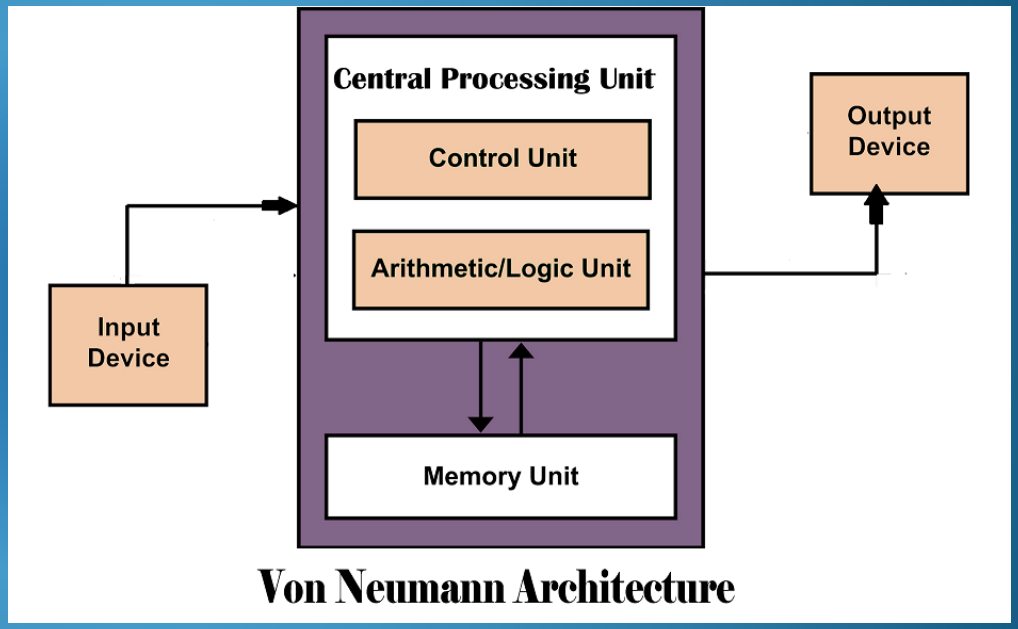
CS240 Operating Systems, Communications and Concurrency

Memory Management

In this section we look at operating system management of electrical memory



The von Neumann architecture refers to a 1945 model for a stored programme computer described by Princeton Mathematician John von Neumann and others which still forms the basis of modern computer operation.

CS240 Operating Systems, Communications and Concurrency

Von Neumann Architecture

The basic von Neumann stored computer has the following components:-

A memory used for holding both instructions and the data required by those instructions.

A control unit for fetching instructions from memory.

An arithmetic processor for performing the specified operations.

A bus for transferring data between memory and the CPU.

Input/Output mechanisms and peripheral devices for transferring data to and from the system.

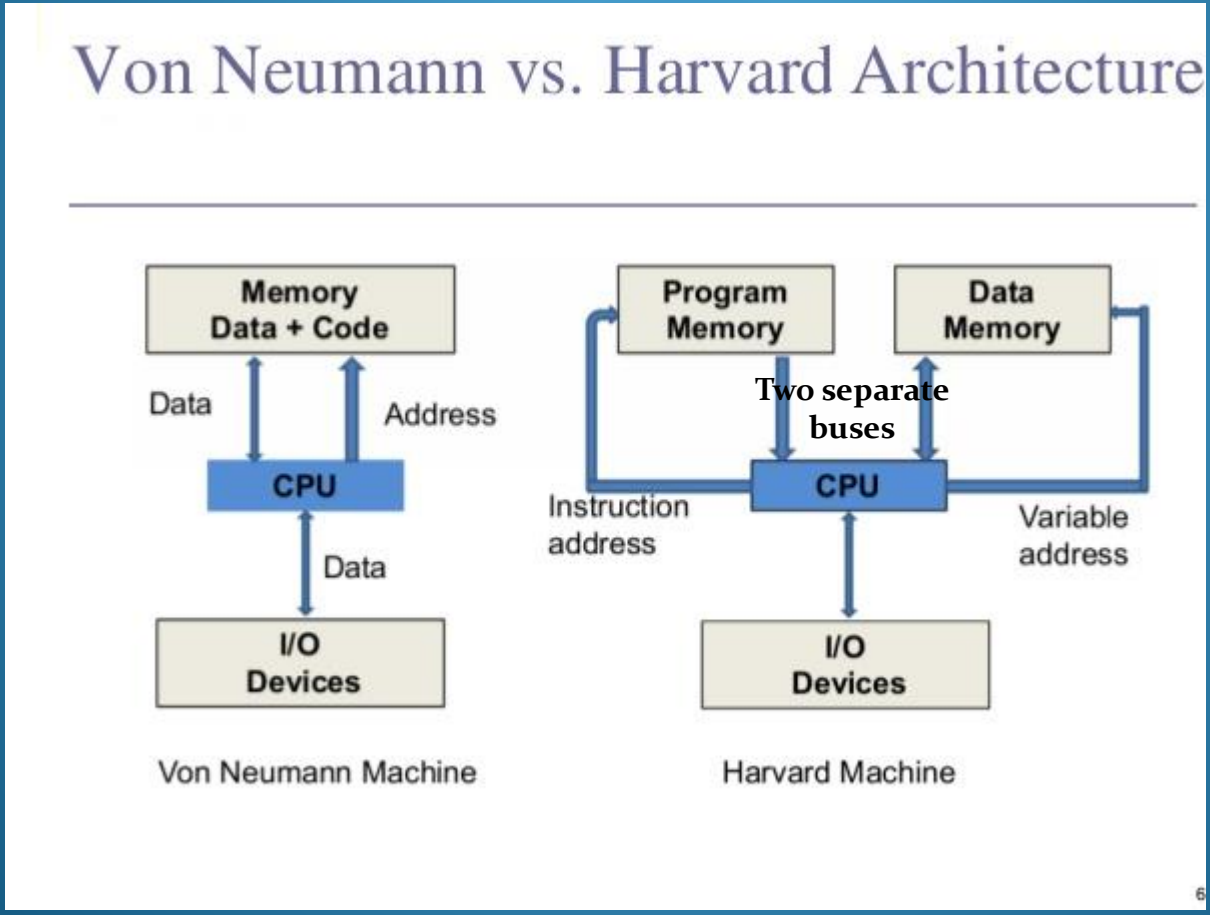
CS240 Operating Systems, Communications and Concurrency

Von Neumann Architecture

In the von Neumann architecture, there is only a single bus between memory and the CPU and so instruction fetching and data operations cannot occur at the same time. The memory is used for storing both instructions and data.

The Harvard Architecture proposed a system with separate pathways for instructions and data and separate memory systems allowing instruction fetching and data loading and storing to occur in parallel and independently.

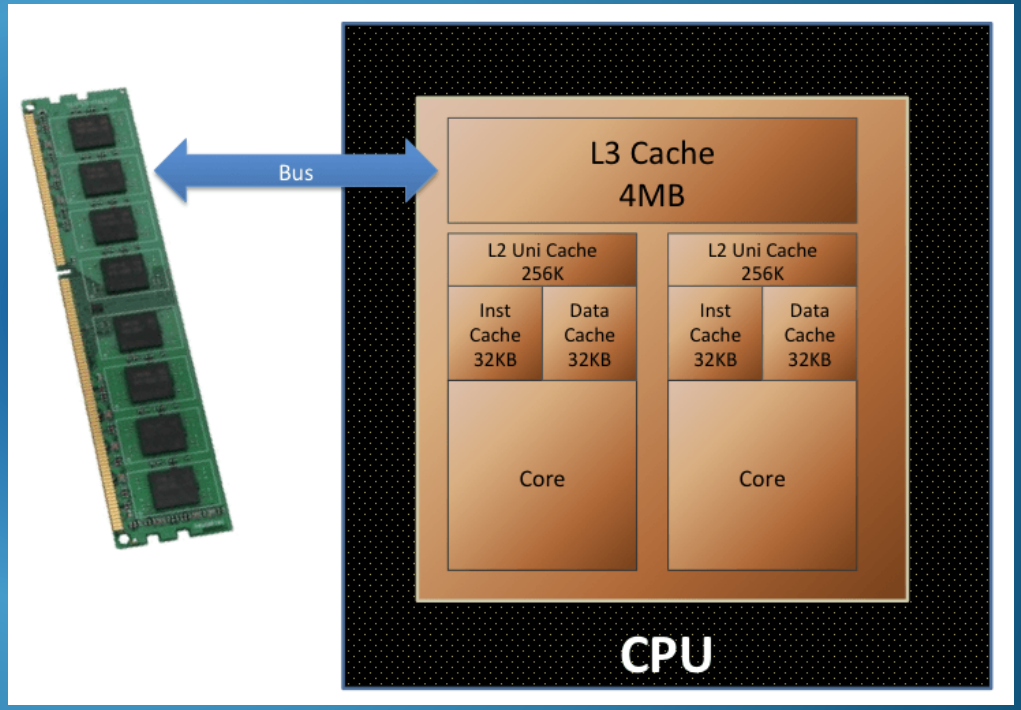
CS240 Operating Systems, Communications and Concurrency



CS240 Operating Systems, Communications and Concurrency

Modified Harvard Architecture

General purpose computers implement a modified Harvard architecture where both data and code can be stored together in a unified main memory, but separate CPU caches are used for instructions and data to achieve better performance.



CS240 Operating Systems, Communications and Concurrency

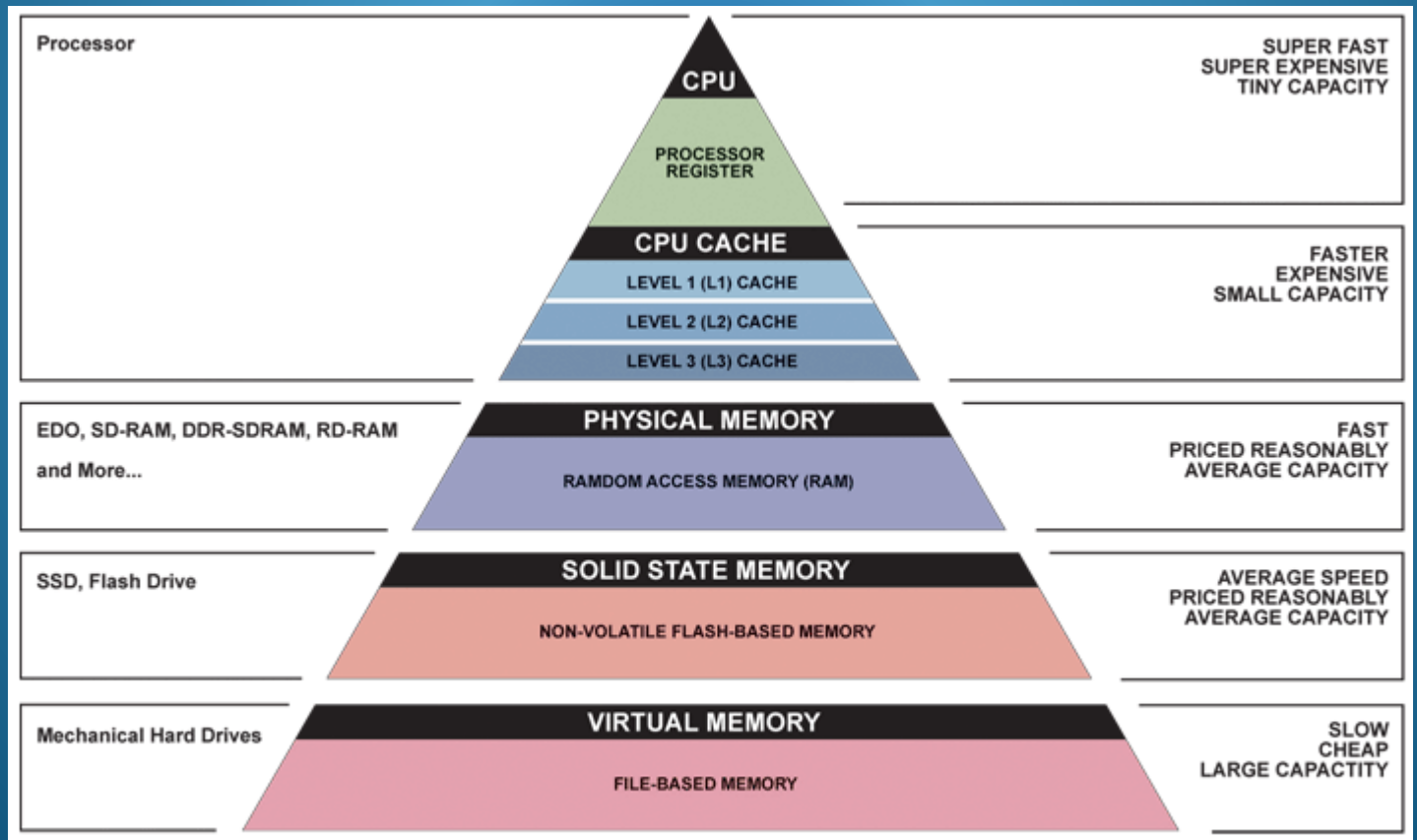
Memory not a component but a hierarchy of components

So, in reality, the system memory is composed of a hierarchy of levels where each level in the hierarchy uses storage technologies and interfaces offering different characteristics.

The memory system is a configuration of different storage technologies that meets a specific cost/capacity/performance objective.

CS240 Operating Systems, Communications and Concurrency

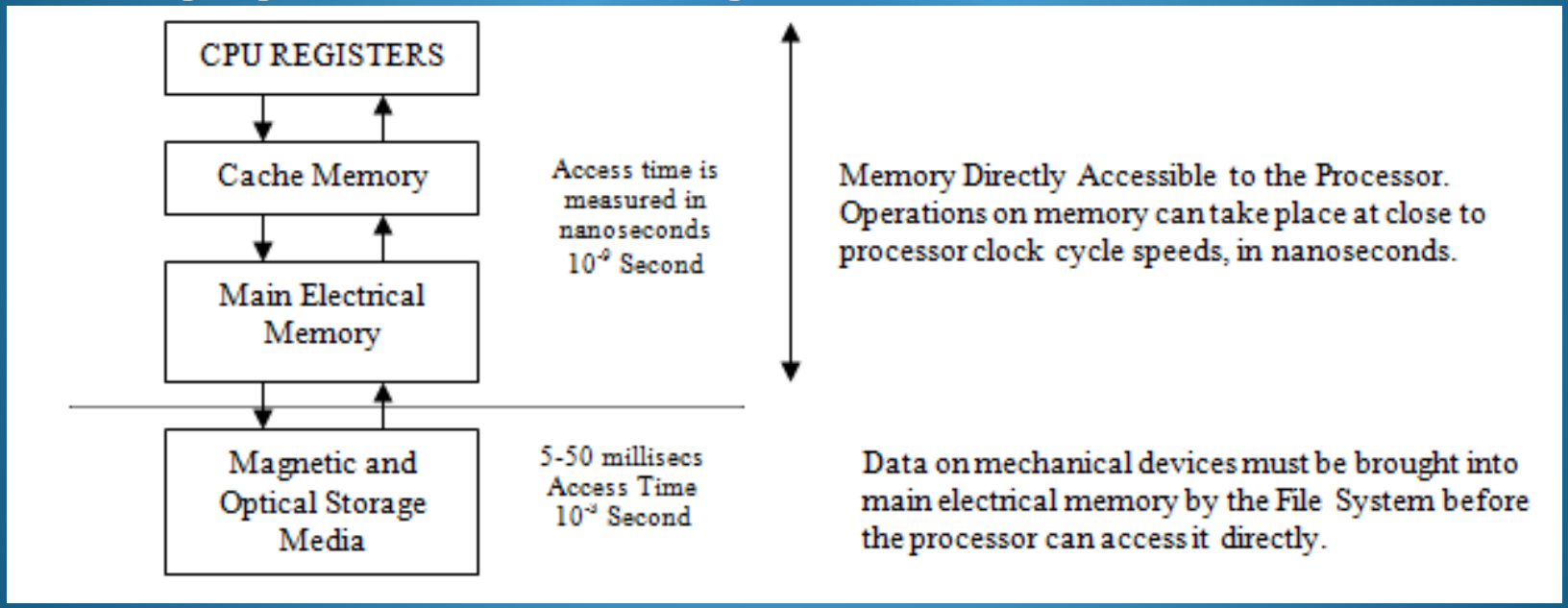
Memory Systems Hierarchy



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng

CS240 Operating Systems, Communications and Concurrency

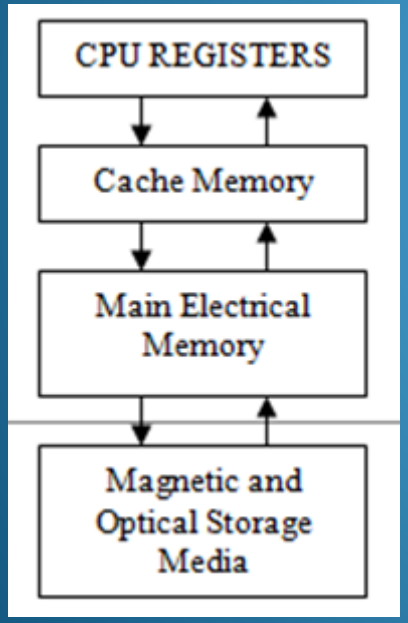
Memory Systems Hierarchy



Need for performance - A processor which is capable of executing a billion instructions per second cannot achieve that performance if these instructions cannot be supplied to the processor at comparable speeds by the memory.

CS240 Operating Systems, Communications and Concurrency

Memory Management – Need for Performance



The Mapping of Main Memory to the processor caches is done by hardware and not under software control, in order to achieve adequate performance.

Operating System Memory Manager is concerned with allocation and management of Main Memory and Secondary Storage Swap Space only.

CS240 Operating Systems, Communications and Concurrency

Memory Management

The electrical memory resource DRAM is quite critical for good system performance and this section focuses on the management of this resource.

The memory manager keeps track of which parts of memory are in use and which parts are free, it allocates memory to processes when they need it and deallocates it when they are finished.

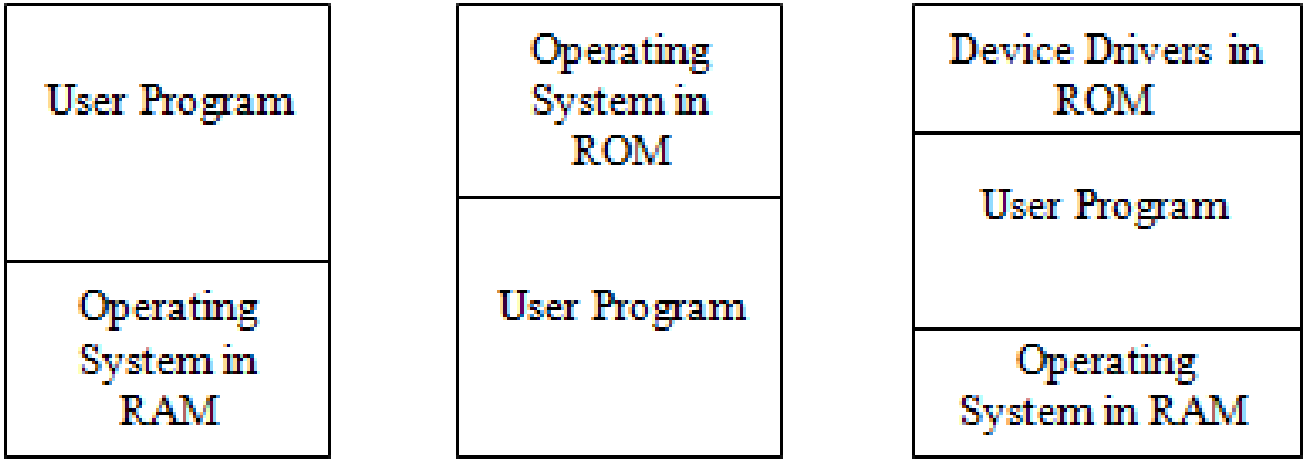
On virtual memory systems, the memory manager will manage swapping between the main memory and disk when the main memory is too small to hold the address space of all the processes.

CS240 Operating Systems, Communications and Concurrency

Memory Management

Monoprogramming

The simplest possible memory management scheme is to run one program at a time sharing the memory with the operating system. Some alternatives for this scheme are shown below.



CS240 Operating Systems, Communications and Concurrency

Memory Management

Monoprogramming

Operating systems that can run only one process at a time yield very poor device utilisation. The computer system's resources are largely idle, waiting for the activity of a single user, and no background activities can take place.

These models were used on very early computers or on embedded systems (systems with a specific application purpose as opposed to general purpose computing environments.)

CS240 Operating Systems, Communications and Concurrency

Memory Management

Multiprogramming

Multi-tasking and multi-user systems introduce some problems for the memory manager.

The memory space of one process needs to be protected from that of other processes. Otherwise malicious users or faulty processes could interfere with the memory space of correct processes.

CS240 Operating Systems, Communications and Concurrency

Memory Management

Multiprogramming

The second problem is that at the time a program is written , compiled and linked into a single binary image, it is unknown where the program is going to be located in memory.

Depending on what other processes are running, a program can be loaded into an arbitrary free area of memory.

The relative memory addresses used by the instructions in the program need to be mapped at run-time to the correct physical locations where the program is loaded.

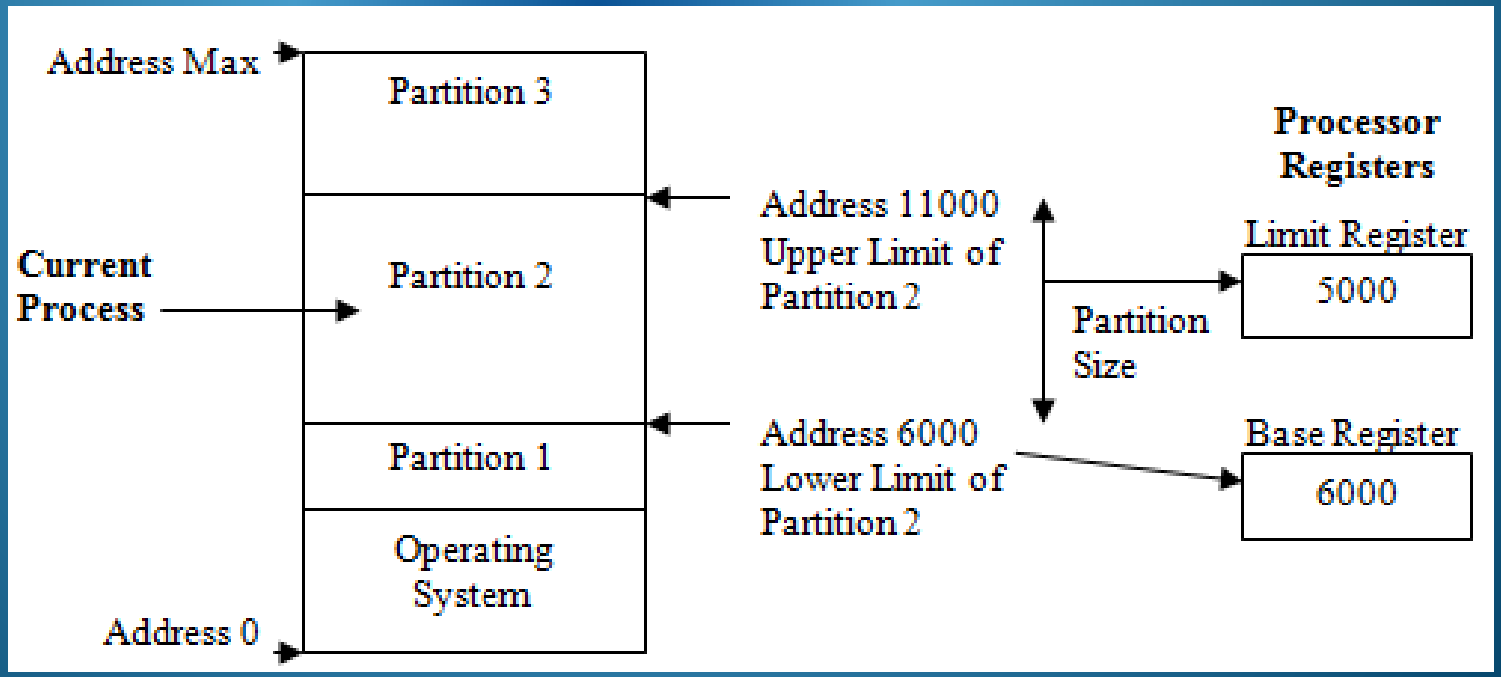
CS240 Operating Systems, Communications and Concurrency

Memory Partitioning needs hardware support in CPU

Access time to memory is vitally important, so the protection features and the dynamic relocation features are built into the hardware design for speed, rather than being implemented in software.

Consider Scheme 1

Fixed Sized Regions



CS240 Operating Systems, Communications and Concurrency

Dynamic Address Generation

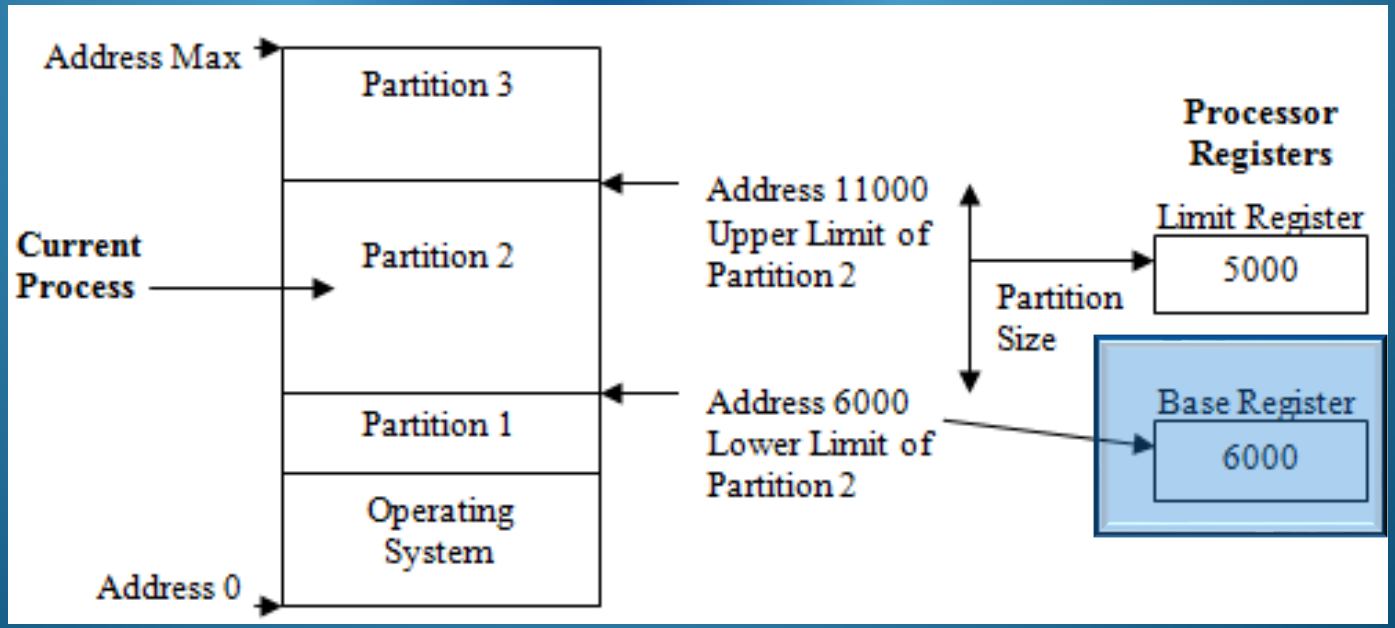
When a program is compiled and linked, it is assumed the program will be loaded beginning at address 0 in memory. All addresses of code and data in the binary image are generated relative to this position.

The process generates logical addresses in a range 0..max.

The hardware then translates these to real addresses by adding on the base register contents.

CS240 Operating Systems, Communications and Concurrency

When the program is actually loaded into a free partition, a base and limit value for the region are associated with the process. When the process is executing, the processor will automatically add on the value of the base to every address generated by the process before access to memory proceeds.



CS240 Operating Systems, Communications and Concurrency

Protection

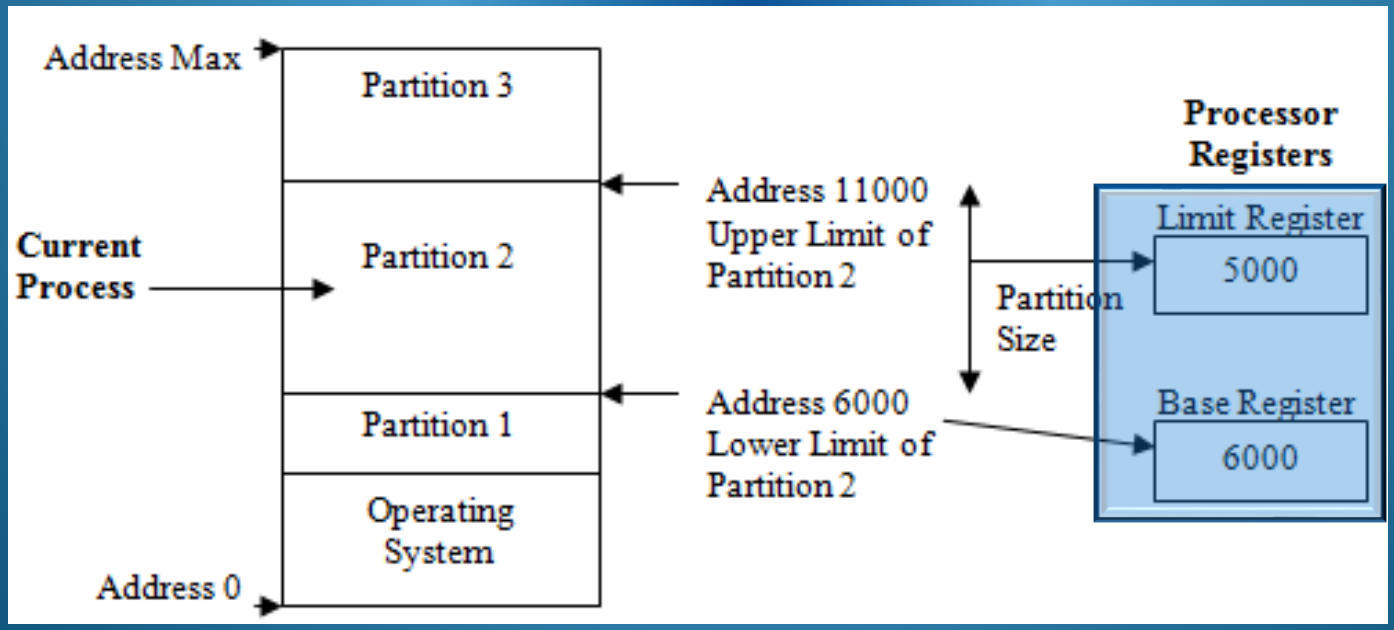
The mechanism prevents a process from accessing any physical memory below its base address as 0 is the lowest logical address the process can generate.

To prevent the process from accessing any memory above its partition area, the processor will compare every address generated by the process with the value stored in the limit register.

If the address generated is bigger than the limit value, the process has generated an address space exception. This would cause the operating system to take control and terminate the offending process.

CS240 Operating Systems, Communications and Concurrency

Hardware places boundary on range of logical addresses a process can generate between 0 and the value stored in Limit Register.

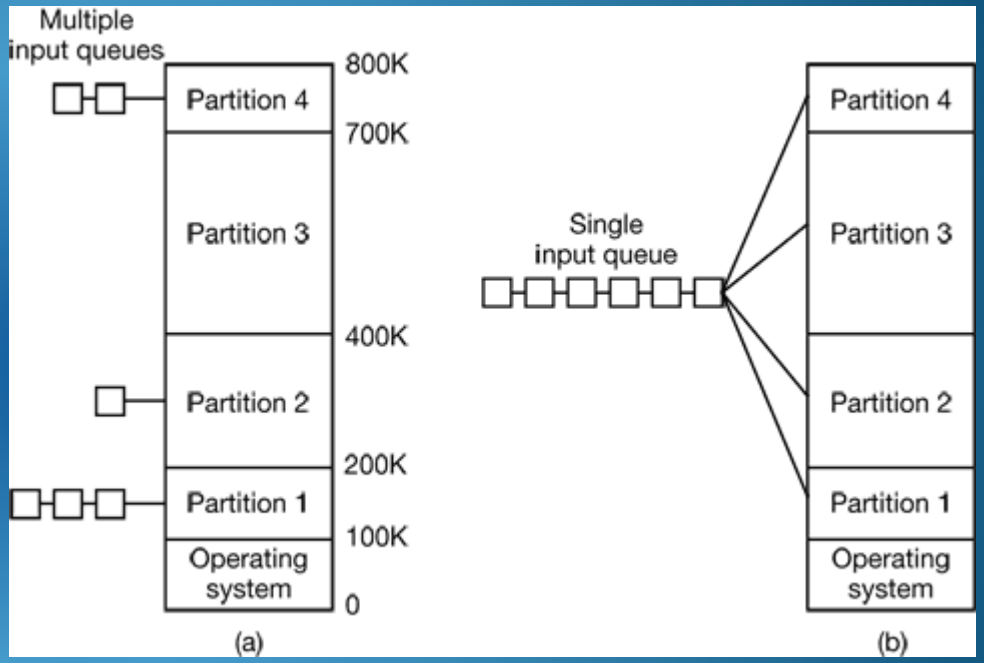


CS240 Operating Systems, Communications and Concurrency

Other Problems

Allocating Processes to Fixed Partitions

As partitions are of different sizes, we need to determine how to allocate processes to partitions.



One approach might be to have a separate queue for each partition for all the processes that can fit in a partition of that size. If the partition is in use, new processes must wait for it to become free before they can be loaded.

Another approach might be to have a single queue of processes waiting to be loaded and to load them into the first available partition that can accommodate them.

CS240 Operating Systems, Communications and Concurrency

Efficiency Concerns with Fixed Partitions Scheme

If a small process is allocated to a large partition, this might be wasteful of memory.

Remember, our hardware scheme only allows one process per partition.

On the other hand, there is no point having a long queue of processes seeking a small partition when queues to the larger partitions are empty.

CS240 Operating Systems, Communications and Concurrency

How big should the partitions be?

If partitions are going to be fixed in size, we also need to determine the number of partitions that should exist and what size each of them should be.

The largest partition size represents the largest process address space available and so our system will not be able to run processes with memory space requirements greater than our largest partition.

PROBLEM: We have enough physical memory space to run the process, but because the process cannot fit in our largest partition, we cannot run it. (External Fragmentation)

CS240 Operating Systems, Communications and Concurrency

How big should the partitions be?

This would argue in favour of a few large partitions rather than many small ones.

On the other hand, the larger the partitions that exist, the more likely that processes will not need all the space of the partitions allocated to them, leading to a wastage of available resources. (Internal Fragmentation)

It also means fewer processes available for multiprogramming possibly leading to poor device utilisation in other areas of the system.

CS240 Operating Systems, Communications and Concurrency

Scheme 2 - Dynamic Partitions

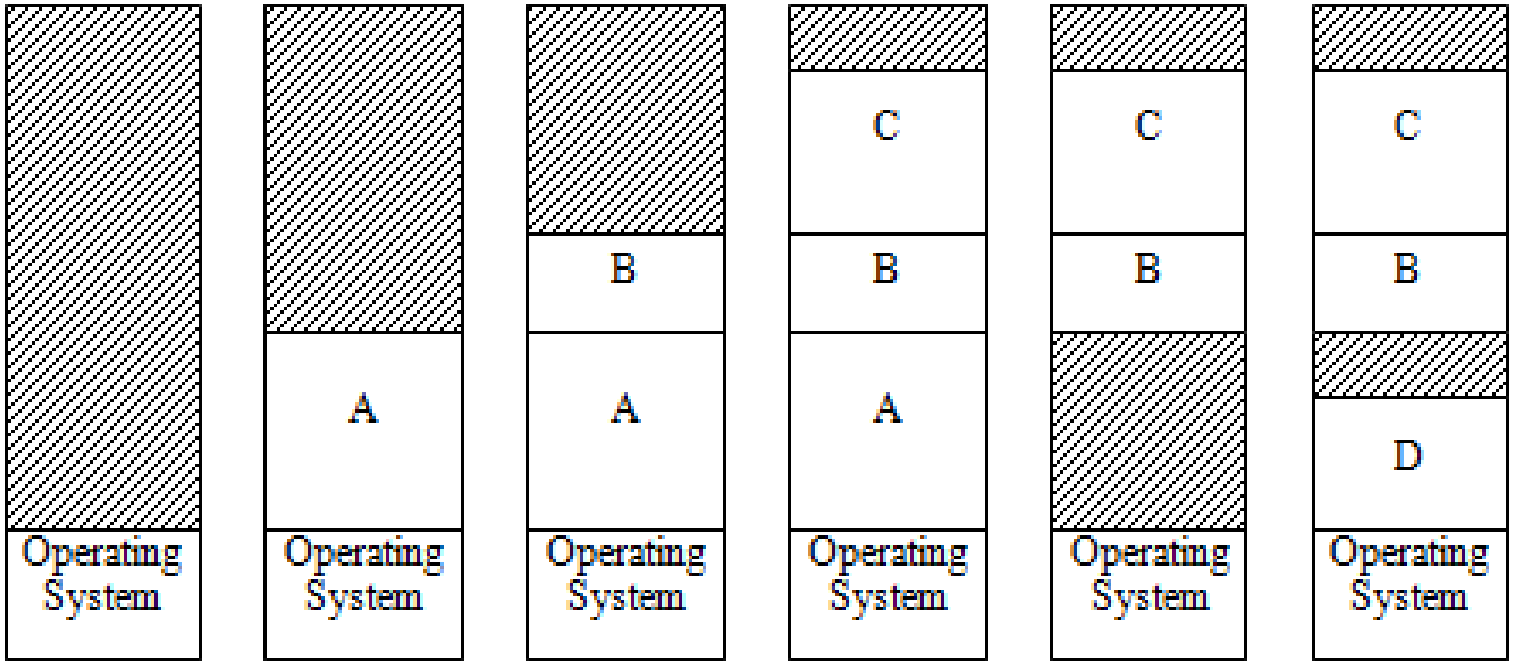
A solution to some of these problems might be for the memory manager to create, dynamically, a partition of the exact size from the available free memory space to satisfy each process request.

The number and size of partitions would be dynamic and the number and size would depend on what processes were actually in the system at a given point in time. With this approach, the memory manager must keep track of where the variable sized blocks are allocated and which areas of memory remain free for future requests.

CS240 Operating Systems, Communications and Concurrency

Scheme 2 - Dynamic Partitions

A possible sequence of memory space allocations and de-allocations over a period of time to processes A-D.



CS240 Operating Systems, Communications and Concurrency

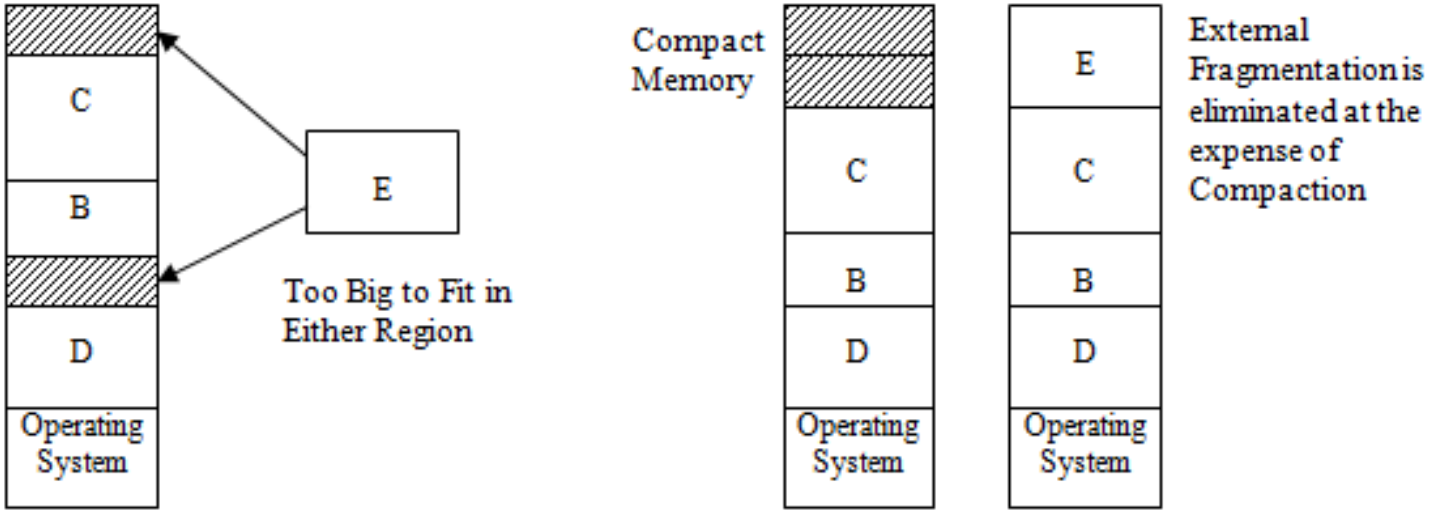
Scheme 2 - Dynamic Partitions

Note that over a period of time, the free regions of memory become fragmented into smaller and smaller pieces as available free areas are divided and apportioned to processes.

This can lead to a situation where the sum of the available free space can accommodate a new allocation request, but because the free space is not all part of one block, the allocation cannot proceed. (External Fragmentation)

CS240 Operating Systems, Communications and Concurrency

Scheme 2 - Dynamic Partitions



Compaction is the process of bringing together the free areas into larger combined free areas by relocating the address spaces of active processes to other regions.

CS240 Operating Systems, Communications and Concurrency

Free Memory Compaction

It is an expensive operation in terms of time as the processor has to physically move (read and write to another location) many words from the physical memory.

This is time spent on system activity rather than time spent executing user processes.

Compaction algorithms would be designed to minimise this movement.