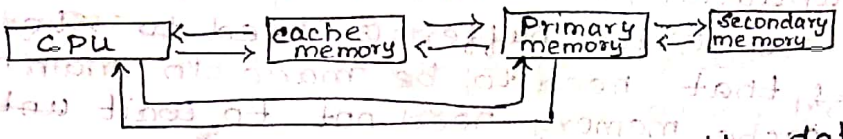
(1) Explain cache Memory? How to improve cache performance? Discuss? Nov'l Dec-17 (Reg 13)

(*) cache Memory is a first level of memory in memory hierarchy.

(*) It is based on the principle of locality of reference - [spatial reference & Temporal reference]

(*) cache Memory is an extremely fast memory type that acts as a buffer between RAM and the CPU.



(*) When CPU needs data/word, it fetches the data from cache memory, if it is available, then CPU access that data from cache memory. It is called cache hit.

(*) If not available in cache memory. It is called cache Miss.

(*) cache miss handle by hardware. ~~The time taken to process~~ on cache miss, the data is moved from main memory to cache memory.

(*) The time taken to process a cache miss, is depend on (i) Memory latency [is the time taken to fetch the first word from a block] (ii) Memory bandwidth [is the time taken to fetch remaining words from the block in cache]

(*) When CPU needs a word it fetches cache memory if it not their, it fetches Main memory. If requested item in a page is not in main memory then such failure is called "page fault".

↳ Instead of waiting, CPU performs some othe task till the pages move from disk to main memory.

↳ They are,

(1) Hit ratio $H_r = \frac{H}{H+M}$ H = cache hit
M = cache Miss,

↳ It is (H_r) is a fraction cache hit to the No of cache look. Higher Hit ratio are always affect the avg memory access time (AMAT).

$AMAT = H_t + M_r * M_p$

M_r => miss rate
M_p => miss penalty
H_t => Hit time
[H_t => time consumed by CPU to hit cache]

(ii) cache Read / Write.

(*) cache Read operation is easy because the copies of data in cache memory and main memory are identical & No need any updation.

(*) cache Write is not easy. It uses 2 strategy / write policy

① Write-through => whenever data in cache updated, it is immediately updated in main memory also. Here write buffer is used to store the changes that need to be made in main memory. So that cache memory need not to wait ~~not~~ while the changes are made to main memory

② Write-back => when item is changed in cache memory, the changes not made immediately in main memory. Rather, before replacing the block, the entire block with all changes is copied to main memory. It also uses write buffer to avoid waiting while writing to main memory.

Write-Miss => CPU wants to write to cache, it checks if element is present. Write miss occur when CPU doesnot find desired element in cache. On write-miss, element copied in two ways

(i) Write Allocate :- A block is loaded in cache & write is made to the element in cache & main memory.

(ii) ~~No~~ No-Write Allocate :- A block is not loaded in cache & no changes made in cache. Rather, the element is changed only in main memory.

CPU Performance Techniques:

[*] CPU performance techniques that can be used to improve the following.

1) Bandwidth
2) Miss Penalty
3) Miss rate
4) Hit time.

Bandwidth :- It refers to the count of bytes read or written per unit time (or per access).

Few techniques to increase Memory bandwidth.

(1) use wider Main Memory

(2) use simple Interleaved memory

(2) Miss Penalty :

↳ The techniques used for reducing the cache miss penalty are,

i) Multilevel caches. → An interface between main memory & cache. Including new cache (level 2) between cache (level 1) & main memory. so speed of level 2 is fast & has large amount of data can quickly accessed. & reduce miss penalty rate.

ii) Assigning higher Priority to Read miss than writes

iii) Merging Write Buffer

iv) victim caches.

(3) Miss rate :

3 types of miss rate

1. compulsary misses
2. capacity misses
3. conflict misses

(4) Hit time

↳ It can be reduced by using the following techniques

1. Small & simple caches
2. avoid address translation during cache indexing.
3. pipelined cache access
4. Trace cache.

(2) Explain Mapping Functions in cache memory to dete... how memory blocks are placed in cache. AP/May -1... Ap/may -16 May/June -16

Ans:-

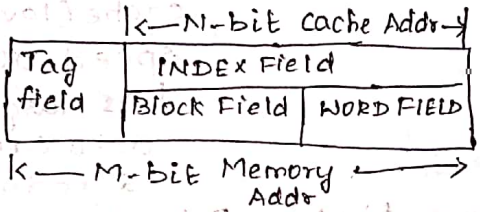
Basically there are three mapping techniques of cache memory.

They are, i) Direct mapping
2) Associative "
3) Set - Associative mapping technique.

(i) Direct Mapping:-

A cache mapping technique that maps block of main memory into distinct cache lines is termed as direct mapping.

Required cache line = Block NO Modulo Total NO of cache line



cache mem add } = N-bits

Mem addr = M bits

Tag field = M-N bits

Word field = $(\log_2 w)$ bit

block field = $(N - \log_2 w)$ bit

eg:- 512 x 12 cache memory.

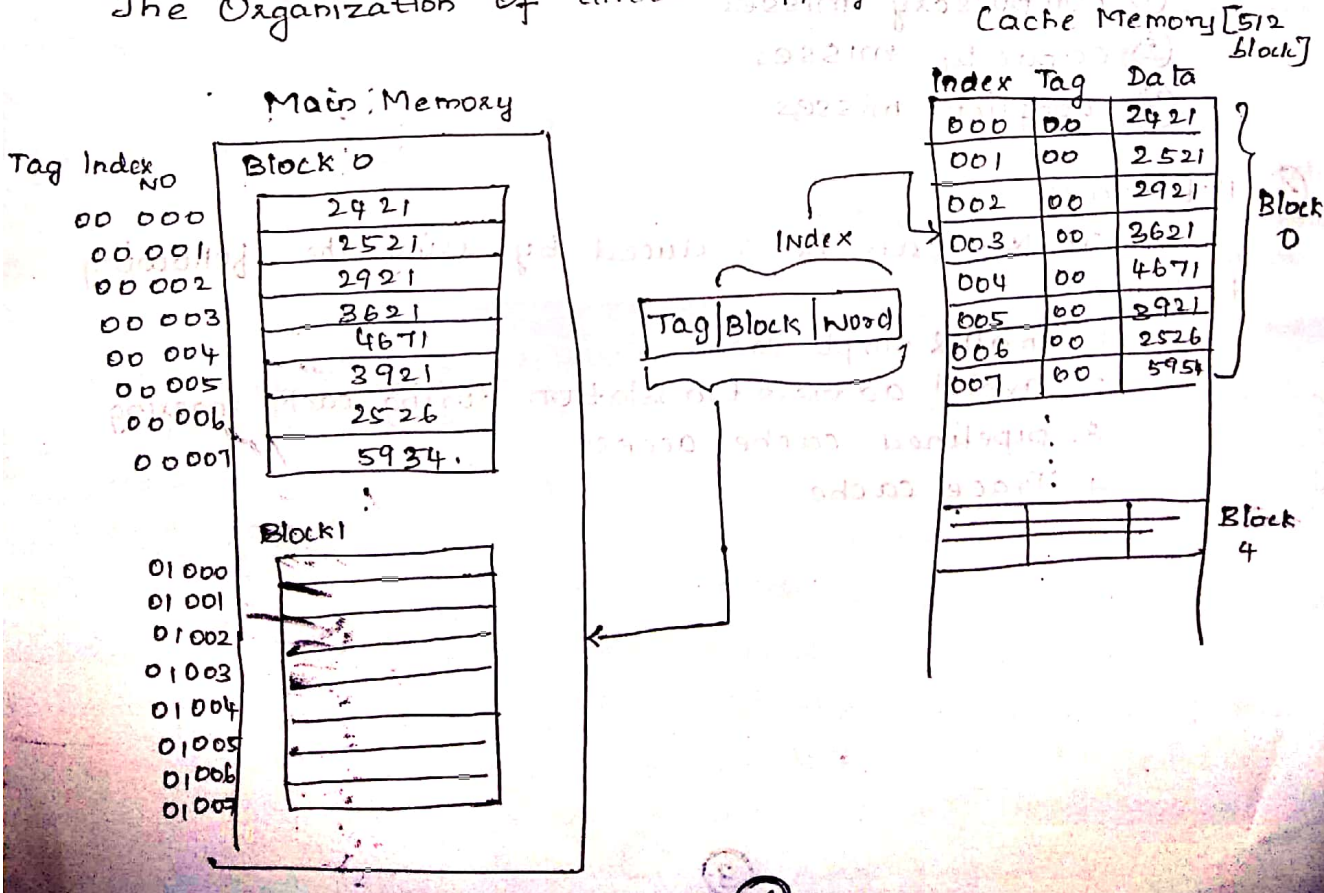
N bits = 9 bits ($2^9 = 512$)

32K x 12 main memory

then M bits = 15 bits ($2^{15} = 32K$).

15 bit addr is read from the main memory, the 9 bits in the Index field is used to access cache Line.

The Organization of direct Mapping.



→ In abv dgm, cache mem stores Block 0 from main memory. If any word required by CPU, then it can be easily read from the cache memory.

→ Suppose CPU wants Block 1 datas/words with an addr 01003. As the index addr is 003, the cache with the addr is accessed.

↳ The tag field of 003 is compared. They don't match, as tag field in cache is 00. So complete block of cache mem is replaced by block 1 of from main memory.

(2) Associative Mapping Technique

→ Quite complicated technique.

→ Memory addr consists of 2 fields.

tag field & word field

TAG	WORD
12 bit	4 bit

15-bit CPU addr

↓

Argument Reg (AR)

Addr	Data
02004	1234
29824	4316
05392	2656
.	.
.	.
.	.

Associative Mapping Cache Organization.

(3) Set Associative Mapping Technique:

→ Combination of direct mapping & Associative mapping.

→ In this Mapping, cache memory divided into 'S' No of Sets.

↳ A single set contains one or more tag data pairs in one word of cache. INDEX will point to 'S' No of tag data pairs

↳ if index addr = N-bits.

then 2^N words.

& length of word L_W = No of sets × (No of tag bit + No of bits in data word) bits

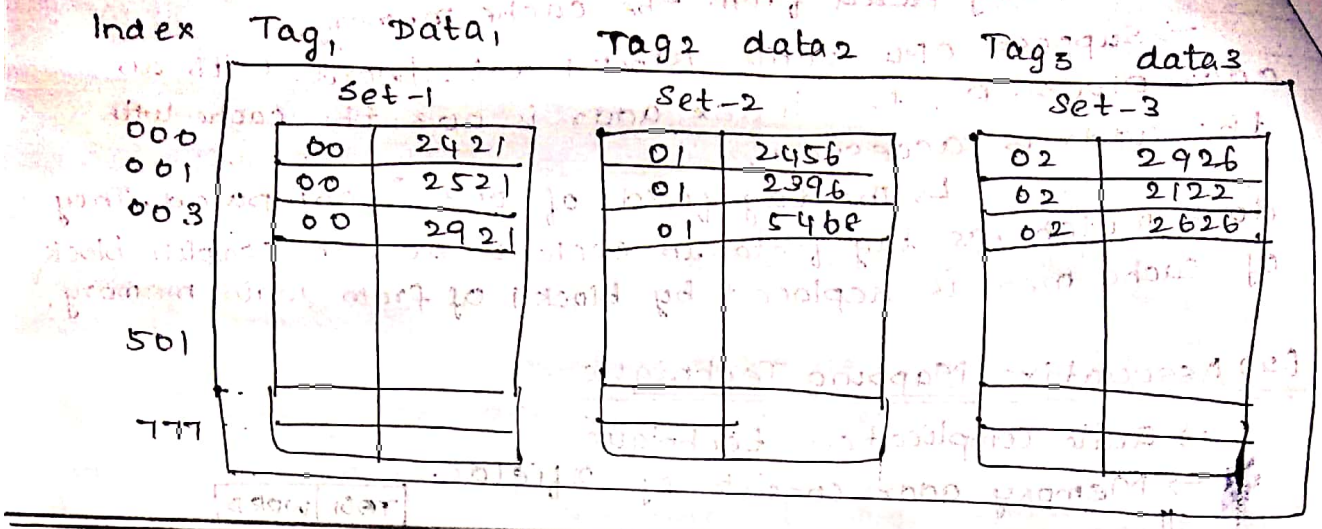
Size of cache memory will be = $(2^N \times L_W)$.

When data word at addr 01003 is requested by CPU. then cache line with index addr 003 is accessed.

↳ At this addr they are 3 diff tag data pairs.

↳ Associative search is carried out to ~~sea~~ compare tag value in current cache line. If match is found, then CPU will access the data word. If match Not found one of the tag data pairs in the set is replaced with the new tag pair using replacement alg [FIFO, LRU].

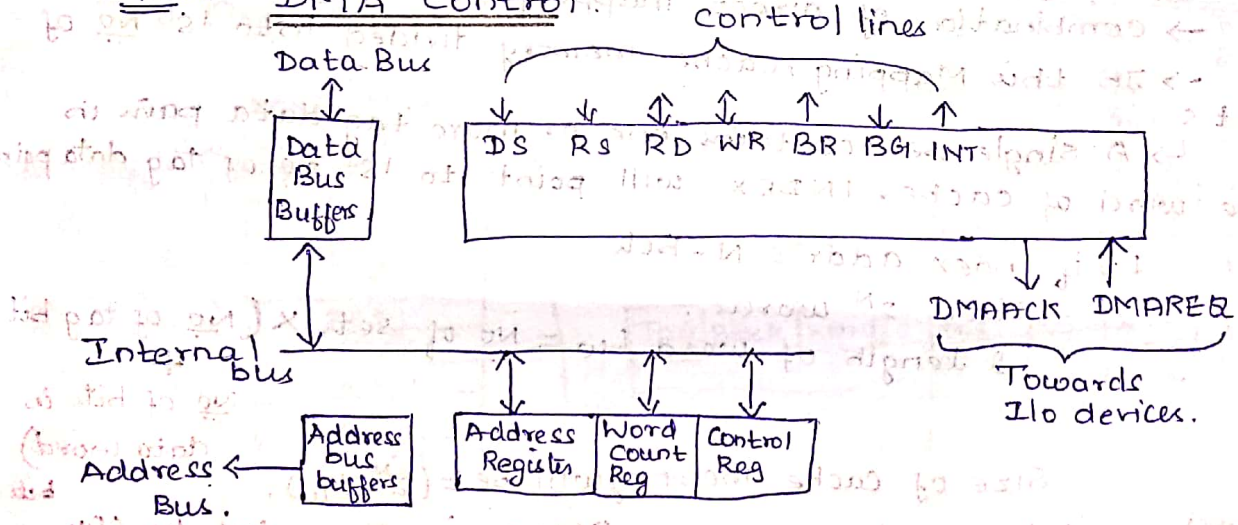
3-way Set Associative Mapping Cache Organization:



(2) Explain about I/O Communication Technique (or) Discuss about different Modes of data transfer.

(3) Explain about DMA controller with the block diagram May/June-16
(or)
Draw the typical block diagram of a DMA controller. and explain how it is used for Direct data transfer between memory & Peripherals ? Nov-dec-15

Ans:- DMA Control



DMA Block Diagram:

DMA Controller Consists of,

1. Control logic
2. Register
3. Data and address bus buffers
4. Address and data bus.

(1) Control logic.

=> It consists of seven control lines. They are,

(a) DMA Select (DS) (b) Register Select (RS) (c) Read RD (d) Write (WR) (e) Bus Request (BR) (f) Bus Grant (BG) (g) Interrupt (INT).

(*) DS & RS lines are two select inputs to the control logic. The CPU enables these line for selecting DMA register.

(*) The RD & WR lines are bidirectional & work as i/ps o/p to control logic.

(*) The BR & BG lines are related to bus.

↳ Request to bus made through BR lines.

↳ If BG i/p line is 0, then CPU will read or write from DMA reg using data bus.

↳ If BG i/p line is 1, means CPU handed over the bus to DMA & DMA will be able to read or write from memory.

(*) Other lines are DMA request and DMA acknowledgment

(2) Registers :- Three types of register used in DMA controller.

(a) Address Reg :- This reg holds the addr to point to a specific location inside the memory.

(b) Word count Reg :- It stores total no of words that are to be transfered into the memory.

(c) Control Reg :- It defines the mode in which the data is to be transferred.

(3) Data and Address Bus Buffers :-

→ The data is passed through various data bus buffers to get placed over the data bus

→ & the addr is placed on the addr bus by passing through the addr bus buffers.

(4) Data and Address Bus.

→ The data is transferred to/from the memory, using the data bus.

→ The addr is transferred to/from the memory using the addr bus

Operation of DMA :

⇒ CPU initializes the DMA using a pgm that consists of IIo instr to specify addr of particular DMA reg.

↳ Then data is transfer between the mem & peripheral device.

The CPU sends the following infor through the data bus to initialize the DMA.

(1) Starting addr of memory block
(2) Word count
(3) Read or write operation control.
(4) Another control that starts DMA transfer.

⇒ The CPU & dma can communicate with each other through addr & data buses.

↳ As soon as DMA controller receives the DMA req from the peripheral devices, it activates BR line requesting CPU to hand over all the buses.

↳ CPU replies through BG line by setting it to '1' & disables all the buses.

↳ The DMA will then place the current value of the addr reg into addr bus & then it initiates either RD or WR signal.

↳ The DMA will then send a DMA ack signal to the peripheral device.

↳ The RD & WR lines of the DMA controller are bidirectional. And their directions depends on BG value.

↳ If BG holds '0' then the RD & WR lines are as i/p to the DMA controller.

↳ If BG holds '1' then the RD & WR lines will be o/p from the DMA controller by indicating the read or write operation within RAM.

⇒ The peripheral devices can directly put a word in the data bus or it can receive a word from the data bus if ack it received from DMA.

⇒ The addr reg is incremented & the word count reg is decremented each time when a word is transferred.

Word Count = 0, then DMA checks whether there is any other req from peripheral device, then dma will process the req. If no req, DMA will hand over buses to CPU.

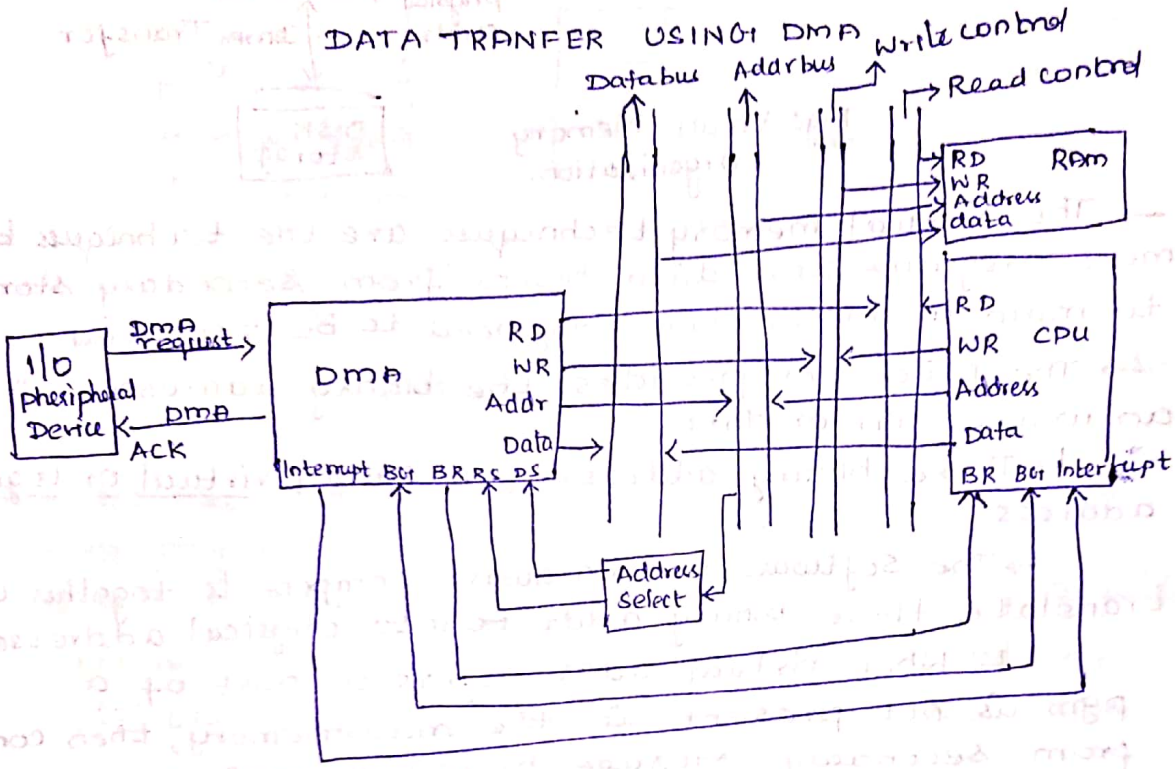
=> Count reg = 0, DMA stops & disables all the buses.

↳ it will send interrupt signal to CPU to take over all bus.

Application :-

1. ↳ used for regularly updating the display of an interactive terminal
2. ↳ used to transfer infor between magnetic disk and memory with high speed.

DATA TRANFER USING DMA



④ Explain Virtual Memory & steps involved in virtual memory address translation AP/may 15 Nov/Dec 16 AP/may 17 AP/may 18.

Virtual Memory :-

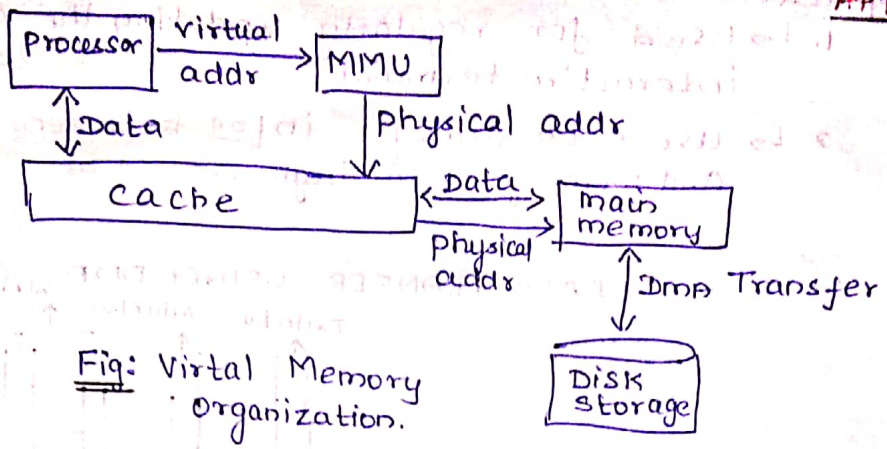


Fig: Virtal Memory organization.

→ The virtual memory techniques are the techniques that move programs and data blocks from secondary storage to main memory, when they need to be executed.

→ The processor provides the binary addresses for an instruction or data.

↳ These binary address are called virtual or logical address.

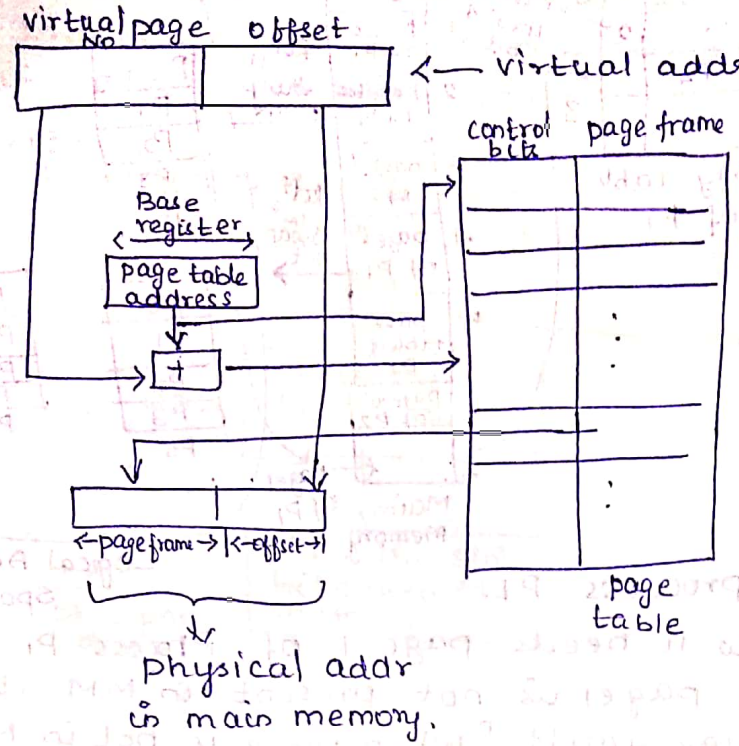
↳ The software and hardware components together will translate these binary addr to into physical addresses.

↳ When virtual addr refers to part of a pgm is not present in the mainmemory, then contents from secondary storage moved to main memory before they accessed. It is called swapping.

⇒ The MMU is Memory Management Unit which is responsible for translating given virtual addr to physical addr.

↳ The data is moved between main memory and disk by employing the DMA Scheme.

Address Translation In Virtual Memories :-



In VM, entire data is divided into fixed sized blocks referred as "pages".

→ The data shifts between Main mem & VM in the form of pages.

→ The VM generated by processor consists of 2 fields

(1) virtual page no → Gives the no of required page

(2) offset → refers to certain byte of data within a given page.

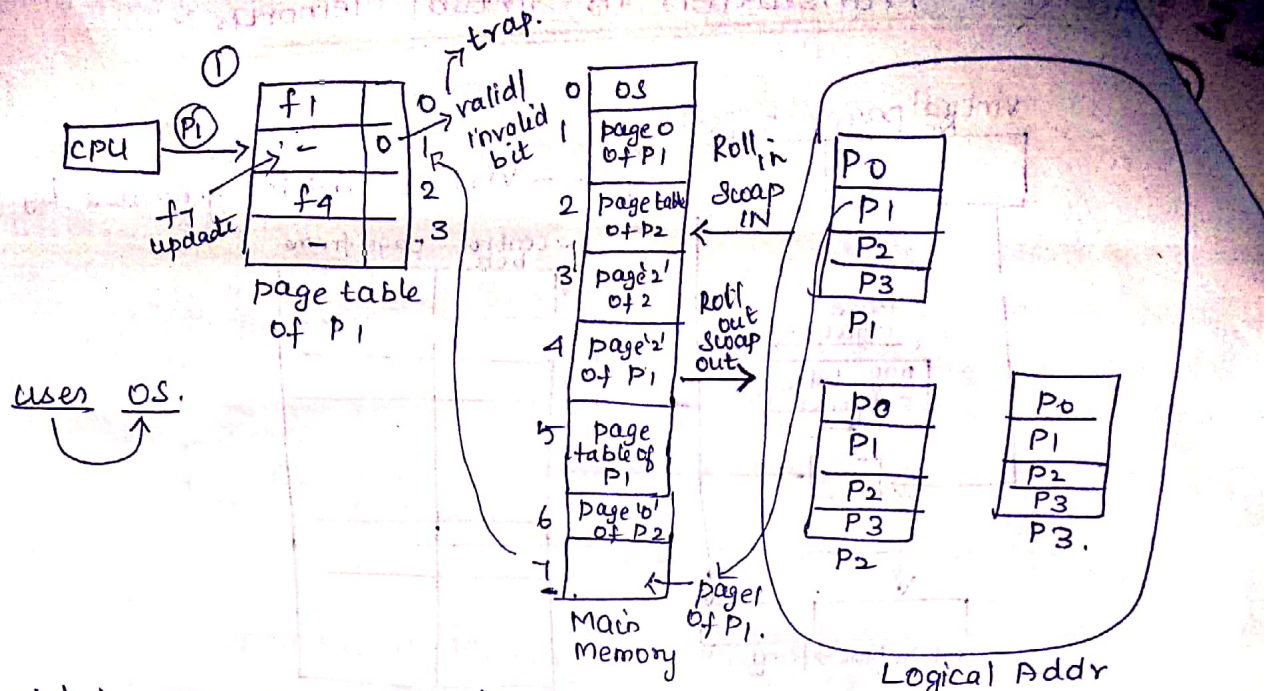
→ In VM translation process, there is a provision of a reg, which is referred as base register. → It holds the starting addr of a data structure called page table.

↳ The data structure of the page table stores the addr of the pages located in main memory.

↳ The page table contains certain control bits which gives the status of the page (i.e time of last modification, etc).

↳ There is another circuitary called a page frame which holds one page in the main memory

⇒ TO generate addr of page, the page no (provided by Processor) is combined along with the contents of base addr reg. This gives the required page entry present in the page table. In this way VM addr is translated to Main memory.



→ let CPU process P1.

↳ Now it needs page 1 of Process P1

↳ If page 1 is not present in MM. it is called as page fault [when page is not in MM when that page is request by CPU].

↳ Trap will generate. (i.e interrupt). Now control is transfered from user to OS (called as context switching)

↳ Now OS check whether user is authenticate or not. (for security purpose)

↳ Now If valid user, then OS checks the data in LAS. Now Page 1 from P1 is placed in MM if there is space in MM.

↳ page No 1 is present at frame 7.

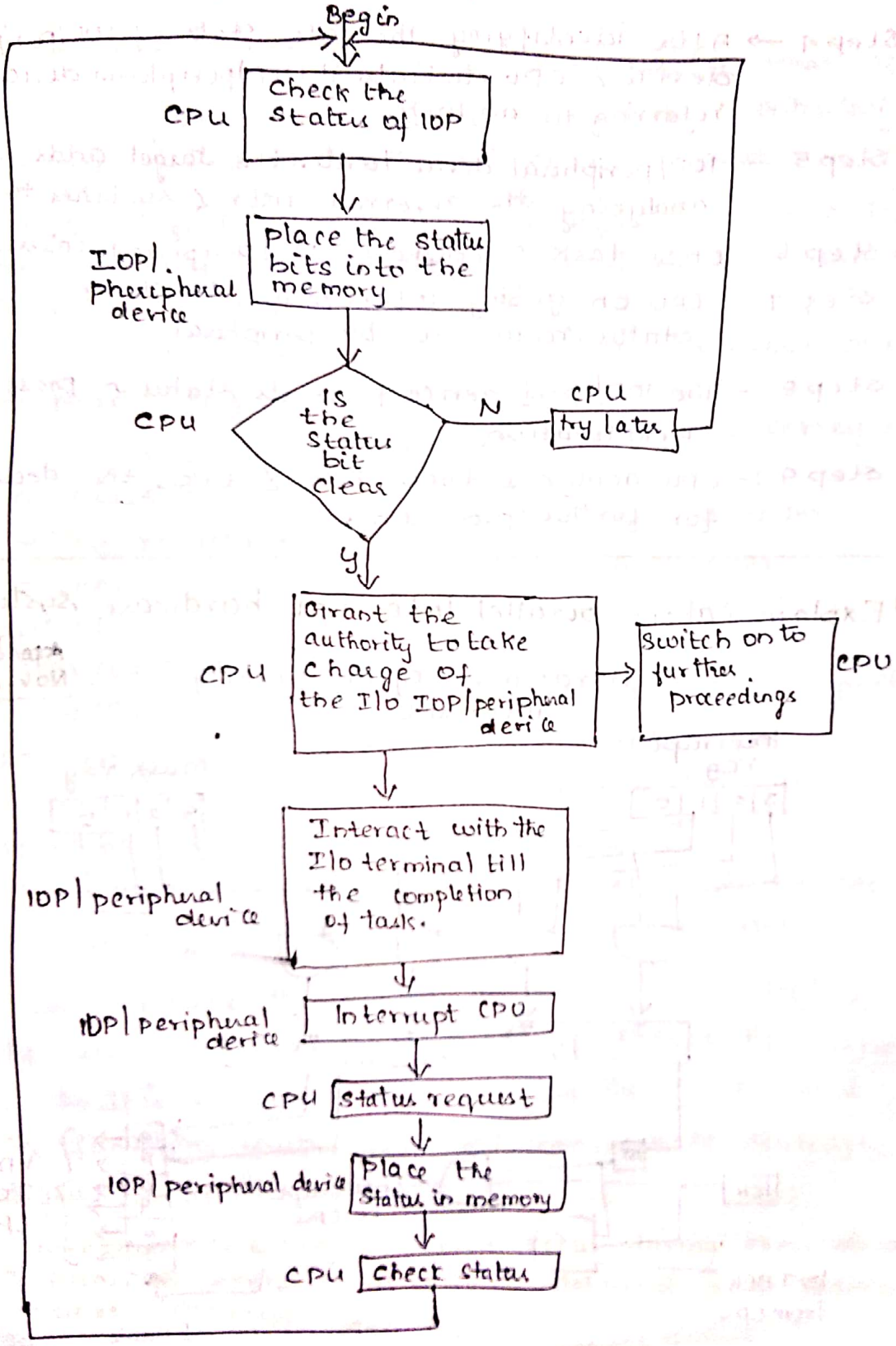
↳ Now ~~OS~~ control shift from OS to user.

---

5) Write the sequence of operation carried out by a processor when interrupted by a peripheral & explain about parallel priority interrup hlw.

Ans:- Fig:- Communication between Peripheral device & CPU

Ap/may-18
Nov/dec-16.



Step 1 → CPU issues a signal to check peripheral devi status

step 2 → IOP/pheripheral provides its status by placing a word in requi memory location.

Step 3 → CPU reads status bit

(i) If IOP/ peripheral device is busy, CPU resumes back

(ii) The IOP/ peripheral device is idle.

Step 4 → After identifying the idle state of IOP/peripheral device, CPU initiates it (IOP/peripheral device) by referring to an instr.

Step 5 → IOP/peripheral device locates the target addr by analyzing the referred instr & switches to given task

step 6 :- once task completed, IOP/peripheral. interrupts CPU.

step 7 :- CPU on geting interrupted sends the status request to IOP/ peripheral.

step 8 :- The IOP/ peri device places its status in prescribed mem location

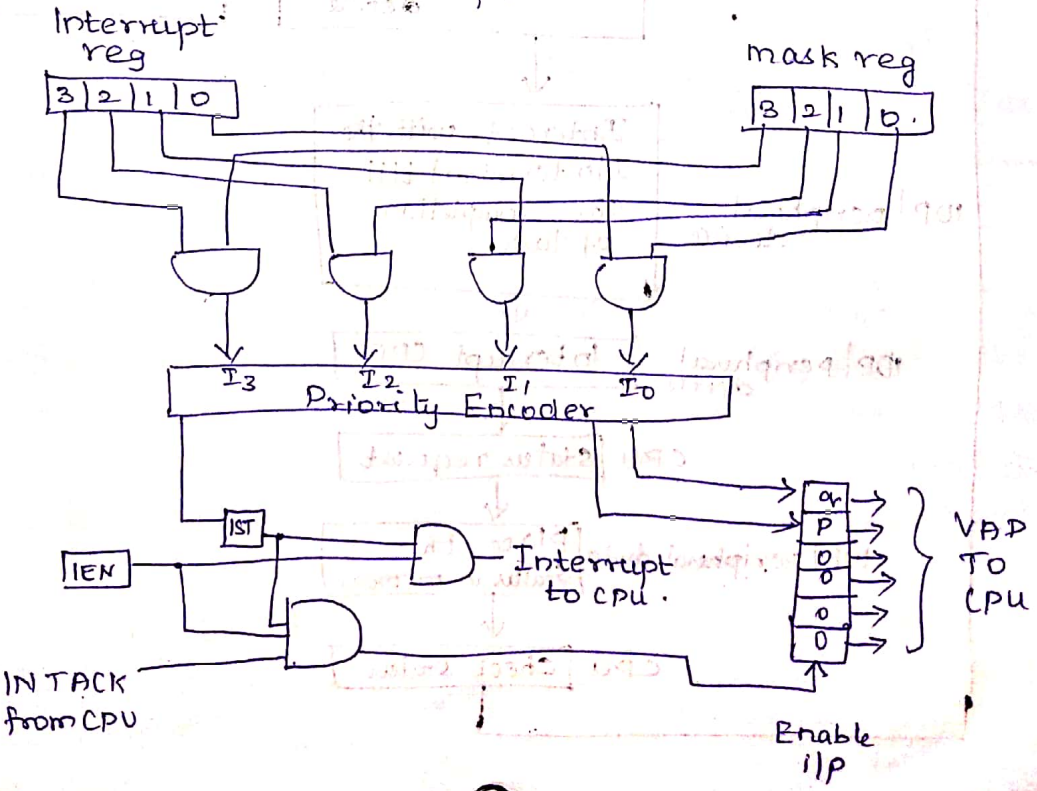
step 9 :- CPU analyzes these bits & takes the decision for further proceedings.

---

(6) Explain about parallel Interrupt hardware system.

Nov-dec 16.

Hardware of the priority Interrupt



⇒ Parallel Priority Interrupt mechanism ; have two register

(1) Interrupt register

↳ Stores the interrupt signals of different devices as bits whose position denotes priority level

(2) Mask register :

↳ hold the same no of bits as that of interrupt register bit is responsible for controlling the status of each of the interrupt request.

⇒ Let disk, printer, reader & keyboard are 4 devices whose interrup signals are stored in interrupt reg with value 0, 1, 2, & 3 respectively.

↳ These value set based on external condition

→ Mask reg also holds same value.

→ The bits of Interrupt reg ANDed with corresponding bit of mask register.

→ The resultant o/p is i/p to priority encoder.

→ There are 3 o/p's from priority encoder. (two of which first form vector address and forwarded to CPU.

→ The third o/p sets the interrup status flip-flop 'IST' only when unmasked interrupt occurs.

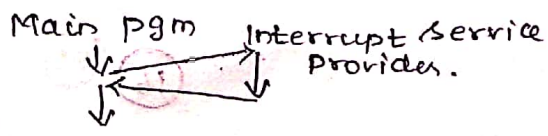
→ The IEN [Interrupt Enable flip flop] is responsible for controlling interrupt system.

→ The interupt to the CPU is produced by AND in the o/p's of IST & IEN flip flops.

→ To place the vector addr into the data bus, the bus buffers of the o/p registers must be enable.

↳ This is done by ANDing the o/p of IST & IEN flip flops with INTACK [Interrupt Acknowledge signal] from the CPU

INTERRUPT :- whenever event occur during execution of a pgm, the execution of pgm is delayed. Such condition indicates Interrupt



(7) Explain I|o communication technique / Explain Different Modes of data transfer.

Ans:- There are 3 different types of I|o communication or 3 modes of data transfer.

1. Programmed I|o
2. Interrupt driven I|o
3. Direct Memory Access (DMA).

Programmed I|o :-

→ It is a method that manages I|o activity so that the processor can grant access rights to a special module called I|o module.

→ This module is responsible for processing all I|o activities. & proride information to ~~user~~ processor.

⇒ In programmed I|o, whenever the processor has to perform certain I|o activity, it calls its respective I|o module & assign it with a task.

↳ The I|o module will execute the I|o activity based on accessibility of processor.

↳ The task of processor also collects data from main memory & proride it to the desired I|o module.

Adv :-

1. I|o command is issued by the processor to I|o module
2. Requested I|o instruction is executed at earliest.
3. I|o module switches to next task once complete it assigns task.
4. It is inexpensive & requiress less h|w.

Disadr :-

1. The processor has to wait until the I|o module is ready for performing data transfer.
2. computation overhead occur.
3. Performace decrease as it takes time to execute process.

(2) Interrupt Initiated Ilo:

→ In the programmed Ilo method the CPU stays in the pgm loop until the Ilo unit indicates that it is ready for data transfer.

→ This is time consuming process because it keeps the processor busy needlessly.

→ This problem can be overcome by Interrupt Initiated Ilo.

↳ In this when the interface determines that the peripheral is ready for data transfer, it generates an interrupt.

↳ After receiving interrupt signal, the CPU stops the task which it is processing & Service the Ilo transfer & then returns back to its previous processing task.

Adv:-

(*) Ilo command is issued by processor to the respective Ilo module.

(*) Processor need not wait for completion of Ilo module i.e mean while it performs other task

Disadv:-

(*) Processor intervention required every time.

(*) The rate of Ilo transfer is restricted by functioning speed of the processor.

(*) More processor time is consumed.

(3) Direct Memory Access (DMA)

=> Removing the CPU from the path and letting pheripheral device manage memory buses directly would improve speed of transfer.

This technique is known as DMA.

→ A DMA controller manages to transfer data between peripherals & memory unit.

→ Many hlw system uses DMA such as disk drive controller, graphics cards, sound cards etc.

→ In DMA, CPU would initiate the transfer, do other operation while transfer is in progress & and receive an interrupt from DMA controller when the transfer is completed.

Adv :-

→ Data in large volume can be moved by utilizing system bus.

→ DMA module transfer entire block to & from the memory & informs the processor.

Disadv :-

→ For transfering data, DMA Module need the total control of the system bus.

→ Processor must wait when it urgently needs the system bus.

→ Processor cannot indulge in other task while waiting for bus.

X — X — X.