Cache size and associativity are not independent in determining cache performance.

VIRTUAL MEMORY:

If a computer lacks the random access memory [RAM] needed to run a program or operation, because of its limited space, this is where virtual memory comes in; ie OS uses the virtual memory to compensate.

Virtual memory combines computer's RAM with temporary space on the hard disk. When RAM runs low, virtual memory moves the data from RAM to a space called paging file. Moving the data frees up RAM so computer can completes its work

The more RAM a computer has the faster the program will generally run. Lack of RAM slowdown computer, tempted to increase virtual memory to compensate.

Computer can read data from RAM much more quickly than from a hard disk So adding RAM is a better solution.

The technique that uses main memory as a cache for secondary storage is called virtual memory.

Major motivations for virtual memory.
1) To allow efficient and safe sharing of memory among multiple pgms.
2) To remove the programming burdens of a small, limited amount of main memory.

Date

Although the concepts at work in virtual memory and in cache are the same, their differing historical roots have led to use of different terminology.

page: A virtual memory block is called pag

page fault: A virtual memory miss

Virtual address: An address that corresponds to a location in virtual space and is translated by address mapping to a physical address when memory is accessed.

Address mapping / Address Translation: The process by which a virtual address is mapped to an address used to access the memory.

The binary addresses that the processor issues either instruction or data are called virtual / logical addresses. These addresses are translated into the physical addresses by a combination of hardware and s/w components. If virtual address refers to a part of the program or data space ie currently

Date

in the physical memory then the contents of the appropriate location in the memory are access immediately. On the other hand, if the referenced address is not in the main memory its contents must be brought into a suitable location in the memory before they can be used.

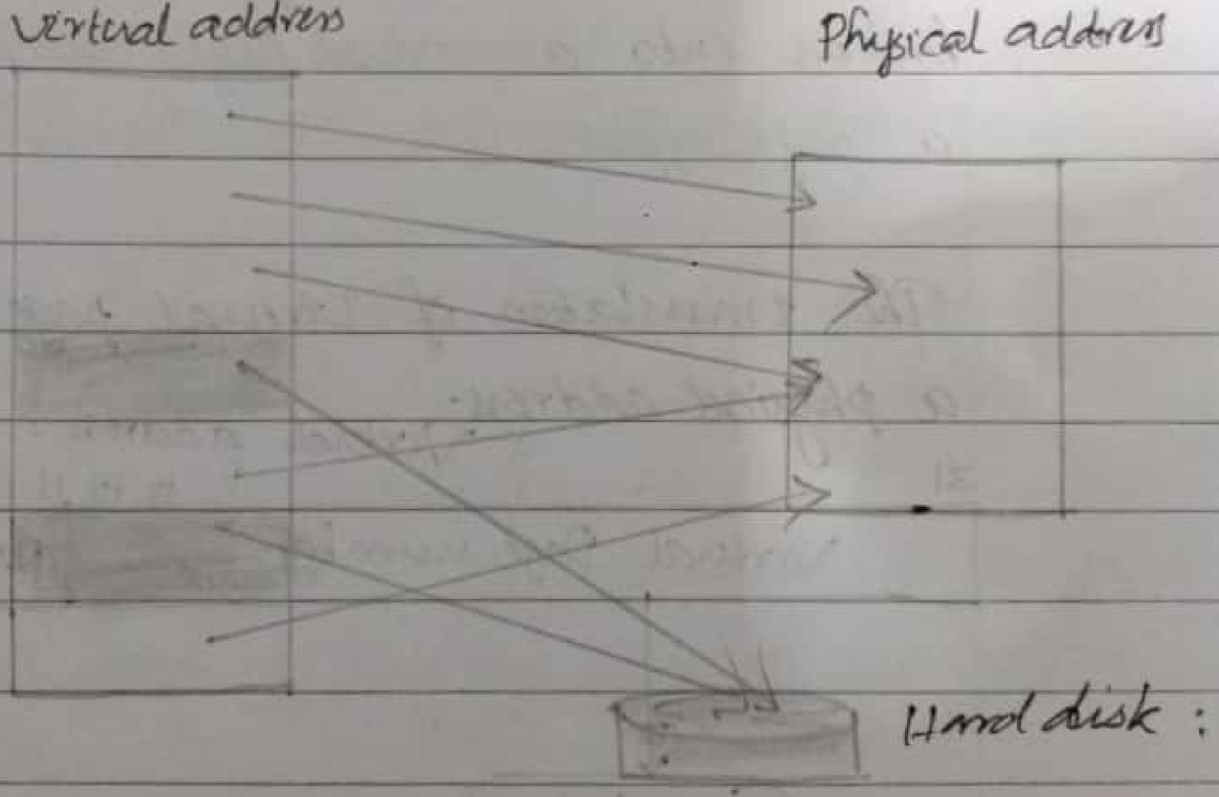


Fig. In virtual memory, blocks of memory (pages) are mapped from one set of addresses (called virtual addresses) to another set [called physical addresses).

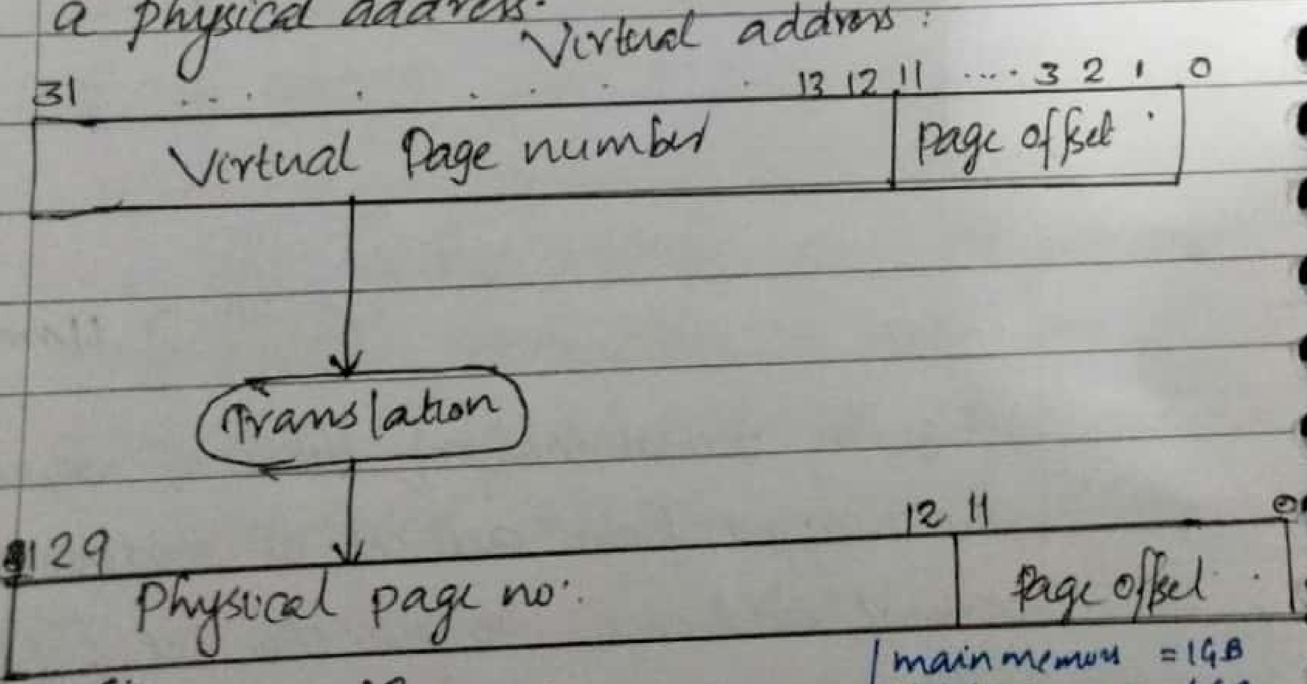
2^{10} - 1KB
2^{20} - 1MB
2^{30} - 1GB

Date

Physical pages can be shared by having two virtual addresses point to the same physical address. This capability is used to allow 2 different programs to share data or code.

In virtual memory, the address is broken into a virtual page number and a page offset.

The translation of virtual page number to a physical address.



In this page size = 2^{12} = 4KB.

main memory = 1GB
virtual memory = 4GB.

No. of pages allowed in memory = 2^{18} ∵ physical page no has 18 bits

" " " virtual addr = 2^{20} = ~~4GB~~

Date

The physical page number constitutes the upper portion of the physical address, while the page offset which is not changed, constitutes the lower portion.

→ The number of bits in the page offset field determines the page size.

→ The no: of pages addressable with virtual address need not match the no: of pages addressable with the physical address.

Paging uses fixed sized blocks. There is also a variable size block scheme called Segmentation.

In segmentation, an address consits of two parts: a segment number & a segment offset.

In virtual memory system, we locate pages by using a table that indexes the memory; this structure is called a page table, and it resides in memory.

A page table is indexed with

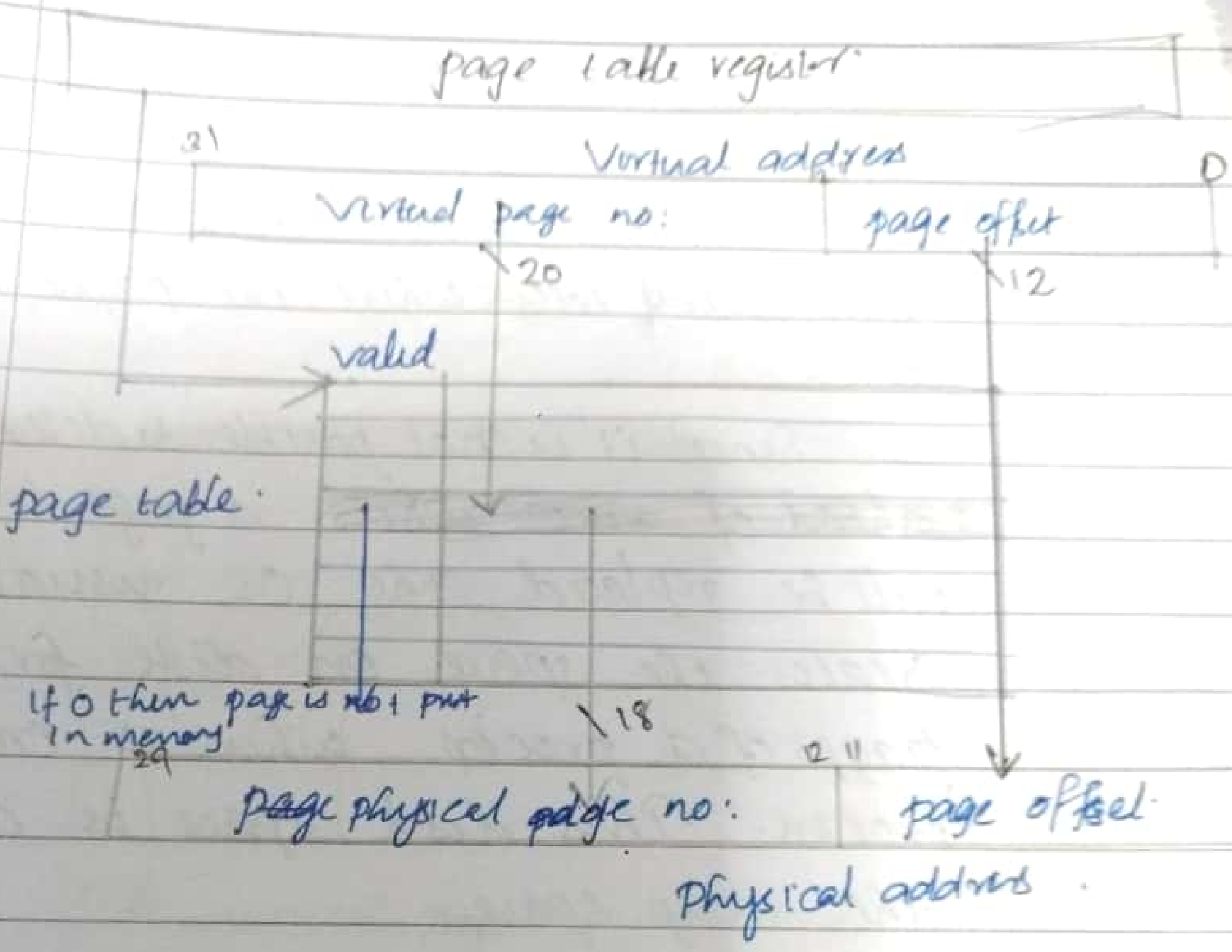
→

the page number from the virtual address to discover the corresponding physical page number.
Each program has its own page table which maps the virtual address space of that program to main memory.

The page table, together with the program counter and the registers, specifies the state of a program.

If valid bit is off (0) the page is not present in main mly and page fault occurs. If the bit is on(1) the page is in memory and the entry contains the physical page number.

In fig: Starting address of page table is given by page table pointer.



page size $= 2^{12}$ Bytes $= 4$ kB

Virtual space $= 2^{32}$ B $= 4$ GB

Physical address space $= 2^{30}$ B $= 1$ GB

No: of entries in the page table $= 2^{20}$

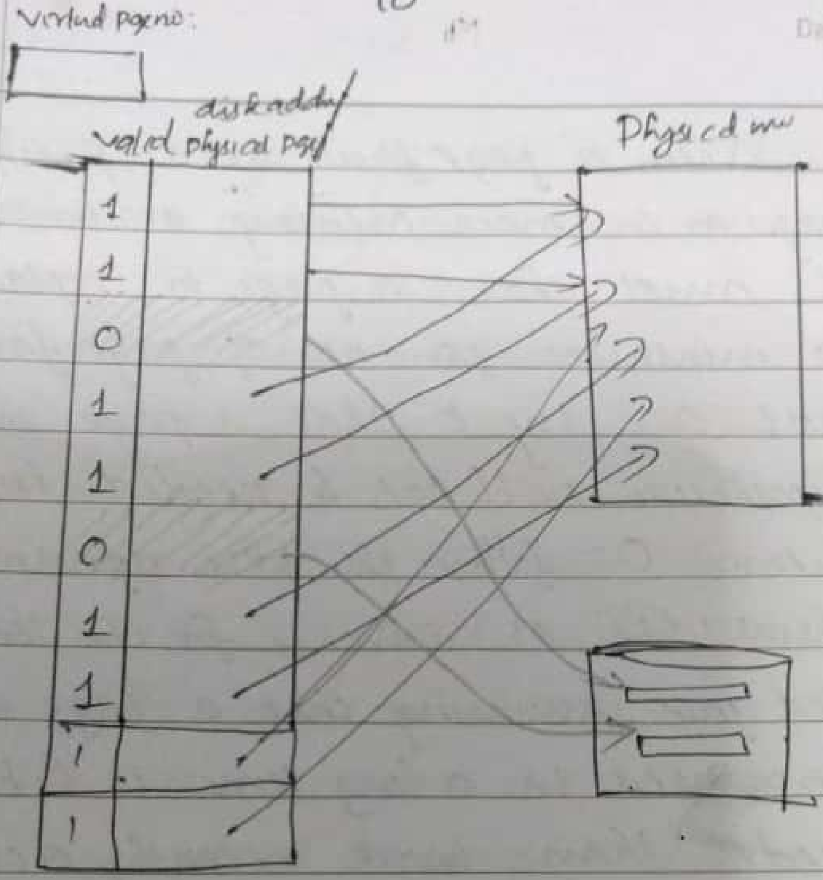
Date

Page faults :

The virtual address alone doesnot immediately tells where the pages is on disk.

Since it is not possible to determine ahead of time when a page in memory will be replaced, the OS usually creates the space on disk for all the pages of a process when it creates the process. This disk space is called the swap space.

It also creates a data structure to record where each virtual page is stored on disk. This data structure may be a part of page table or may be auxiliary data structure indexed in the same way as the page table.



The virtual page no: is used to index the page table. If the valid bit is on, the page table supplies the physical page no. [starting address of page in m/y] corresponds to virtual page. If valid bit is off, the page currently resides only on disk, at a specific disk address. In many s/m the table of physical page address & disk page address, ~~to~~, while logically one table, is stored in 2 separate data s/m.

. When a page fault occurs, if all the pages ar in main memory are in use, the OS must choose a page to replace. Becaus To minimize the no. of page faults, the most OS try to chose a page that they hypothesize will not be needed in the near future. OS follow the LRU replacement scheme. The OS searches for the least recently used page, assuming that a page that has not been used in a long time is likely to be needed than more recently accen page. The replaced pages are written to swap or space on the disk.

To help the operating system estimate the LRU pages, some computers provide a reference bit or use bit which is set whenever a page is accessed. The operating system periodically clears the reference bits and later records them so it can be determine which pages were touched during a particular time period. With this usage info, the os can select a page that is among the least recently referenced.

Virtual Memory Writes:

Virtual mly s/m must use write back, performing individual writes into the pages in memory, & copying the page back to the disk when it is replaced in the memory.

To track whether a page has been written since it was read into the memory, dirty bit is added to the page table. The dirty bit is set when any word in a page is written. If the operating s/m chooses to replace the page, dirty bit indicates whether the page needs to be written out before its location in memory can be given to another page. Hence modified page is often called a dirty page.

Date

TLB [Translation look aside buffer]

Since page tables are stored in main memory, every mly access by a program can take at least twice as long

a) One mly access to obtain the physical address.

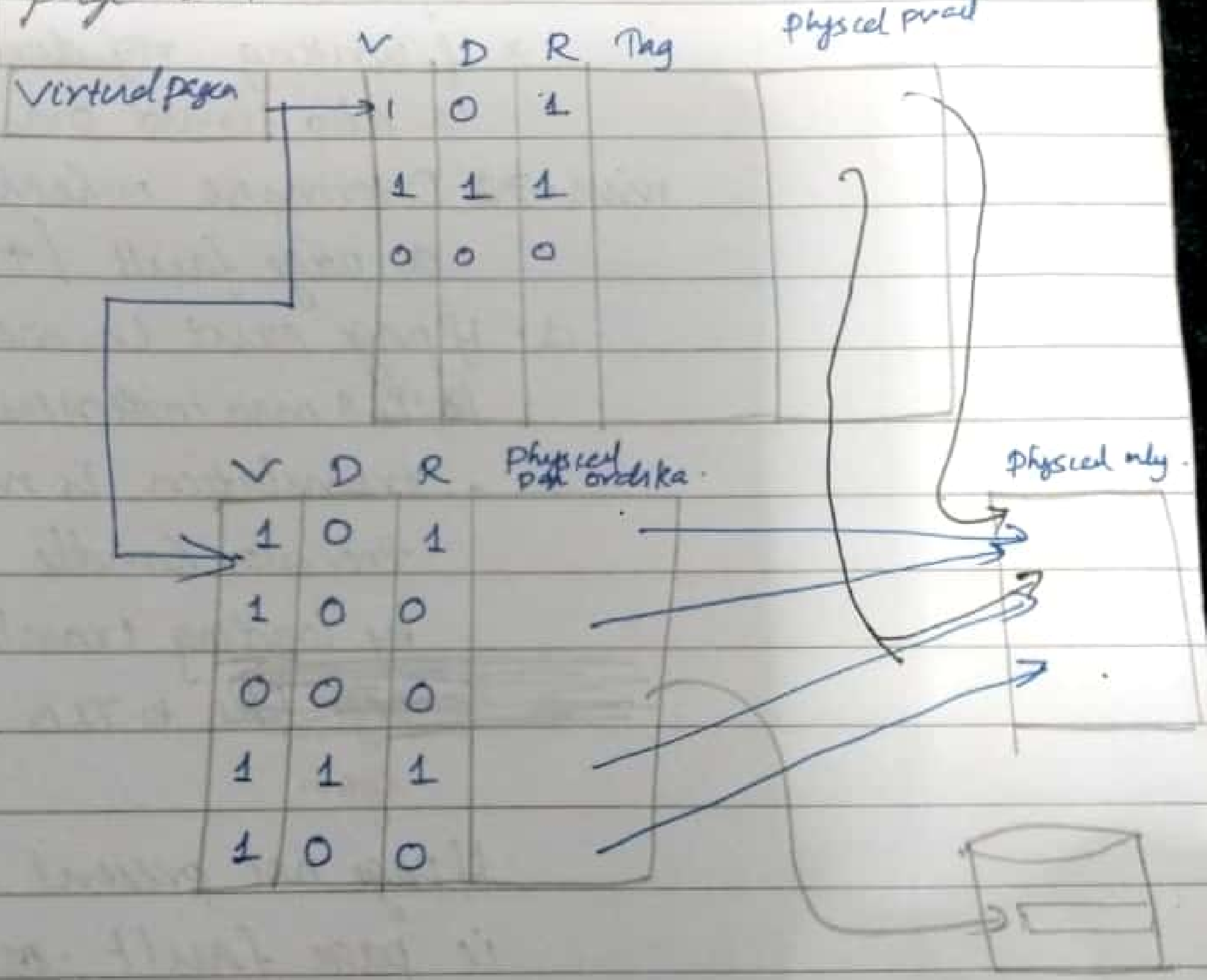
b) Second access to get the data.

The key to improve access performance is to relay on locality of references to the page table. ~~When~~

When a translation for a virtual page number is used, it will probably be needed again in the near fure because the references to words on the page have both temporal spatial locality.

For that modern processors include a special cache that keeps track of recently used translation. The special address translation cache is traditionally reffred to as a TLB, although it would be more

accurate to call it a translation cache.
Fig: shows that each tag entry in the TLB holds a portion of virtual page no: & each data entry of the TLB holds physical page no:



TLB is access instead of page table on every references, the TLB will need to be include other status bits such as dirty & reference bits.

Date

On every references, look up virtual page no: in TLB ~~if~~ get a hit → ~~x~~ physical page no: is used to form address

* Reference bit is turned on.

* If writing ~~ref~~ dirty bit is also turned on

miss → ~~1~~ Determine whether it is a page fault / TLB miss.

~~1~~. If page exist in memory, ie TLB miss indicates only that translation is missing, Processor handle TLB miss by loading translation from page table to TLB then try ref again

If page not present in memory ie page fault, ~~os~~ invokes OS using exception.

Becase the references & dirty bits are contained in th TLB entry, we need to copy these bits back to the page table entry when we replace an entry. These bits are the only portion of TLB that can be changed.