VIRTUAL MEMORY

→ Virtual memory is a concept used in some large computer system that permits the user to construct programs as though a large memory space were available, which is equal to locality of auxillary memory.

→ Virtual memory is used to give programmer the illusion that they have a very large memory at their disposal even though the computer system has a relatively small memory.

→ A virtual memory is a computer system technique which gives an application program the impression that it has contiguous working area or address space. But in actual the address space may be fragmented & may even overflow onto disk storage. Hence virtual memory is more than just using disk space to extend physical memory or main memory size.

Role of virtual memory in multitasking →

1. Each process has its own address space, thereby not required to be relocated or relative addressing mode is not required.

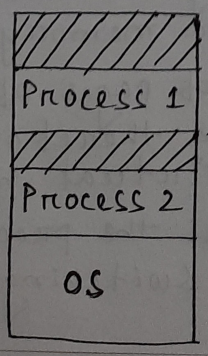
2. Each process require one contiguous block of free memory upon launch.

Need of virtual Memory →

A virtual memory is used because there is shortage storage of memory.

→ Each application program creates a process & these processes may be too big for physical memory. So a efficient memory management is required to hold or store active process.

→ Each processes have their own contiguous virtual or logical address group called address space.



→ contiguous virtual or logical address group called address space.

→ Unused space.

Structure of virtual Memory →

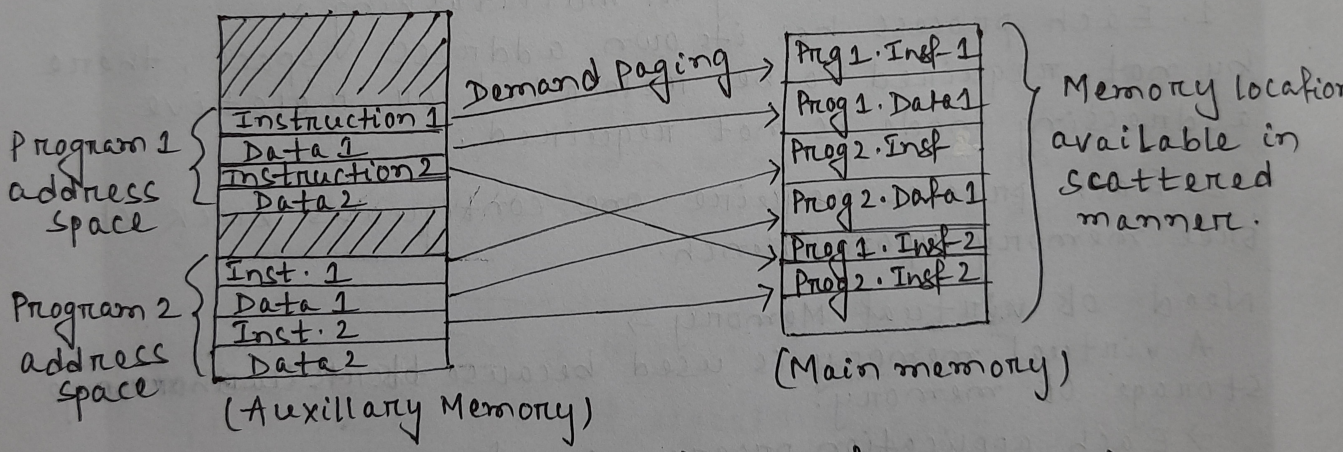
→ An address used to identify the location of virtual memory is called virtual address & the set of such addresses is called address space.

→ An address in main memory is called physical address & the set of such addresses is called memory space.

→ Address space is generated by program as they reference instructions & data. Hence the processor provides reference to the virtual address register (VAR).

→ The contiguous virtual memory address or address space of an application program is divided into pages of equal or same size.

→ In multiprogramming system each page of a process is brought into main memory from virtual memory only when it is needed by the processor & this concept is called demand paging.



→ If the program generates a reference that is not present in main memory then page fault occur. Or if the searched page is not found in main memory then page fault occur. So the operating system is then instructed by processor to bring the desired page from auxillary memory to main memory.

→ At any one time, only a few no. of pages of any process can be present in main memory & therefore more no. of process can be maintained in virtual memory.

→ The operating system (OS) manages the processes or the OS manages continuously the switching among processes.

→ operating system manages process specially when one process waits for resources. e.g. retrieval of page from disk in case of page fault.

→ when a page is brought into main memory in case of page fault & the main memory size is full then the new incoming page from virtual memory replace a existing page from main memory block by using the replacement algorithms. Several types of replacement algorithms are,

1. FIFO
2. LFU
3. LRU
4. Optimal
5. Random

1. FIFO - Replace that page from main memory that has been in the main memory for longest time.

2. LFU - Replace that page from main memory that has least or fewest references. To know how many references each page have, a counter is set for each page.

3. LRU - Replace that page from main memory block that has been in the main memory for longest time with no or least references to it. It gives best hit ratio.

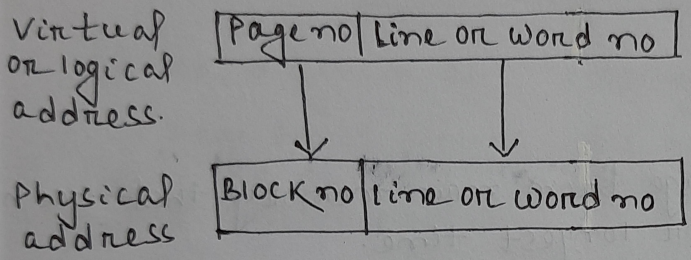
4. Random - This algorithm pick up any page from main-memory randomly & replace it.

5. Optimal - An optimal algorithm replace that page from main-memory that has least frequency of reference in near future. An optimal algorithm gives also best hit ratio.

→ If a page is thrown out just before it is used then it leads to increase in percentage of page fault. If such a phenomena repeats frequently then it leads to thrashing problem.

→ Address mapping -

Each reference generated by processor goes through address mapping process where the virtual address generated by processor get translated to the corresponding physical address. This address mapping is performed dynamically while executing program.



a) Page number is used to identify a particular page in virtual memory.

b) Block number is used to identify a block in main memory.

c) line or word no is used to identify a word in the specified block or page.

Though page size = block size, so line or word no. in the virtual address is same as the line or word no in the physical address.

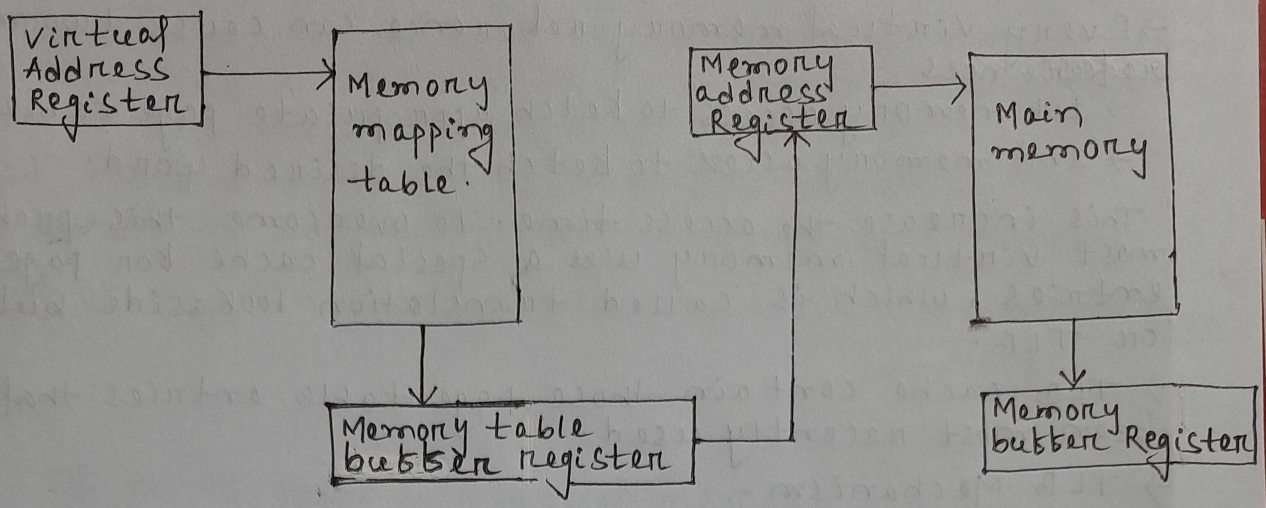
→ The operating system maintains page table for each process. The page table shows block or frame or page frame location for each page of the process in main memory.

The page table can be placed or created either in the same main memory or it may need one separate memory. Therefore the access time of processor depends on 2 factor,

a) access time to access page table.

b) access time for retrieval of word of page from main memory.

But the access time of page table stored in main memory is very less than that of the separate memory page table.



(Block diagram of memory page table mapping)

→ When page fault occurs, the execution of present program is suspended until it attempts to brought the required page into main memory.

→ Loading a page from auxillary memory to main memory is an IO operation. So the operating system assign this task to IO processor. On the mean time control is transfer to the next program in the main memory that is waiting to be processed by CPU.

→ When the main memory block has been assigned to a page & the transfer is completed then the original program can resume its operation.

structure of page table →

→ In most system page table is present in main memory. The size of page table is of variable length because the processes or application programs are of variable length.

→ There is a page table for each process. When a perticular process is running, a register holds the starting address of page table for that process.

→ As the size of the process may large, so as the size of page table. To overcome the problem of storing page table of bigger size, virtual memory is used rather than main memory.

→ Some processors make use of two-level scheme to organize large page table. In this scheme there is a page directory in which each entry points to a page table.

Translation Lookaside Buffer → (TLB)

→ Every virtual memory reference can cause two memory access.
  → 1st memory access to fetch appropriate page table.
  → 2nd memory access to fetch the desired word.

This increase the access time. To overcome this problem, most virtual memory uses a special cache for page table entries, which is called translation lookaside buffer or TLB.

→ TLB cache contain those page table entries that have been most recently used.

→ TLB Mechanism -

  → when a virtual address is generated, CPU initially checks TLB. If the page table entry is in TLB, then CPU generates physical address. But if the page table entry is not in TLB, then the page table is search-ed in next & the corresponding physical address is generated.

  → If the required page is in main memory block then update TLB. If the required page is not present in main memory then operating system instructs CPU to read that page from disk & as it is an IO operation so the IO processor takes care of it. Now transfer page from disk to main memory. Now update the page table for the new incoming page & then update the TLB also.

  → If during page transfer the main memory size is FULL then use replacement algorithm to replace any one existing page from main memory block by the new incoming page.

Address Mapping using page table →

Consider a virtual memory of size 8 KB, main memory of size 4 KB & page size of 1 KB. Therefore block size is also of 1 KB because block size = page size.

∴ virtual memory size = 8 KB = $2^3 \times 2^{10} = 2^{13}$

⇒ No. of addressing bits for virtual memory = 13 bits.

No. of pages in virtual memory = $\frac{\text{virtual memory size}}{\text{page size}}$

$= \frac{8\text{ KB}}{1\text{ KB}} = 8$

∴ no. of pages in virtual memory = $8 = 2^3$

⇒ Page no = Page identifier = 3 bits

Similarly,

∴ main memory size = 4 KB = $2^2 \times 2^{10} = 2^{12}$

⇒ No. of addressing bits for main memory = 12 bits

No. of blocks in main memory = $\frac{\text{Main memory size}}{\text{Block size}}$

$= \frac{4\text{ KB}}{1\text{ KB}} = 4$

∴ no. of blocks in main memory = $4 = 2^2$

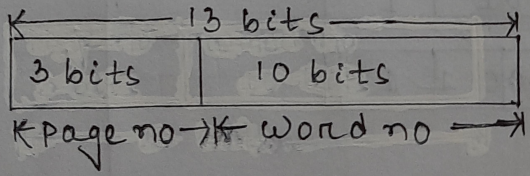
⇒ no. of bits for block identifier = block no = 2 bits.

The size of page or block = 1 KB = 2^{10}

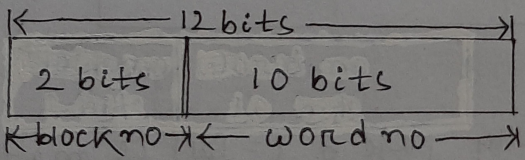
⇒ No. of words in a page or block = 2^{10}

⇒ word identifier = word no = line no = 10 bits.

virtual address



physical address



There are two different types of address mapping in virtual memory. These are -

1. address mapping using memory page table.
2. address mapping using associative memory page table.

a) Address mapping using memory page table →

→ The memory page table contain two different column. These are - block no & presence bit. Or each entry in Page table contain two different values - block no & Presence bit.

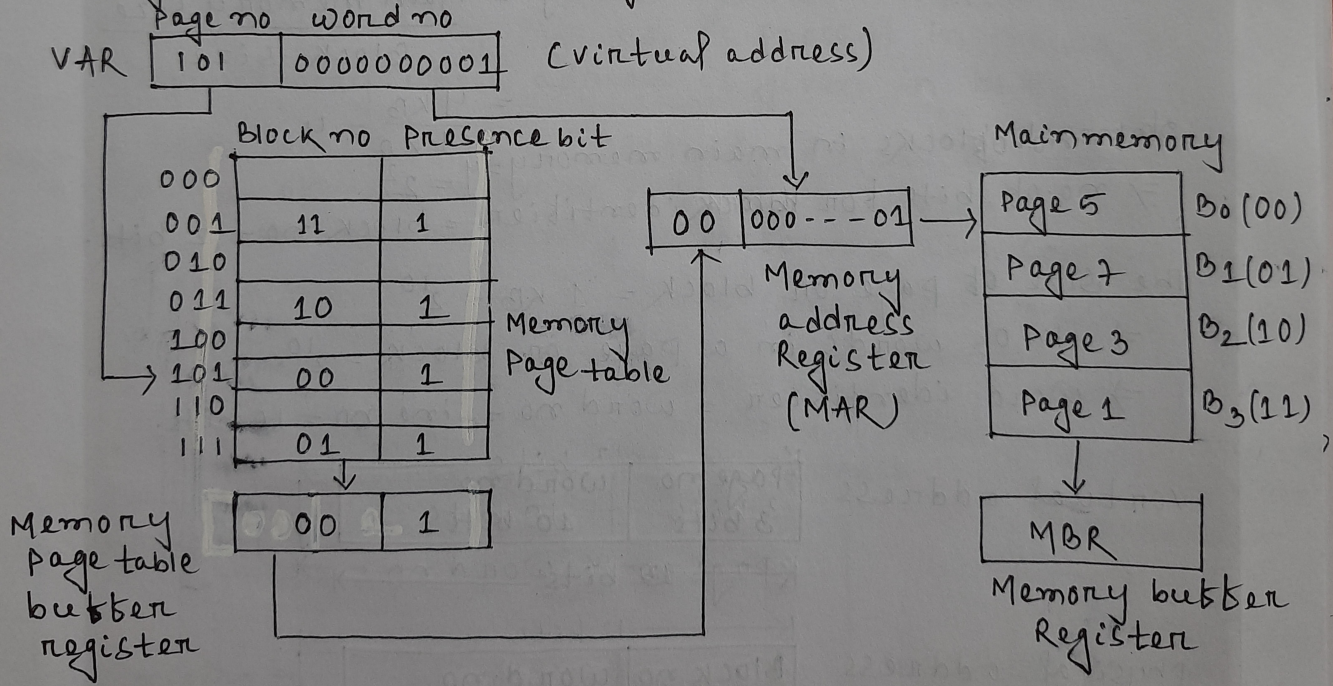
→ The block no specifies the block within which the page is present in main memory & the presence bit is used to show whether the page has been transferred from auxillary memory to main memory or not. Or, the presence bit shows whether a page is present in main memory block or not.

Presence bit

= 0 : Page is not available in main memory.

= 1 : Page is available in main memory.

→ No. of entries in a page table of memory is equals to no. of pages in virtual memory for the process.



→ Initially page 5 (= 101) is matched to address of memory page table. The content of matched address at 101 is read which is [00 | 1]. This means page no 5 is present in block no. 0 (B0). Now the word no, of the virtual address 0000000001 & block no 0 (=00) are transferred to MAR and the corresponding word at location 0000000001 of block number 00 is read from

main memory & transferred to MBR.

b) Address mapping using associative memory page table →

The major drawback of memory page table is that it is inefficient with respect to storage utilization. i.e: in memory page table no. of entries that are utilized is equals to no. of blocks in main memory.

→ Due to the above reason an efficient way of organizing page table is to put no. of entries or words in a page table is equals to no. of blocks in main memory

→ By doing this size of memory requirement for page table is reduced. This method is implemented by using associative memory.

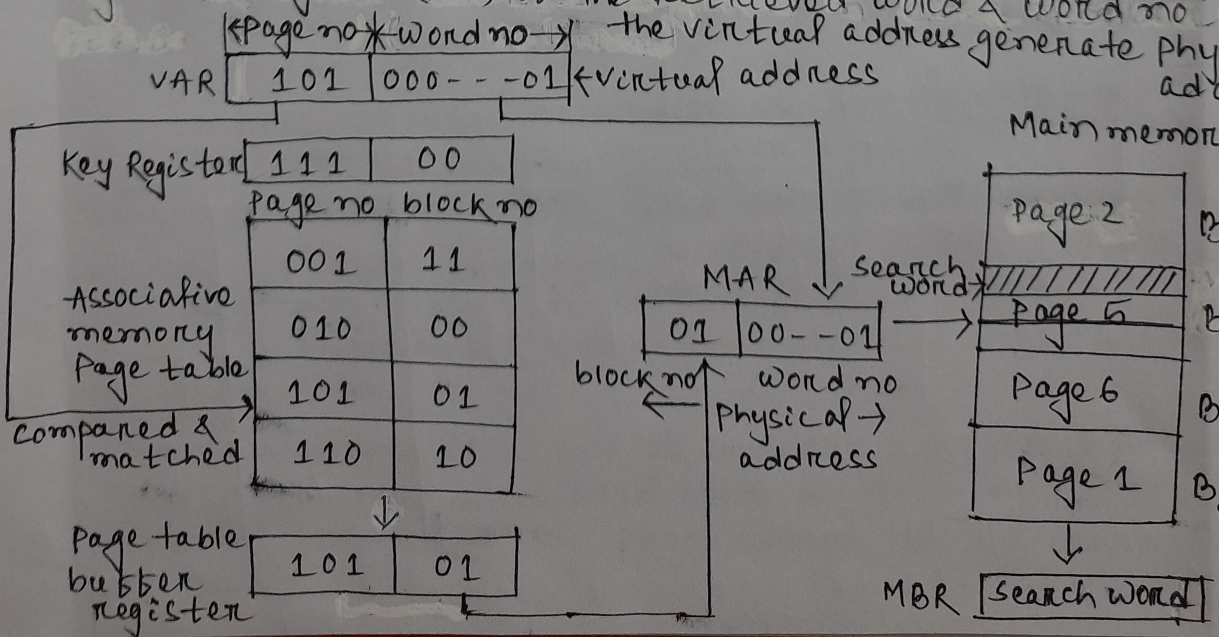
→ Each word of associative memory page table contains 2 different fields – page number & block number.

word of page table →	Page no	Block no.

Each word or entry of page table indicates that the page identified by page no. is present in the block of main memory whose address is given in block no field.

→ Page no. present in virtual address is compared with page no. of each word or entries of associative memory page table. Here matching is performed with content or word

If a match is found after comparison then the word for which match has found is read from page table. And by combing block no. of the retrieved word & word no of the virtual address generate physical address



→ Now by using physical address give reference to the main memory to read the search word & place it in the MBR.