Memory Management

Md. Asifur Rahman
Lecturer
Department of CSE, RUET

Addresses

- **Logical Address /Virtual Address:** Address Generated by CPU.
  - 32 bit processor can generate 2^32 address.
  - These addresses can be used to map to both physical and virtual address space.
- **Physical Address:** Address that actually exists in RAM.

User-level process has its own virtual address space. This term means simply: the set of memory addresses that the process can use without error. The address space is called “virtual” because the address numbers are not directly related to physical RAM addresses where the data resides.

When 32-bit addressing is in use, addresses can range from 0 to 0x7fffffff; (Numbers greater than 2^31 are in the IRIX kernel address space.) When 64-bit addressing is used, a process's address space can encompass 2^40 numbers. (The numbers greater than 2^40 are reserved for kernel address spaces.)

Logical Vs Physical Address

Logical Address : Address generated by CPU.

Physical Address: Address seen by the memory unit (which is loaded into the memory address register of memory)

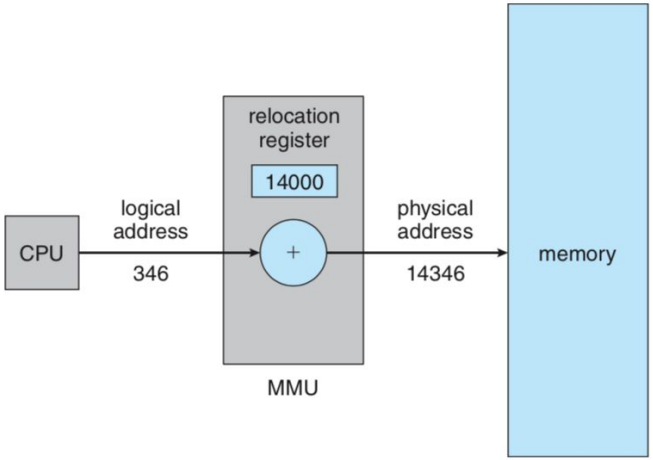


Figure 8.4 Dynamic relocation using a relocation register.

Address Binding

- Compile Time
- Load Time
  - Dynamic Loading (Only reference is loaded)
  - Static Loading (Entire program is loaded)
- Execution Time

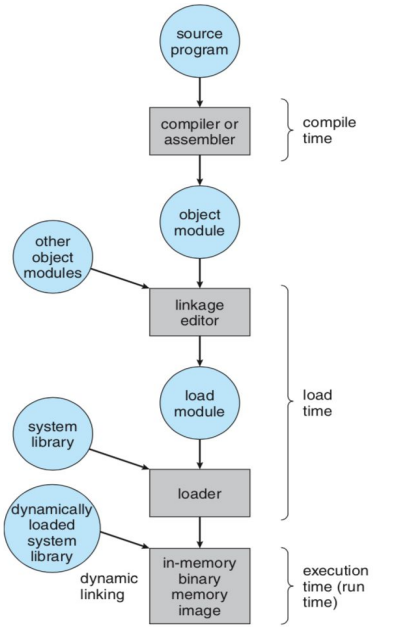
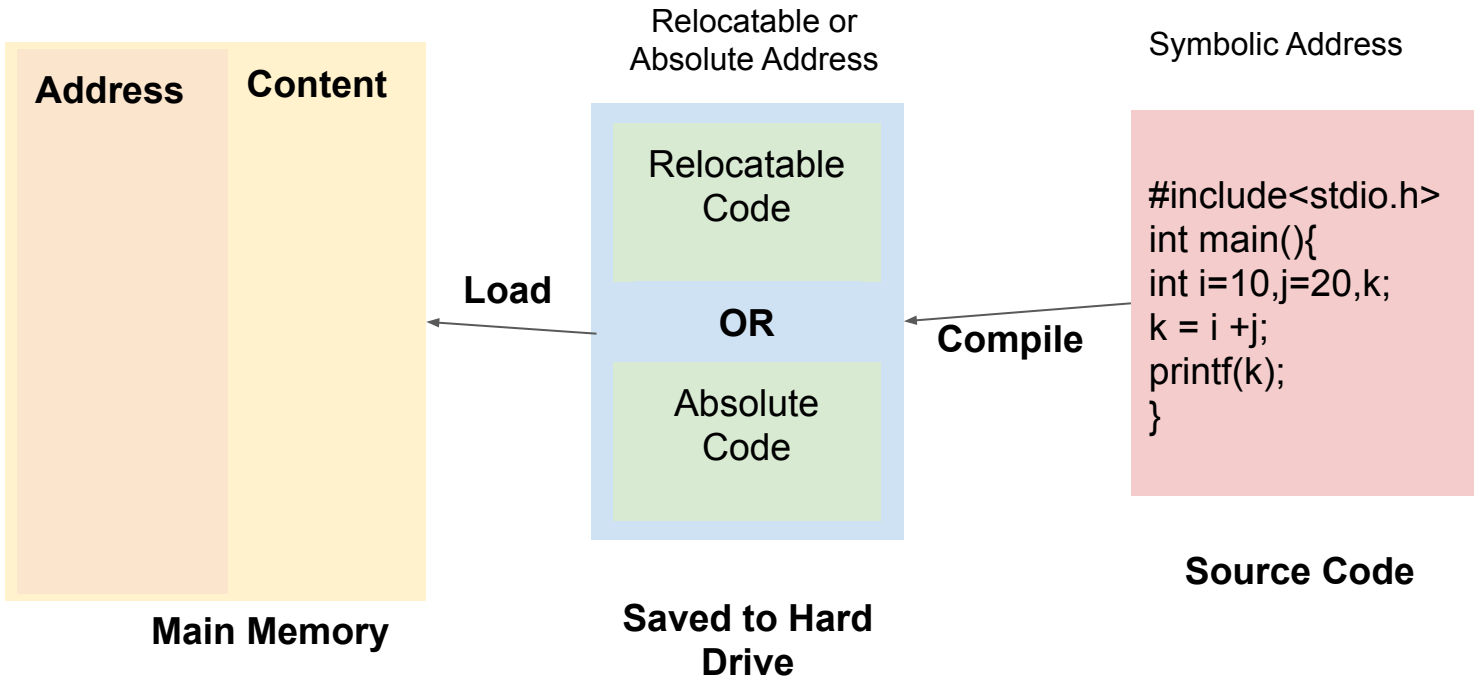


Figure 8.3 Multistep processing of a user program.

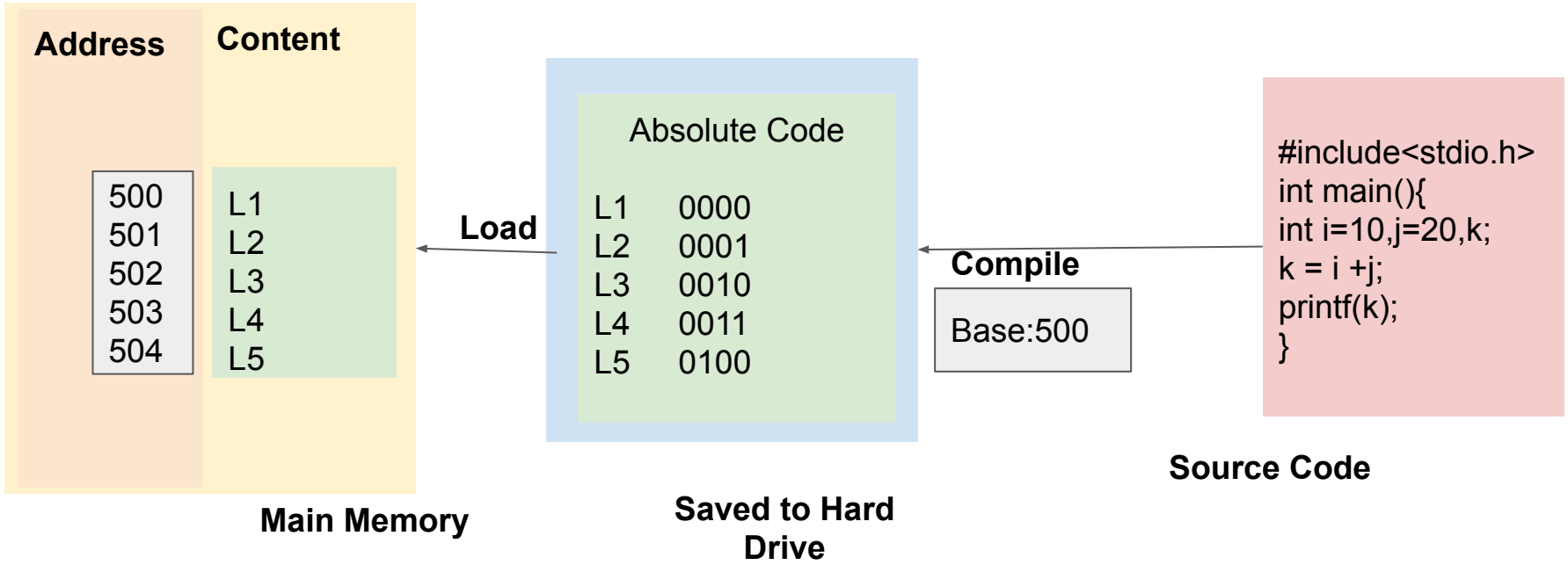
Address Binding



Address Binding

Compile Time Address Binding:

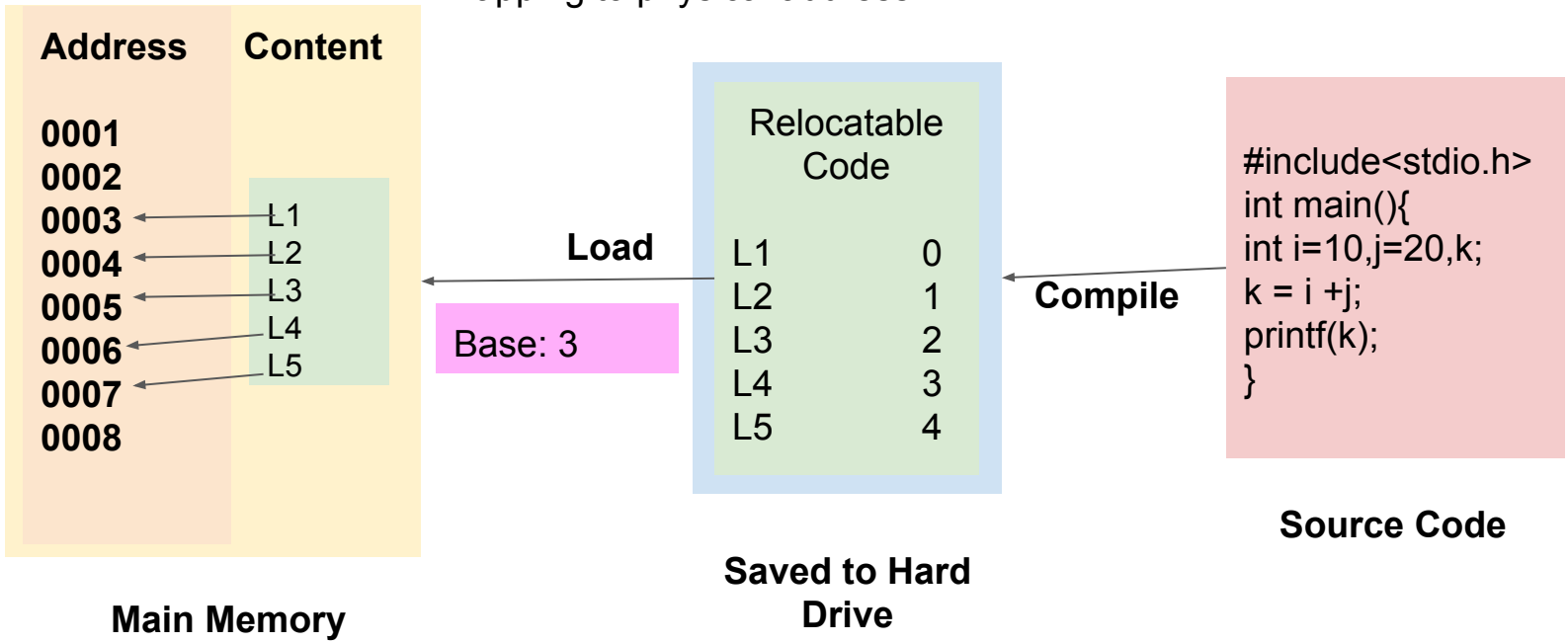
- The base address is known to compiler at compile time.



Address Binding

Load Time Address Binding:

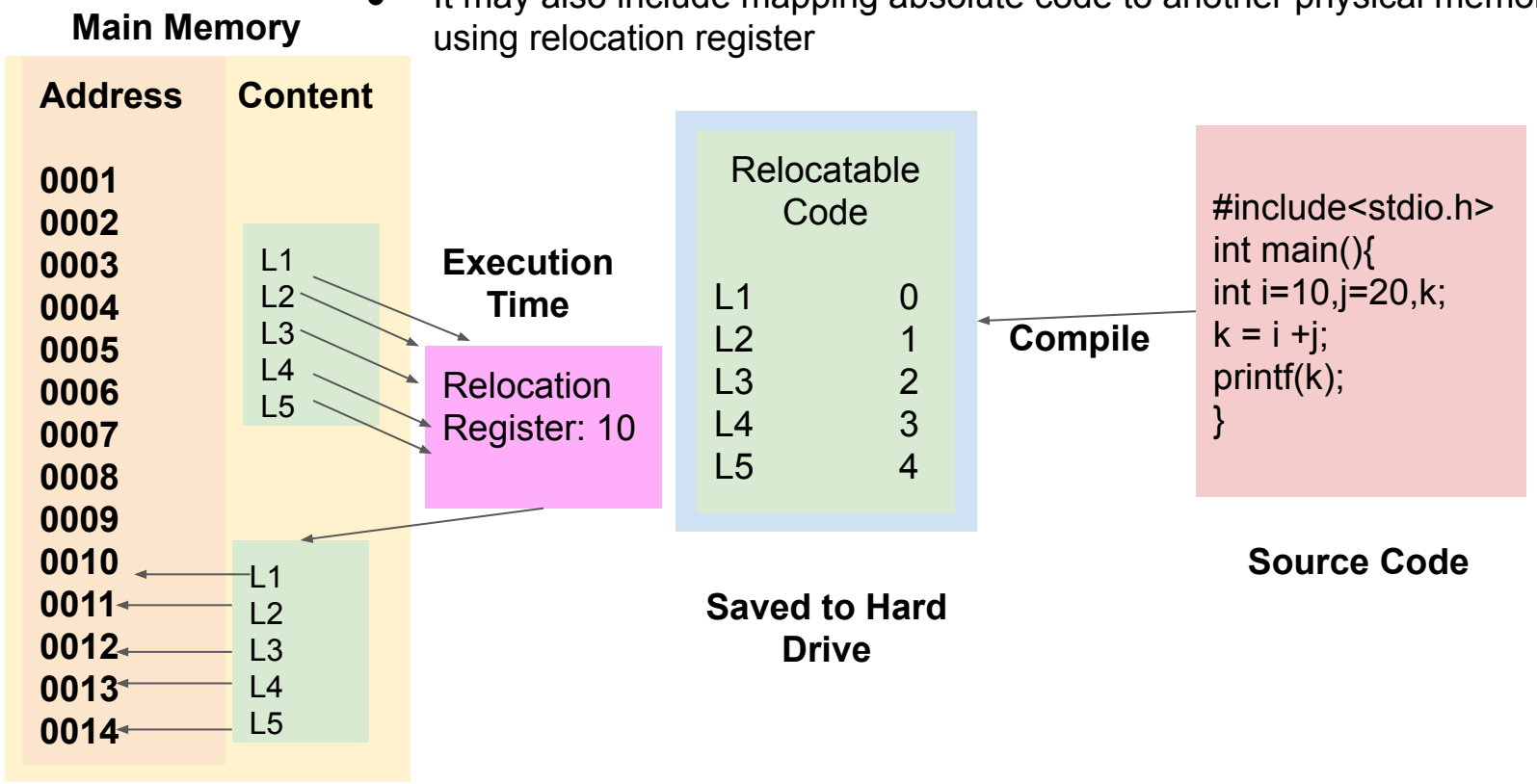
- The base address is known to memory management unit only for mapping to physical address.



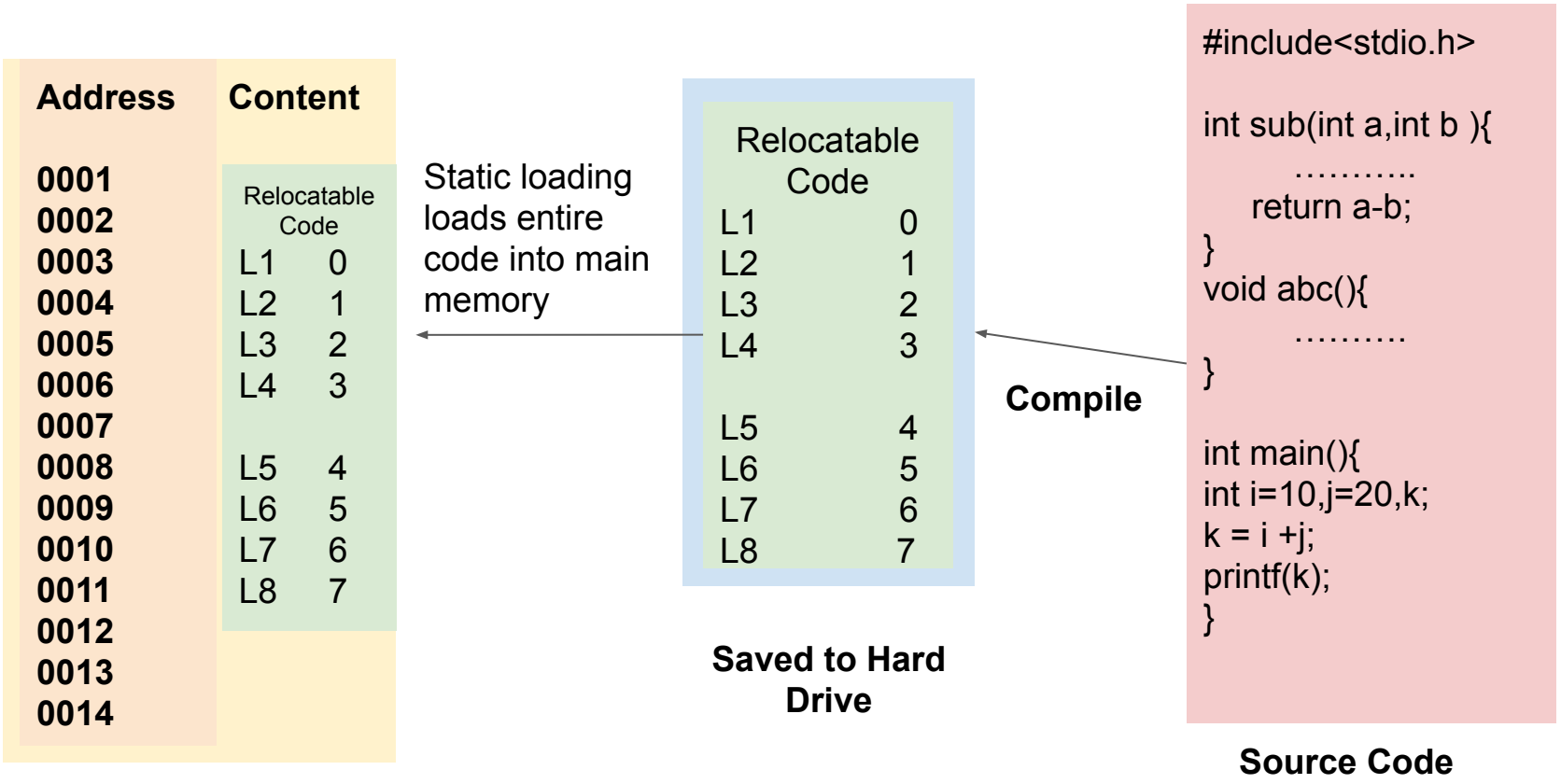
Address Binding

Load Time Address Binding:

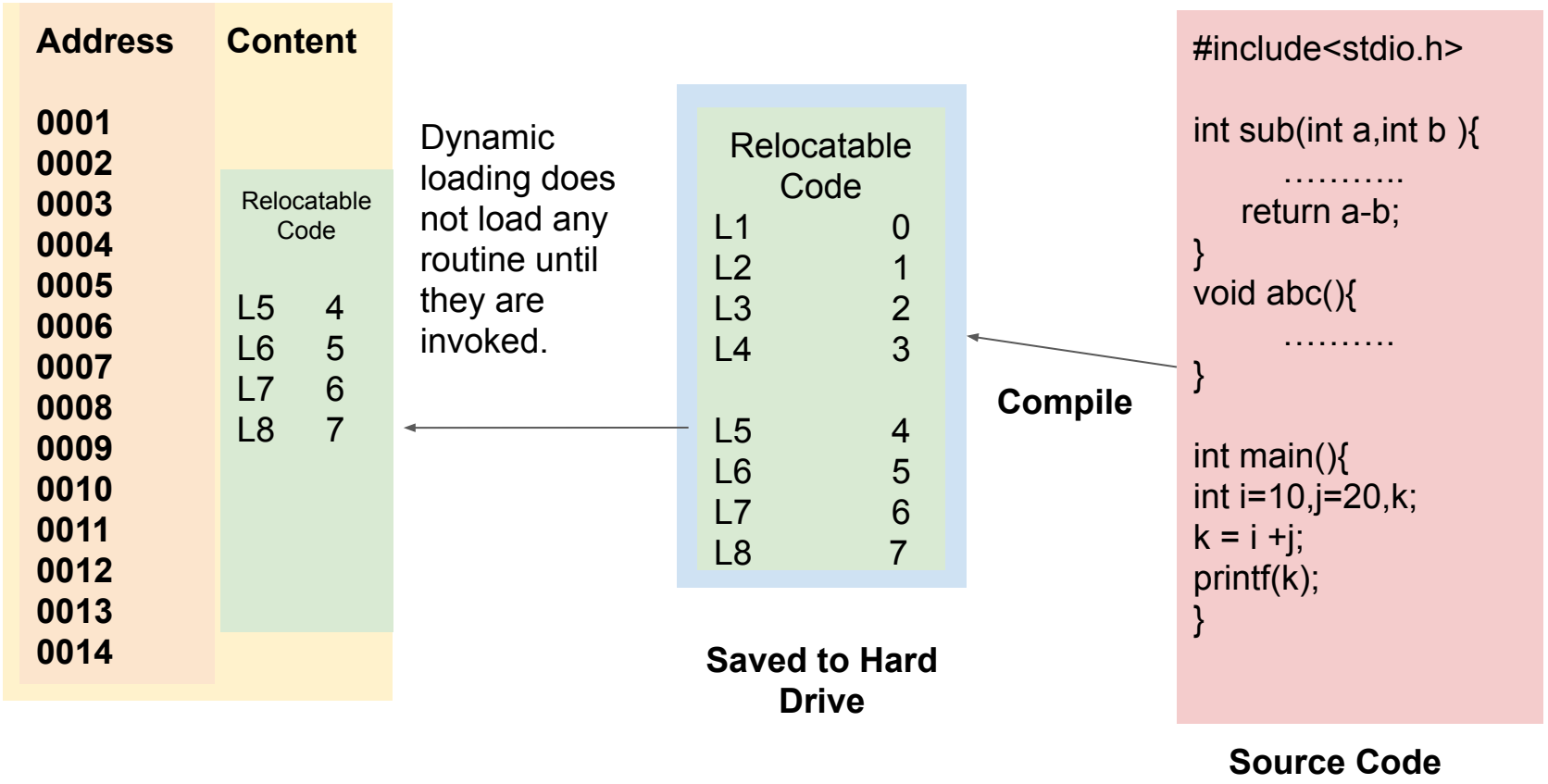
- It may also include mapping absolute code to another physical memory using relocation register



Static vs Dynamic Loading

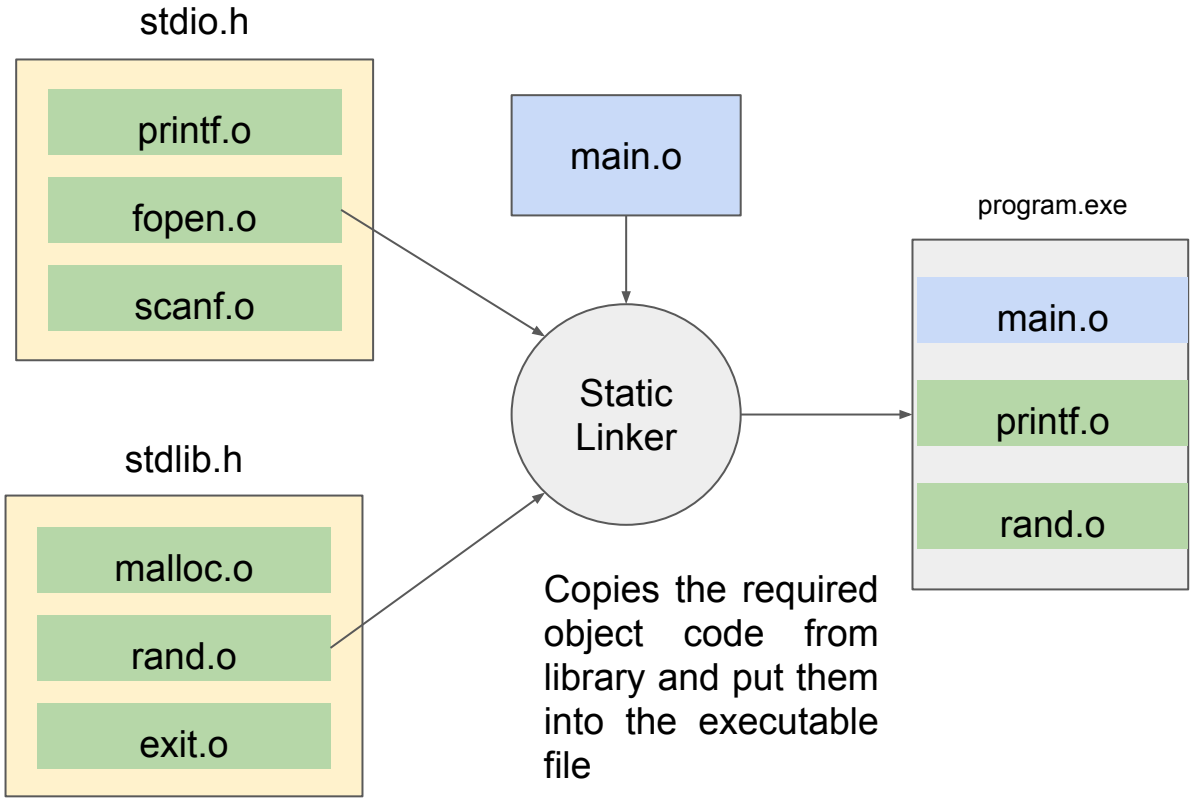


Static vs Dynamic Loading

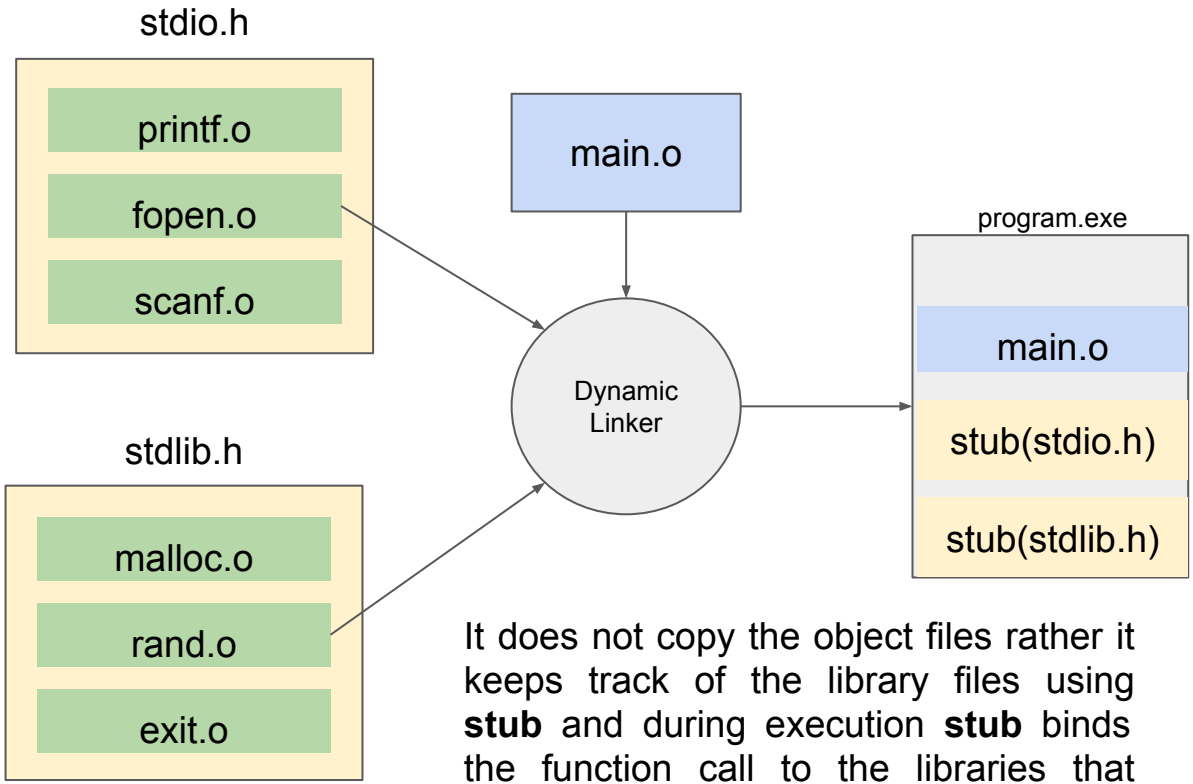


Static VS Dynamic Linking of Library

Execution Time Address Binding:



Static VS Dynamic Linking of Library



It does not copy the object files rather it keeps track of the library files using **stub** and during execution **stub** binds the function call to the libraries that reside in the disk

Self Study

- Difference and pros vs cons between **Dynamic loading and Static Loading** (Self Study)
- Difference and pros vs cons between **Dynamic linking and Static linking** (Self Study)
- Difference between physical and logical address (Self Study).

Swapping

- A backing store is a fast disk which is capable of accommodating all copies of memory images for all users and can provide direct access to those memory images.

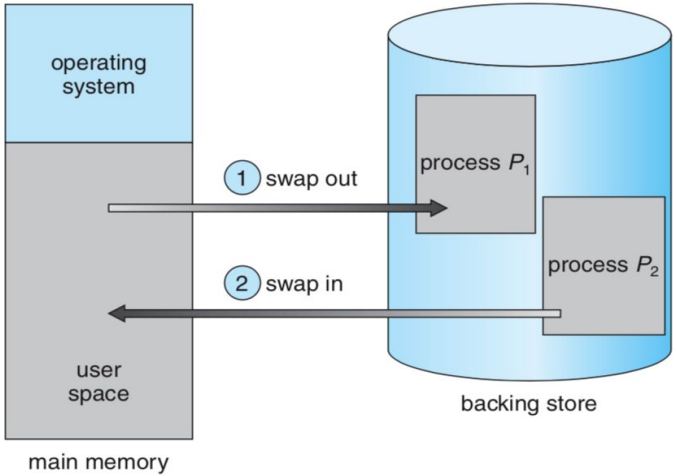
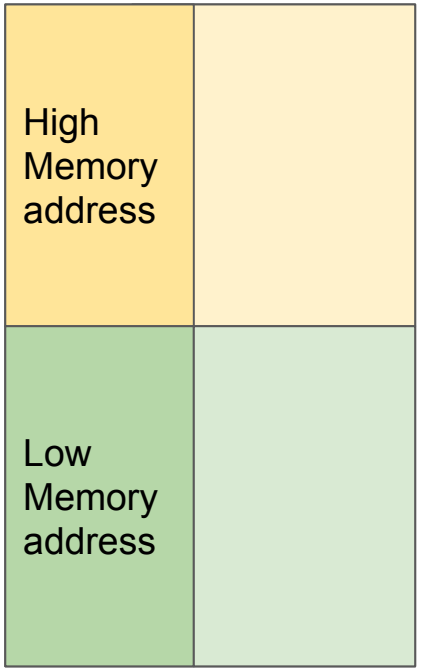


Figure 8.5 Swapping of two processes using a disk as a backing store.

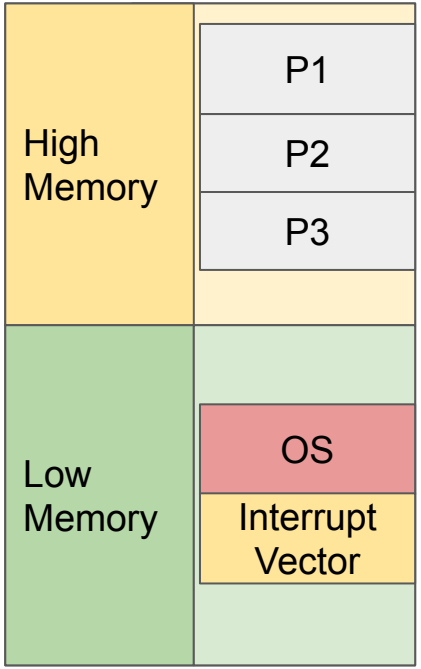
- System maintains a **ready queue** of processes whose memory images are on main memory or on backing store.
- **CPU scheduler** schedule a process and call **dispatcher** to run the scheduled process.
- Dispatcher checks whether the **scheduled process** is present in the main memory.
- If not then there is no free space in main memory.
- Dispatcher **swaps out** a process currently residing in main memory to backing store and **swaps in** the schedule process into main memory.

Main Memory

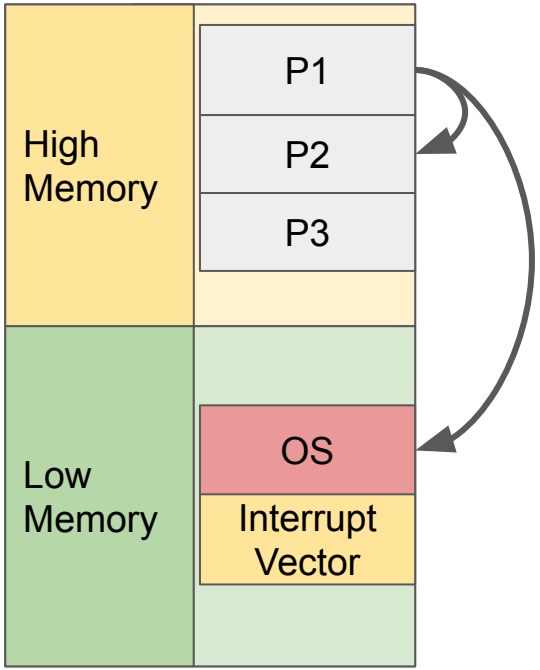


Main memory divided into two partition: one for OS and the other for user process

Contagious Memory



Memory Protection Issue



We need to prevent scenario Like :

- User process P1 trying to access memory portion where OS is residing
- User process P1 trying to access memory portion where other user process P2 resides.

Memory Protection

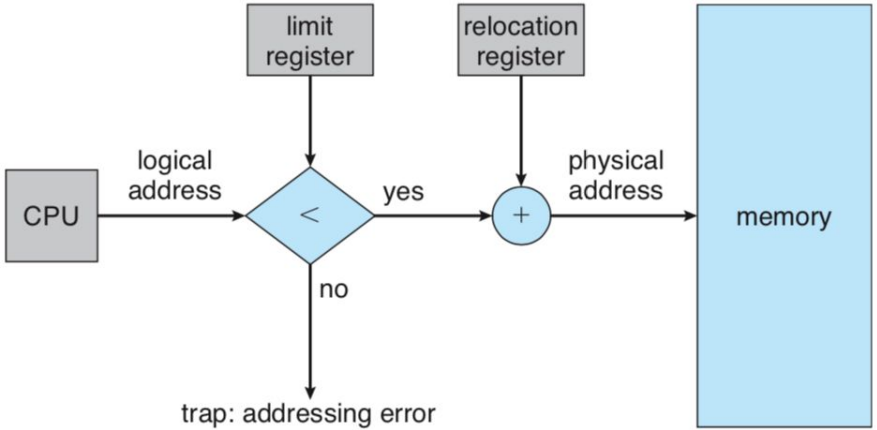
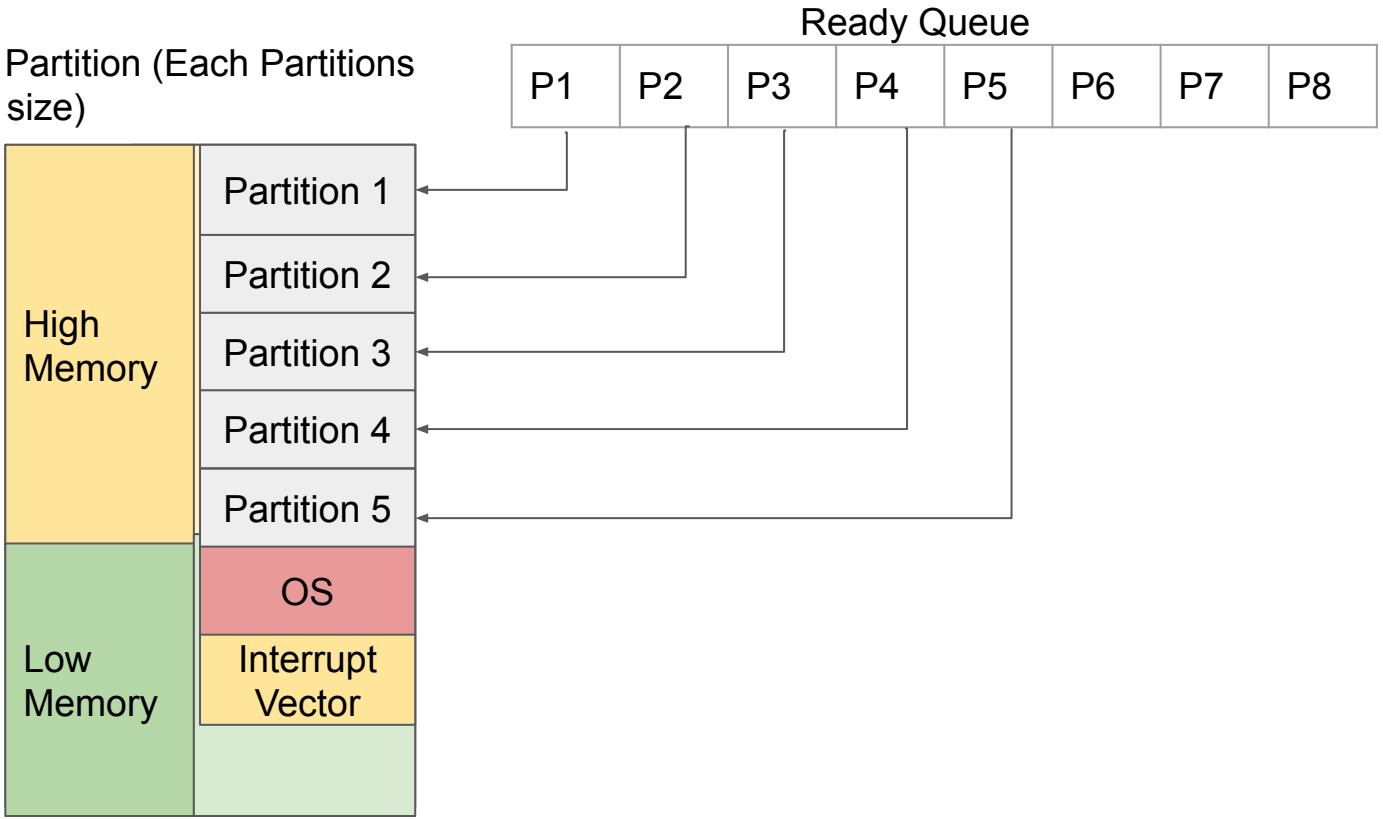


Figure 8.6 Hardware support for relocation and limit registers.

- When the CPU scheduler selects a process for execution, the dispatcher loads the relocation and limit registers with the correct values. MMU maps each logical address of the process to a physical address by checking relocation and limit registers.
- The relocation register contains the value of the smallest physical address
- The limit register contains the range of logical addresses. Each logical address must fall within the range specified by the limit register

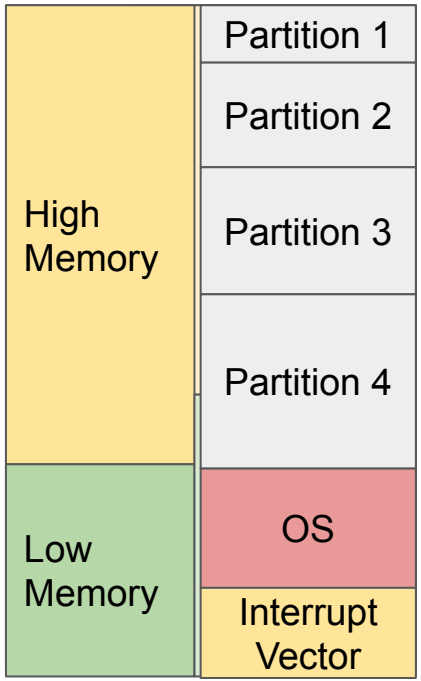
Memory Allocation

- Fixed Sized Partition (Each Partitions are of same size)



Memory Allocation

- Variable Sized Partition (Each Partitions are of different size)



Partition	State	Size
1	Free	100
2	Free	200
3	Free	300
4	Free	400

Solution to **Dynamic storage allocation problem**

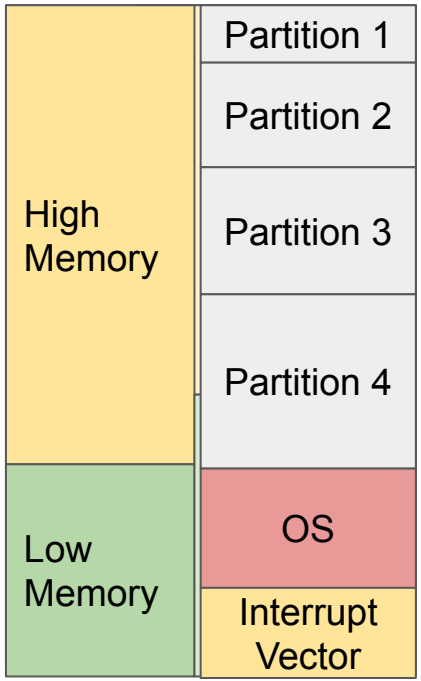
- First Fit
- Best Fit
- Worst Fit

Ready Queue

Process	P1	P2	P3	P4	P5	P6	P8
Require Memory	110	80	100	200			

Memory Allocation

- **First Fit:** Allocate the first hole that is big enough.Searching can start either at the beginning of the set of holes or at the location where the previous first-fit search ended.



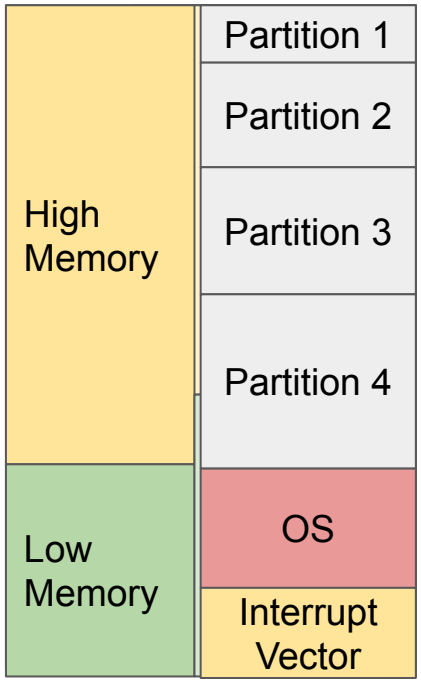
Partition	State	Size
1	Free	100
2	Free	200
3	Free	300
4	Free	400

Process	P1	P2	P3	P4	P5	P6	P8
Require Memory	150	90	390	280			

Ready Queue

Memory Allocation

- **Best Fit:** Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.



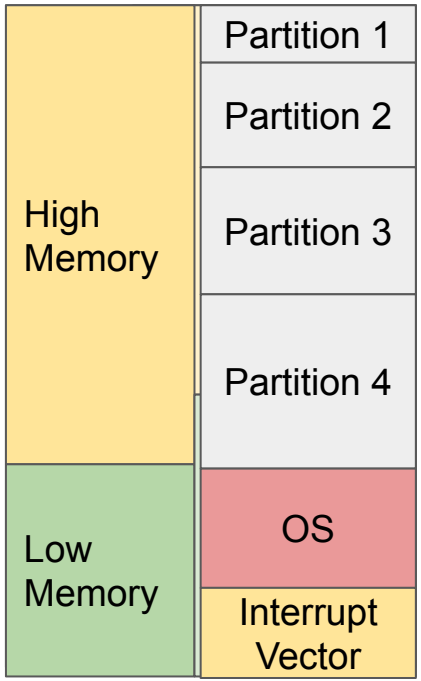
Partition	State	Size
1	Free	100
2	Free	200
3	Free	300
4	Free	400

Process	P1	P2	P3	P4	P5	P6	P8
Require Memory	150	90	390	280			

Ready Queue

Memory Allocation

- **Worst Fit:** Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole.



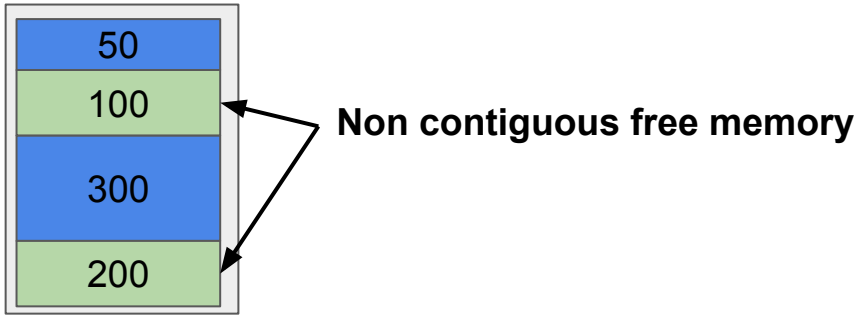
Partition	State	Size
1	Free	100
2	Free	200
3	Free	300
4	Free	400

Process	P1	P2	P3	P4	P5	P6	P8
Require Memory	150	90	390	280			

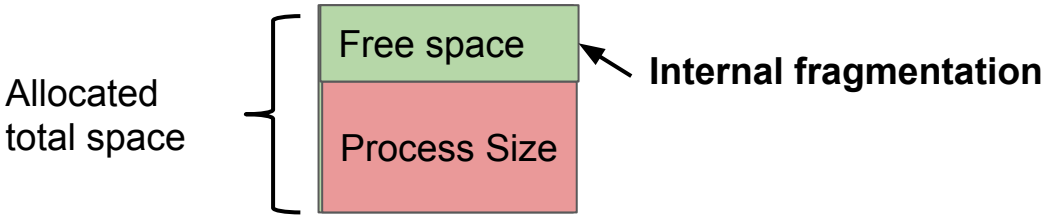
Ready Queue

Fragmentation

- **External Fragmentation:** When there is enough total memory space to satisfy request but the available spaces are not contagious.



- **Internal Fragmentation:** If memory allocated to a process is slightly bigger than the process, it leads to internal fragmentation



Solution to fragmentation

- **Compaction**:
  - Shuffle all the memory in order to place all the holes together to make a big chunks of contiguous free memory
  - Compaction is only possible if relocation is dynamic , that means it can be done in execution time.
- **Permit logical address of process non contagious**
  - Paging
  - Segmentation

Segment

- A logical address space is a collection of segments where each segment has a name and a length.
- The address specify both the segment name and the offset within the segment.
- A logical address therefore consists of two tuple:

<segment-number, offset>

- A **C compiler** might create separate segments for the followings:
    - The Code - Code Segment
    - Global Variable - (Global constant Object Code Segment), (Global Non Constant Object- Data Segment)
    - The Heap - Heap Segment
    - The Stack - Stack Segment
    - The Standard C library -

Segmentation Hardware

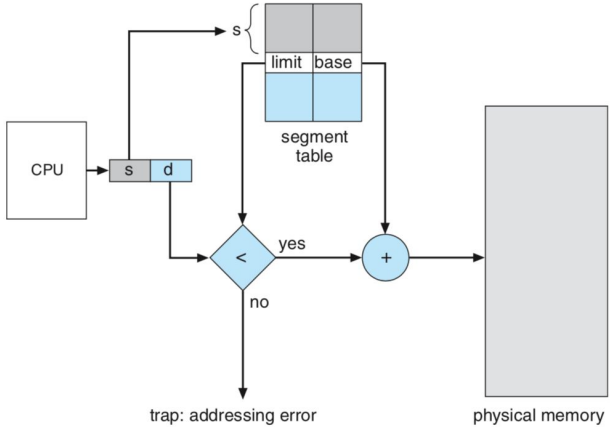
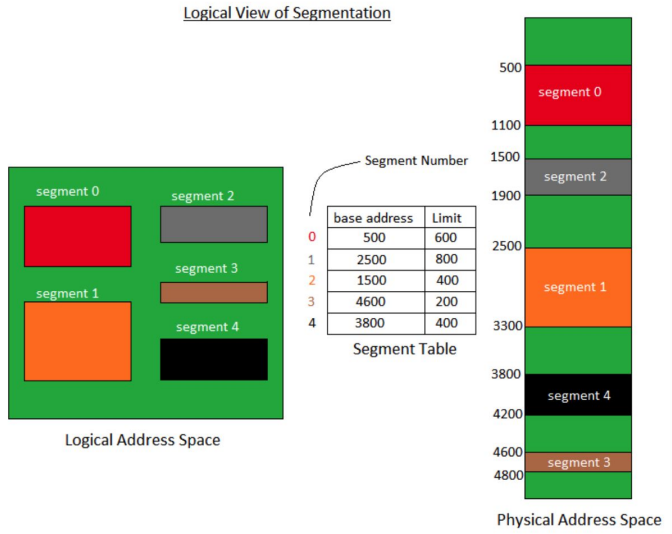


Figure 8.8 Segmentation hardware.



- Segment Table
  - Segment Base
  - Segment limit
- Logical Address :
  - Segment Number
  - Offset
- **Advantage** : Permits physical address space to be non contiguous. No internal fragmentation.
- **Disadvantage**: Can not avoid external fragmentation

Paging

- Physical Memory is broken into fixed sized blocks called **frames**
- Logical Memory is broken into same sized blocks called **pages**.
- Main memory and Backing store is divided into **frames**.
- **Page table** maps pages to frames
- **Frame table** keeps track of free of allocated frames.
- During execution corresponding pages of a process are loaded into available memory frame (Also swapped between frames of back store and main memory if necessary)

In contrast segmentation does not uses a fixed size partitioning scheme for logical and physical memory.

Paging Hardware

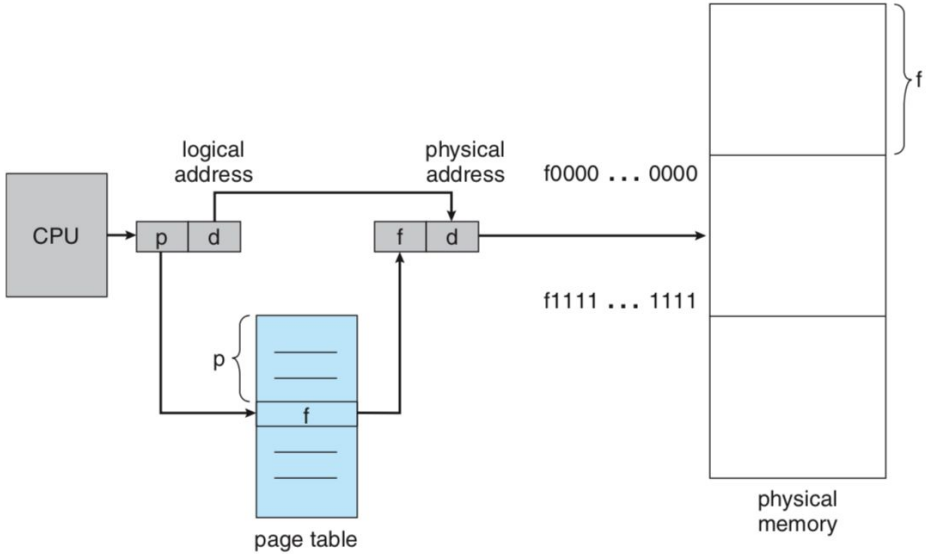


Figure 8.10 Paging hardware.

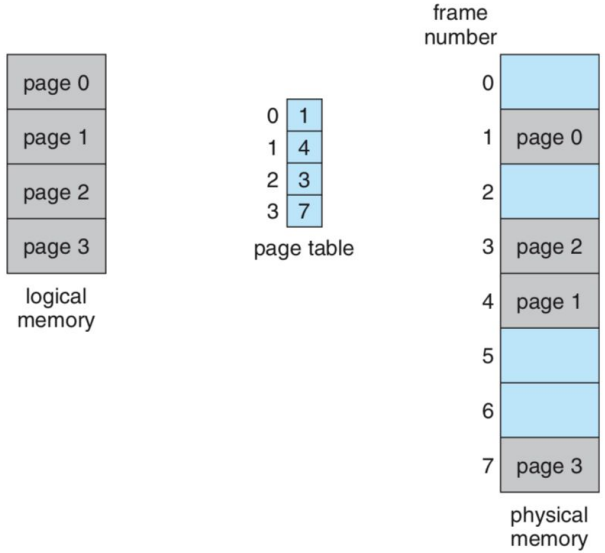


Figure 8.11 Paging model of logical and physical memory.

- Each logical address generated by CPU is broken into: **Page Number (p)** and **Page offset (d)**
- Page number is used as an index into **page table**
- From there corresponding **base address** of particular frame is combined with the **page offset** in order to generate the physical memory address

Paging Hardware

- Size of page size is a power of 2, varying between 512 bytes and 1 GB.
- If the size of logical address space is 2^m
- Then logical address represents as follow

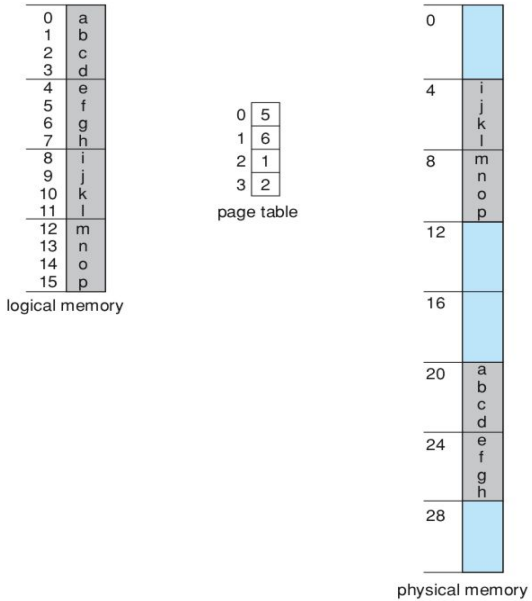


Figure 8.12 Paging example for a 32-byte memory with 4-byte pages.

Translation Look-Aside Buffer (TLB)

- In paging memory management scheme a page table is created and entry of each process is maintained in the page table.
- But this page table itself needs to be allocated in any one of the memory, RAM, Cache or Register
- For faster access page table is placed on a high speed cache named TLB
- Each entry in the TLB contains two parts: a **Key (or Tag)** and a **Value**
- **TLB lookup** : TLB is given a item, the item is compared with all the keys and if the item is found then corresponding value field is returned.

Translation Look-Aside Buffer (TLB)

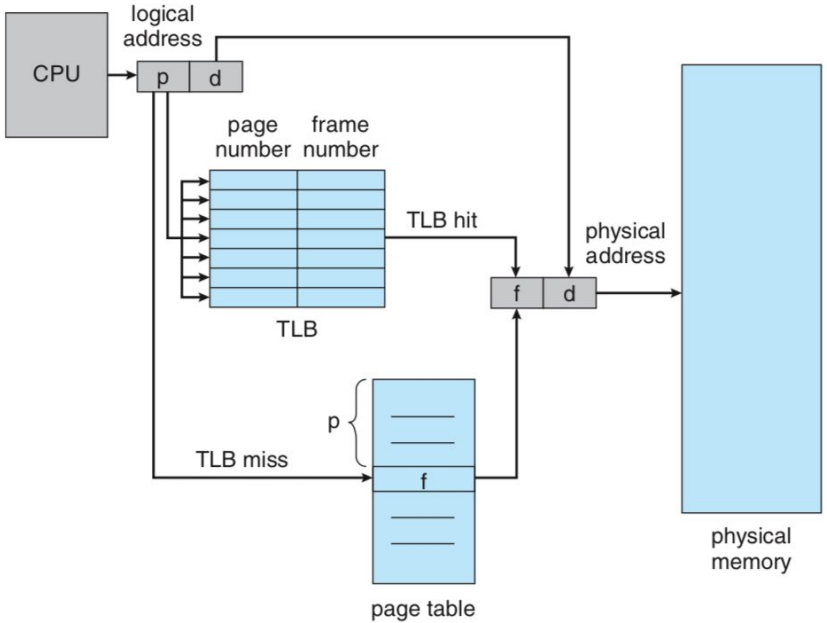


Figure 8.14 Paging hardware with TLB.

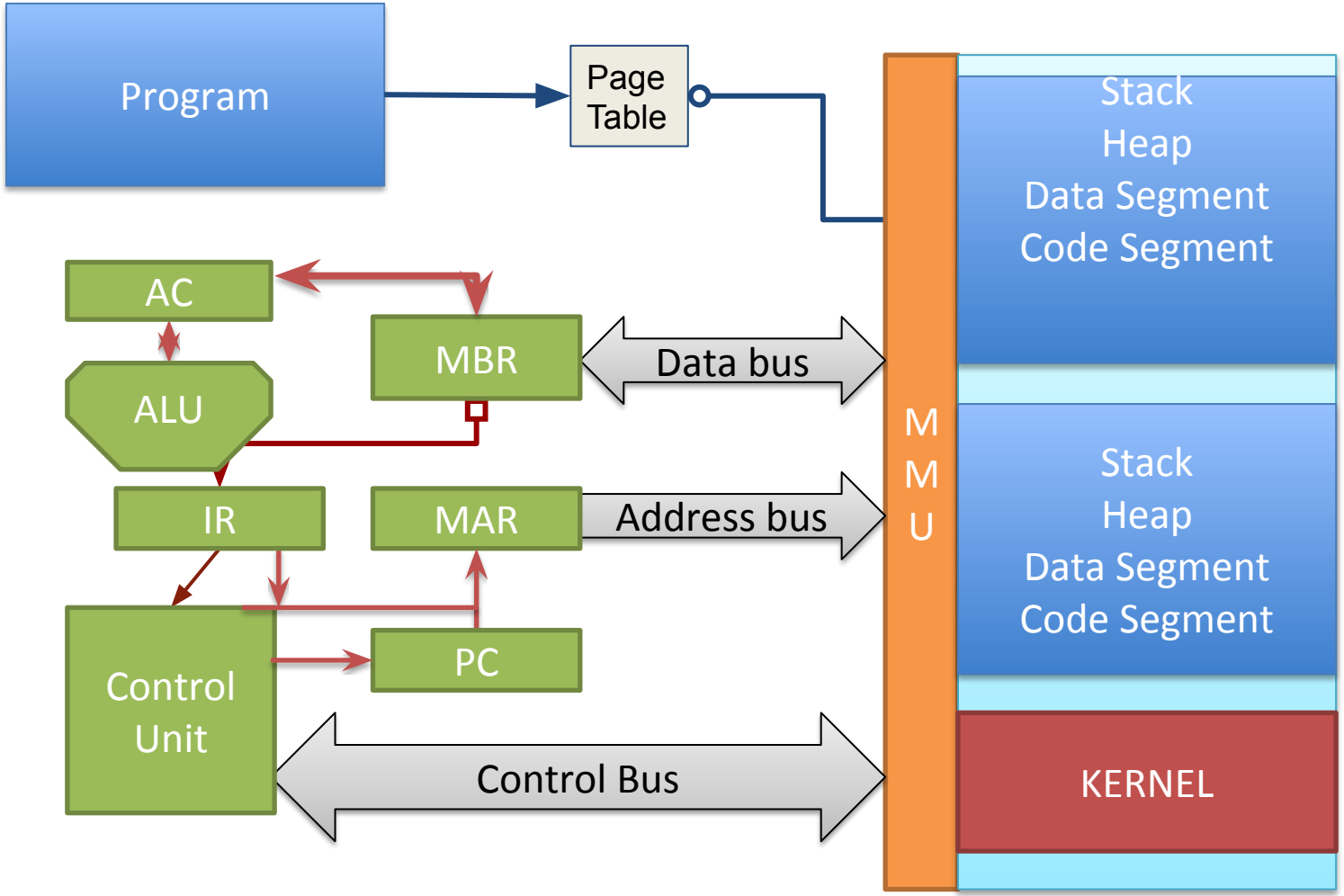
1. When a logical address is generated by the CPU its page number is presented to the TLB.
2. If the page number is found (Page hit) then its frame number is immediately available for use.
3. If the page number is not found (TLB miss), a memory reference to the page table **(In main memory kernel address space)** must be made. Later after obtaining the frame number it is used to access memory and at the same time the page number and the frame number is added to TLB. Now if the TLB is full then existing entry must be replaced.

Self Study

- Advantage and Disadvantage of Paging and Segmentation
- Protection is paging (8.5.3)
- Shared Pages (8.5.4)

References

1. Book -
    i. Modern Operating System – by Tanenbaum
    ii. Operating System Concepts - by Abraham Silberschatz
2. website: http://dthain.blogspot.com


Program
Page Table
Stack
Heap
Data Segment
Code Segment
AC
MBR
Data bus
ALU
IR
MAR
Address bus
MMU
Stack
Heap
Data Segment
Code Segment
PC
Control Unit
Control Bus
KERNEL