \times \text{clock cycle time.}$$

where,
→ Instruction Count (IC) = total no of

instruction executed.

→ CPI = average cycles per instruction

$$\text{Clock cycle time} = \frac{1}{\text{clock frequency}}$$

② Clock Frequency:

→ measured in Hz (usually GHz)

→ Higher frequency means more cycle per second

③ CPI (cycle per instruction)

→ average no. of clock cycles each instruction takes.

→ lower CPI → better performance.

→ Depends on processor architecture and instruction mix.

④ Ipe (Instruction per cycle)

$$\rightarrow Ipe = \frac{1}{CPI}$$

→ measure how many instruction are executed per clock cycle.

Higher Ipe → better performance

⑤ Throughput:

→ Amount of work done per unit time
(instructions/sec, FLOPS)

→ useful for evaluating overall processor efficiency on a workload

⑥ Benchmarking:

→ Running standardized or application specific program to measure performance

ex: SPECint, Geomark, PassMark

⑦ Memory and pipeline efficiency:

→ cache, branch prediction, and pipeline hazards affect performance.

So, For faster execution, lower CPI, higher clock frequency, and efficient memory or pipeline design improve processor performance.

5

20-21 (2nd mid)

main memory size = 16 mbytes
 $= (2^4 \times 2^{20}) \text{ bytes.}$

$= 2^{24} \text{ bytes.}$

$\therefore \text{no. of physical address bits} = \log_2 2^{24}$
 $= 24 \text{ bits}$

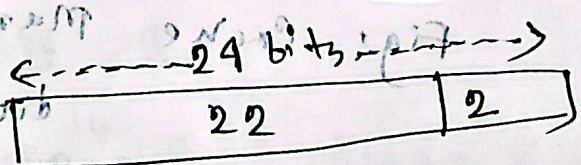
Block size = 4 bytes

$(2^{24}) / 4 = 2^{22} \text{ blocks}$

Block offset = $\log_2 4 = 2$

No. of block in main memory = $\frac{2^{24}}{2^2} = 2^{22}$

each block no. number bits = $\log_2 2^{22} = 22$



cache size = 64 Kbytes

$$= 2^6 \times 2^{10} = 2^{16} \text{ bytes}$$

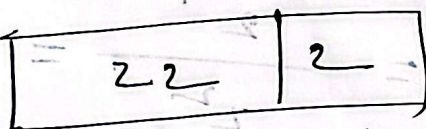
$$\text{No. of lines in cache} = \frac{2^{16}}{(2)^2}$$

$$\therefore \text{line no. bits} = \log_2 2^{14} = 14$$

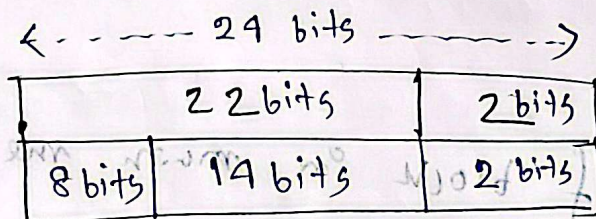
$$\therefore \text{no. of tag bits} = 24 - (14 + 2)$$

$$= 8$$

Physical address (24)

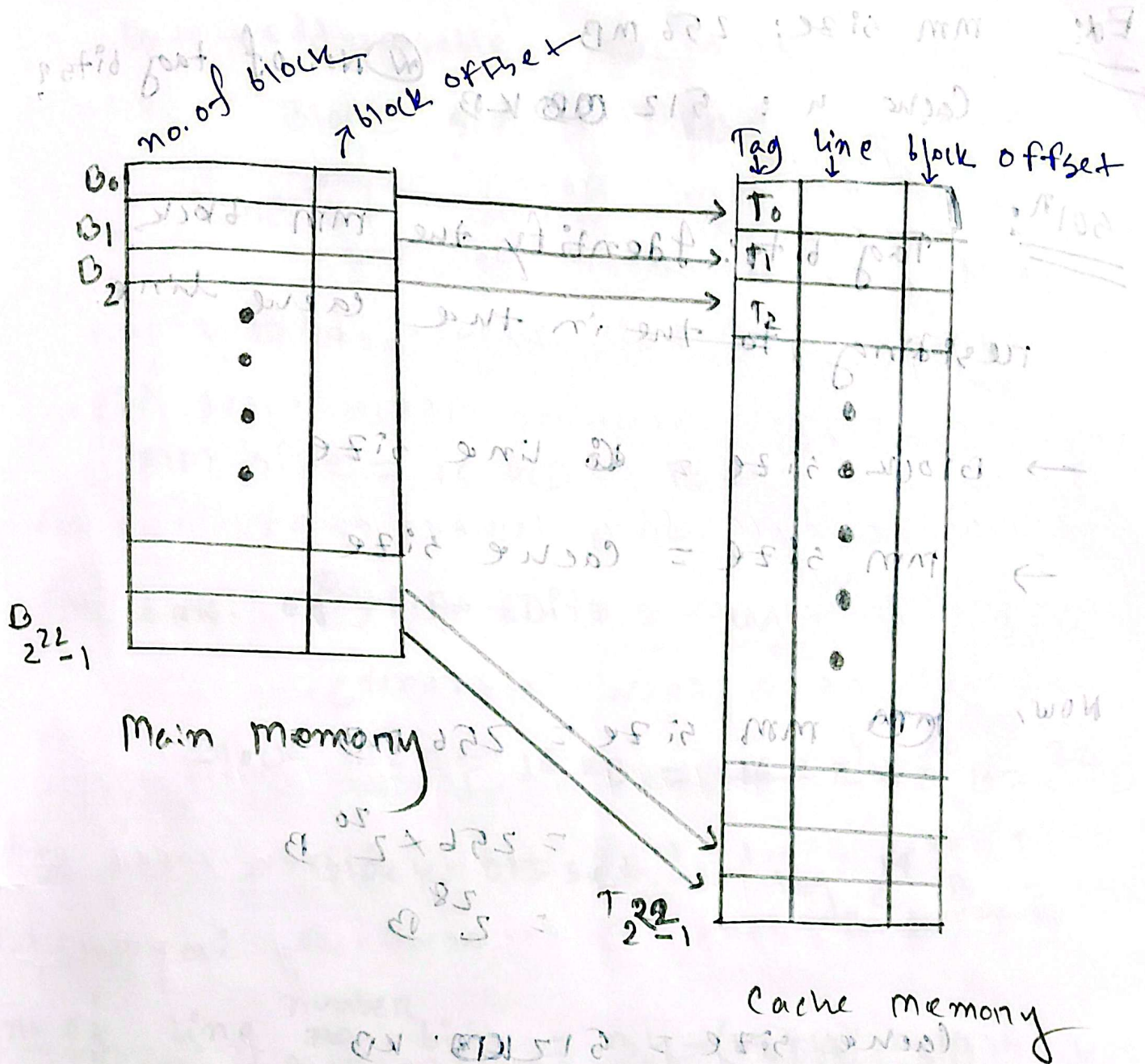


No. of block block offset



↓ ↓ ↓
tag line block offset

Fig: Cache Memory Design with direct mapping.



Right (Direct) mapping. cache/main memory structure.

extra

tag directory:

No. of cache in line = 2^{14}

No. of tag bits = 8

$\therefore$ tag directory size = $2^{14} \times 8 = 131072$ bits

Ex:

mem size: 256 MB

Cache n : 512 ~~MB~~ KB

Q No. of tag bits, E.

Soln:

Tag bits: identify the mem block
residing ~~to the~~ in the cache line.

→ Block size = line size

→ mem size = cache size

Now,

~~mem~~ mem size = 256 MB

$$= 256 \times 2^{20} \text{ B}$$
$$= 2^{28} \text{ B}$$

Cache size = 512 ~~MB~~ KB

$$= 2^8 (2^9 \times 2^{10}) \text{ B}$$
$$= 2^{19} \text{ B}$$

No. of tag bit: $-\log_2 \frac{2^{28}}{2^{19}}$

$$= \log_2 2^9$$
$$= 9$$

$8 = \log_2 2^9$

$8 + 1 = 9$

Ans

Ex: Byte-addressable mem size : 16 GB

Block size = 16 KB

No. of Tag bits = 10

→ Find cache size?

✓ mem size = 16 GB = $2^4 \times 2^{30}$ B = 2^{34} B

No. of P.A. bits = $\log_2 2^{34}$ B = 34 B

Block size = 16 KB = $16 \times 2^4 \times 2^{10}$ B = 2^{14} B

Block offset = $\log_2 2^{14}$ B = 14

no. of line no. bits = $34 - (14 + 10) = 10$

∴ No. of cache line = 2^{10}

line size = 2^{14} B

∴ cache size = $2^{10} \times 2^{14}$ B = 2^{24} B = $2^4 \times 2^{20}$ B

= 16 MB

Ans:

6

Building a computer that use only SRAM (static Random Access memory) as its main memory is technically possible.

SRAM has low latency, incredibly fast and does not require refresh cycle (it's static) that's why SRAM is redundant. The entire memory hierarchy would be flat and fast and not need cache hierarchy.

→ However, the cost both silicon area and power make it economically impractical.

Now, Tile = (cost + $\frac{1}{\text{tile}}$) must be analyzed in three dimension

(i) Financial

(ii) physical

(iii) power.

① Financial cost :- (core of the cost issue)

→ SRAM cell: consist of 6^{or} transistors (6T Design) to store a single bit. This complex structure

make speed and stability but use a large amount of silicon real estate. ~~and less density~~

→ DRAM cell: consist of only one transistor and one capacitor ^{per bit} (1T1C Design), which is very simpler and smaller and massive bit densities.

② Physical cost (size and density)

→ The larger 6T cell structure of SRAM means it has a much lower density than DRAM.

→ Die size: A 64MB SRAM die is physically larger than an 8GB DRAM - yet hold $\frac{1}{128}$ th the data

→ scaling: DRAM density keep improving rapidly, but SRAM density grows slowly, so the size gap

only increases over time.

(3) Power cost (Energy & ^{Heat} cost) :-

SRAM speeds comes at a high consumption cost.

(i) Static power leakage :- A Billion of SRAM cell leak current even when idle,

creating a large continuous power draw.

(ii) Dynamic power :- Frequent read/write across so many cells add active power consumption.

(iii) Thermal Impact :- The resulting heat demands heavy duty cooling $\Rightarrow$ large heat sink & liquid system non fire refrigeration

- rising total cost and complexity far beyond that of DRAM system.

3.1 (4) mid

computers Architecture

- ① refers to the functional design of a computer
 - ② Deals with what the system does
 - ③ Focus on instruction set, addressing modes, and I/O mechanisms
 - ④ abstract or logical level
 - ⑤ Related to programmer and compiler
 - ⑥ Instruction execution capabilities
 - ⑦ It's high level design
- Ex: RISC vs VISC architecture

computers Organization

- ① refers to the implementation details of the computer
 - ② Deals with ^{implementation} how the system
 - ③ Focus on hardware details like control signal, memory technology and bus instruction.
 - ④ physical or hardware level.
 - ⑤ Hardware engineer
 - ⑥ Circuit layout & hardware control
 - ⑦ It's low level - Design
- Ex: cache memory design, ALU structure

Structure

- ① refers to physical arrangement of component.
- ② Focus on how component are connected
- ③ concerned with hardware elements like CPU, memory, buses
- ④ level: Hardware
- ⑤ A change involves redesigning hardware
- ⑥ Independent of function & concept

Ex: CPU with register

ALU, cache

function

- ① operation perform by each component
- ② Focus on what the components to do
- ③ Data processing, instruction execution
- ④ level: Operational logic
- ⑤ can modify operation without changing structure
- ⑥ dependent

Ex: Fetch, decode, execute, etc.

2.10

$$f = 80 \text{ MHz}, T = \frac{1}{f} = \frac{1}{80} = 12.5 \text{ nsec}$$

from table, total Instruction count,

$$I_{\text{total}} = 100,000$$

⑥ Find CPI:

$$CPI_{\text{eff}} = \frac{\sum (\text{instruction count} \times CPI)}{I_{\text{total}}}$$

$$= \frac{45000 \times 1 + 32000 \times 2 + 15000 \times 2 + 8000 \times 2}{100,000}$$

$$\therefore \text{effective CPI} = 1.55$$

Ans:

⑦ MIPS rate:
 millions of instruction per second.

$$MIPS = \frac{\text{Clock Frequency (MHz)}}{CPI_{\text{eff}}}$$

$$= \frac{80}{1.55} = 51.61$$

Ans:

③

Execution time = ?

$$T_{\text{Exe}} = \frac{\text{Total cycle}}{\text{clock frequency}}$$

$$= \frac{I_{\text{total}} \times CPI_{\text{eff}}}{\text{clock frequency}}$$

$$= \frac{(193 \times 100000) \times 1.55}{80 \times 10^6}$$

$$= 2.410375 \text{ msec.}$$

Ans

④

$$= \frac{1000000}{1000000}$$

Cache Design Bottleneck:

- ① Cache size Access time: Larger caches store more data but increase access time.
- ② Cache size limitation: Small size cannot hold all frequent used data, leading to high miss rate. More misses mean more accesses to slower main memory, slowing the system.
- ③ Cache access time: Cache access must be faster than memory access. If the cache is too large then lookup time increases reducing performance benefits.
- ④ Mapping function conflict: In direct-mapped cache, multiple memory blocks can map to the same cache line, causing conflicts/misses.
- ⑤ Cache Hit & miss rate: A high miss rate forces the processor to access slower main memory, increasing latency.

⑥ memory bandwidth: cache miss increase memory traffic, causing delays in data retrieval.

⑦ Cache - coherency: In multicore system maintaining consistency between caches leads to additional overhead.

⑧ power consumption: larger and faster caches consume more power.

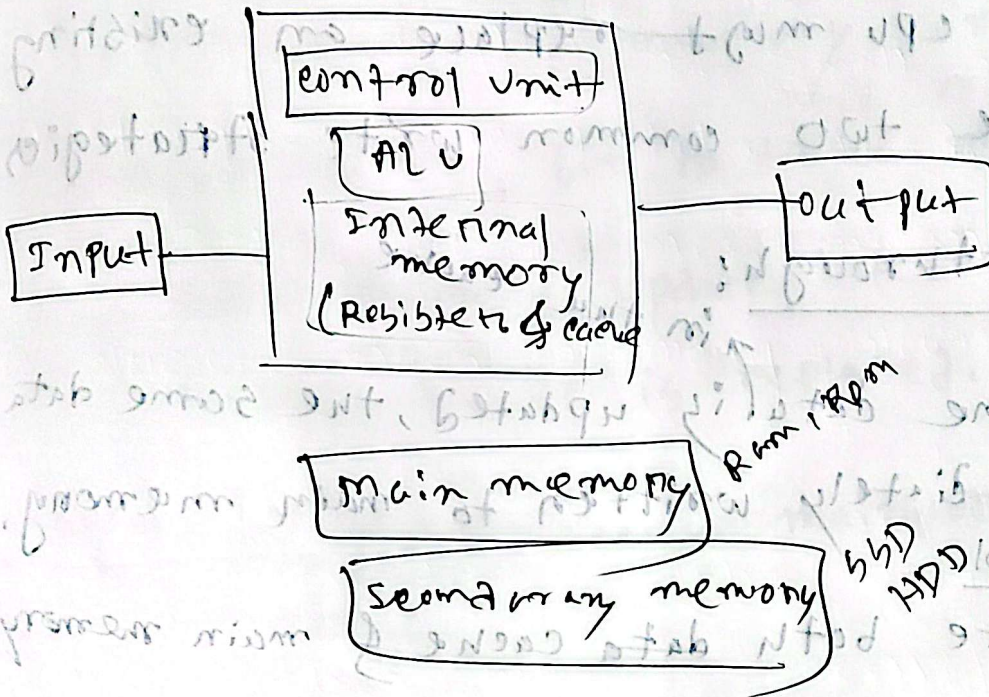
⑨ Associativity Trade-off:

It reduces conflicts but increase lookup time and hardware complexity.

⑩ Write strategy: write-through update main memory on every time which is slow.

⑪ Bus contention: The cache share the memory bus with the CPU. Frequent cache misses

misses or writes can create bottlenecks on the bus.



Cache write through:

Cache Block replacement: when a cache block is full and a new block from memory needs to be added, the CPU must replace an existing block. The two common write strategies

① Write-through:

Every time data ^{in the cache} is updated, the same data is immediately written to main memory.

Characteristics:

→ It updates both data cache & main memory.

Advantages

→ simple to implement and safe.

→ Ensure memory always has the latest copy of data.

→ slower for writes because every update goes to memory.

Replace:

→ old cache block can be replaced safely because memory already has the updated data.

② write-back: Data is only written to main memory when the cache block is replaced.

characteristics:

- Faster because memory writes are reduced
- need dirty bit to track modified blocks.
- main memory update later when the cache blocks is replaced.

→ I/O must access main memory through cache.

Replacements

→ If the block is dirty, → write it back to main memory

→ If block is clean → just discard it

03

2nd md

Range of 4 bit 2's complement

→ with 4 bit, the left most bit is the sign bit.

Range: -2^3 to $2^3 - 1$

$\Rightarrow -8$ to $+7$

4 bit binary

Decimal value

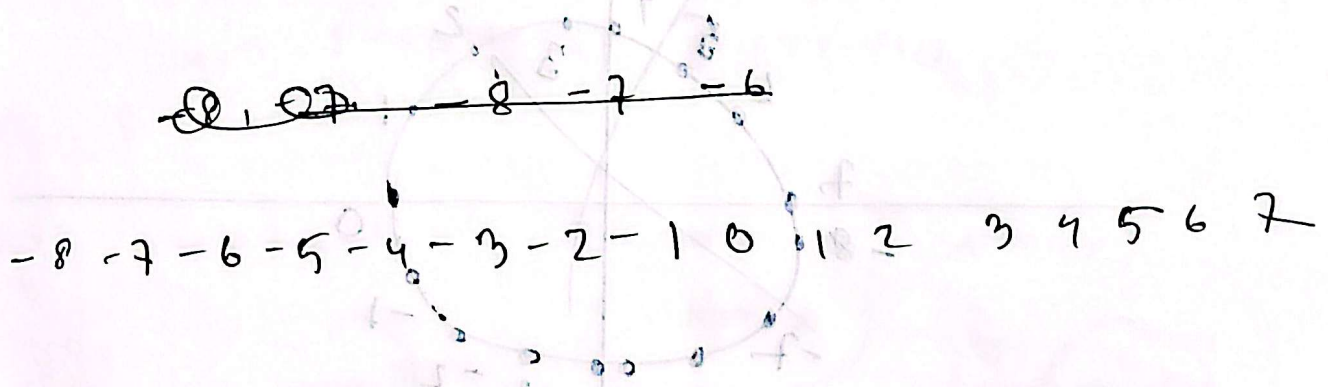
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Geometric duplications

→ imagine a straight line -8 to $+7$

→ negative numbers (1000 to 1111) appear to the left of 0

→ positive numbers (0001 to 0111) to the right of 0



the binary code wraps around at the ends because of modular arithmetic.

9 bit circle: (mod 16 Geometry)

→ Draw 16 point circle (like clock) labeled with binary numbers 0000 to 1111

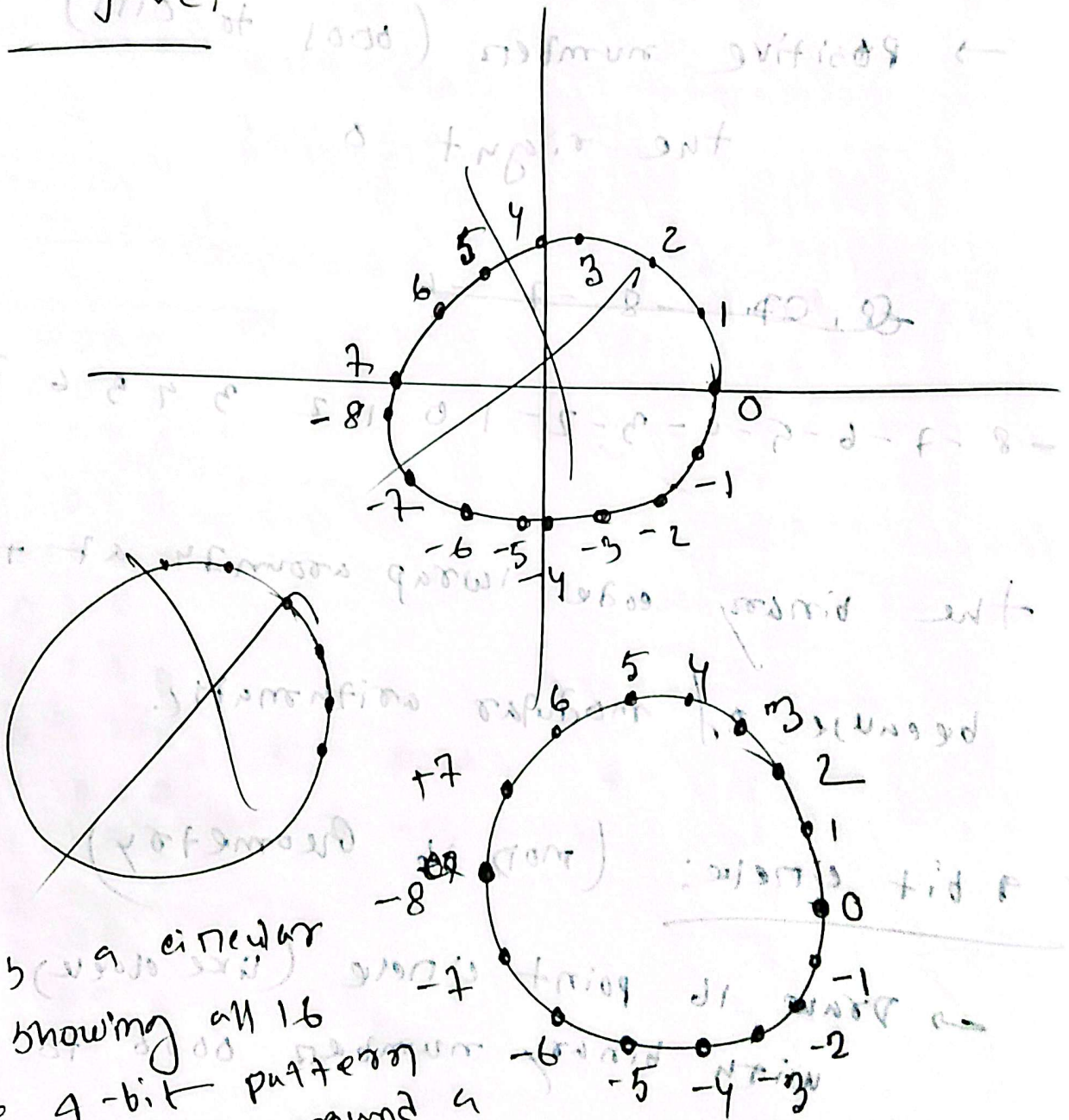
→ moving clockwise increase the unsigned value $0 \rightarrow 15$

For 2's complement:

Value 0-7 as 0 to +7

8-15 as -8 to -1

Figure:



here is a circular graph showing all 16 possible 4-bit patterns placed evenly around a circle.

function

Data processing

storage

movement

control

Data storage facility

Data movement

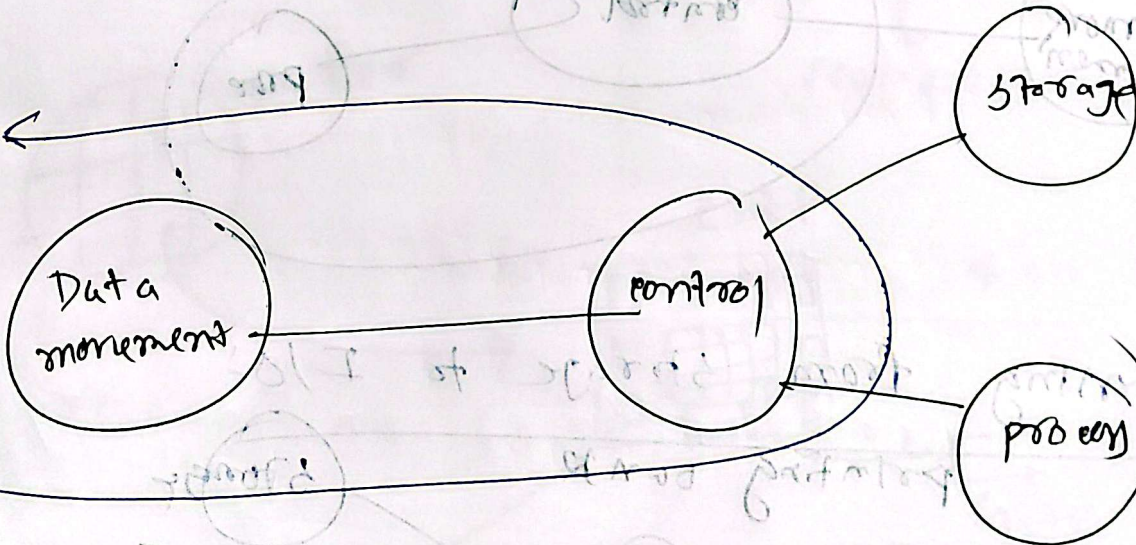
control mechanism

Data processing

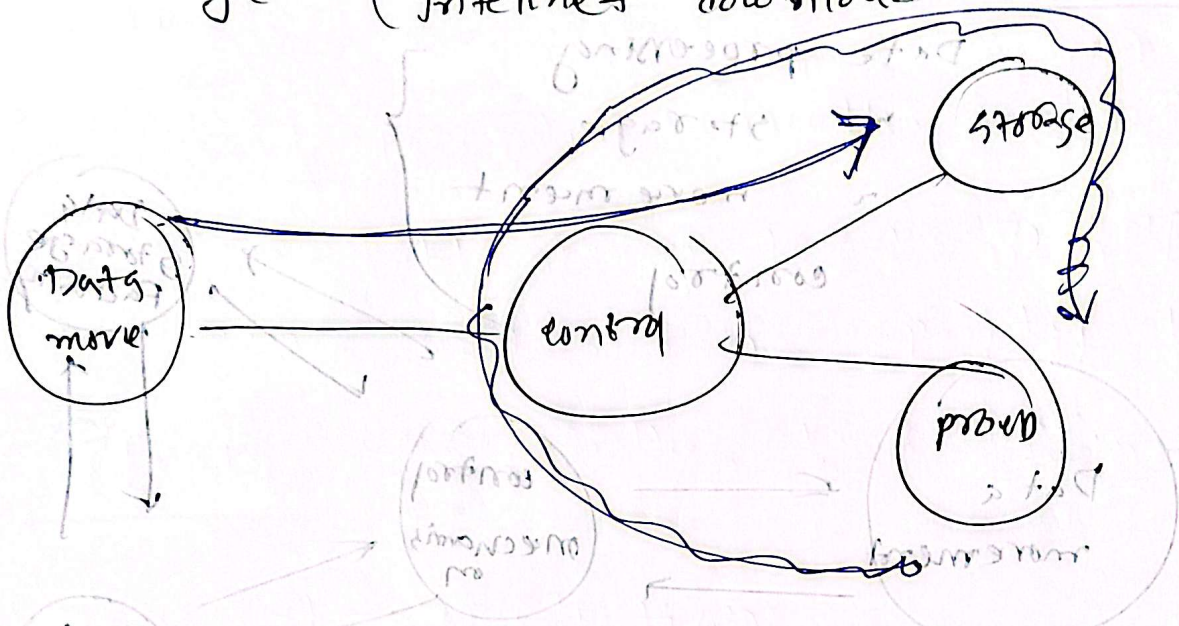
functional

overview

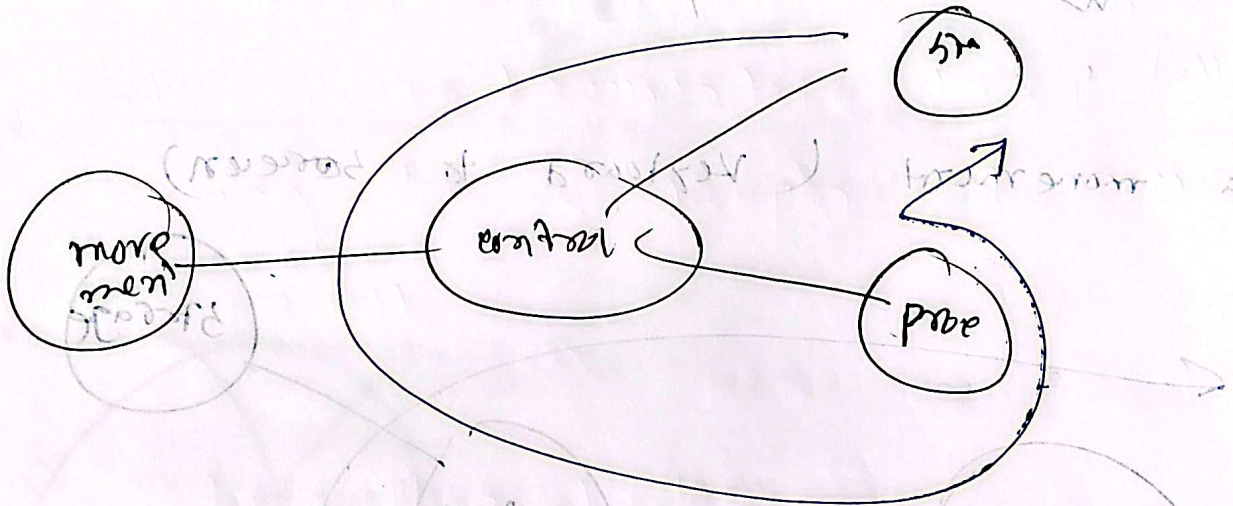
Data movement (Keyboards to screen)



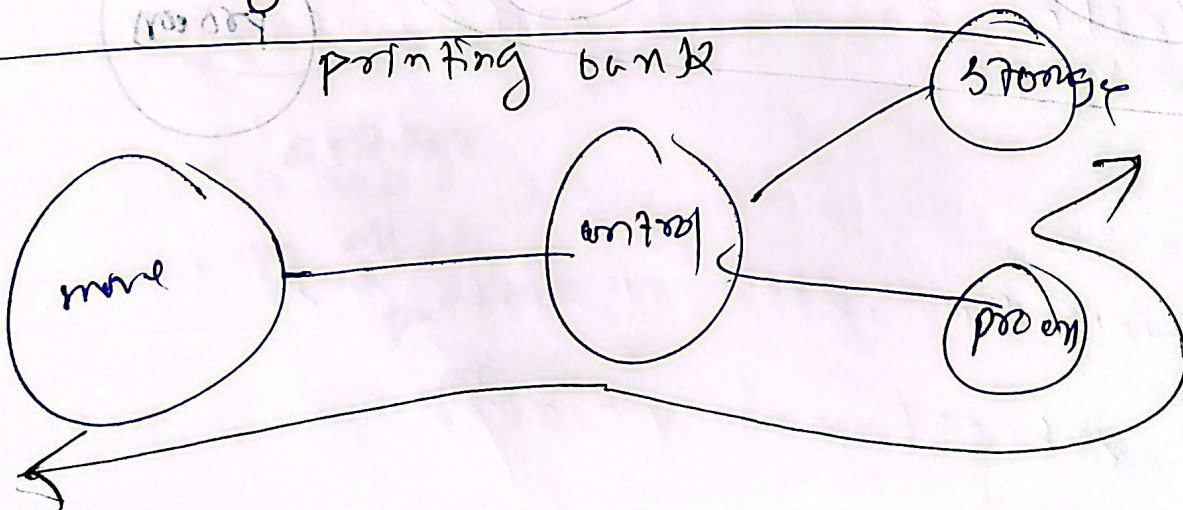
Storage (Internet download to Disk)



processing / storing updating bank statement



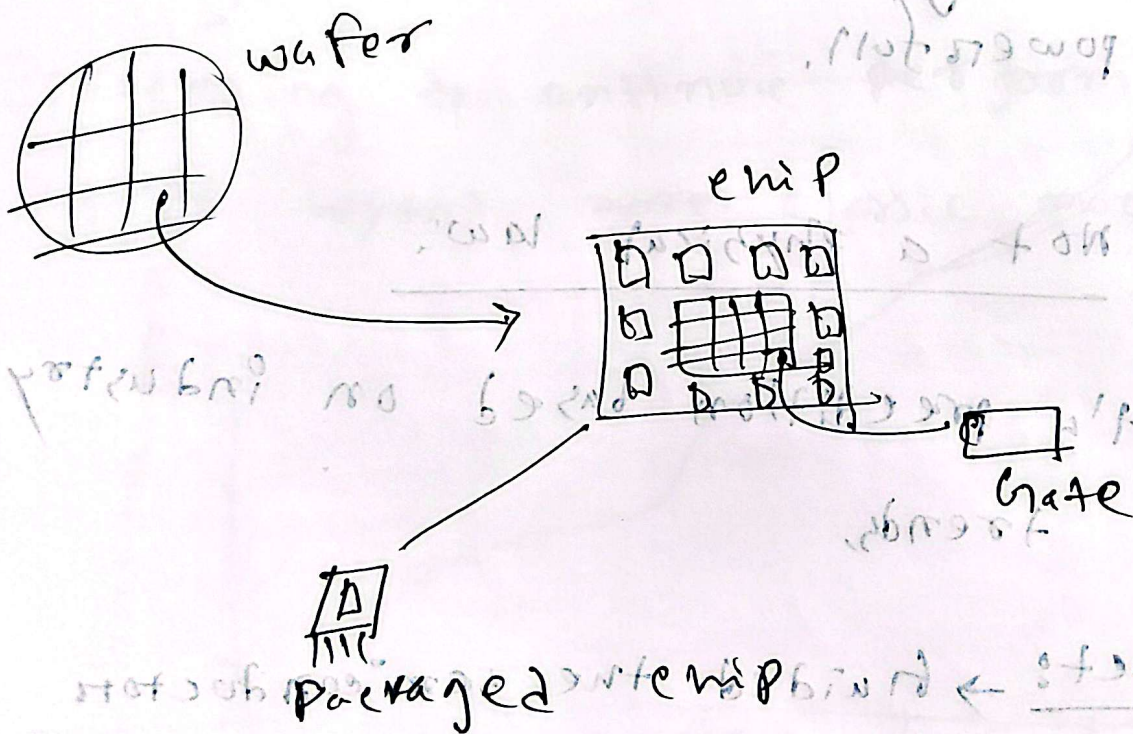
processing from storage to I/O
printing bank



Moores' Law:

Moores' Law is the observation that the number of transistors on a microchip (integrated circuit) double approximately every two years, resulting in a corresponding increase in computing power and a decrease in cost per transistor.

It was proposed by in 1965 by Gordon Moore, co-founder of Intel.



relationship - wafer, chip, gate

Key Points

① Exponential Growth: If the no. of transistors double every two years, computing performance and storage capacity also increase exponentially over time.

② Cost Reduction:

As chips become more dense, the cost per transistor keep falling making devices cheaper and more powerful.

③ Not a physical law

- It's prediction based on industry trends.

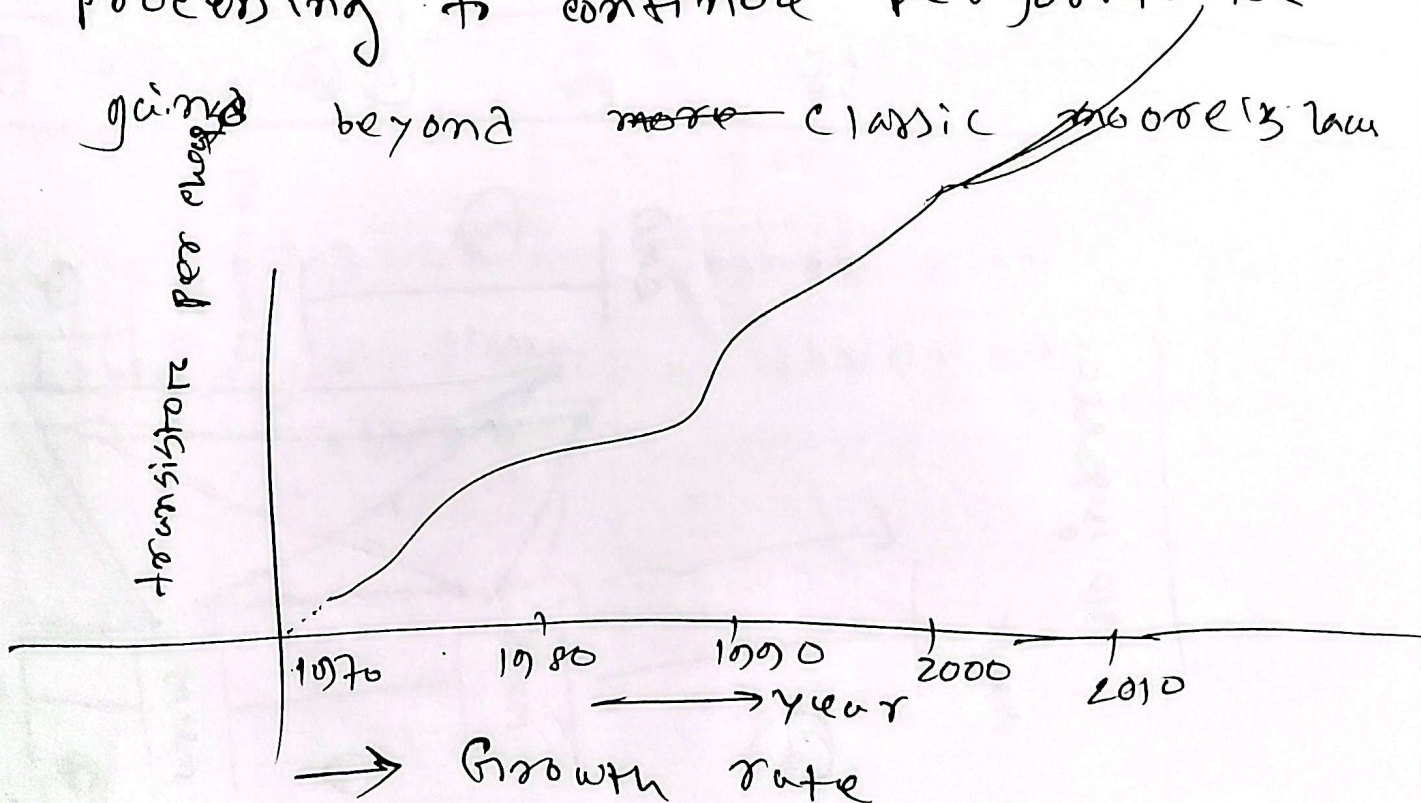
Impact: → Guided the semiconductor industry planning for decades

→ Enables rapid advance for personal computers, smartphone etc

present situation:

→ In recent years, physical limit (heat, quantum effect, manufacturing complexity) have showed the traditional pace.

→ The industry now explore 3D chip design, new materials, and parallel processing to continue performance gain beyond ~~more~~ classic Moore's law



Minimum chip: The inter connection on the integrated circuit are much more reliable than solder connection with more circuitry on each chip, there are fewer interchip connection.

Reduce power: There is a reduction in power & cooling requirements.

The integrated circuit was replaced by the monolithic integrated circuit which was replaced by the microprocessor.



Dynamic DRAM

- store each bit in a capacitor (need periodic refresh)
 - data lost if power off
- volatile memory
- slower speed (capacitors wear out over time)
- higher density
- low cost per bit
- low power consumption
- simple structure (one transistor + one capacitor per bit)
- called main memory in pc, laptops.

Static SRAM

- store each bit in a flip-flops (doesn't need refresh)
- volatile memory
- faster speed
- lower density
- more expensive per bit
- high power consumption
- more complex design (6 transistors per bit)
- cache, register, small high speed memory.

RAID 4

→ Redundant Array of Independent Disks - level 4

→ data is striped at the block level across all disks except the parity disk.

→ Dedicated parity disk stores all parity information.

→ can tolerate one disk failure

→ read performance similar to RAID 5

→ parity disk is less of a bottleneck

→ write performance bottleneck

→ system with more read than write operation

RAID 5

→ Redundant Array of Independent Disks - level 5

→ across all disk with parity disk.

→ parity is distributed across all disk

→ same

→ read performance similar to RAID 4

→ avoid single parity disk bottleneck

→ better

→ system with balanced read/write operation.

⊗

computer architecture:

Computer Architecture is the functional design & structure of a computer system. It focuses on how a computer's hardware components, like the CPU & and memory, work together to execute instructions efficiently.

computer Organization: computer Organization

focuses on the physical implementation of a computer system based on its architecture. It deals with how different hardware components, like CPU, memory and I/O devices, are connected & work together to execute tasks.

→ while computer Architect explains what a computer does, computer organization describes how ^{it} does it.

① Architecture describes what the computer does

② CA deals with the functional behaviors of computer system

③ It deals with high level design issues

④ Architecture indicates its hardware.

⑤ for designing a computer its architecture is fixed first

⑥ CA is ^{also} called Instruction set Architecture (ISA)

⑦ Determines overall system structure

① Organization describes how it does it

② CO deals with structural relationship.

③ It deals with low-level design issues.

④ Organization indicates its performance.

⑤ for designing a computer, an organization is decided after its architecture.

⑥ CO is frequently called micro architecture.

⑦ Implements specific hardware

LA 00

⑧ It acts as an interface between hardware & software.

⑨ The software developers is aware of it.

⑩ It makes the computer hardware visible

CO 10

⑧ It deals with the components of a connection in a system.

⑨ It escapes the software programmer detection

⑩ It offers details on how well the computer performing.

5/6-1
5/6

Major component in CPU for stack

① stack pointer: points to top of stack

② Register/memory: stores the stack data

③ Control Unit: executes push & pop instructions.

When an operation is performed the stack pointer is updated.

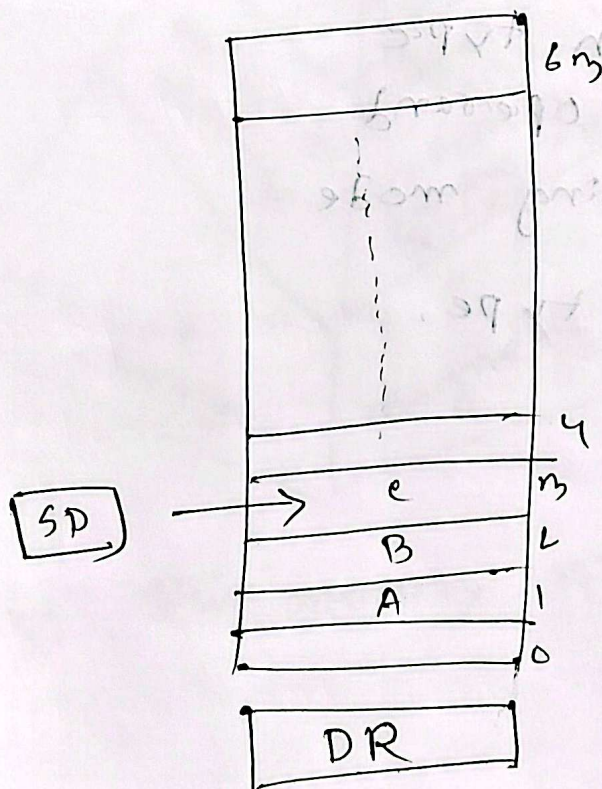


Fig: 64 word stack

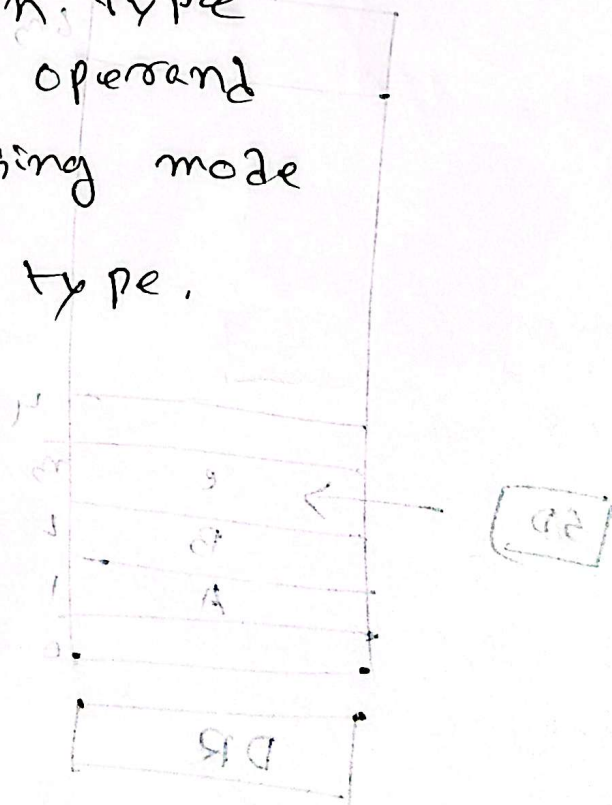
Ex.

Trade-off: A trade-off is a situation where gaining one advantage requires giving up another advantage.

In other way, to increase something on one side, we must compromise on another side.

✓ Instruction class type :-

- operation type
- no. of operand
- addressing mode
- data type.



memory hierarchy :- The memory hierarchy is the organization of memory components in a computer system based on speed, cost & size to achieve the best performance at the lowest cost.

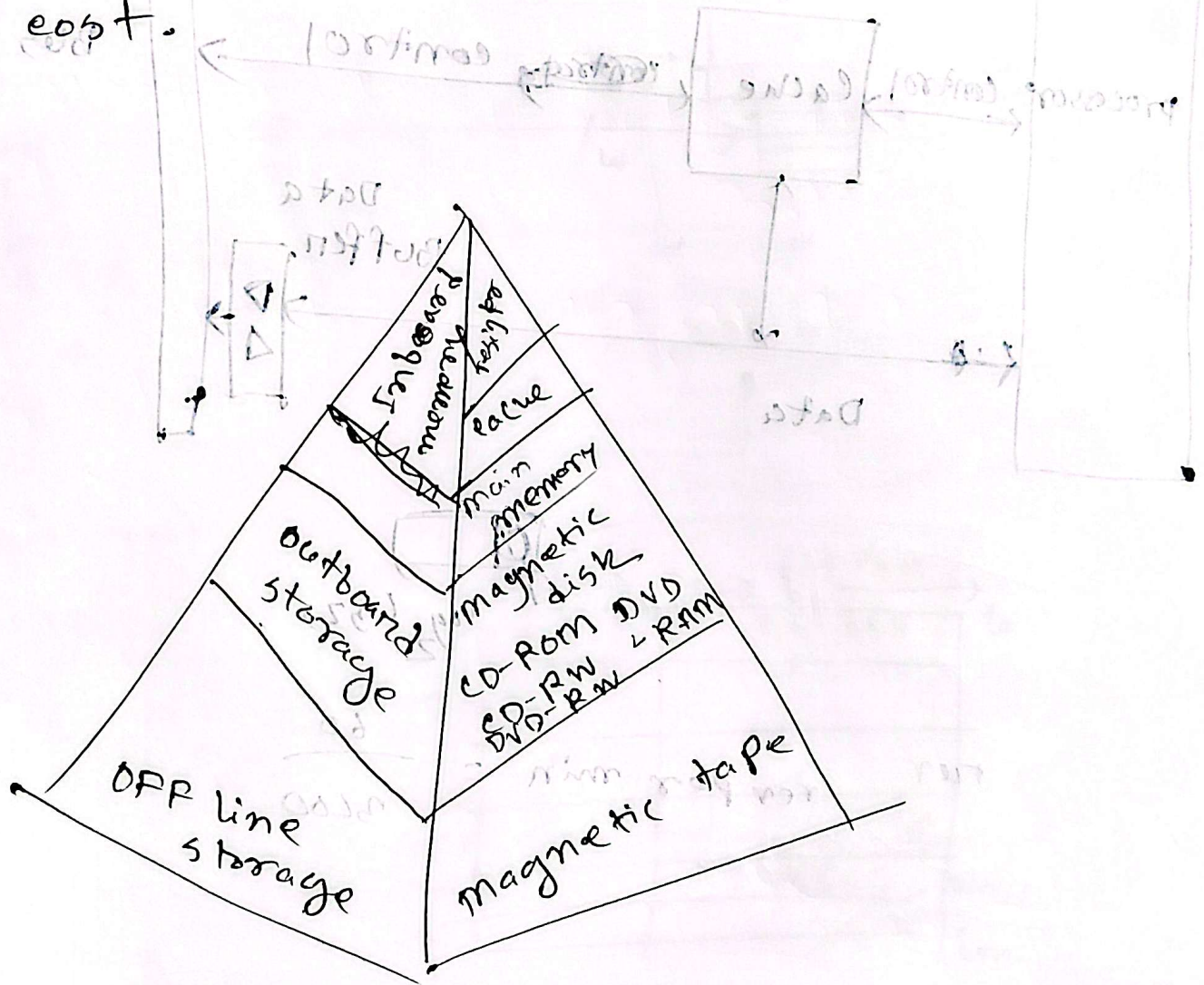


Fig: The memory hierarchy.

7(a)

① Interrupt - Driven I/O:

- CPU repeatedly checks the I/O device status to see if it's ready
- I/O device send ~~an~~ in an interrupt signal when ready.
- CPU can continue ~~at~~ executing other tasks until interrupted.
- more efficient than polling

② Buffering:

- temporarily store I/O data in memory.
- CPU can continue processing while I/O fills or empties the buffer.

→ Types:

Single buffer : One buffer btw CPU & I/O
Double " : two " used alternately
Circular " : multiple " in a loop for continuous flow.

③ DMA

- DMA controller transfer data directly btw memory & I/O without cpu involvement
- cpu only initiates the transfer & continues other work.
- cycle stealing or Burst mode can be used
- Best for large data transfer.

④ Multiprogramming with cpu scheduling

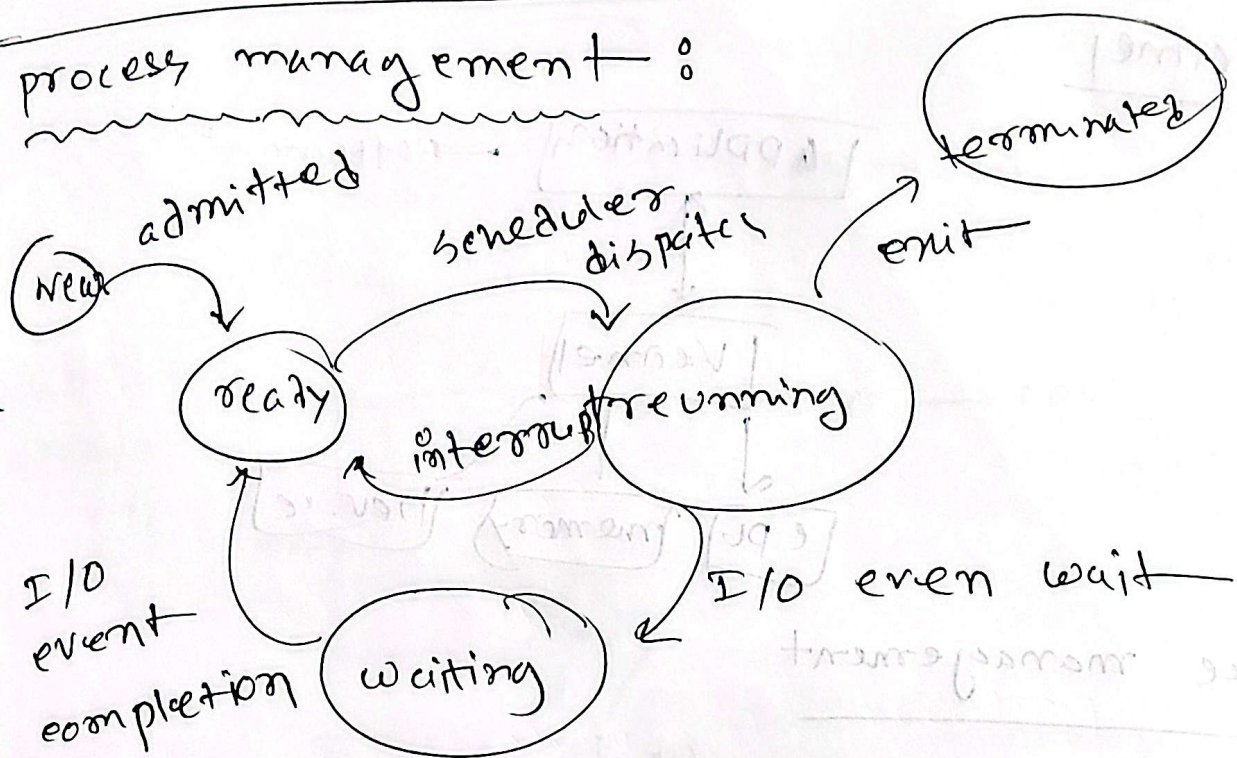
- OS switch to another ready process when one is waiting for I/O
- use short-term scheduler to maximize cpu utilization
- prevent cpu from idling during I/O wait.

⑤ Spooling (simultaneous peripheral operation on-line)

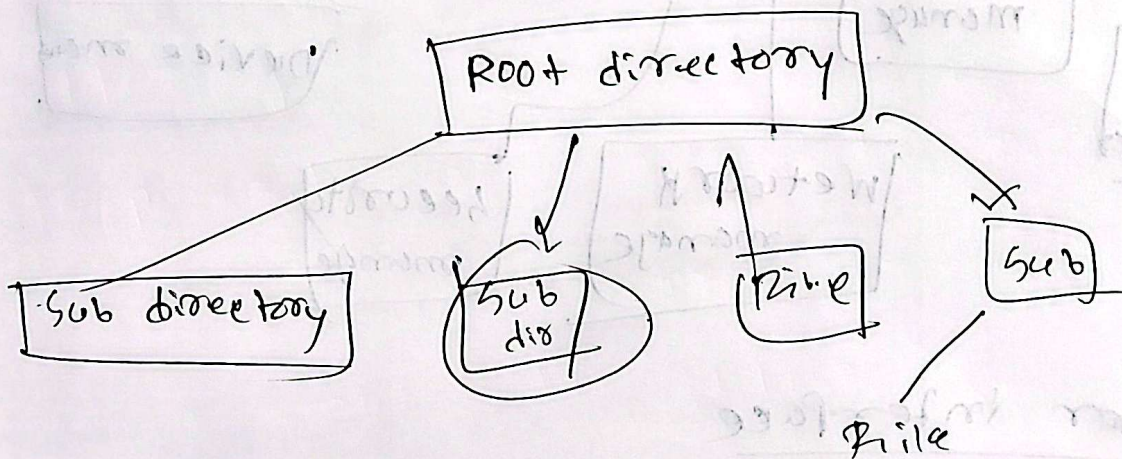
- used for device like printers.
- cpu doesn't send data directly to the I/O device
- data is first stored in spool (disk or buffer)
- I/O device reads data from the spool at its own pace.

Key elements of OS:

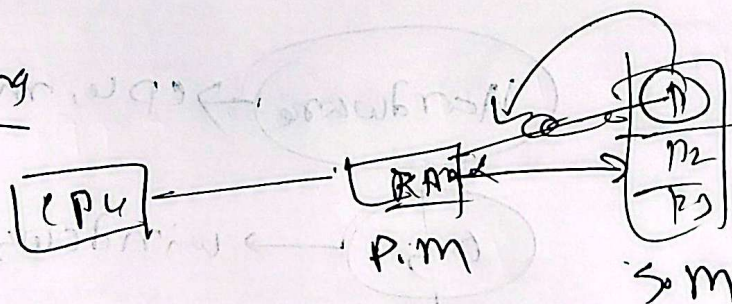
① process management:



② File management:



Memory mang



Direct mapping	Associative mapping	Set-associative
→ fixed single cache line	→ Any cache line allowed	→ Any line inside a fixed set
→ lowest flexibility	→ Highest flexibility	→ moderate
→ High conflict miss	→ no conflict miss	→ less conflict miss than direct mapping
→ simple Hardware	→ Very complex	→ moderate
→ Fast access	→ slowest	→ Balanced speed
→ uses: index + tag + offset	→ index + tag + offset	→ Tag + set + word
→ uses: Tag → line → word	→ Tag + word	
→ cheap to implement	→ Expensive	→ medium cost
→ require @ one comparator	→ require comparator for entire cache	→ comparator for each line in the set
→ mapping formula $= \text{block no. mod } \text{cache line}$	→ Formula No mapping formula	→ Formula $= \text{Block no. mod } \text{No. of sets}$

Direct	Associative	Set associative
→ Tag storage is small → more thrashing → Good for small cache → one-to-one mapping → suitable for low cost CPUs	→ largest → least thrashing → Good for small-but high speed system → many-to-many mapping → suitable for high-end system	→ medium → reduced thrashing → Good for general purpose system → one to many within set mapping → widely used in modern CPUs (2-way, 4-way, 8-way)

0, 8, 0, 6, 8

① Direct mapped

$$\text{cache line} = \text{Block number} \bmod 8$$

Step	Block number	line number	cache state	Hit/miss	Explanation
1	0	0 % 8 = 0	[0-----]	m	line 0 empty
2	8	0 % 8 = 0	[8-----]	m	line 0 overrites
3	0	0 % 8 = 0	[0-----]	m	replace by 0
4	6	6 % 8 = 6	[0-----6]	m	
5	8	8 % 8 = 0	[8-----]	mi	
total misses = 5					

② Associative map

Step	Block no	cache state	Hit/miss	Explanation
1	0	[0-----]	m ✓	
2	8	[0, 8-----]	m ✓	
3	0	[0, 8-----]	mi	
4	6	[0, 8, 6-----]	m ✓	
5	8	[0, 8, 6-----]	H	

total miss = 3

② 2 way set associative

8 block = 4 sets

each set has 2 line.
mapping : set = Block no. mod 4

step	Block	set	cache	H/M
1	0	$0 \% 4 = 0$	[0] → set 0	M
2	8	$8 \% 4 = 0$	[0, 8] → set 0	M
3	0	$0 \% 4 = 0$	set 0 [0, 8] set 0 [0, 8]	H
4	6	$6 \% 4 = 2$	set 2 [6]	M
5	8	$8 \% 4 = 0$	set 0 [0, 8]	H

Block	Cache	Hit/Miss	Set
0	M	Hit	0
8	M	Hit	0
0	H	Miss	0
6	M	Hit	2
8	H	Miss	0

End of the program