Operating System Concepts

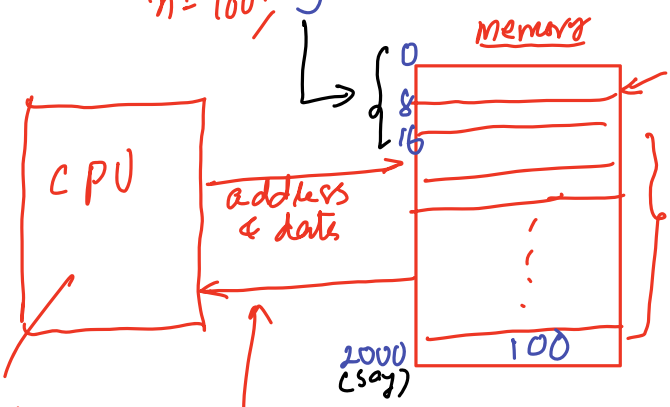
Memory management: Logical versus physical address space, swapping, contiguous allocating, paging, segmentation,)
Slides Set #14

By Prof K R Chowdhary

JNV University

January, 2024

Program: main(){
int n;
n = 100;
}



process
↓
User prog & OS, compilers

loader
↓
loads the progs. in RAM from disk

these locations are provided by OS,

Registers
- Accumulator
- Prog. counter
- MAR (Memory address) register

Instructions & data

n = Symbolic address
2000 = actual " physical

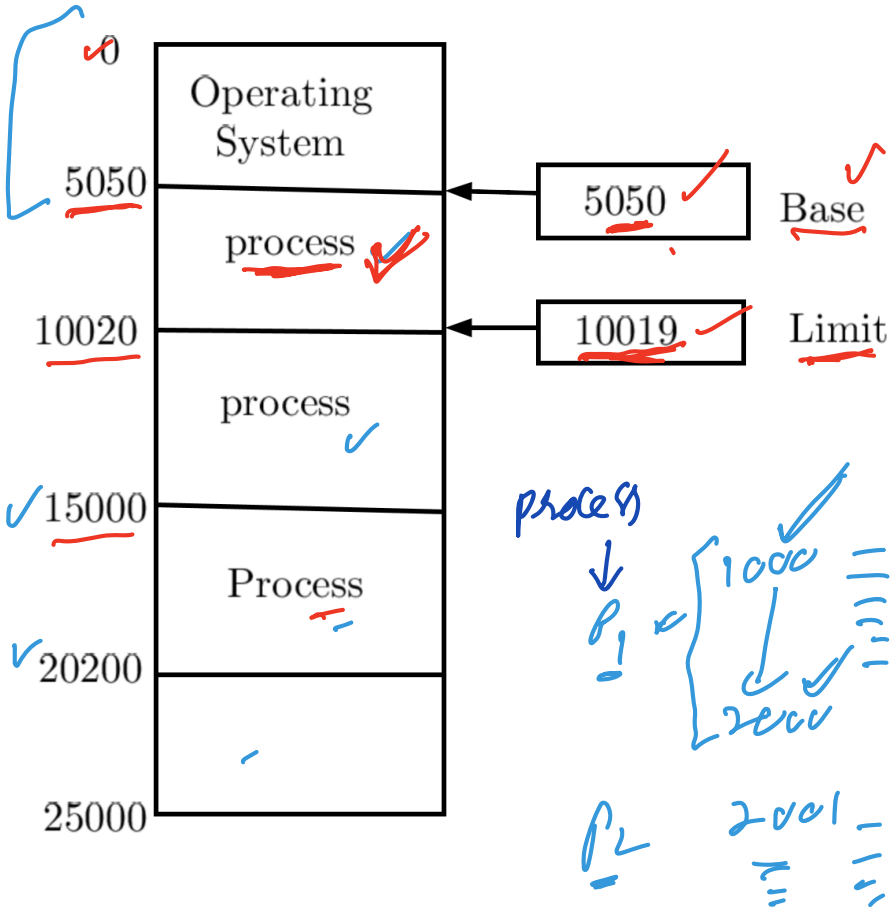
mapping of actual addresses to physical addr is done by compiler while translating prog.

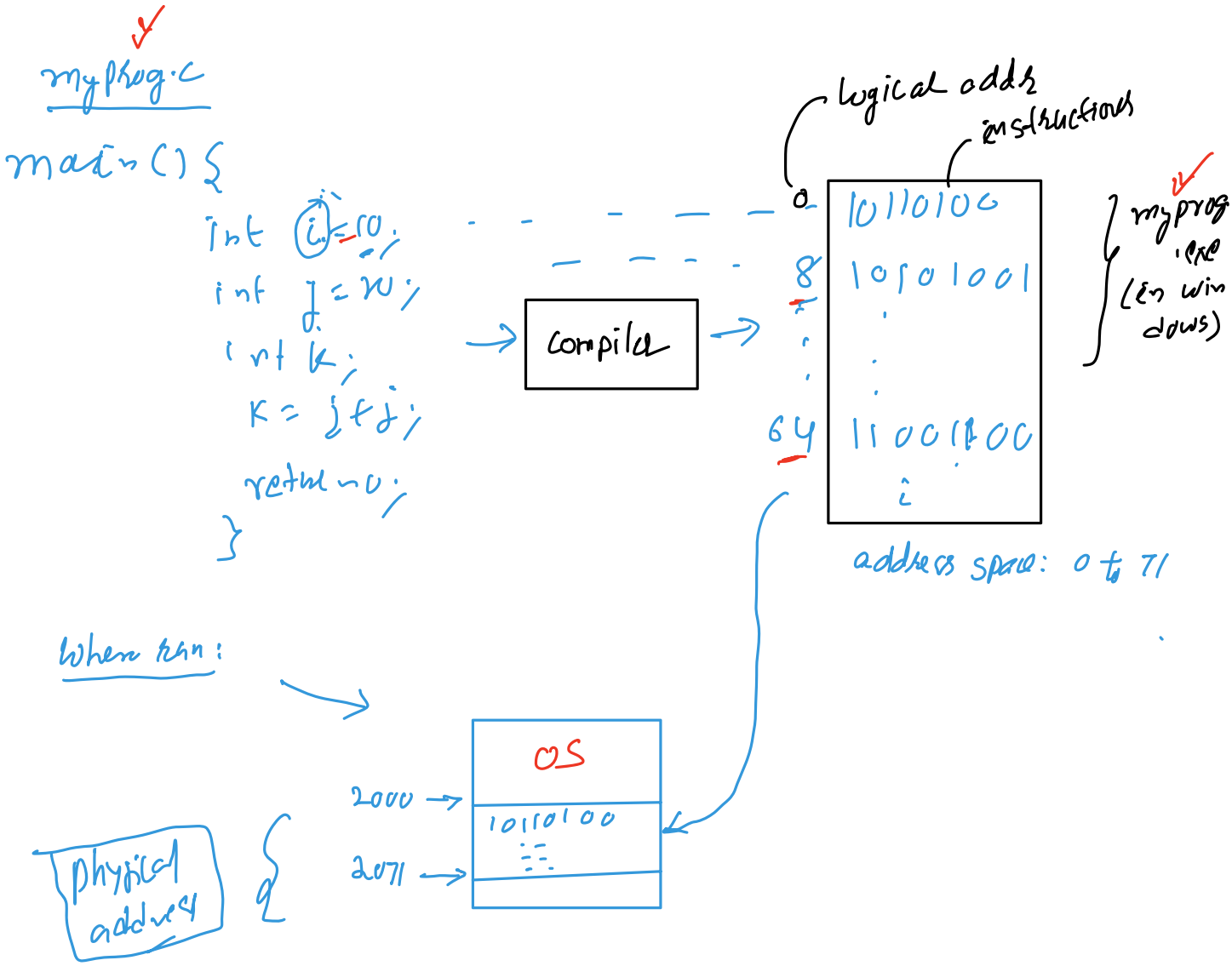
prog. loaded
↑
being

Memory Management: Basic Hardware

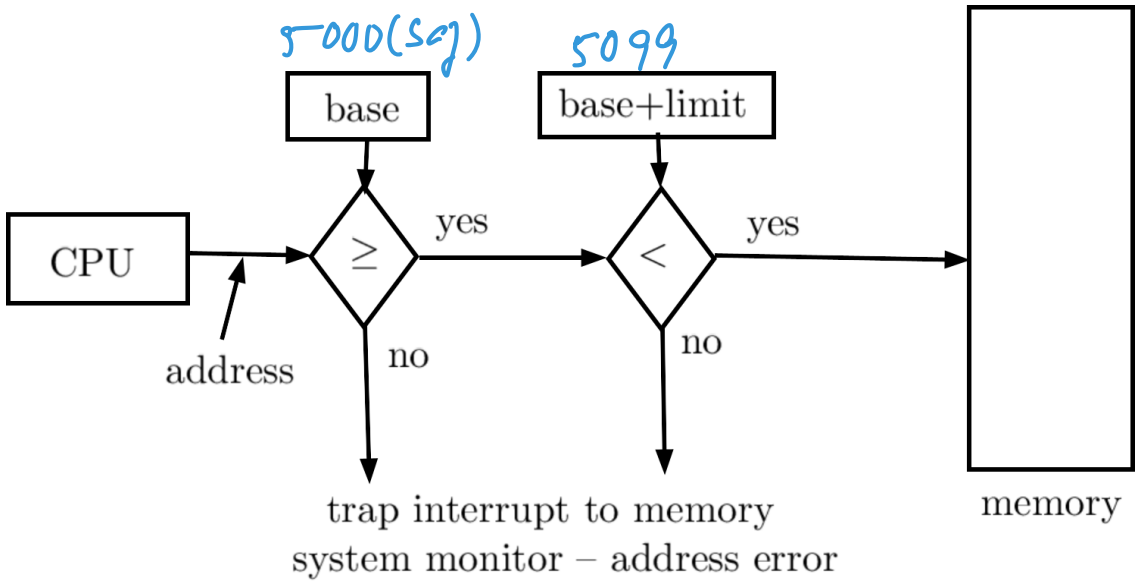
Issues that are particular to managing memory: basic hardware, the binding of symbolic memory addresses to actual physical addresses, and the distinction between logical and physical addresses.

- Main memory and the registers built into the processor itself are the only general-purpose storage that the CPU can access directly.
- We first need to make sure that each process has a separate memory space.




myprog.c
main() {
int i=10;
int j=20;
int k;
k = i+j;
return 0;
}
compile
logical addr
instructions
0
8
64
10110100
10101001
11001100
myprog.exe (in windows)
address space: 0 to 71
when ran:
OS
2000
2071
10110100
physical addrs

Memory protection



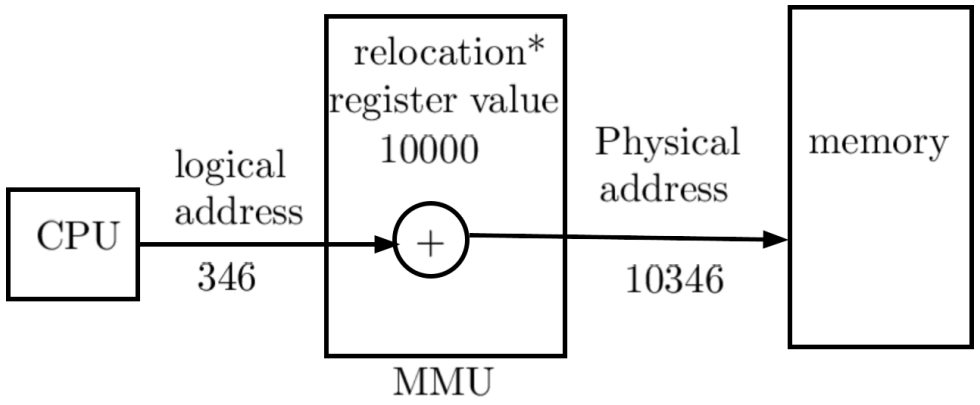
- Protection of memory space is accomplished by having the CPU hardware
- The operating system, executing in kernel mode, is given unrestricted access to both operating-system memory and users' memory.
- **Address Binding**: Usually, a program resides on a disk as a binary executable file.
- Most systems allow a user process to reside in any part of the physical memory.
  - Compile time.
  - Load time.
  - Execution time.

In DOS there is only one user program/one process and the OS in RAM

- So every time, the program is loaded, it is loaded in the same locations
- when a compiler translates a program in DOS (to run in DOS), the addresses in the translated program are same at which it will be running.
- so, Compile time addresses
  = load time addresses
  = run time addresses

Logical versus physical address space

- Address generated by CPU is referred to as **logical address**,



* Also called base register

- An address seen by the memory unit (i.e., one loaded into the memory-address register) is referred to as a **physical address**.
- The run-time mapping from virtual to physical addresses is done by a hardware device (memory-management unit)
- The user program never sees the real physical addresses.
- So, there are two different types of addresses: logical addresses (range: 0 to max) and physical addresses (range: R + 0 to R + max)

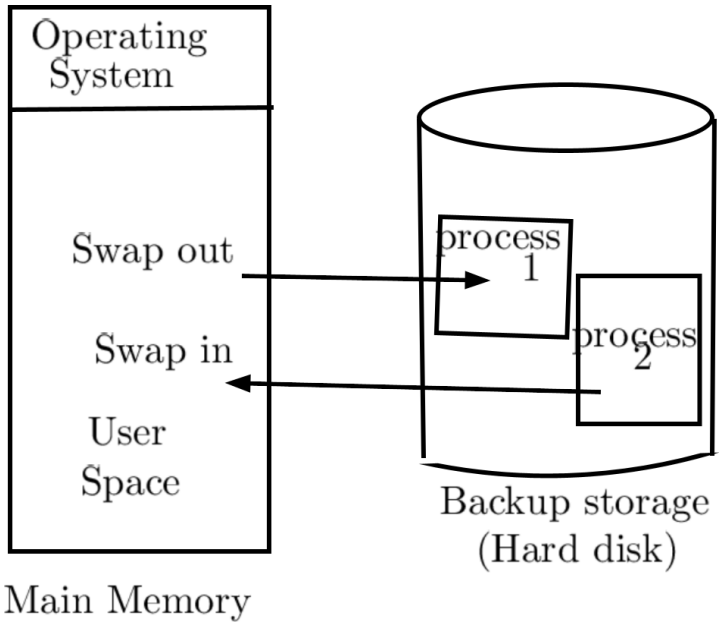
Dynamic loading and linking

- **Dynamic loading**: It is necessary for the entire program and all data of a process to be in physical memory for the process to execute.
- When a routine needs to call another routine, the calling routine first checks to see whether the other routine has been loaded. If it has not, the relocatable linking loader is called to load..
- The advantage of dynamic loading is that a routine is loaded only when it is needed.
- Dynamic loading does not require special support from the operating system.
- **Dynamic Linking**. Dynamically linked libraries are system libraries that are linked to user programs when the programs are run

Swapping

A process must be in memory to be executed. However, it can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution.

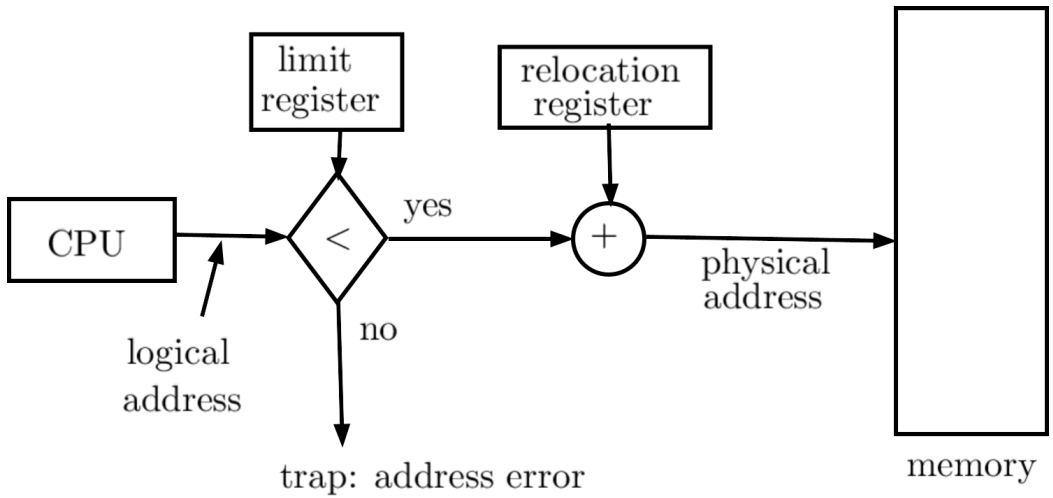
- **Standard swapping** involves moving processes between main memory and a backing store.



- The backing store is commonly a fast disk.
- Standard swapping is not used in modern operating systems. It requires too much swapping time and provides too little execution time
- **Swapping on Mobile Systems**.

Contiguous Memory Allocation

- The main memory must accommodate both the operating system and the various user processes.
- We can place the operating system in either low memory or high memory.
- We usually want several user processes to reside in memory at the same time.
- **Memory Protection**: We can prevent a process from accessing memory it does not own



- When the CPU scheduler selects a process for execution, the dispatcher loads the relocation and limit registers