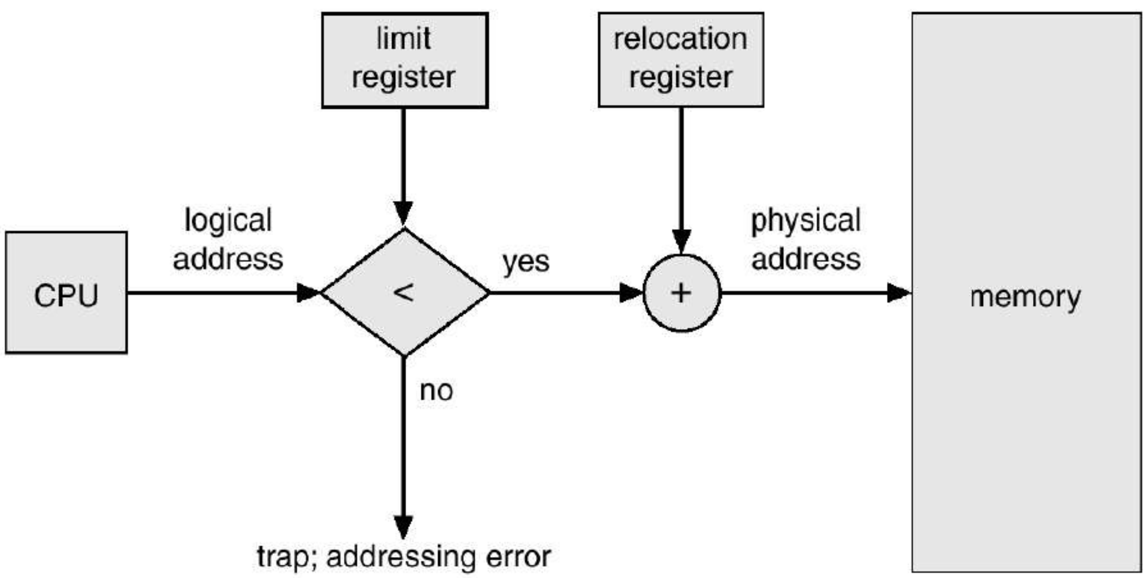
Lesson 19, 20

Objectives

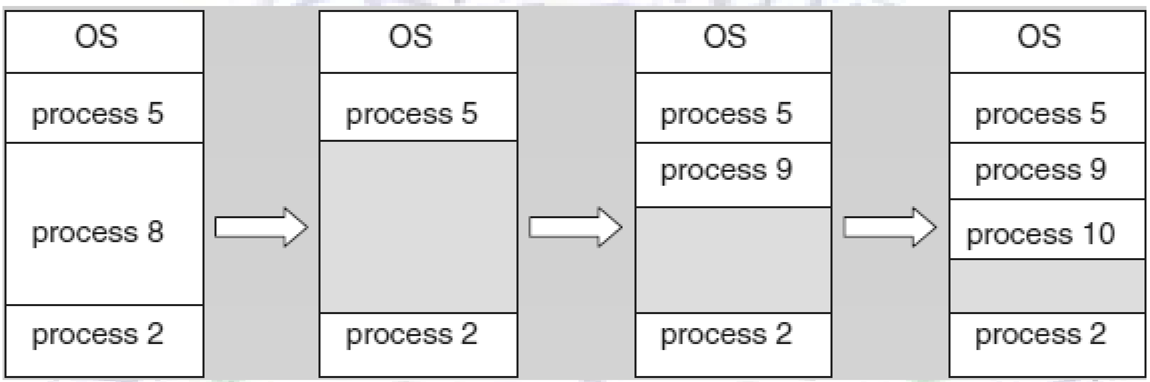
- Contiguous Allocation
- Fragmentation
- Paging
- Implementation of Page table

Contiguous Allocation

- Main memory usually into two partitions:
    - Resident operating system, usually held in low memory with interrupt vector.
    - User processes then held in high memory.
- Single-partition allocation
    - Relocation-register scheme used to protect user processes from each other, and from changing operating-system code and data.
    - Relocation register contains value of smallest physical address; limit register contains range of logical addresses – each logical address must be less than the limit register.



- Multiple-partition allocation
    - Hole – block of available memory; holes of various sizes are scattered throughout memory.
    - When a process arrives, it is allocated memory from a hole large enough to accommodate it.
    - Operating system maintains information about:
      a) Allocated partitions
      b) Free partitions (hole)



Dynamic Storage Allocation Problem

How to satisfy a request of size n from a list of free holes?

- First-fit:
    - Allocate the first hole that is big enough.
- Best-fit:
    - Allocate the smallest hole that is big enough;
    - Must search entire list, unless ordered by size.
    - Produces the smallest leftover hole.
- Worst-fit:
    - Allocate the largest hole;
    - Must also search entire list
    - Produces the largest leftover hole.
- First-fit and best-fit better than worst-fit in terms of speed and storage utilization since in first fit we need not to scan entire memory and in case of best fit wasted space is minimal.

Fragmentation

Scattered memory holes that cannot accommodate a process individually but if they put together a process may be accommodated, phenomenon is called fragmentation. There are two types fragmentation.

External Fragmentation

- Total memory space exists to satisfy a request, but it is not contiguous.
- Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block.
  - Compaction is possible only if relocation is dynamic, and is done at execution time.
  - I/O problem
  - Latch job in memory while it is involved in I/O.
  - Do I/O only into OS buffers.

Internal Fragmentation

- Allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.
- Solution is Paging; which will also a solution to external fragmentation

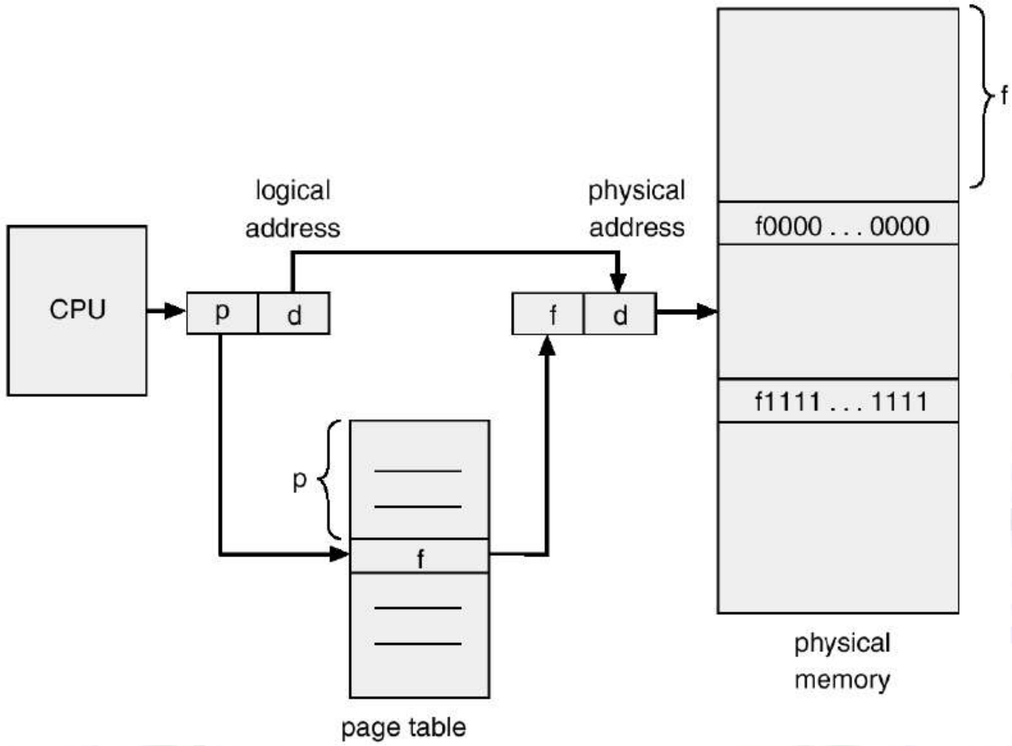
Paging

- Logical address space of a process can be noncontiguous;
- Process is allocated physical memory whenever the latter is available.
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8192 bytes).
- Divide logical memory into blocks of same size called **pages**.
- Keep track of all free frames.
- To run a program of size n pages, need to find n free frames and load program.
- Set up a page table to translate logical to physical addresses.
- Internal fragmentation but negligible when page size is small.

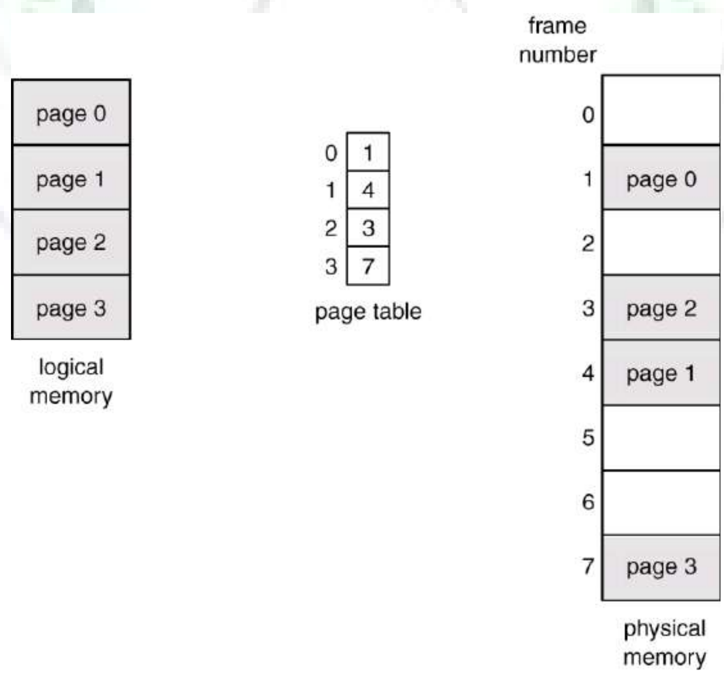
Address Translation Scheme

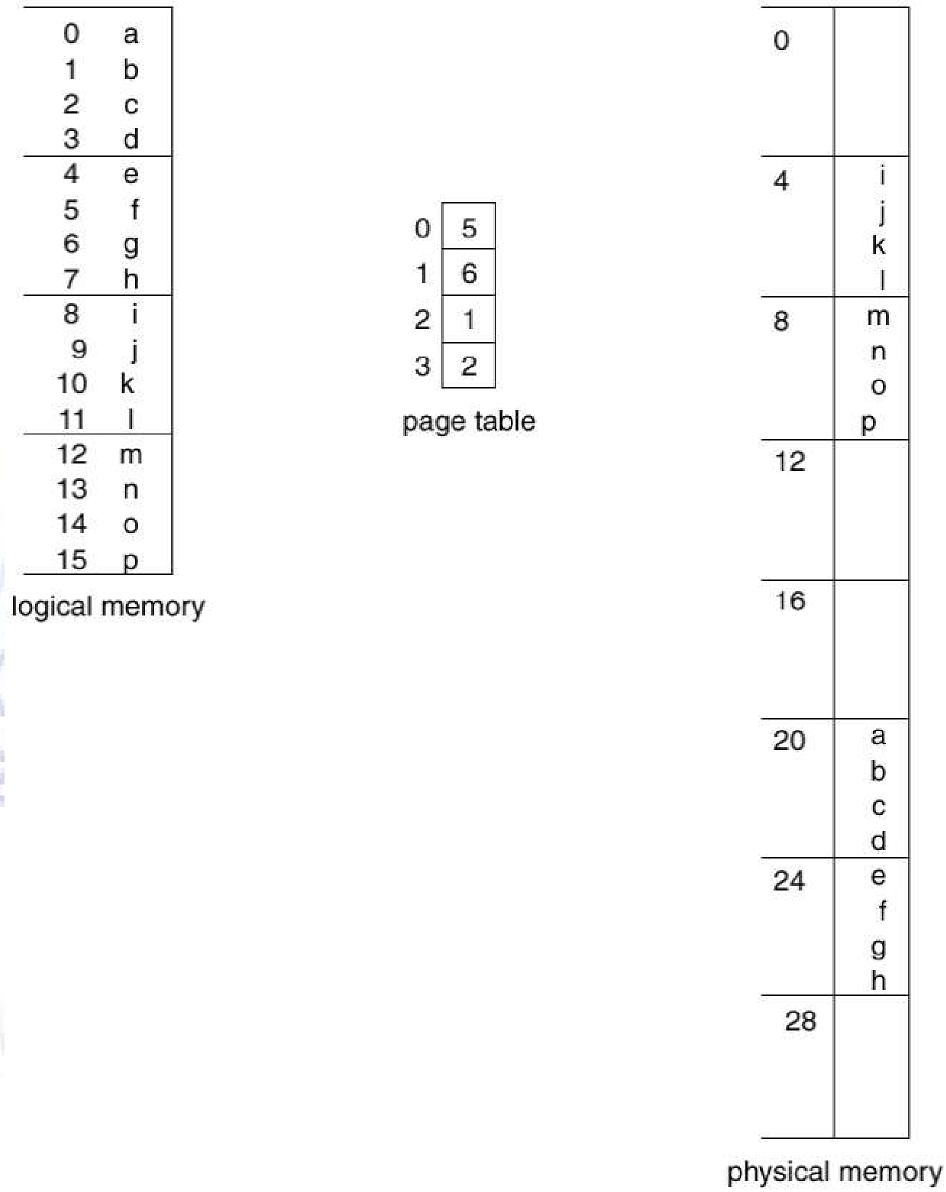
- Address generated by CPU is divided into:

- Page number (p) – used as an index into a page table which contains base address of each page in physical memory.
- Page offset (d) – combined with base address to define the physical memory address that is sent to the memory unit.



Paging Example





Implementation of Page Table

- Page table is kept in main memory.
- Page-table base register (PTBR) points to the page table.
- Page-table length register (PRLR) indicates size of the page table.
- In this scheme every data/instruction access requires two memory accesses. One for the page table and one for the data/instruction.

- The two memory access problem can be solved by the use of a special fast-lookup hardware cache called associative memory or translation look-aside buffers (TLBs)

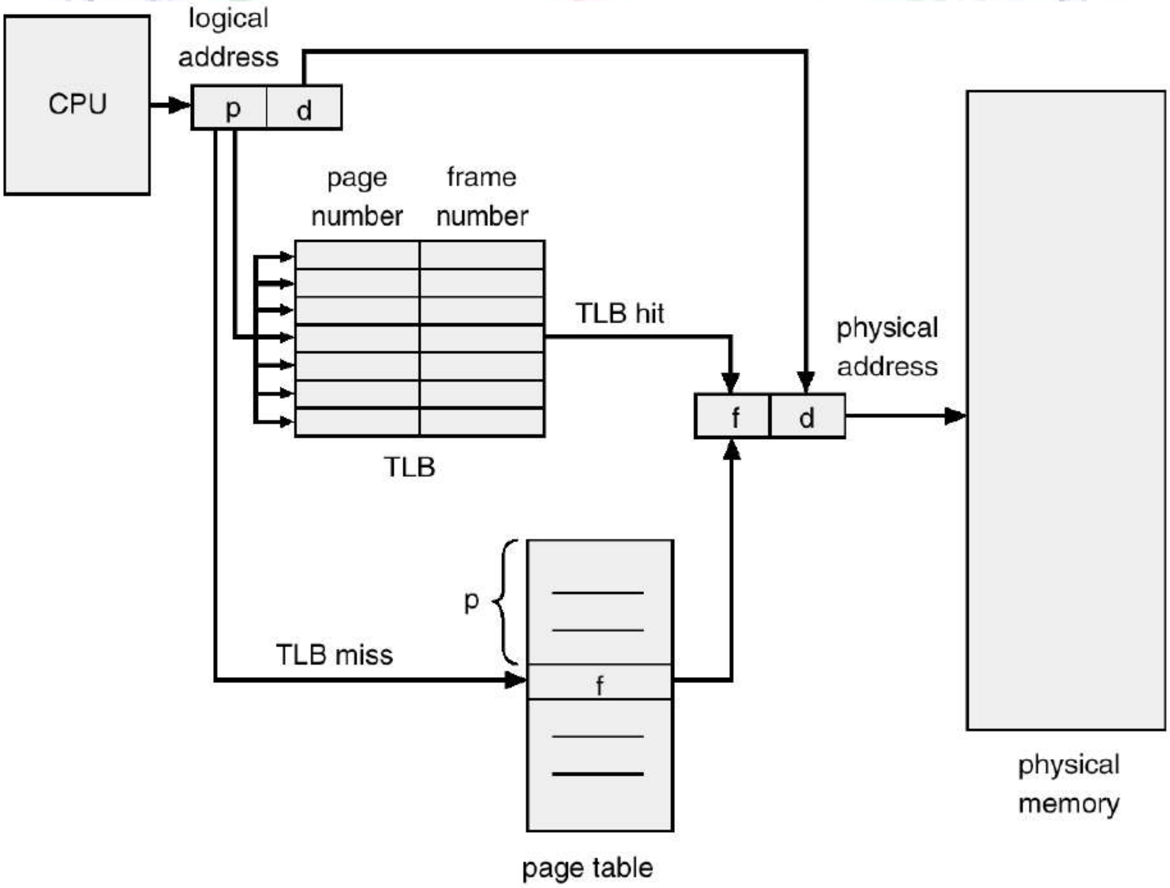
Associative Memory

- Associative memory – parallel search

Page #	Frame #

Address translation (A´, A´´)

- If A´ is in associative register, get frame # out.
- Otherwise get frame # from page table in memory



Performance of Associative memory

What can be achieved by using an associative memory can be figured out by calculating the Effective Access Time (EAT). The procedure is given below.

- Associative Lookup = ε time unit
- Assume memory cycle time is 1 microsecond
- Hit ratio – percentage of times that a page number if found in the associative registers; ration related to number of associative registers.
- Hit ratio = α
- Effective Access Time (EAT)

  $EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$

  $\quad = 2 + \varepsilon - \alpha$

Example

Assuming Hit ratio 80% and 98% respectively and comment the results.