8/9/23

Syllabus

1) Overview of Computer System → Basic elements, Instruction, Interrupts
2) Basic introduction (of OS)
3) Process description & process scheduling → FCFS, SJF
4) Mutual exclusion & process synchronization → Peterson's algorithm, semaphore
5) Threads, Processes, Deadlock & Starvation
6) Memory management → Virtual Memory, Paging, Segmentation
7) Disk Scheduling & i/o Management → LOOK, SCAN, CSAN
8) File management & security

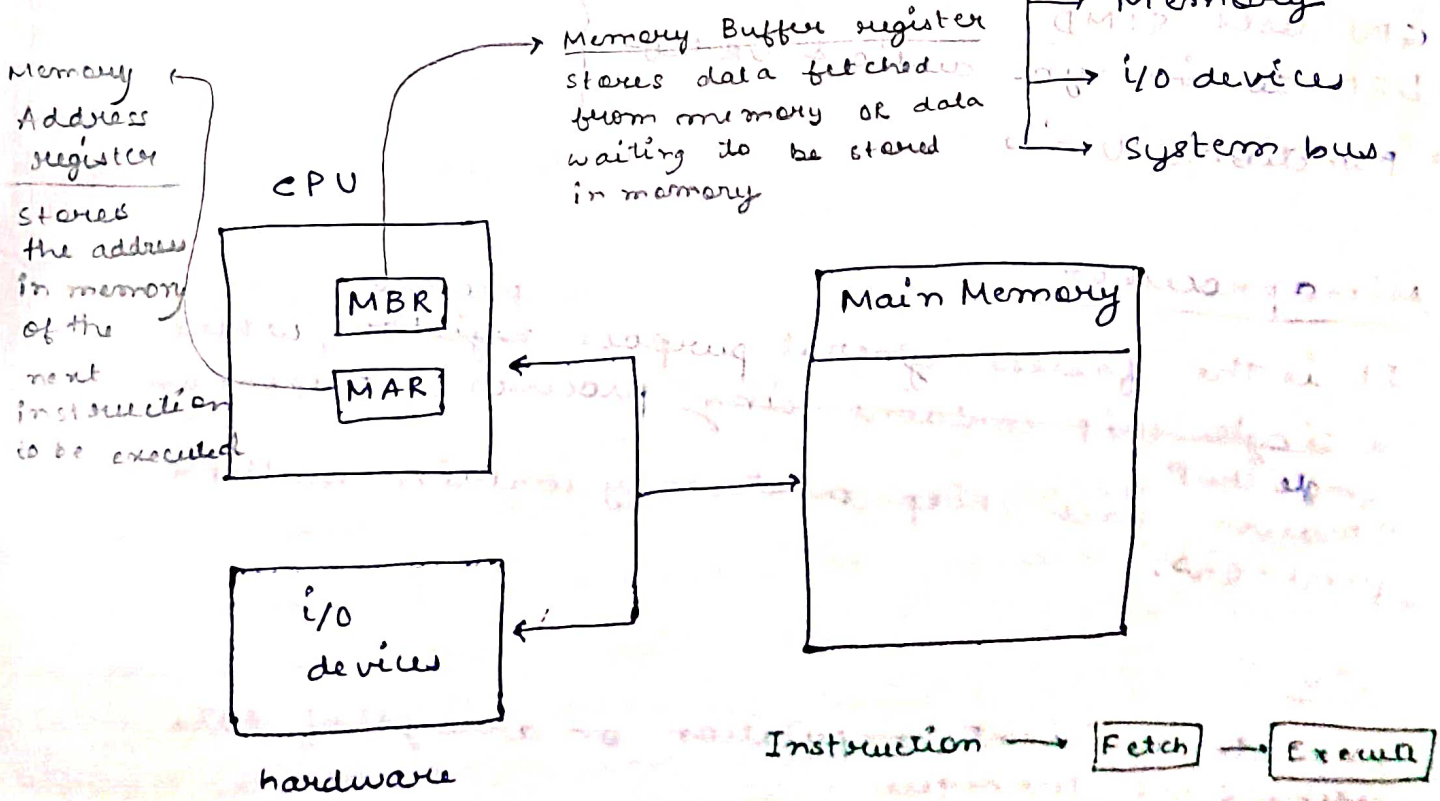
Intro.

- OS → interface b/w user & hardware

  First OS - MS DOS

  Then WINDOWS took over becoz of its advantage of Graphical User Interface

- Basic elements of comp. sys. → CPU, Memory, i/o devices, System bus.



12/9/23

Q) What is an OS?

Ans It exploits the hardware resources of 1 or more processors to provide a set of services to system user.

It acts as an interface b/w user hardware & user applications

Basic elements of comp.

1. Processor - controls operation of comp. & perform data processing
2. MM - stores data & program
3. i/o - moves data b/w computer & its external environment
4. System bus - provides communication b/w processor, i/o modules & main memory

* 2 main registers present in process - MAR, MBR
→ MAR - addressing memory for next read / write
→ MBR - contains the data to be written onto the memory or receives the data read from memory

EVOLUTION OF MICRO PROCESSOR

→ GPU uses SIMD (Digital signal processing) single inst^n / Multiple data
→ DSP to sync audio & video
→ Functional Units

1. Microprocessor

It is the fastest general purpose processor, where a single chip contains single processor is present on single chip
However each chip may contain multiple processors.

2. GPU

provide efficient computation on arrays of data using SIMD technique.

used for general numerical processing

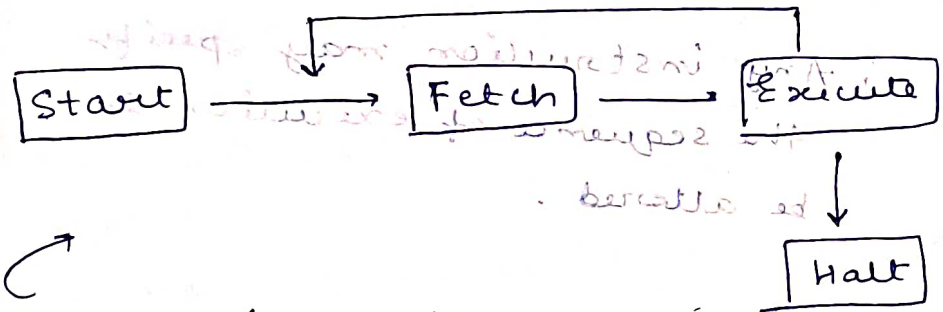
DSP (Digital signal processor)

deals with streaming signals (i.e. synced audio & video) such as audio & video used in embedded systems like modems

They provide support for security through encryption technique.

SOC (System on chip)

To satisfy the requirements of hand held devices (devices you carry in hand like mobile phone.) the microprocessor is giving weight to SOC components such as GPUs, cordec & MM in add^n to processor are on the same chip.



Instruction cycle

PC → addr of next instruction

IR → data in address of PC

★ Difference b/w PC & MAR ?

Ans PC - points to the address of next inst^n to be fetched

MAR - points to the address of data to be fetched to execute the current inst^n

Instruction Execution

A program consists of a set of instruction

It consists of 2 steps :

1. The processing required for a single instruction is called instruction cycle. In the inst. cycle the PC (program counter) holds the address of the next instruction to be fetched, The processor always increments the PC after each instruction fetch so it will fetch the next instruction in PC. The fetched instruction is loaded in IR. It contains

the bits that specify the action processor has to take. It has 4 action :

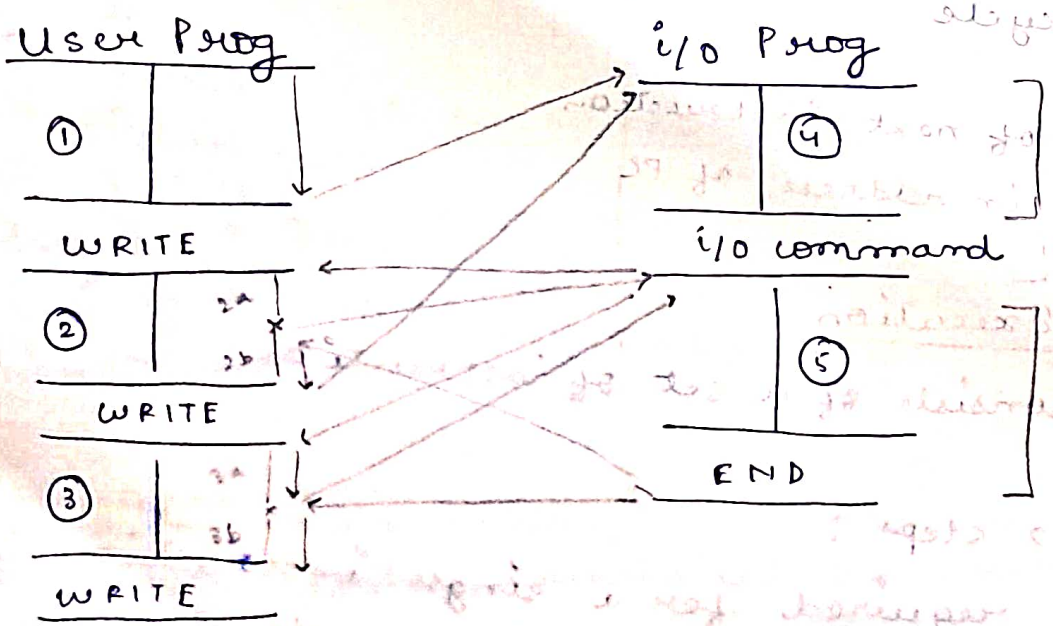
→ Processor memory : Data may be trasferred from processor to memory / vice versa

→ " i/o : Data may be transfered to or from any periferal device.

→ Data prossing : The processor may perform some arithmetic / logic operation on data

→ Control : Any instruction may specify the sequence of execution to be altered.

Interrupts



without int.

waiting time of procs

1
4
5
2
4
5
3

with interrupt/ end

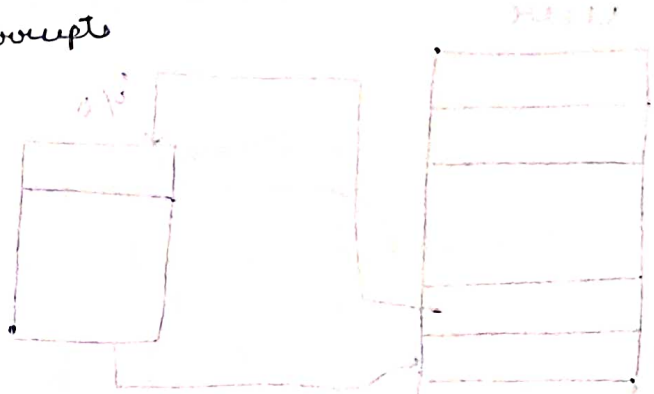
1
4
2a
5
2b
4
3a
5

No waiting time of processor.

A mechanism is provided by all computers by which other modules may interrupt the normal sequencing of the processor. They are provided as a way to improve processor utilization.

There are 4 classes of interrupts. :

1. Program {aka software interrupts
2. Timer
3. i/o
4. Hardware failure.



In the eg above :

The user program provides a series of WRITE program interleaved with processing. Code segments 1, 2, 3 refer to instruction that do not need i/o. The write calls are to an i/o routine which is a system :

The i/o program consists of 3 steps of instruction :

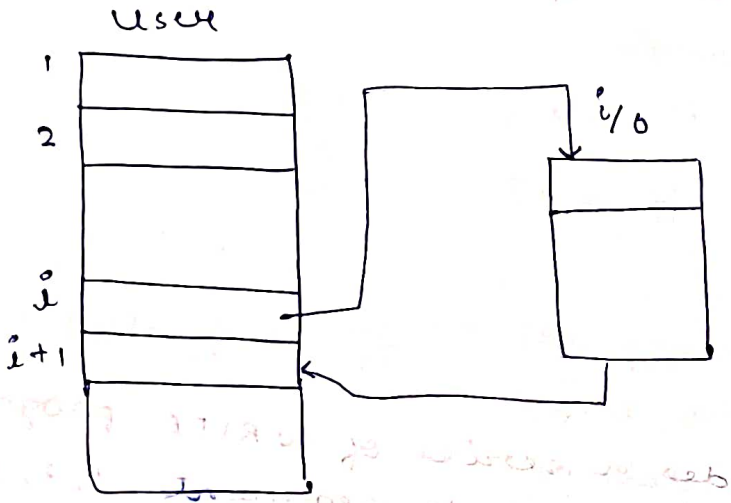
i) The preparation code to prepare for the actual i/o
ii) " actual i/o command
iii) The inst. to complete the operation.

Without interrupts cond

Without interrupts, the processor has to wait until the i/o routine is finished. It take a long time to for the external device to finish processing until then the time of processor is wasted but with use of interrupts the processor is engaged in executing other instructions while ~~in~~ an input-output operation is in progress.

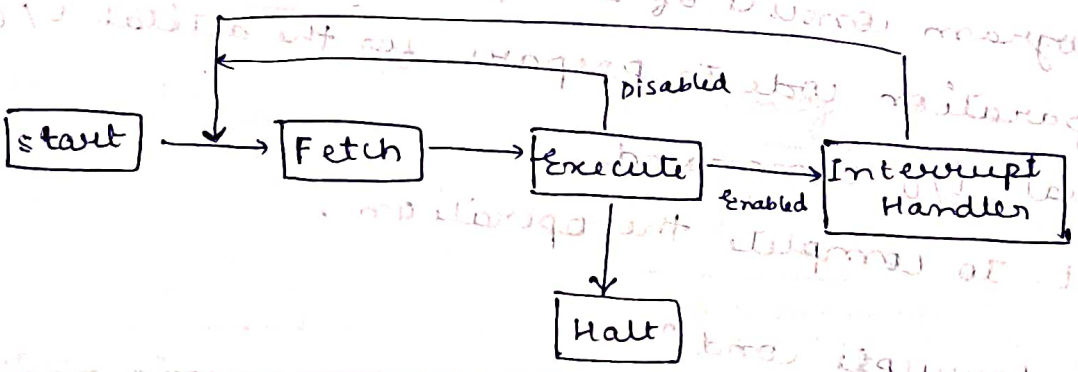
In eg we can see, when the external device completes its processing, the i/o module for that external device sends an interrupt request signal to the processor.

The i/o program i.e. invoked consists only of preparation code & actual i/o command. This way processor works concurrently with the i/o program & improves its utilization.



13/9/23

Instruction cycle with interrupt:

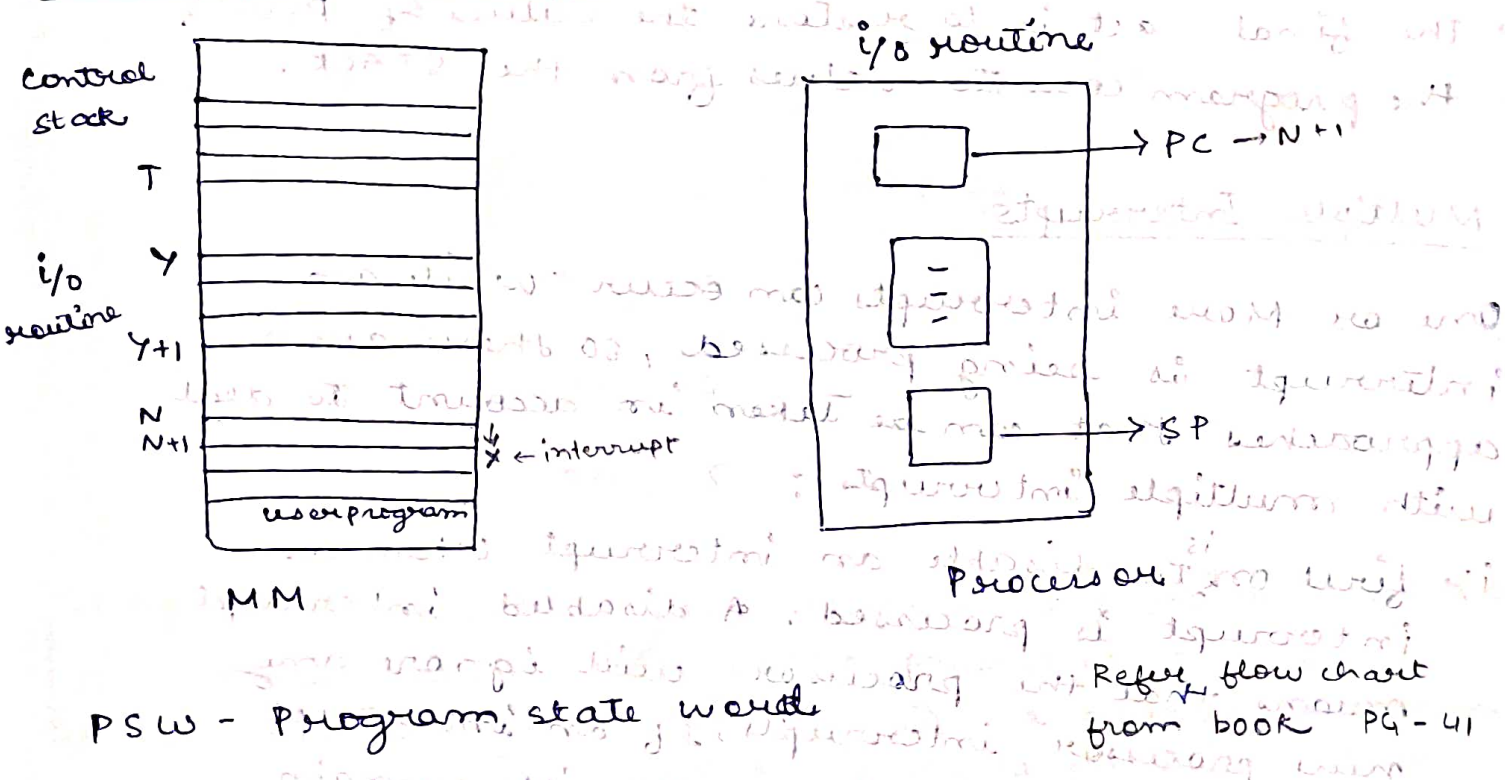


When the interrupt processing is completed, the user program does not have to contain any special code to accomodate interrupts. The processor & OS is responsible for suspending the user program & then resuming it from the same pt. To accomodate the interrupts, an interrupt stage is added to the instruction cycle. In the interrupt stage, the processor checks to see if any interrupts have occured which are indicated by the presence of interrupt signal. If no interrupts are pending, the processor proceeds to the fetch stage, and fetches the

next instruction of the current program. ~~If an~~ interr

- If an interrupt is pending, the processor suspends execution of current program & executes the interrupt handler routine. This routine determines the nature of interrupt & performs whatever actions are to be taken

Interrupt Processing



PSW - Program state word

Refer flow chart from book PG- 41

- The device issues an interrupt signal to the processor.
- The processor finishes the execution of the current instr^n before responding to the interrupt.
- The processor tests for a pending interrupt request, determines that is 1, & sends an acknowledgement signal to the device that has issued the interrupt.
- Now processor needs to prepare to transfer control to the interrupt routine. It saves the information needed to resume the current program which are PSW (program status word) & the location of next instruction to be executed. The processor then loads the program counter with the entry location of interrupt handling routine that will handle the interrupt. Once the PC has been loaded, next instruction cycle is started from interrupt handler routine. Now interrupt handler routine would ~~need~~ general purpose registers

so it starts by saving the contents of all registers to STACK.

- The interrupt handler now proceeds to process the interrupt, when the interrupt processing is complete, the same values are taken back by the control STACK.
- The final act is to restore the values of PSW & the program counter values from the STACK.

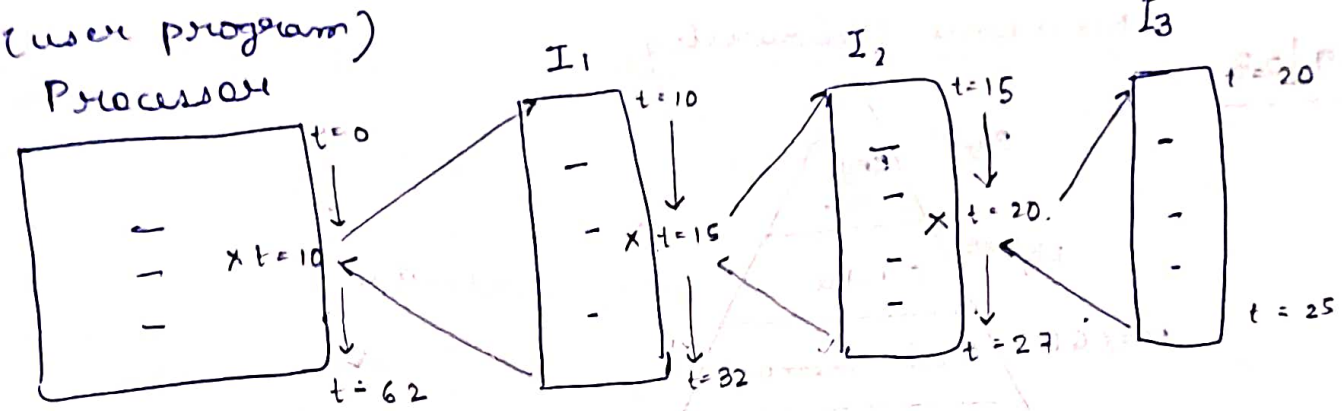
Multiple Interrupts

One or More interrupts can occur while an interrupt is being processed, so there are 2 approaches that can be taken in account to deal with multiple interrupts:

i) first one is to disable an interrupt when an interrupt is processed. A disabled interrupt means that the processor will ignore any new processor interrupts. If an interrupt comes during this time, it will remain pending & will be checked by processor after the processor has enabled interrupts. Thus if an interrupt occurs when a user program is executing, the interrupt is disable immediately

Drawbacks - It does not take account, relative priority / time critical needs.

Eg - When an input arise from communication line, its priority is higher & it needs to be serviced immediately. But as interrupts are disabled it has to wait for a very long time until the previous interrupt has executed its execution.



Priority | Total time to complete

$t = 0$, 40 ← user prog

2 $t = 10$, 10

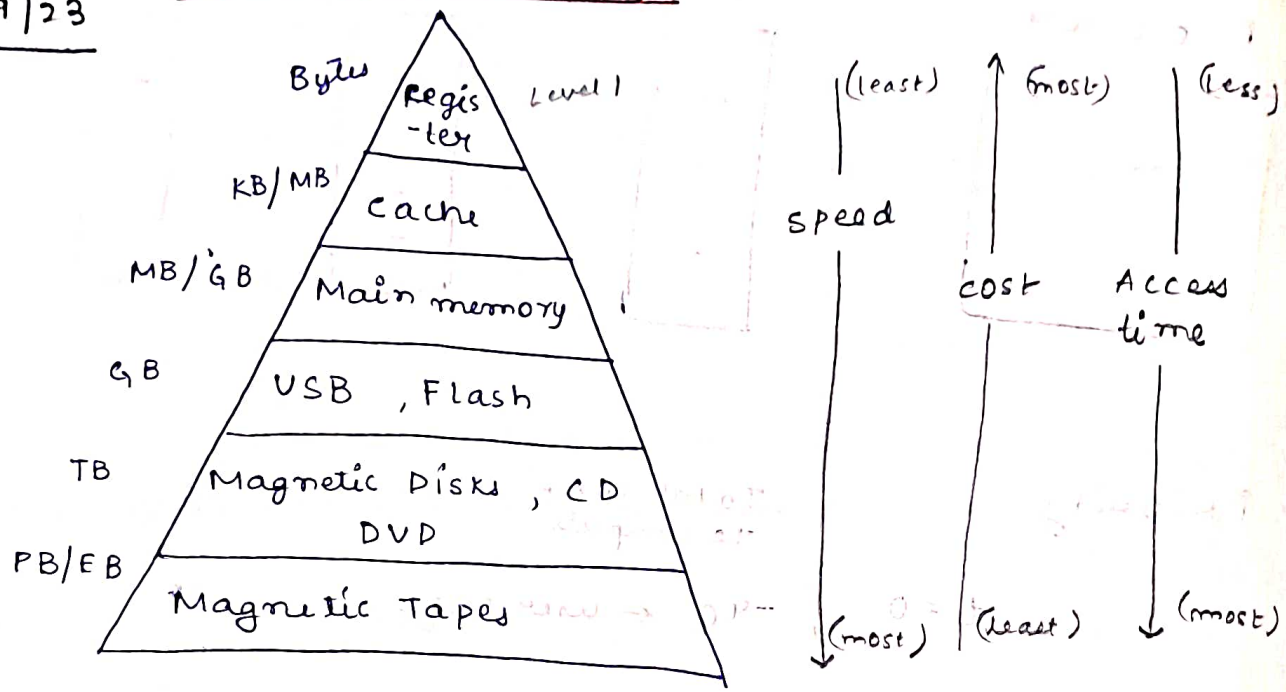
4 $t = 15$, 7

5 $t = 20$, 5

Suppose there are 3 i/o devices, a printer, a disk & a communication line with the priorities 2, 4, 5. A user program begins at $t = 0$, a printer interrupt occurs the user information is stored on control stack & execution continues at interrupt service routine. While this routine is still ~~co~~continuing, another interrupt from communication line arise at $t = 15$. As the communication line has higher priority, the interrupt request of this i/o device is serviced. At $t = 20$, an interrupt with much higher priority arise, so the processor services the interrupt request of this i/o device. When all the interrupts are handled, the previous processor state is restored which is the execution of printer interrupt. After the interrupt routine of ~~in~~ prin is completed the execution of user program continues.

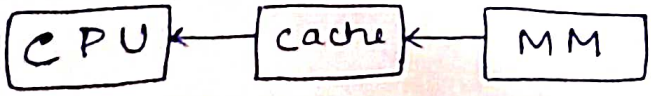
15/9/23

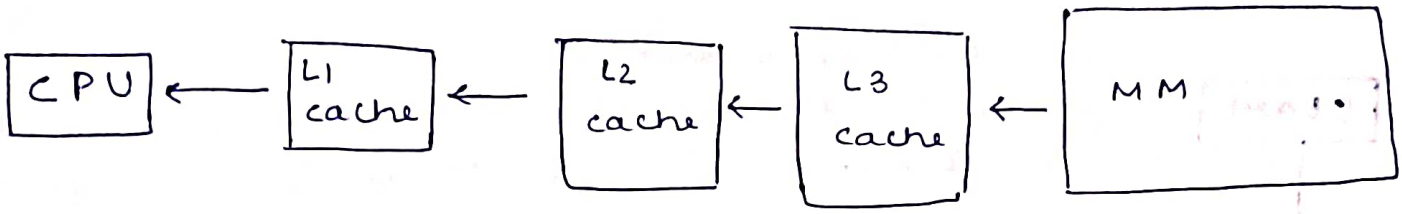
Memory Heirarchy



- The memory in computer system is divided in the form of levels. The smaller, more expensive, faster memory are supplemented by larger, cheaper & slower memory.
- The smallest, fastest & most expensive type of memory consists of registers. Skipping to 2 levels main memory is the principle internal memory of the computer. The cache is usually visible to the programmer for staging the movement of data b/w MM & registers. These 3 types of memory is volatile.
- External non-volatile memory is referred to as secondary memory which include USB & flash memory. They store program & data files.
- The magnetic disks & magnetic tapes are usually used to take back up of the data & are slower than other memories in the above level of the memory heirarchy.

Cache Memory



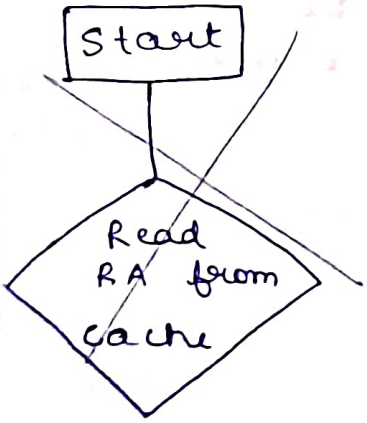


LRU : Least Recently Used

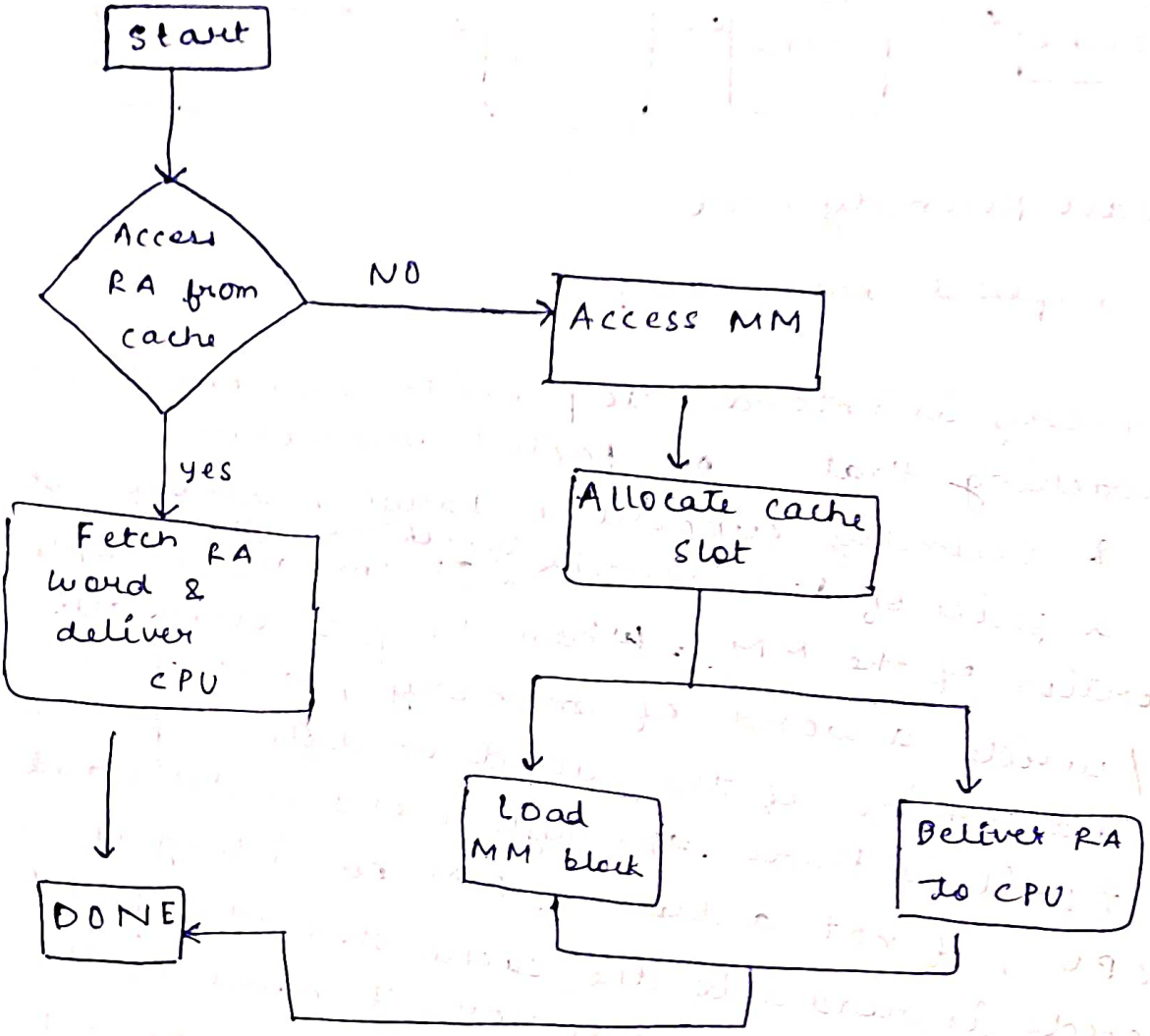
↳ data replaced from cache

- Cache Memory is intended to provide memory access time approaching that of fastest memories available & secondly support a large memory size that has a price of less expensive types of memory. It contains a copy of the portion of the MM. When the processor attempts to read / write a word of memory, a check is made to determine if the word or byte is present in cache. If it is there, the byte / word is delivered to the CPU, if not a block of MM consisting of fixed words is read into the cache then the byte / word is delivered to the processor. Because of the concept of locality of reference, when a block of data is fetched into the cache to satisfy a single memory reference, it is likely the near future memory references will be to other bytes of the block

RA = read address



RA : read address



- The fig illustrates the read operation: The processor generates the address RA of a word to be read. If the word is contained in cache it is delivered to the processor otherwise the block containing that word is loaded in the cache & the word is delivered to the processor.

** Now there are design issues that must be addressed in dealing with virtual memory & cache design.

- Points to consider:
  1) Cache size
  2) Block "
  3) Replacement Algo
  4) Mapping fn
  5) write policy
  6) No. of cache level.

(i) Cache size

According to heirarcy, small caches have significant impact on performance cache is closure to CPU. Hence size of cache is usually smaller.

(ii) Block Size

It is the unit of data exchanged b/w cache & main memory

(iii) Replacement Algo

It chooses which block to replace when a new block is to be loaded into the cache & cache already has all slots filled other blocks. One of the way is replacing the block that is least likely to be needed again in the near future such as LRU

(iv) Mapping fn

When a new block of data is read into the cache, the mapping fn determines which cache location the block will occupy.

(v) Write Policy

It the contents of a block in cache are altered then it is necessary to write it back to the MM before replacing it.

→ One way is writing can occur everytimer, when block is updated

→ Another way is writing only when block is replaced

(vi) No. of cache levels

Multiple levels of cache provide fast access time to CPU & enhance the performance.

Techniques ~~DMA~~ to execute I/O operation

When the processor is executing a program and encounters an instruction related to I/O. 3 techniques are possible for I/O operations:

i) Programmed I/O

ii) ~~DMA~~ Interrupt driven I/O

iii) ~~DMA~~ DMA

→ In case of programmed i/o, the i/o module performs the requested action & sets the respective bits in i/o status register & takes no further action to allot the processor. The processor periodically checks the status of i/o module, until it finds the operation is complete.

→ Interrupt driven i/o : It is more efficient than programmed i/o. It still requires active intervation of the processor to transfer data b/w memory & i/o module. It's drawback is the processor is busy in managing i/o transfer and its utilization in running current program decreases. So the solution to this problem is DMA, where large volumes of data is to be moved. DMA is more efficient technique. The DMA f^n can be performed by a separate module or it can be ~~incorp~~ combined with i/o module. The processor issues a command to the DMA module by sending these 4 informations :

i) whether a read/write is requested
ii) The address of i/o device involved.
iii) The starting location of memory to read data from & write data to
iv) The no. of words to be read/written

The processor then continues with other work