Main Memory Manager

Read Text 9.1, 9.2

Allocating main memory space to processes

Protecting each process main memory space

Using the disk as extension to main memory

A few basic concepts

An instruction must be in main memory for it to be executed by the CPU

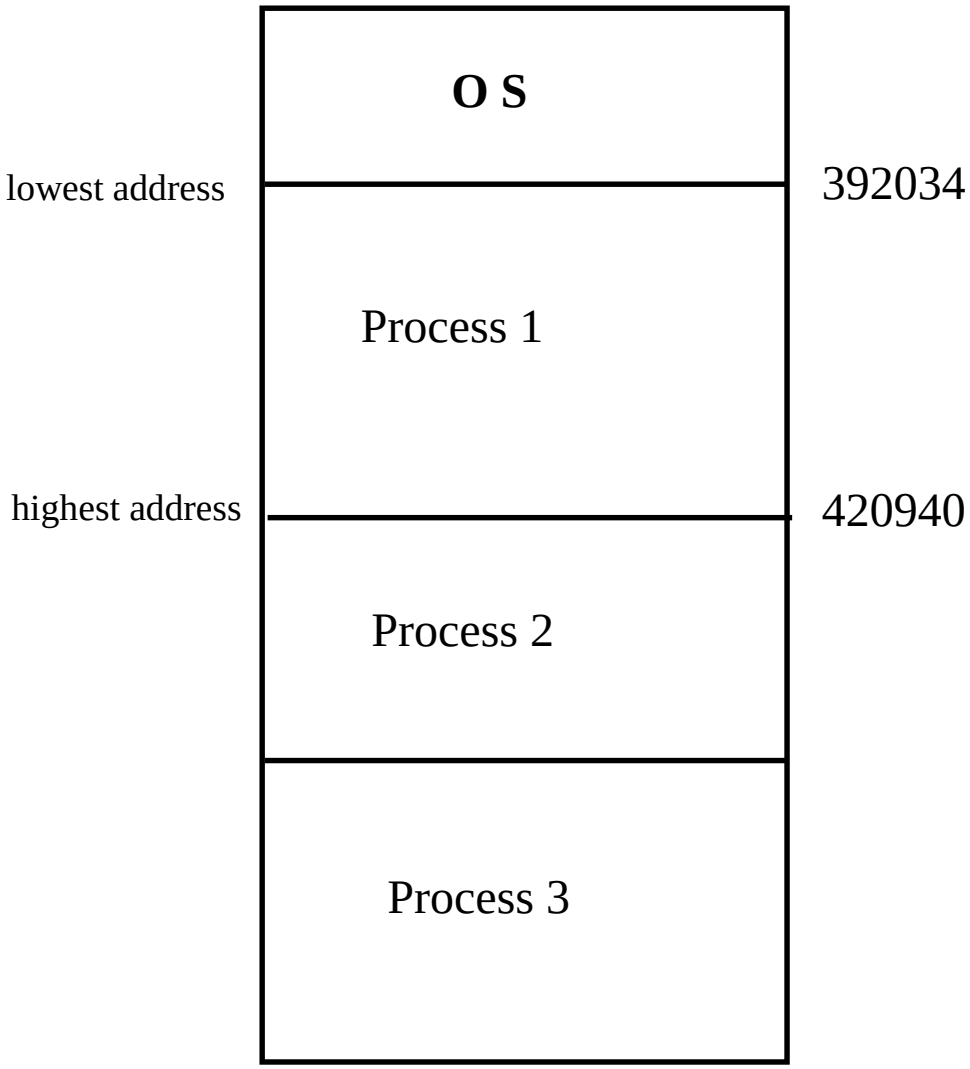
Data must be retrieved from main memory or stored back into main memory by the CPU

Only the operating system will assign memory space to a process

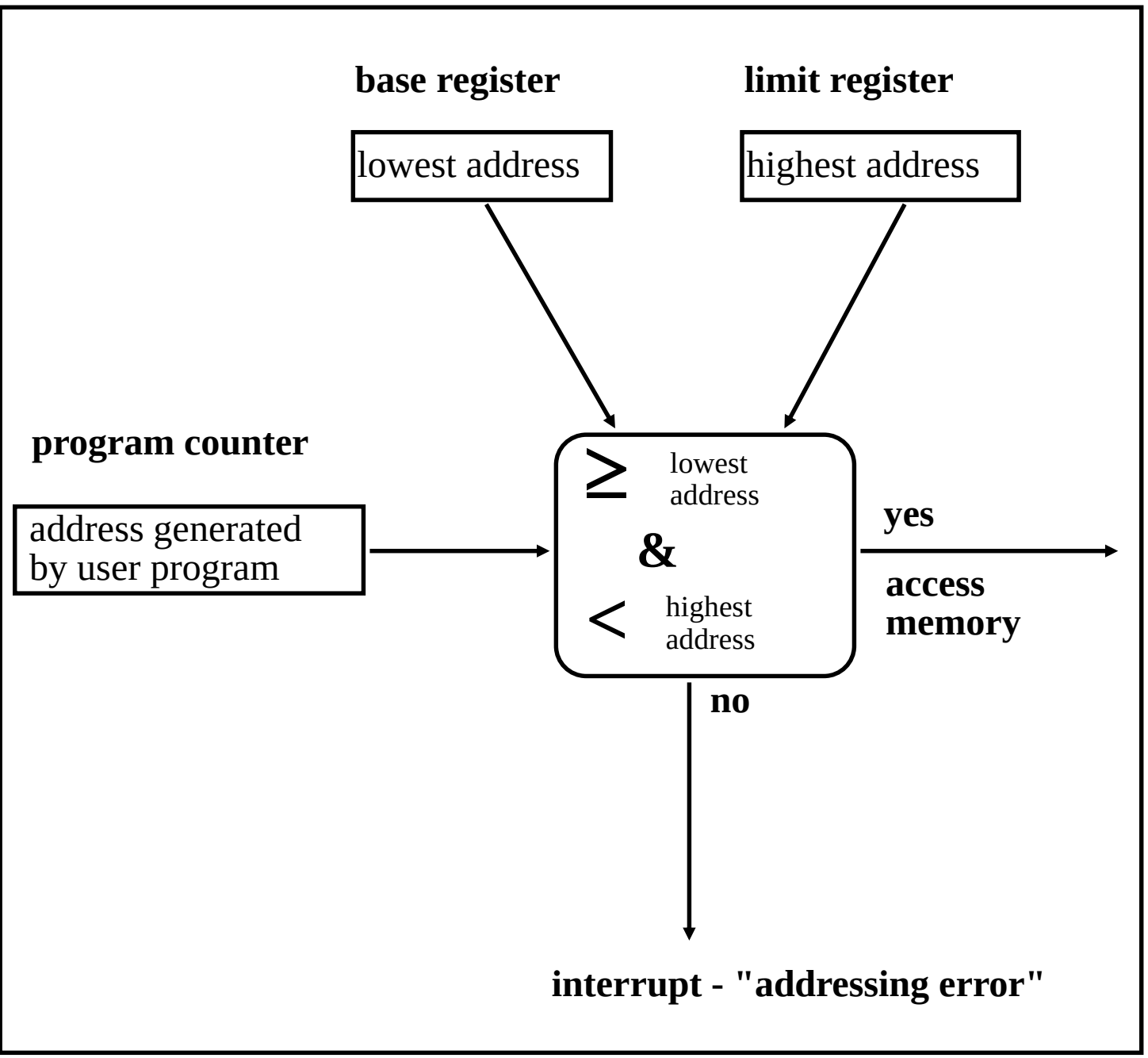
The OS knows where each concurrently executing process is in main memory.

The OS must be protected from any user process accessing its memory space and processes' memory spaces must be protected from each other. This protection mechanism is implemented within the CPU to make it as fast as possible since it must occur every time main memory is accessed and also because the OS is not running when the process is accessing memeory.

Sharing Physical Address Space



Protecting Physical Address Space



Base and limit register values are loaded during context switching

The OS has access to all of the physical memory space to:

- Load user programs into memory
- Remove programs from memory
- Perform I/O to an from user memory

Translation to Absolute Addresses

Source Program B

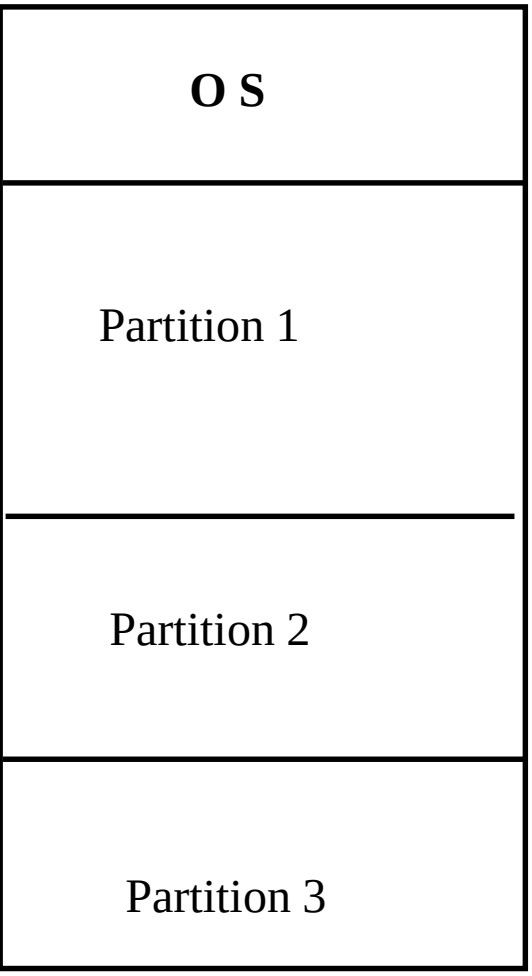
Source Code

Var *X*
….
Call procedure F

COMPILER

Module B

Absolute addresses	Translated code
8000	Variables
8002	*X*
8028	Program instructions
8040	*jump to 10000*
9230	



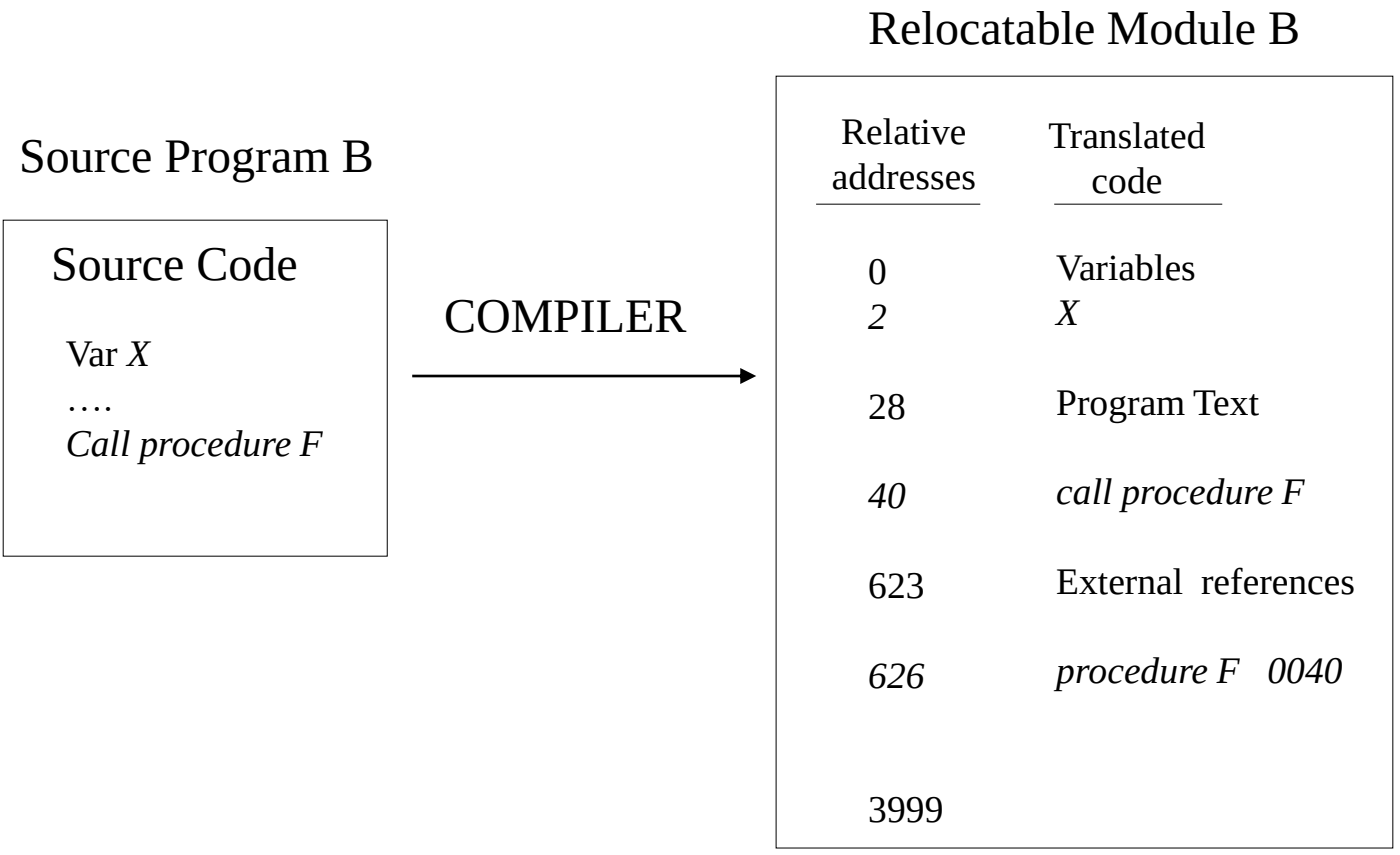
8000 - base address

19999 - limit address

Program must always run In same partition or will need to be compiled again

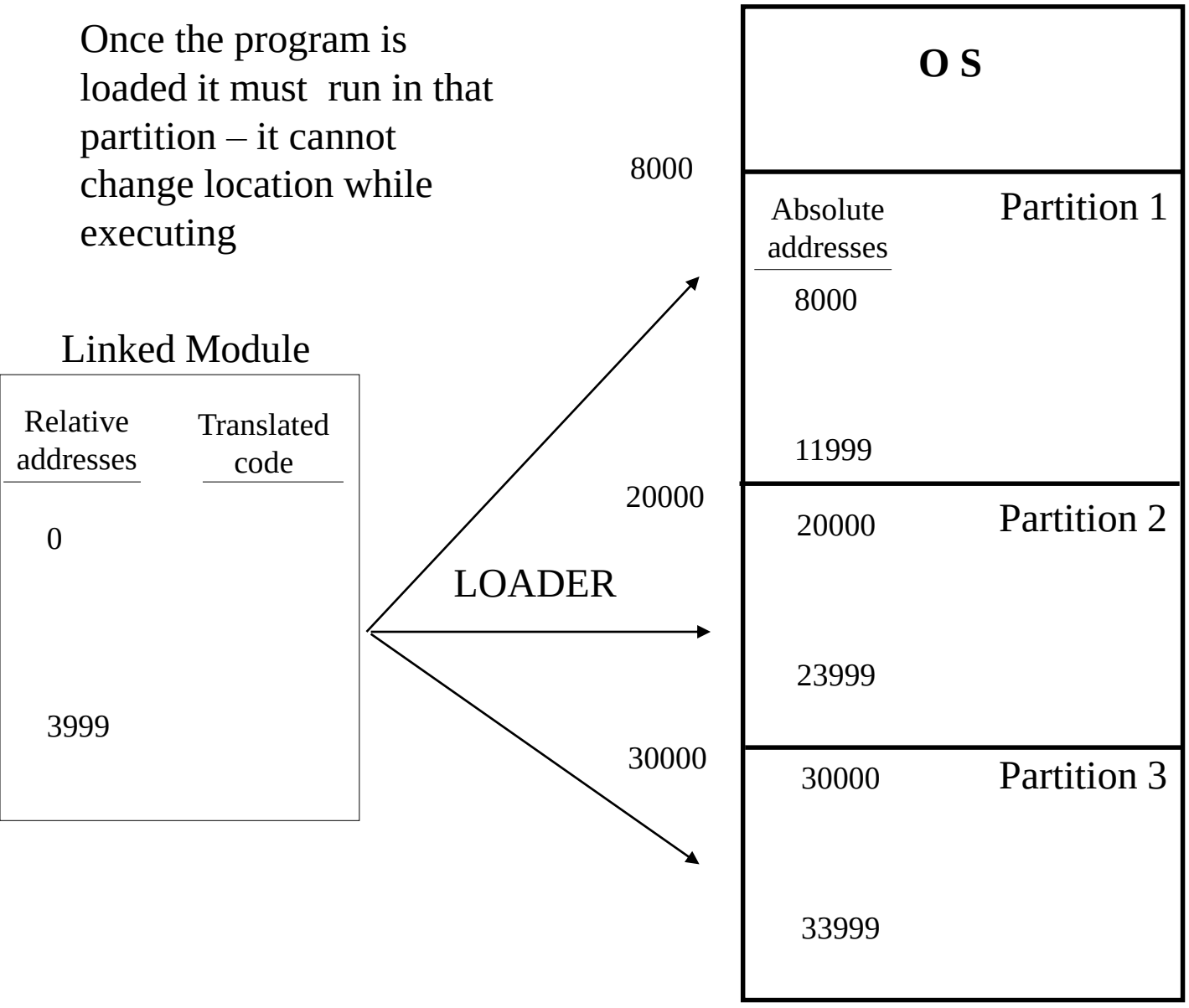
Translation to Relocatable Addresses

Binding logical addresses to physical addresses is done when the program is loaded



Loading Relocatable Modules

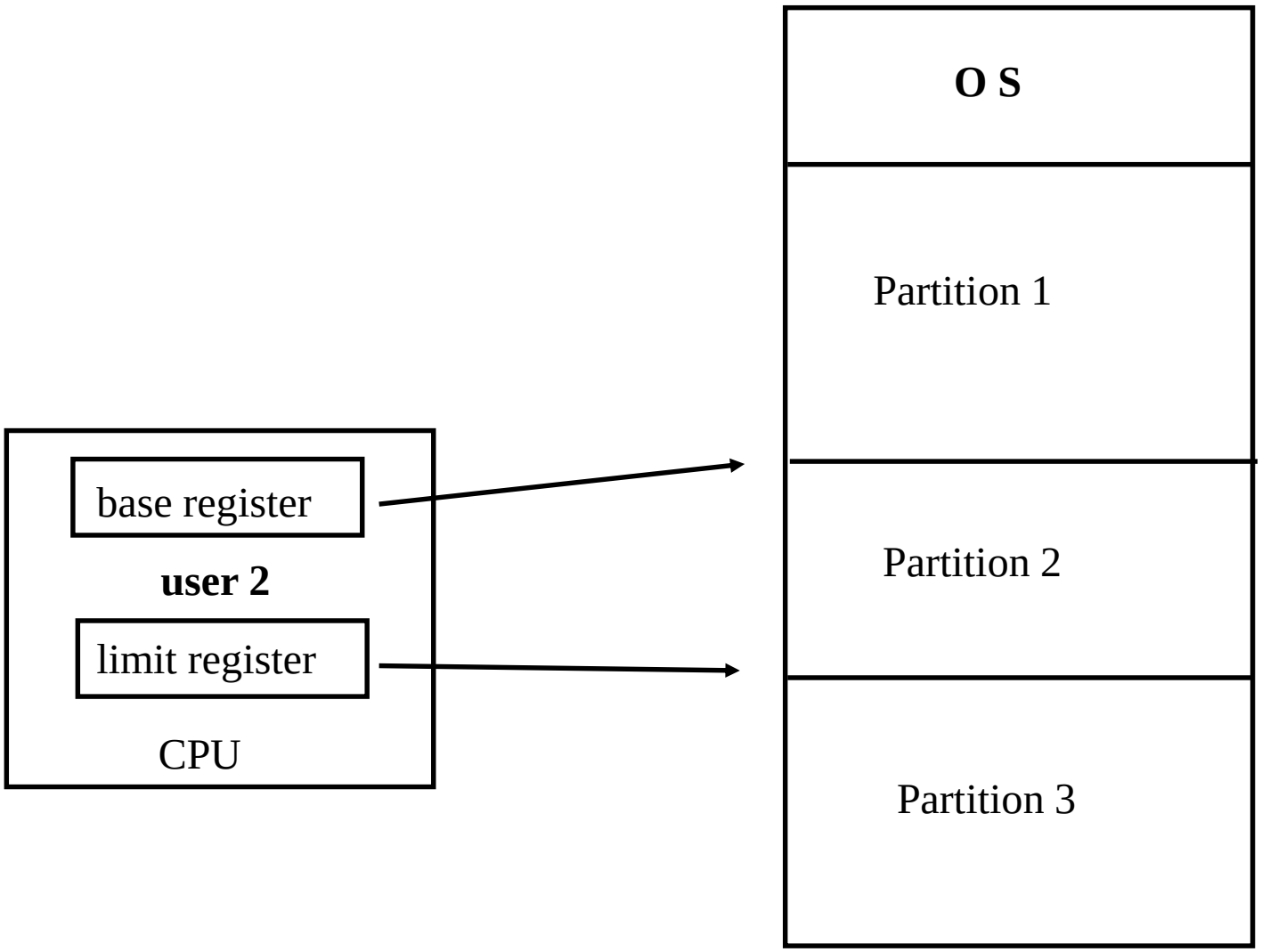
Once the program is loaded it must run in that partition – it cannot change location while executing



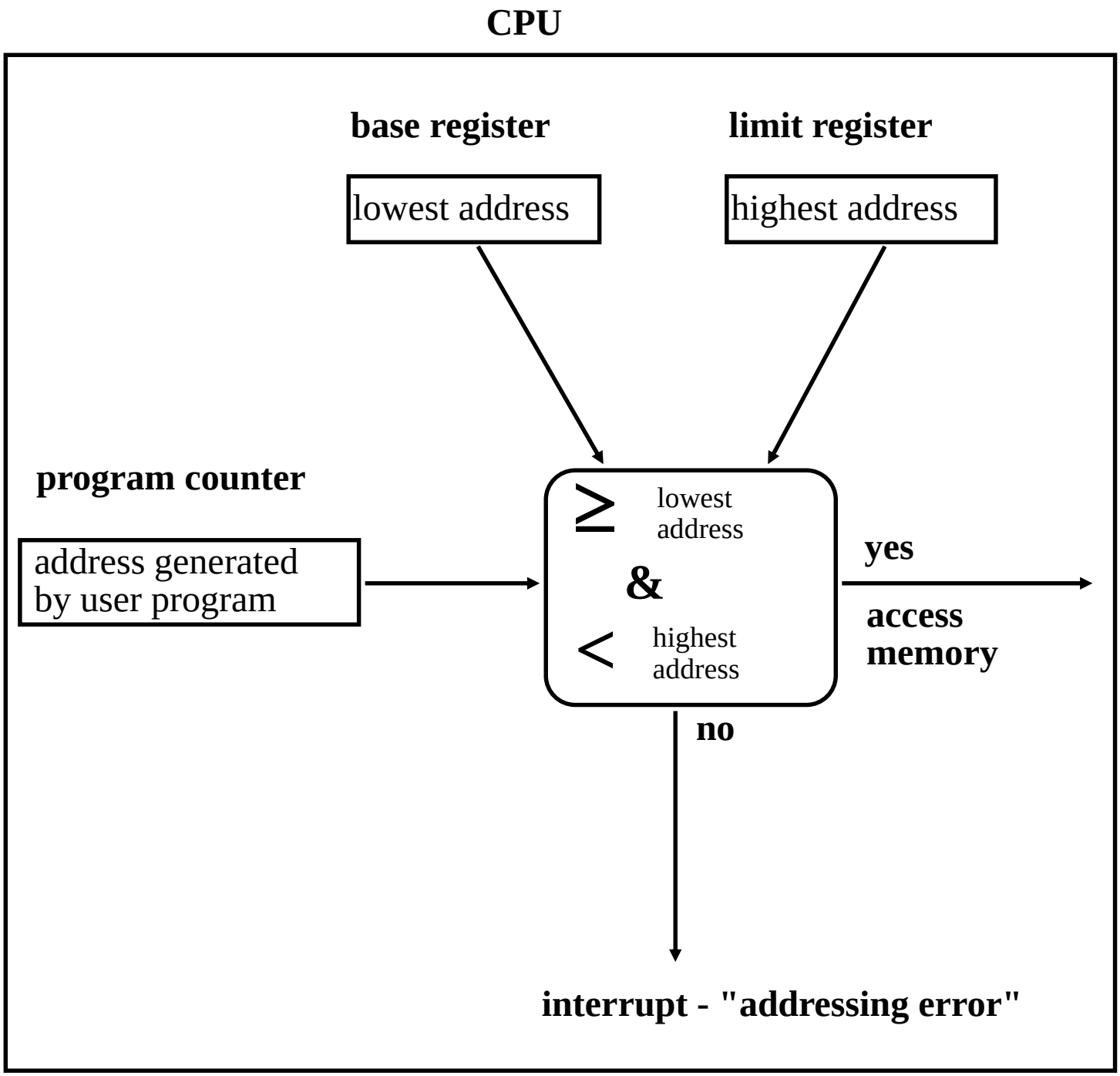
Main Memory Protection

The low and the high physical memory addresses of the partition used by the process that is assigned the CPU are loaded into CPU registers by the operating system using privileged instructions.

Each memory address generated by the process is checked against these registers



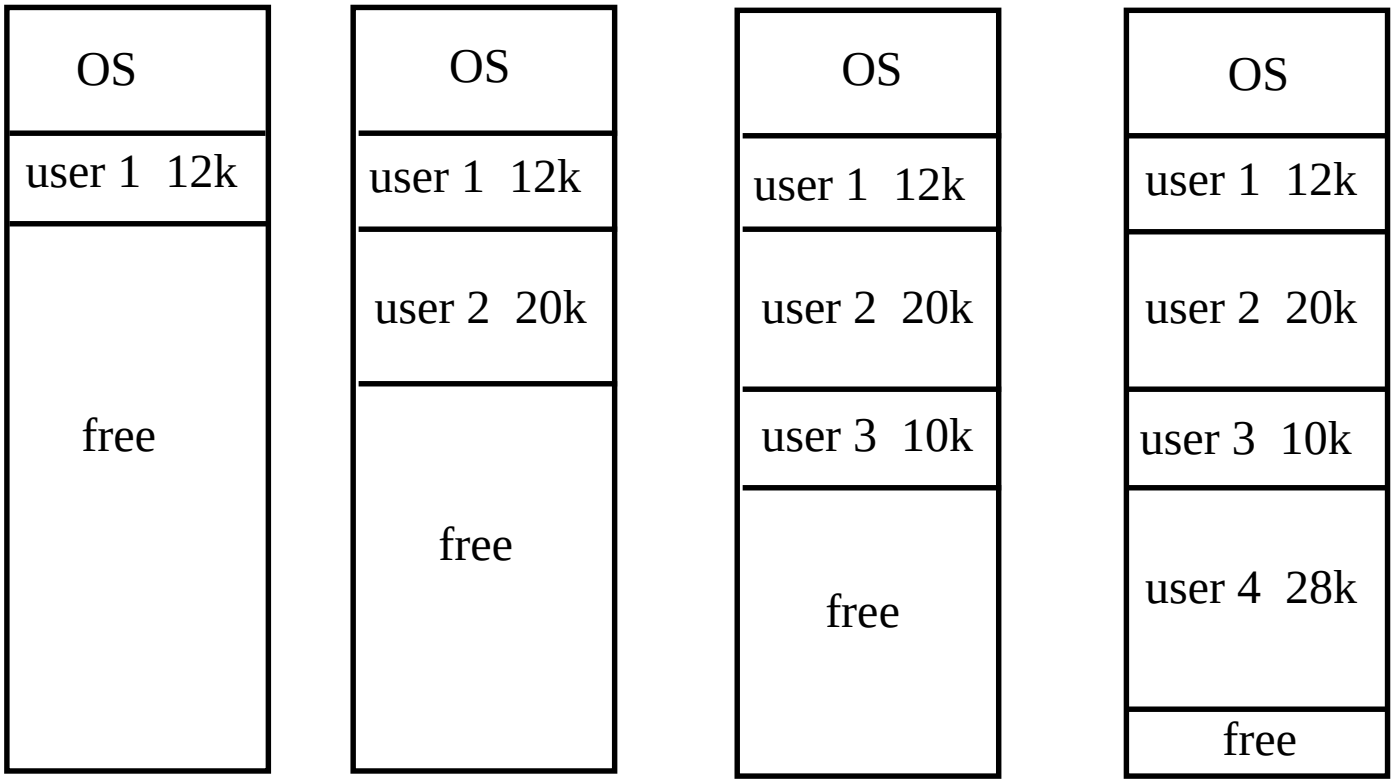
Protection in Multiple Partition System

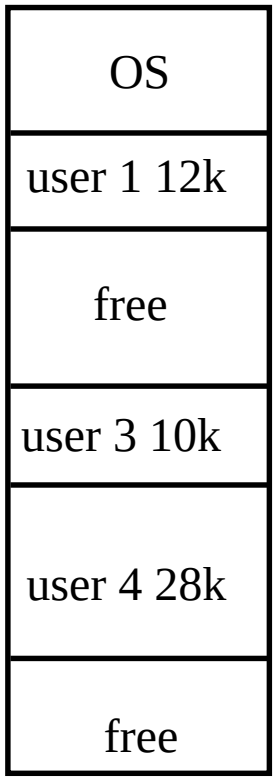


The low and the high physical memory addresses of the partition used by the process that is assigned the CPU are loaded into CPU registers. Depending on which partition is executing, the registers are loaded with different values

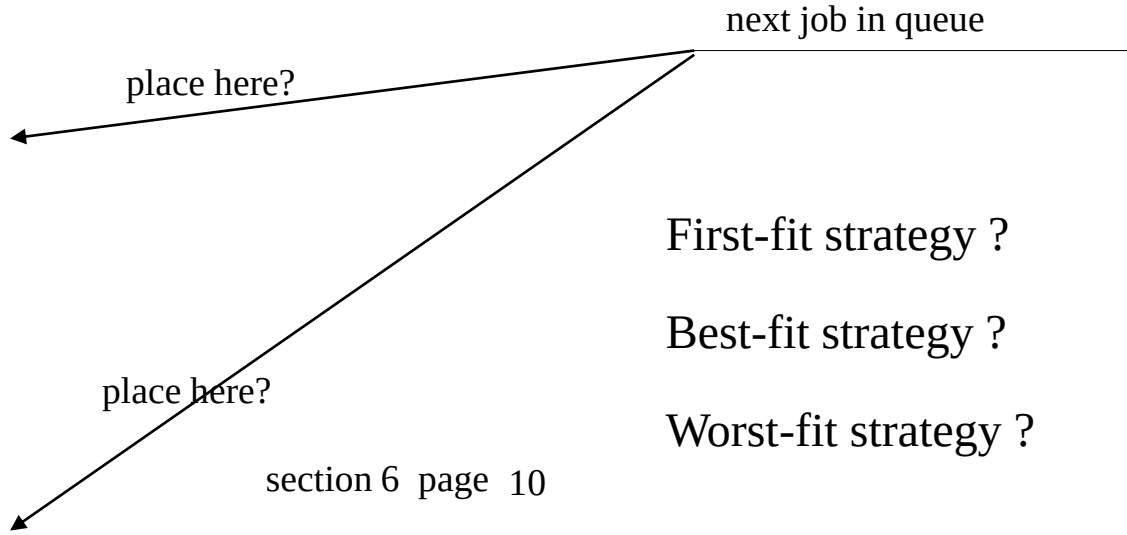
Multiple Variable Partitions

Processes are assigned as much memory as they need. There are no fixed boundaries among partitions but uninterrupted memory is still needed





As a process finishes its memory becomes available for the next job in queue



First-fit strategy ?

Best-fit strategy ?

Worst-fit strategy ?

Main Memory Placement Strategies

First Fit Strategy

Assign the process to the first free hole big enough that appears in the free hole list

Best Fit Strategy

Assign the process to the free hole that fits it the closest

Worst Fit Strategy

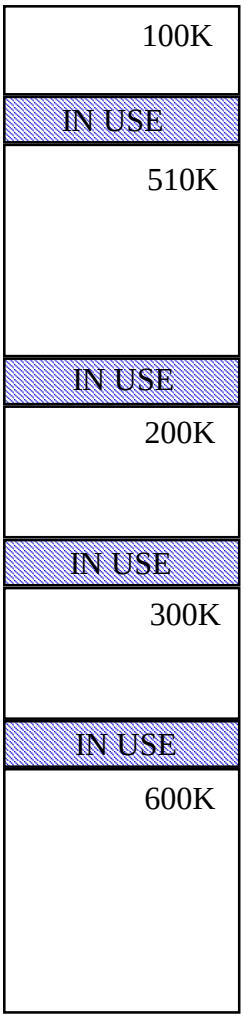
Assign the process to the free hole that fits it the worst

Example of Main Memory Placement Strategies

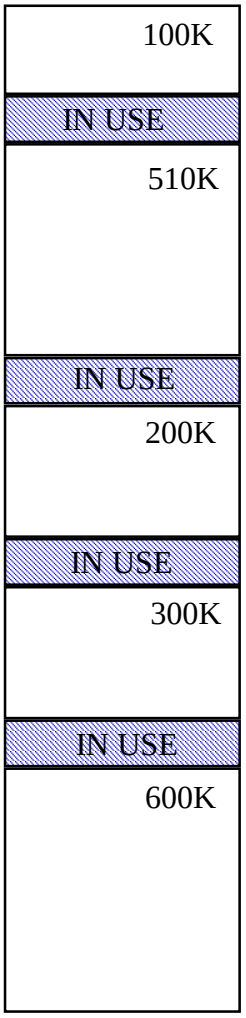
A particular Operating System manages real memory organized as multiple variable partitions. At the present time there are free memory partitions of 100K, 510K, 200K, 300K, and 600K. In the input queue we have several processes waiting to be assigned memory. Process 1 is at the head of this queue and requires 212K, next in line in the queue is process 2 which requires 417K, followed by process 3 requiring 112K, and then process 4 requiring 426K. Assume that the free storage list is appropriately updated right after a process is assigned main memory.

Indicate below to which partitions in memory will the processes be assigned to according to the following strategies.

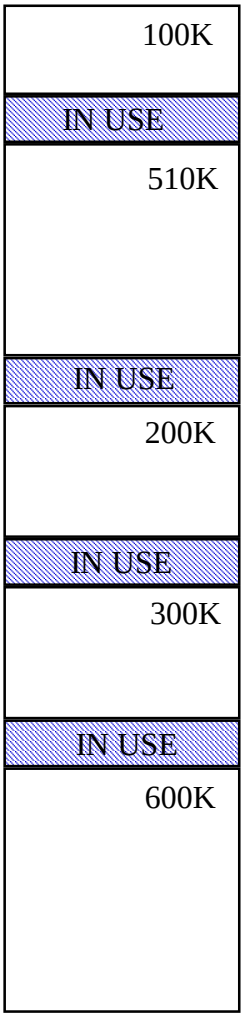
FIRST FIT



BEST FIT

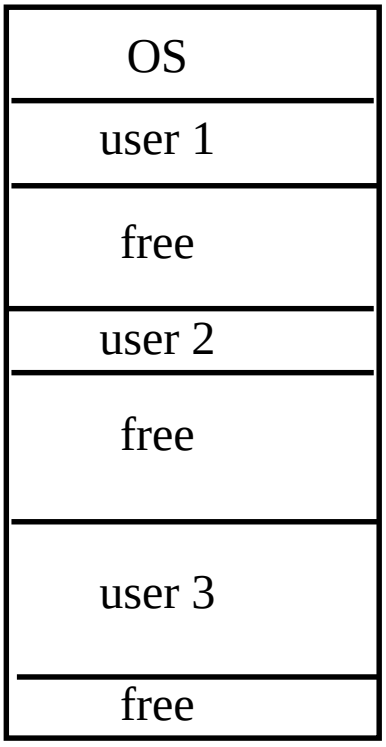


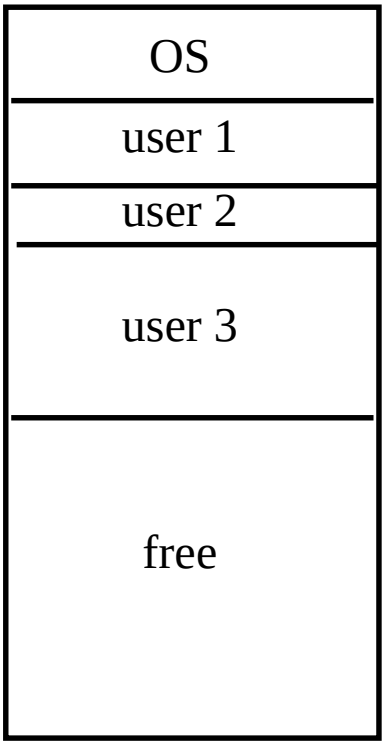
WORST FIT



External Fragmentation and Compaction

External Fragmentation occurs when there is enough memory that has not been assigned to any process to fit a new process in, but it is in separate pieces so that the new process cannot fit contiguously

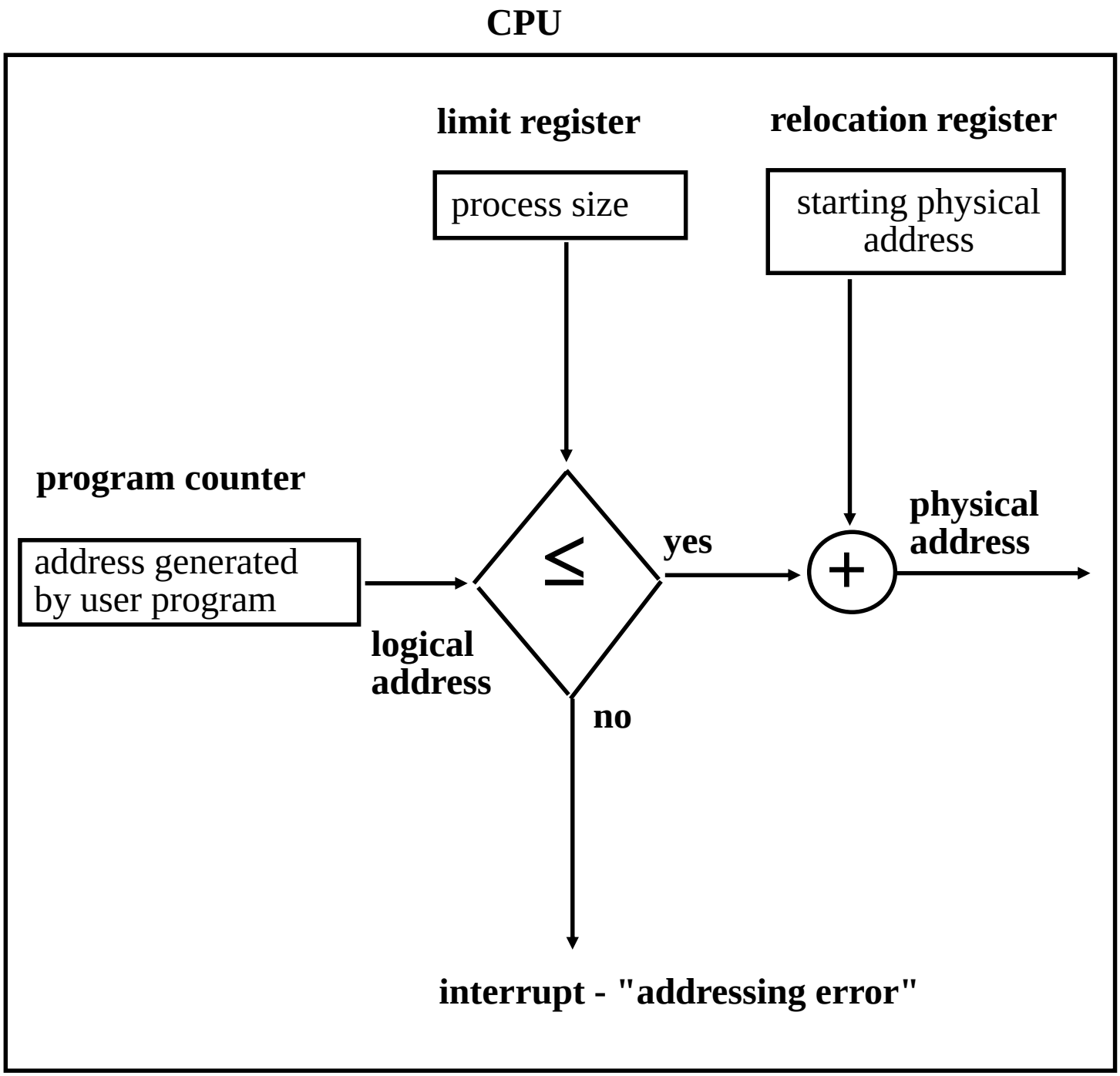




Compaction moves the processes to one end and collects all the free spaces at the other creating one large uninterrupted block of memory.

Since processes are moved during execution from one memory segment to another, physical memory addresses are assigned at execution time.

Relocation Hardware Support



The compaction process must save the new starting physical address for every process in main memory