**CS 340**
Lecturer: Dr. Simina Fluture

# MEMORY MANAGEMENT

## PART 1

### Types of memory allocation schemes
### Logical vs. Physical Address Space
### Contiguous Allocation

Contiguous Allocation
The address space of the process must be entirely in the main memory before the process is scheduled for execution, and it must be contiguous.

Paging The address space of the process must be entirely in the main memory before the process is scheduled for execution, however it doesn't need to take a contiguous block.

Demand Paging
Theoretically the process can start executing with no address space yet in the main memory. Its address space doesn't need to be contiguous.

Logical versus Physical Address Space
*logical address*: address generated by the CPU
*logical address space* is the set of all logical addresses generated by a program.

*Physical address*: the address loaded into the memory address register of the memory. Physical address or absolute address is an actual location in main memory.
*Physical address space* is the set of all physical addresses.

The user program deals only with logical addresses and never sees the physical address.

The mapping from logical to physical address is done by the *memory-management unit ( MMU )* which is a hardware device.

**Contiguous Allocation**
In most schemes for memory management, the operating system occupies some fixed portion of the main memory; the rest of main memory is available for use by multiple processes. The operating system is placed in low memory most of the time. The core task of any memory management system is to bring programs into main memory for execution.

Single Partition Allocation
We need to protect the operating-system code and data from changes done by user processes and also need to protect user processes from one another.
This is done using:
- relocation register (contains the value of the smallest physical address)
- limit register (contains the range of logical addresses)
The MMU maps the logical address dynamically by adding to the value in the relocation register.

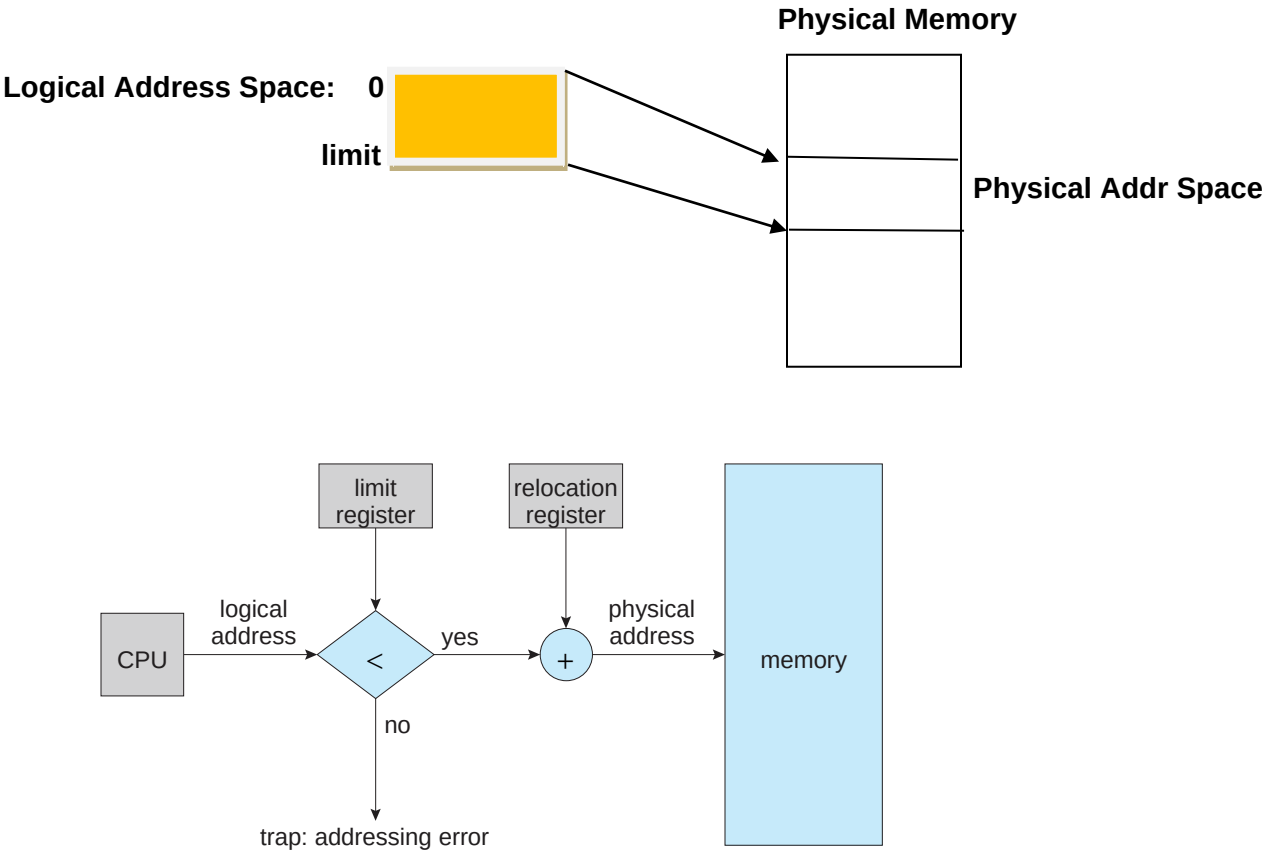


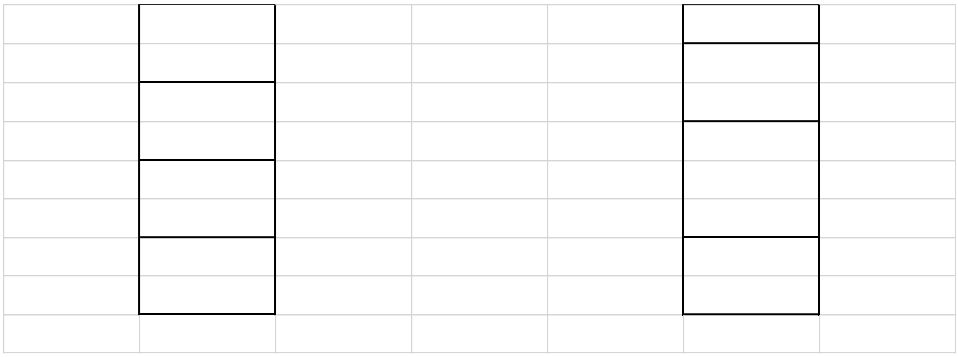

**Fig: Hardware support for relocation and limit registers**

Multiple-Partition Allocation
We need to consider the problem of how to allocate available memory to various processes that are in the job queue waiting to be brought into memory.

**Fixed Partitioning** (one on one correspondence)
One partition can contain at most one process;
One process can take at most one partition.

Equal-Size Partitioning
Divide main memory into equal fixed-sized partitions.
*Disadvantages*:

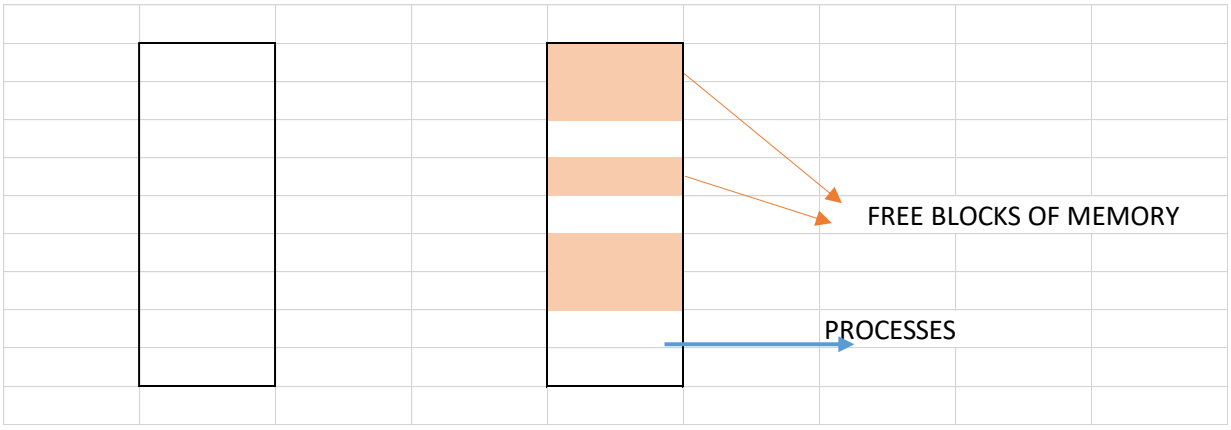
Unequal-Size Partitioning
Divide main memory by using unequal-size partitions.
*Disadvantages:*



a) Equal-size partitions b) Unequal-size partitions

**Internal Fragmentation:** Internal Fragmentation occurs when there is available memory internal to a partition that cannot be used.

**Dynamic Partitioning** (OS/MVT - Multiprogramming with a Variable Number of Tasks)
The partitions used are of variable length and number.
When a process is brought into main memory, it is allocated exactly as much memory as it requires.
The OS keeps a table indicating which parts of memory are available and which are occupied.
Initially, memory is considered as one large block of available memory, a *hole*.
As processes enter the system, they are put into an input (job) queue. At any given time, there is a list of available block sizes in the input queue.
The OS can order the input queue according to a scheduling algorithm. When a process arrives and needs memory, we search for a large enough memory hole.
If the hole is too large, it is split in two: one part is allocated to the process, the other is returned to the set of available blocks.
When a process terminates, it releases its block of memory.




Dynamic Storage Allocation Problem: How to satisfy a request of size *n* from a list of free holes.
Strategies used:

**First-fit**: allocate the first hole big enough. Searching can start either at the beginning of the set of holes, or where the previous first-fit search ended.

**Best-fit**: allocate the *smallest* and big enough hole.
**Worst-fit**: allocate the *largest* hole. This strategy produces the largest leftover hole.

Example given in the video.

External and Internal Fragmentation
**External Fragmentation**: External fragmentation exists when enough total memory space exists to satisfy a request but is not contiguous.
Solution: Compaction: The goal is to shuffle the 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.
- selecting an optimal compaction strategy is difficult.
- swapping can be combined with compaction.

Video Link: http://youtu.be/Ro1vU8sSpWw?hd=1