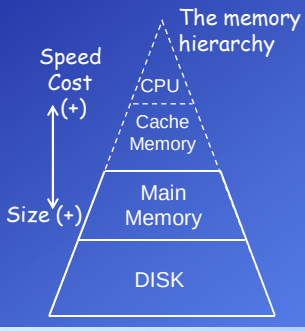
9 Memory Management, Virtual Memory

Purpose:

- Providing a continuous (linear) address space to users/processes that is larger than the physical memory.
  Each process gets the illusion that the system has a separate, continuous, large memory, dedicated only to itself and no other processes are executing or consuming resources.
- Providing security/protection on memory blocks.
- Sharing program and data in multiuser /multiprogram systems.



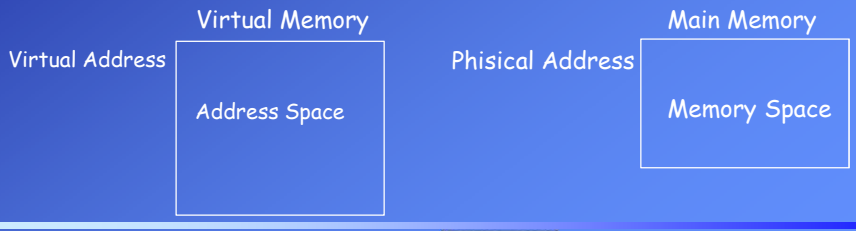
The concept of virtual memory is in principle similar to that of the cache memory.
A relation similar to cache and main memory is established between main memory and the disk. (Locality!)

Using virtual memory, your computer addresses more memory than it actually has, and it uses the hard disk to hold the excess.

Programs and data are stored in disk; the necessary data are brought to the main memory.

Addresses and memories:

- Programs are written and addresses are generated according the virtual memory (linear and large).
- The addresses generated in programs are called logical address or **virtual address**.
  In the systems with the virtual memory the CPU generates virtual addresses.
- The set of virtual addresses (the space of the virtual memory) is called *virtual address space* or shortly *address space*.
- The real address of the main memory is called **physical address**.
- The space of the main memory is called *memory space*.

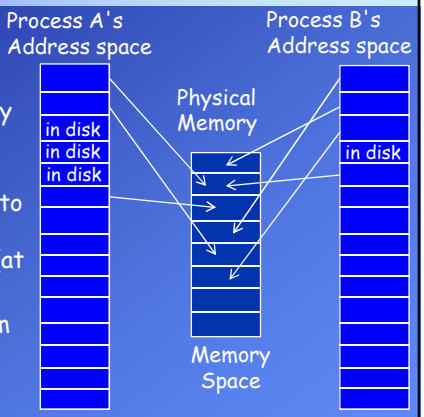


Addresses and memories: (cont'd)

Portions of the process address space are scattered about physical memory and are likely to be not contiguous at all (unused data are in disk).

The operating system maintains a map (table) to determine the real location of data in virtual space; it can be in disk or in physical memory (at which address).

The process is unaware of where in the system any particular portion of its address space is being held (disk / memory).



Each running process generates addresses as if it has the entire machine to itself; as if the computer offers an extremely large and linear memory.

One of the main **advantages**:

The job of the programmer and compiler is simplified, because no details of the hardware or memory organization are necessary to build a program.

The program can be compiled independently from the properties of the physical memory.

9.1 Paged Mapping:

To benefit from the spatial locality and to transfer data using a DMAC (IOP) movement of data between the disk and the main memory takes the form of pages.

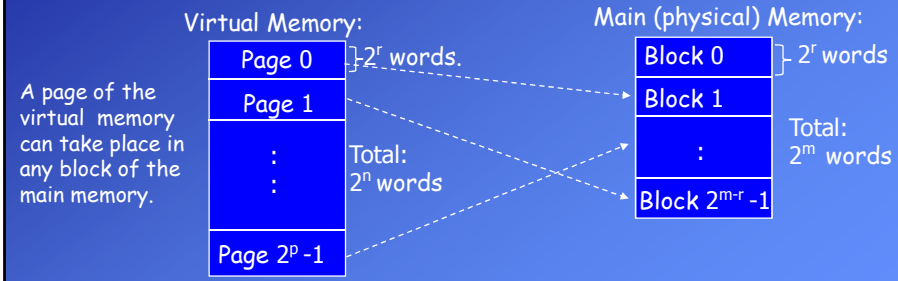
Virtual memory (address space) is divided into fixed size *pages*.

Main memory is divided into blocks (*page frame*) with the same size of pages.

Page size: 2^r

Address space: 2^n Number of Pages: 2^{n-r}, p = n-r

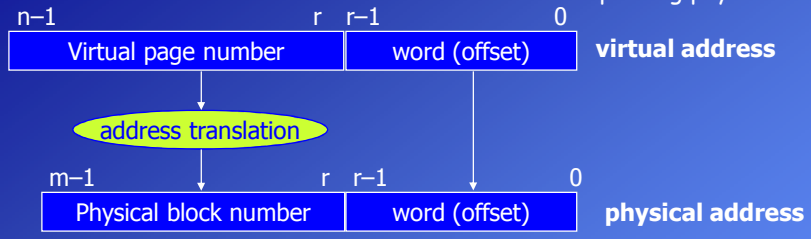
Memory space: 2^m



Virtual memory $2^{(n-r)} = 2^p$ pages.

Main memory $2^{(m-r)}$ blocks

When the CPU generates a virtual address the memory management unit (MMU) is responsible to translate the virtual addresses to its corresponding physical address.



The information about which page is currently residing in which memory block is kept in a **page table (PT)**.

Page table is stored in the main memory and each process has its own table.

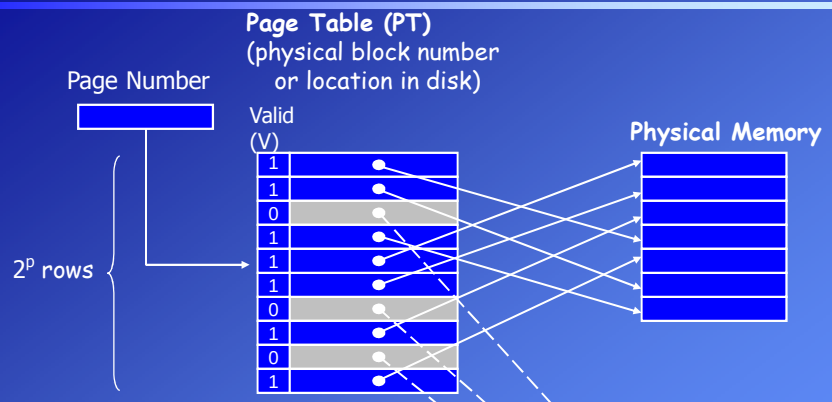
If the referenced page is not in the main memory *page fault* occurs.

The operating systems blocks the current process and configures the DMAC to transfer the referenced page from the disk to the main memory.

During the transfer of the page the CPU runs another process.

Only the necessary pages are brought to the main memory (***demand paging***).

Due to locality it is expected that the process will spent a significant time on the same page.

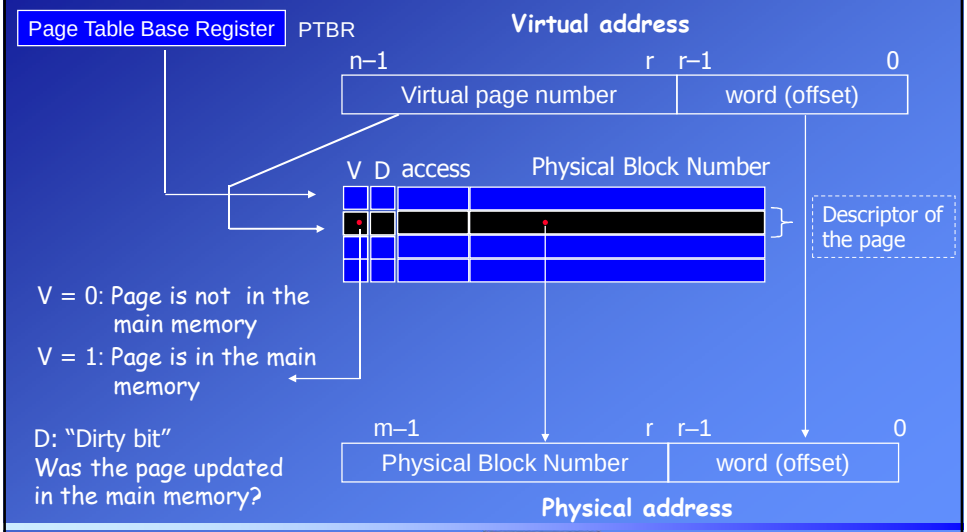


In the page table there may be also additional information:

- Access rights related to the page (RWX)
- Dirty bit (D): Was the page updated in the main memory?

The location of the page in disk (*track-sector*) can be kept in the page table or the operation system can maintain an additional table.

The starting address of the page table of the currently running process is held in Page Table Base Register (PTBR) of the memory management unit.

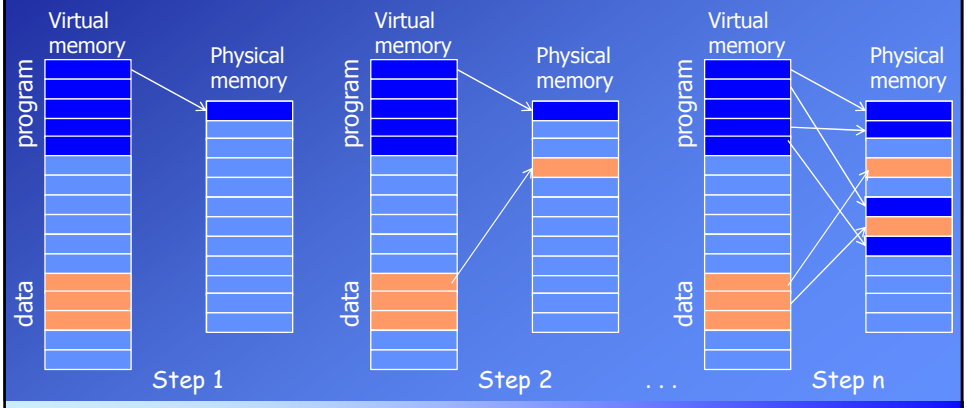


Demand Paging:

When a program first begins executing, the operating system copies a small portion of the process address space from the disk into main memory.

This typically includes the first page of instructions in the program and possibly a small amount of data that the program needs at startup.

Then, as more instructions or data are needed, the operating system brings in pages from the process's address on demand.



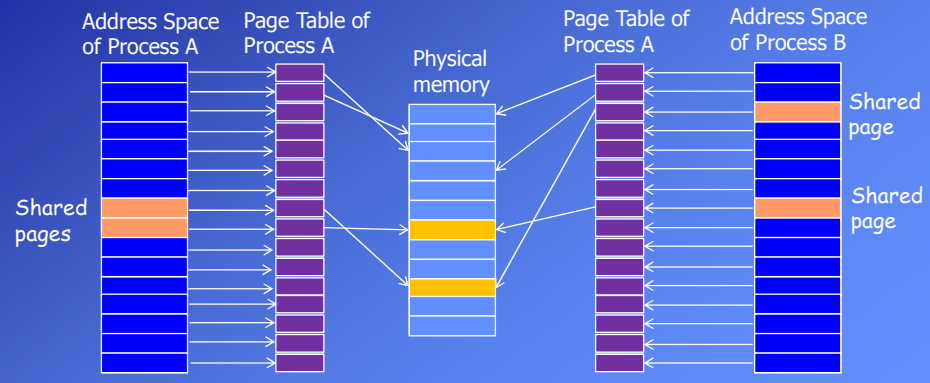
Shared Memory:

In default address spaces of two processes are protected from each other.

With the shared memory mechanism two address spaces can intersect at some points to provide inter-process communication.

Shared pages map to the same physical page.

The same page frame number is placed in the page tables of two processes sharing a page.



Accelerating the address translations (*Translation Lookaside Buffer* – TLB) :

Because of the lookup to the page table each virtual address – physical address translation requires an extra memory access.

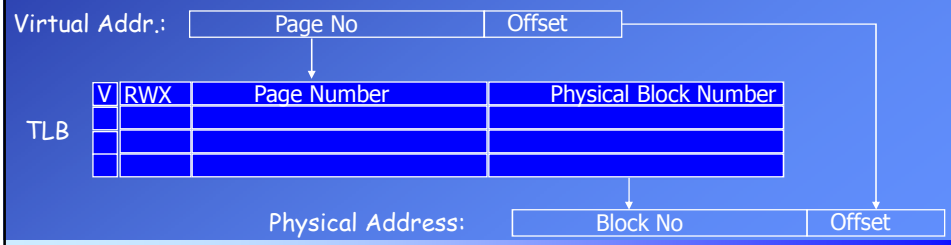
To make the translation faster the most frequently used pages (locality) and their descriptors (main memory block number, access rights) are stored in an associative memory (cached): **Translation Lookaside Buffer (TLB).**

When a virtual address is generated the hardware searches the TLB.

If the mapping exists in the TLB, the hardware can translate the address without using the page table (faster).

Otherwise (TLB miss) necessary mapping information is obtained from the page table and loaded to the TLB.

When TLB is full replacement algorithms are applied (such as FIFO, LRU).



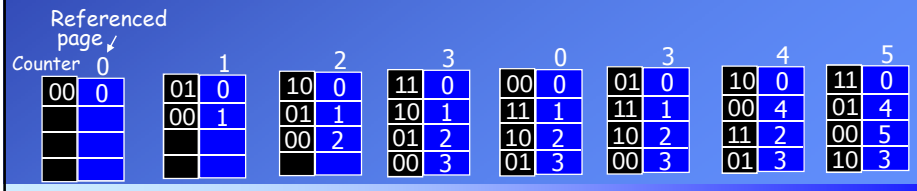
Page Replacement

If a page is referenced which is currently not in the main memory and all blocks of the memory are occupied then a page replacement algorithm must be applied to replace a block in the main memory with a page from the virtual memory.

The LRU (*Least Recently Used*) is one of the efficient algorithms which assigns "aging counters" to the pages (blocks) residing in the main memory.

1. When a page is referenced its counter is cleared.
2. Only the counters of the pages that have lower values than the referenced page are incremented.
3. When a replacement is necessary the page with the highest counter value is replaced; the new page is brought to the main memory, its counter is cleared and the other counters are incremented.

Example: Referenced page numbers: 0, 1, 2, 3, 0, 3, 4, 5
Main memory has 4 blocks. We need 2-bit counters.



Fragmentation

Remember, to benefit from the spatial locality and to use the DMAC the data are transferred in the form of pages.

Process memory is divided into fixed size pages.

The process may not use the entire (last) page.

For example; in a system with the pages of 1K x words, a process with the size of 5K x words + 1 will occupy 6 pages.

Actually the last page contains only 1 word of data but it will copied as an entire page to the main memory and it will occupy an entire block there.

Other process may not use this block.

It might also happen that the process itself requires less than one page in its entirety, but it must occupy an entire block when copied to memory.

In the paged systems pages are not partitioned into smaller chunks; they are always transferred as a whole.

The problem that some space in a page can not be used is called **internal fragmentation**.

Besides, some process may not need the entire virtual space therefore they do not use all the 2^p lines of the page table.

9.2 Segmented and Paged Mapping (Paged Segmentation):

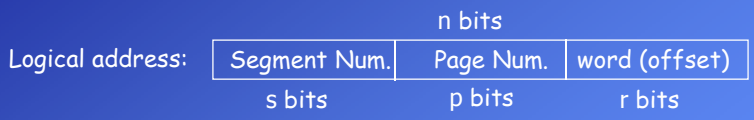
The virtual address space is divided into logical, variable-length units called segments.

Each segment can include different numbers of pages.

A segment corresponds to logical part of a program, such as an entire program, a subroutine, an array.

Segmentation supports sharing and protection on logical parts of the programs (segments), both of which are very difficult to do with paging.

Logical address is divided into three fields:



Properties of the system:

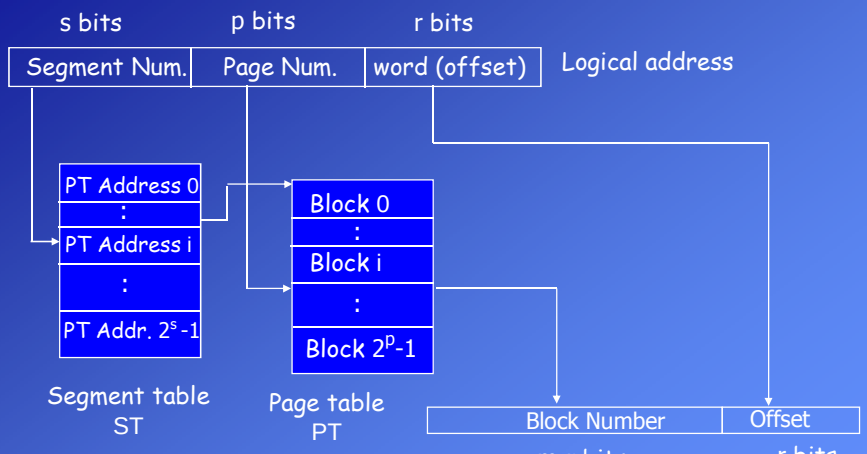
Logical address space: 2^n words

Number of Segments: 2^s

Number of pages in a segment: Between 1 and 2^p

Size of a page: 2^r words

Address translation with paged segmentation:

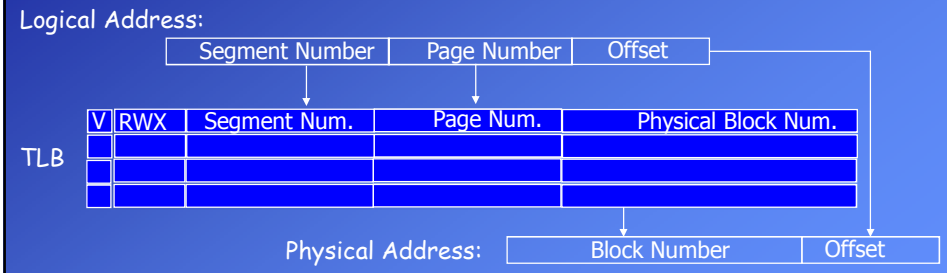


In the segment table other necessary information about the segment is held, such as length, access rights.

Accelerating the address translations (*Translation Lookaside Buffer* – TLB):

Each logical address – physical address translation requires two extra memory access, because of two tables.

To make the translation faster the most frequently used segments, their pages (locality) and related information (physical block number, access rights) are stored in an associative memory (cached): **Translation Lookaside Buffer (TLB).**



9.3 Putting It All Together: Paged Segmentation, TLB, Cache Memory

1. CPU generates the logical address: s, p, w (segment, page, word).
2. Search in TLB.
   a) Match of segment and page numbers in TLB and access rights are OK
      - Get the physical block number from TLB and construct the physical address.
      - Search data in the cache memory. See 3.
      - If LRU is used aging counters in TLB and page table are updated.

   b) Searched Segment-page pair is not in TLB
      - Get the starting address of page table from the segment table using s.
      - Search in table the referenced page number p.

      i. Page is in main memory
         - Get the physical block number from the PT and construct the physical address.
         - Search data in the cache memory. See 3.
         - Update the TLB: New s,p info into TLB, replacement if necessary.
         - Update the aging counters in page table.

b) Searched Segment-page pair is not in TLB (cont'd)

ii. Page is not in main memory

- **Page fault** occurs.
- The referenced page is brought to the physical memory.
- The page table is updated: Address, counters.
- The physical address is constructed
- Search data in the cache memory. See 3.
- Update the TLB: New s,p info into TLB, replacement if necessary.

3. Using the physical address the referenced data is searched in the cache memory according the applied mapping technique (set- associative, direct).

a) The data is in the cache memory

- Data is read from the cache memory.
- If LRU is used the counters in the cache are updated.

b) The data is not in the cache memory

- Data is read from the main memory.
- A block of data is transferred from the main memory to the cache.
- Counters in the cache are updated.