Final – 2020-21

(1) How to determine the performance of a processor?

- professor performance metric:

$$\text{performance} \uparrow = \frac{1}{\text{execution time} \downarrow}$$

- Execution time:

$$\text{execution time} = \text{Instruction Count} \times \text{CPI} \times \text{clock cycle time}$$

or,

$$\text{execution time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{clock Rate}}.$$

- Instruction count (IC): total instructions in a program.
- CPI (cycle per Instruction): Average cycles taken per instruction.
- clock cycle time: time for one cycle (seconds)
- clock Rate: frequence of processor.

- Average CPI $= \frac{\sum(\text{count} * \text{CPI})}{\sum \text{count}} =$

- MIPS (Million Instruction per second):

$$\text{MIPS} = \frac{\text{Instruction Count}}{\text{execution time} \times 10^6}$$

- MFLOPS (million floating point operations per second) is used for floating-point intensive programs.

(2) Difference between SRAM and DRAM:

	SRAM (Static RAM)	DRAM (Dynamic RAM)
1. storage technology	: Uses flip-flops	- uses transistor + capasitor
2. Volatility	: Volatile (losses data when data off)	- Volatile
3. Speed	: faster	- slower
4. Density	: lower	- higher
5. power consumption	: consume less power when idle, more when active.	- consume more power due to periodic refresh.
6. Refresh needed?	: No (data static)	- Yes
7. Cost	: expensive	- Cheaper
8. Typical use	: cache memory (L1, L2, L3)	- main memory

(3) Write short notes on the following
i) Hit time
ii) Miss penalty

Hit time:

Defination: the time it takes to access data from the cache when the requested data is already present.

Components:
- time to check the cache
- time to deliver data to the CPU.

↳ lower hit time → faster cache performance.

Miss penalty:

Defination: the extra time required to fetch data from the next level of memory when it is not found in the cache.

Components:
- time to access main memory or lower cache
- time to transfer data back to cache and CPU.

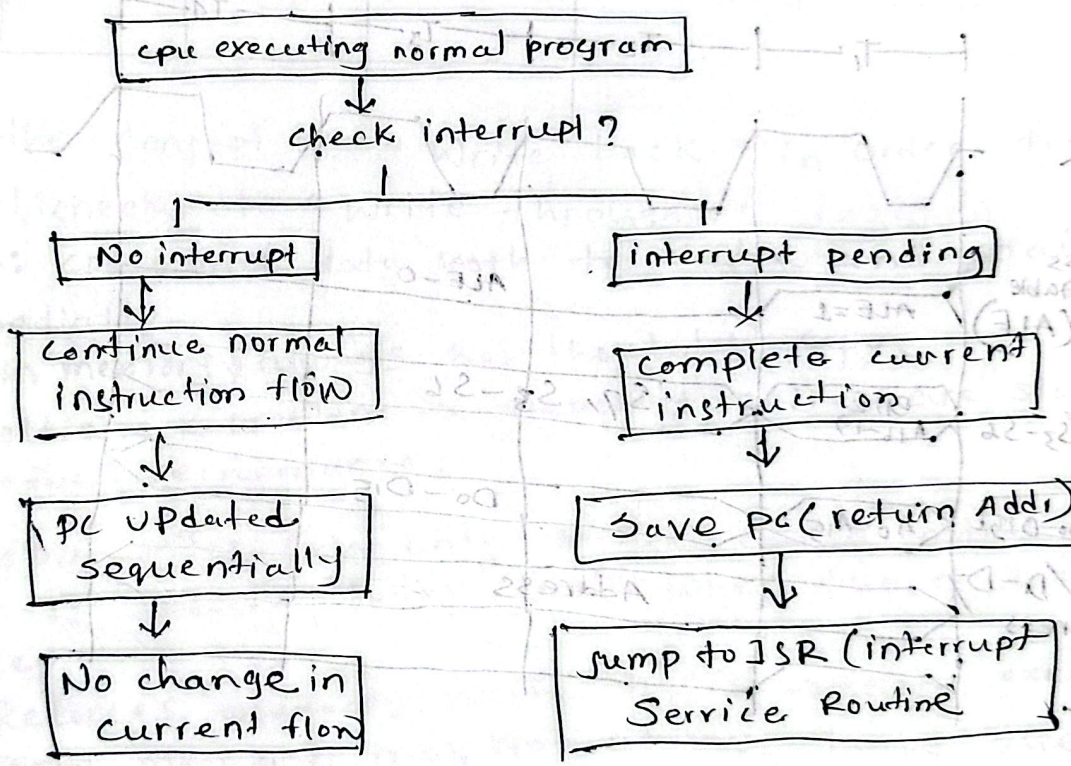
↳ Higher miss penalty → slower overall memory performance

(2) (i) Analyze the step(s) (i) no interrupt (ii) interrupt pending

No interrupt:
- the processor is currently executing the normal instruction flow.
- there is no interrupt request from any device.
- the CPU does not need to stop or change it's current program.
- the interrupt enable (IE) flag can be ON, but since no device has requested an interrupt, nothing happens.
- the program counter (PC) continues sequential execution.
- the stages operate normally without interruption.

Interrupt pending:

- At least one device has sent an interrupt request
- the interrupt line becomes active, indicating service is needed
- the cpu checks:
  - Interrupt enable flag (IE)
  - priority
- If interrupts are enabled → cpu will:
  - complete the current instruction
  - save the current pc
  - switch to the interrupt Service Routine (ISR)
- the normal program execution is temporarily paused
- After finishing the ISR, cpu returns to the saved pc and continues normal work.



(b) the ENIAC was a decimal machine, where a register was represented by a ring of 10 vaccum tubes.....
ENIAC used 10 vacuum tubes, arranged in a ring, to represent only one decimal digit (0-9). But only one tube was ON at any time.
If each tube could independently be ON or OFF, then 10 tube could represent:

$2^{10} = 1024$ possible states.

But ENIAC used only 10 states.

So, it used 10 tubes but only 10 out of 1024 possible combinations, which is: $10/1024 \approx 0.98\%$ this, means more than 99% of the representable states were Wasted.

If all tubes could be ON or OFF independently

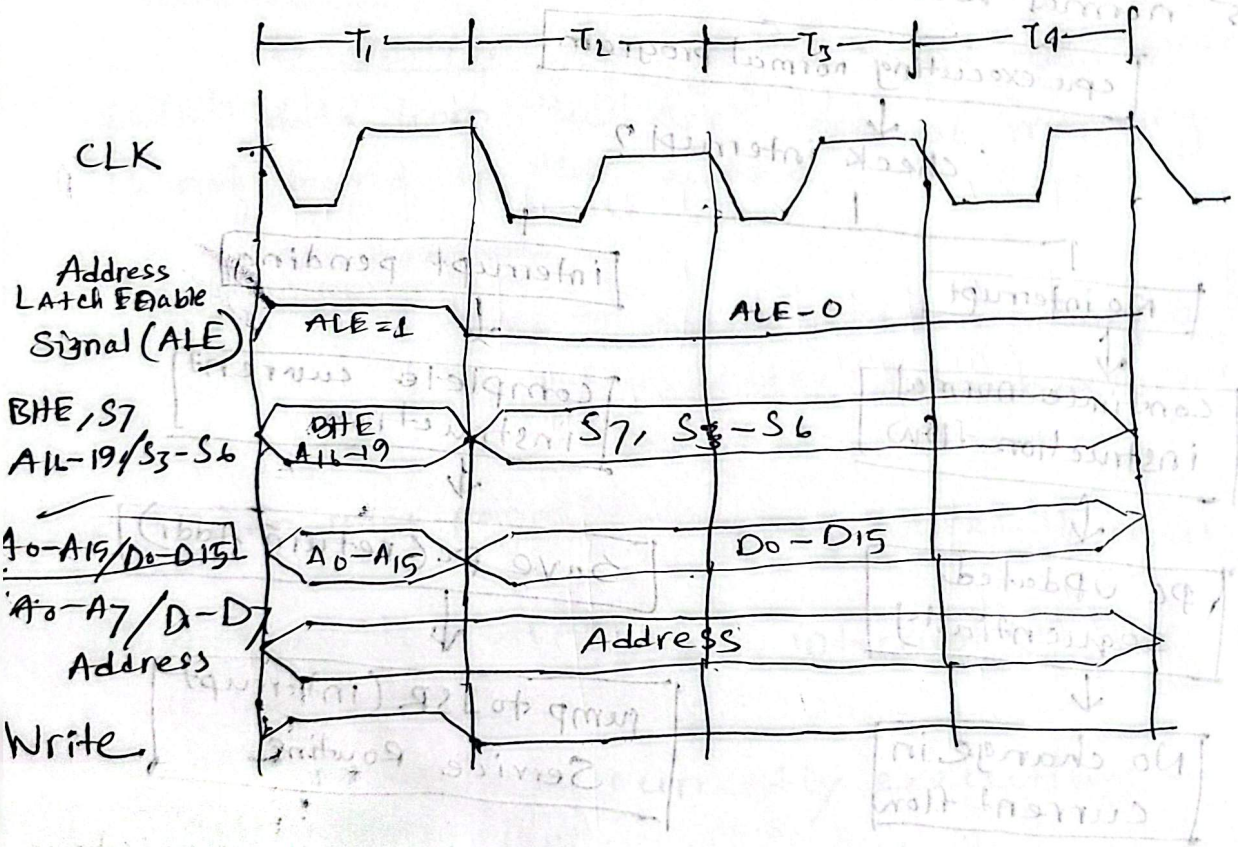
total states $= 2^{10} = 1024$

Integer representable: 0 to 1023.

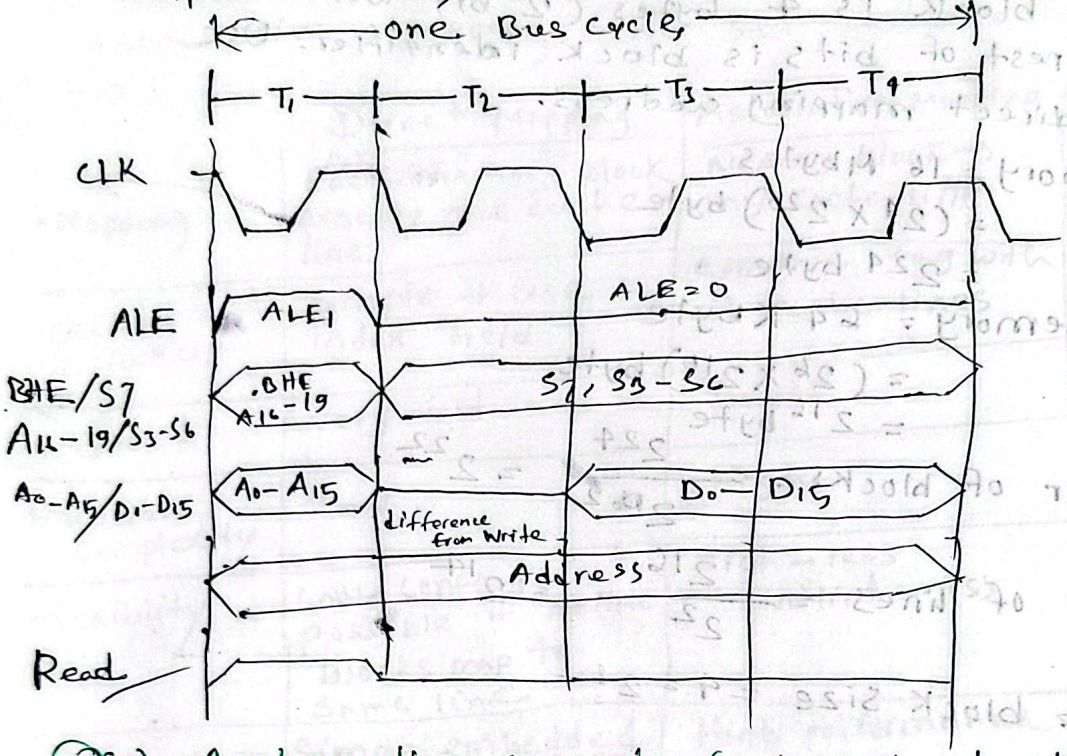
Because a 10-bit binary number has a maximum value of: $2^{10} - 1 = 1023$.

(c) show the Asynchronous timing diagram

Simplified 8086/8088 Write bus cycle



Simplified 8086/8088 read bus cycle.



(3)(a). Analyze the concept of "Write back" in order to avoid the bottleneck of "write through".

Write through: CPU writes data both to cache and main memory immediatly.

pros: main memory is always has the latest copy.

cons: Bottleneck problem → memory writes are slow → reduce performance.

Write back: CPU writes data only to the cache initially. Main memory is updated later, only when the cache line is replaced

pros: Reduces memory write traffic → faster execution

cons: Main memory may not always have the latest data.

- Write through problem: Every CPU write → memory write → show bus → CPU waits → performance bottleneck.
- Write back solution:
  - multiple writes to the same cache line only update the cache.
  - memory is written once per dirty line, not for every write.
  - Result → less bus traffic, higher speed, avoids bottleneck.

⑥ Let's design a cache of 64-KByte. Main memory is 16 Mbytes. Cache block is 4 bytes (2 bit word identifier) and rest of bits is block identifier. Use the concept of direct mapping address.

Main memory = 16 Mbytes
$= (2^4 \times 2^{20})$ byte
$= 2^{24}$ byte

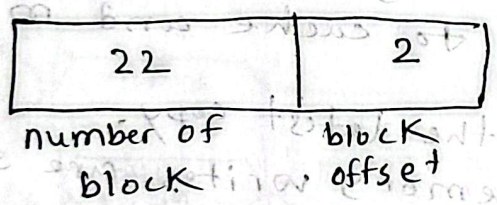
Cache memory = 64 Kbyte
$= (2^6 \times 2^{10})$ byte
$= 2^{16}$ byte

∴ total number of blocks = $\frac{2^{24}}{2^2} = 2^{22}$

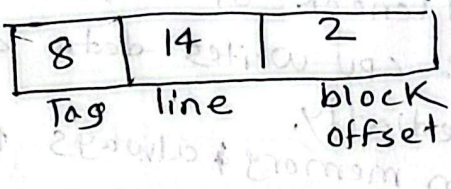
total number of lines = $\frac{2^{16}}{2^2} = 2^{14}$

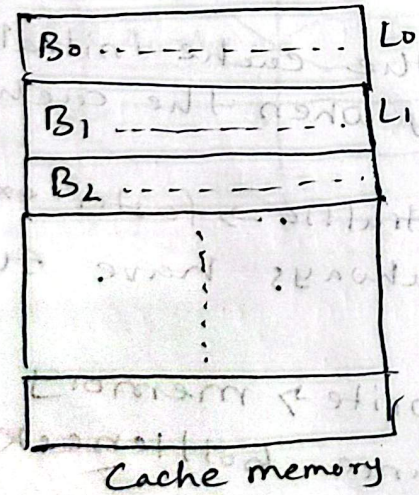
line size = block size = 4 = 2^2

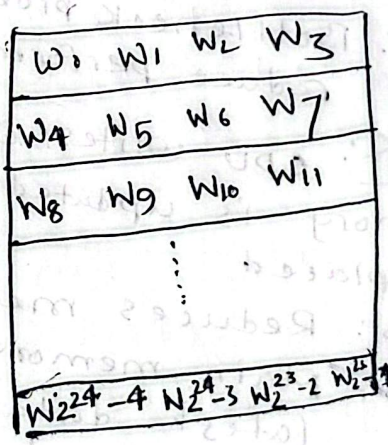
Physical Address (24)



PA (24)







(4)

(a) What are the difference among direct mapping, associative mapping and set associative mapping?

→

	Direct Mapping	Associative Mapping	Set-Associative Mapping
• Mapping	Each memory block exactly one cache line	Memory block → any cache line	Cache divided into sets; block → any line within a set
• Cache lookup	Simple → use index field	Compare tag with all cache lines	Compare tag with a set
Speed	very fast	slower	medium
Hardware complexity	Low	high	Medium
flexiblity	Low- conflicts possible if multiple blocks map to same line	High — less conflict miss	medium → reduces conflict compared to direct mapping
Example	Simple embedded Systems, older pcs.	High performance CPUs	Modern cpus
	simple but more conflict misses	flexible but hardware expensive	less conflict + moderate hardware

(b) 8 0, 8, 0, 6, 8.

Fully associative cache (8 blocks)

Access	Cache content	Miss/Hit
0	0	Miss
8	0, 8	Miss
0	0, 8	Hit
6	0, 8, 6	Miss
8	0, 8, 6	Hit

number of miss = 3.

Two way set associative cache (8 blocks → 4 sets x 2 lines)

mapping table: Index (block mod 4)

Block	
0	0
8	0
6	2

Access	Set Index	Cache content	Miss / Hit
0	0	0	Miss
8	0	0, 8	Miss
0	0	0, 8	hit
6	2	6	Miss
8	0	0, 8	hit

number of miss = 3.

- Direct Mapped cache (8 blocks - 1 line per index)

Access	Index = block % 8	Cache content	Miss / hit
0	0	0	miss
8	0	8	miss
0	0	0	miss
6	6	6	miss
8	0	8	miss

Number of miss = 5

(c) List and briefly define three techniques for performing I/O.

- programmed I/O : cpu contineously checks the I/o device status until it is ready for data transfer. simple but cpu is busy-waiting, wasting cycles.
- Interrupt-Driven I/O: I/O device signals cpu using an interrupt when it is ready. cpu can do other work in the meantime, improving efficiency.
- Direct memory address (DMA): Special DMA controller transfer data directly between memory and I/o device, bypassing CPU, cpu is free during transfer → Efficient for large data.

(5) (a) A computer has a 256 KByte, 4-way set associative, write back data cache with block size of 32 KBytes. The processor sends 32-bit addresses to the cache controller. Each cache tag directory entry contains, in addition to address tag, 2 valid bits, 1 modified bit and 1 replacement bit. Find the number of bits in the tag field for an address.

Cache size = 256 KB = 262,144 bytes.
Cache type = 4-way set associative
Block size = 32KB = 32,768 bytes.
Address = 32 bit.

Number of cache blocks

$$\text{Cache blocks} = \frac{\text{Cache size}}{\text{Block size}} = \frac{262,144}{32,768} = 8 \text{ blocks.}$$

$$\text{number of sets} = \frac{\text{Cache blocks}}{\text{Associativity}} = 8/4 = 2 \text{ sets}$$
(2 sets, each set has 4 lines)

Addres breakdown:

Addres = tag + set Index + Block offset.

-block offset bits

- Block size = 32KB → 32 × 1024 = 32768 bytes.

  Block offset bits: $\log_2(32768) = 15$ bits

- set index bits:

  Number of sets = 2

  Set index bit = $\log_2(2) = 1$ bit.

- Tag bits:

  tag bits = Address bits − (set index bits + Block offset bits)

  = 32 − (1+15)

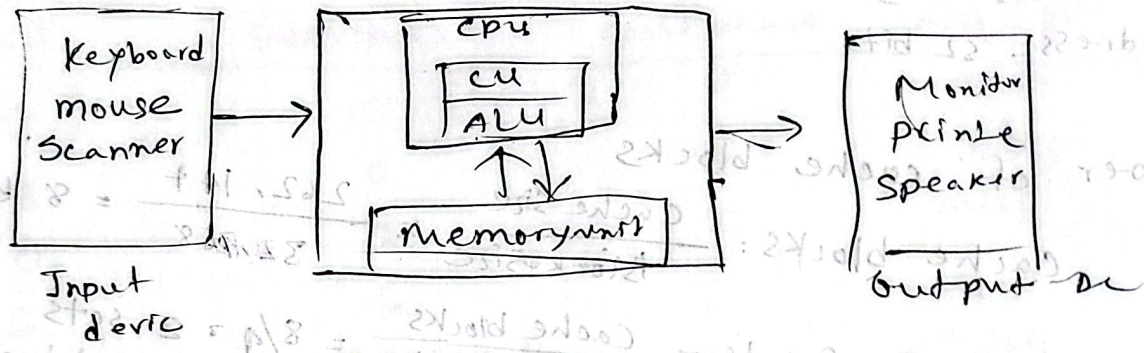
  = 16 bits.

tag field = 16 bits

Additional bits = 2 valid bits + 1 modified bit
+ 1 replacement bit
= 4 bits.

total bits per tag entry = 16 + 4
= 20 bits.

5(b) Von neumann ~~ar~~ compute Architecture is the basic computer Architecture fo the modern computer. explain this——



Von-neumann architecture is a computer design model where program instructions and data share the same memory and are accessed sequentially by the processor. this is the foundation of most modern computers.

Key features:

- single memory for data and instruction.
- Stored program concept.
  - cpu featches instruction from memory sequentially
- Sequential execution
- CPU components (ALU, CU, Registers)
  - → fetches, decodes and execute instructions.
  - → temporary storage inside cpu for fast access.
- Input / output system.

(c) Describe the multiplexed bus with adv and disadv.

A multiplexed bus is a bus in which the same set of physical lines is used to transmit multiple types of information at different times, usually address and data, to reduce the number of bus line required.

Working:

- In the first part of the cycle, the bus carries the address.
- In the next part, the same bus lines carry the data.
- Requires control signals to differentiate between address and data phases.

Adv:

- Reduced number of lines → cheaper and simpler hardware.
- Less physical space required on the circuit board.
- Can be used for both address and data transmission efficiently.

Disadv:

- slower performance: address and data use the bus sequentially not simultaneously.
- Requires extra control logic to manage timing and distinguish address/data.
- More complex design compared to a dedicated bus system.

(b) (a). Describe the necessity of I/O modules

I/O modules are essential components of computer system that act as a bridge between the cpu - main memory and the external devices. they are needed for the following reasons:

- Device communication: I/O modules handle communication, converting device signals into a form the cpu can understand
- Speed mismatch handling: cpu is very fast, but I/O device are slow. I/O modules use buffers and interrupts to synchronize speed differences so the cpu does not waste time waiting.

- Data format conversion: CPU handles digital, binary data. Many I/O devices use analog or character-based data. I/O modules convert formats.
- Device Control: Each I/O device works in its own way. the I/O module performs device-specific operations such as:
  - control commands
  - status reporting
  - error detection.
- CPU Offloading:
  - Data transfer
  - Error checking
  - Interrupt handling
- Support for Different I/O techniques: I/O modules support:
  - programmed I/O
  - Interrupt-driven I/O
  - DMA (Direct Memory Access).

(b) In direct Memory Access (DMA) operation, describe about "Cycle Stealing":

Cycle stealing is a method used in DMA where the DMA controller temporarily 'steals' one memory access cycle from the CPU to transfer a small amount of data.

How it works:

- Normally, the CPU and DMA both need access to main memory.
- During cycle stealing, the DMA controller pauses the CPU for just one memory cycle.
- It uses that single cycle to transfer one word/block of data between memory and an I/O device.
- After that cycle, the CPU continues its work.

Why it's called cycle stealing:

Because DMA steals one cycle from the CPU's memory access time, the CPU has to wait for that cycle. However, the delay is very small; so, CPU performance is only slightly affected.

Key points:

- DMA steals one bus cycle at a time.
- CPU is momentarily halted only during that cycle.
- No long interruption like full bus takeover.
- It allows overlap of CPU execution and I/O operations, improving performance.

Adv:

- Better performance than programmed I/O.
- minimize CPU idle time.
- Smooth, continuous data transfer.

Example:

When transferring data from disk to memory, the DMA takes control for one cycle, transfers one word, the returns control to the CPU - repeating this quickly.

(c) Describe O/S as a resource Manager in I/O controller.

the operating system manages all hardware resources of a computer, and one of the most important tasks is managing I/O devices and I/O controllers, it ensures that multiple programs can safely and efficiently use I/O devices without conflict.

- Device Allocation: the OS decides which process gets which I/O device and when. it prevents two processes from using the same device at the same time.
- Buffering and Caching.
- I/O scheduling.
- Interrupt handling
- Device drivers
- error detection and recovery
- protection and security.