



Memory Management

Operating System (CSC1036 & INF1036)



Dr. Hasin A. Ahmed
Assistant Professor
Department of Computer Science
Gauhati University



Operating System Types

- Batch operating system
 - Multi programming operating system
 - Multi tasking operating system
 - Multi processing operating system
- Real time operating system
- Network operating system
- Mobile operating system
- Distributed operating system





Batch Operating System

- Batch operating system: In the 1970s, Batch processing was very popular.
- In this technique, similar types of jobs were batched together and executed at a time.
- Example: Initial versions of OS/360 DOS/360 operating system released in 1960s
-

Multi programming operating system

- In a multiprogramming environment, when a process does its I/O, the CPU can start the execution of other processes.
- Therefore, multiprogramming improves the efficiency of the system.

Multi tasking operating system

- Multitasking OS enables a user to execute multiple computer tasks at the same time.

Multi processing operating system

- In Multiprocessing, there are more than one processors present in the system which can execute more than one process at the same time.

Real time operating system

- In Real-Time Systems, each job carries a certain deadline within which the job is supposed to be completed
- A delay incurs either a huge loss or the delayed result is completely useless.
- Examples: PSOS (Portable Software On Silicon), VRTX (Versatile Real-Time Executive), RT Linux (Real time operating system), Lynx OS, Windows CE

Hard and Soft Real Time System

- Hard and Soft real-time systems are the variants of real-time systems where the hard real-time system is more restrictive than the soft real-time system. The hard real-time system must assure to finish the real-time task within the specified deadline.
- In a soft real-time system, the meeting of deadline is not compulsory for every task, but the process should get processed and give the result. Even the soft real-time systems cannot miss the deadline for every task or process according to the priority it should meet the deadline or miss the deadline.

Network Operating System

- A network operating system connects multiple devices and computers on the network and allows them to share resources on the network.

Mobile Operating System

- A mobile operating system (OS) is software that allows smartphones, tablet PCs (personal computers) and other devices to run applications and programs.
- Examples: Android, iOS, Symbian, KaiOS, Sailfish OS, Harmony OS

Distributed Operating System

- In this type of operating systems, various autonomous interconnected computers communicate with each other using a shared communication network.
- Independent systems possess their own memory unit and CPU.
- These are referred to as loosely coupled systems or distributed systems.
- Examples: Solaris, OSF/1, Micros, Dynix, Locus, Mach



Memory Management

- In a uniprogramming system, main memory is divided into two parts: one part for the operating system (kernel) and one part for the program currently being executed.
- In a multiprogramming system, the “user” part of memory must be further subdivided to accommodate multiple processes.
- The task of subdivision is carried out dynamically by the operating system and is known as memory management.



• **Requirements of Memory Management**

- **Relocation**

- **Protection**

- **Sharing**

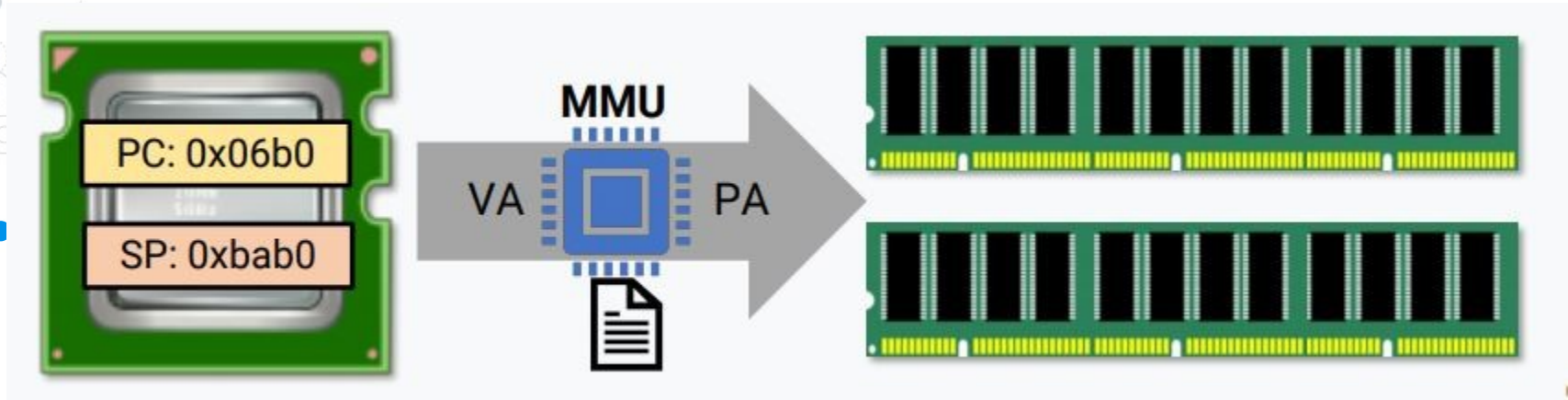
- **Logical organization**

- **Physical organization**

Physical vs Logical (Virtual) Address

- Logical Address space: An address generated by the CPU is known as “Logical Address”. It is also known as a Virtual address. The set of all logical addresses is known as Logical address space.
- Physical Address space: An address seen by the memory unit (i.e the one loaded into the memory address register of the memory) is commonly known as a “Physical Address”.
- A Physical address is also known as a Real address. The set of all physical addresses corresponding to these logical addresses is known as Physical address space.
- The run-time mapping from virtual to physical addresses is done by a hardware device **Memory Management Unit(MMU)**.

Physical vs Logical (Virtual) Address





Relocation

- We cannot know ahead of time where a program will be placed, and we must allow for the possibility that the program may be moved about in main memory.
- These facts raise some technical concerns related to addressing.
- The processor hardware and operating system software must be able to translate the memory references found in the code of the program into actual physical memory addresses, reflecting the current location of the program in main memory.



Protection

- Each process should be protected against unwanted interference by other processes, whether accidental or intentional
- Thus, programs in other processes should not be able to reference memory locations in a process for reading or writing purposes without permission
- All memory references generated by a process must be checked at run time to ensure that they refer only to the memory space allocated to that process.



Sharing

- Any protection mechanism must have the flexibility to allow several processes to access the same portion of main memory.
- For example, if a number of processes are executing the same program, it is advantageous to allow each process to access the same copy of the program rather than have its own separate copy.
- Processes that are cooperating on some task may need to share access to the same data structure.
- The memory management system must therefore allow controlled access to shared areas of memory without compromising essential protection.



Logical Organization

- Most programs are organized into modules, some of which are unmodifiable (read only, execute only) and some of which contain data that may be modified.
- If the operating system and computer hardware can effectively deal with user programs and data in the form of modules of some sort, then a number of advantages can be realized.
 - ➔ Modules can be written and compiled independently
 - ➔ Different degrees of protection (read only, execute only) can be given to different modules
 - ➔ It is possible to introduce mechanisms by which modules can be shared among processes.

Physical Organization

- In a two-level scheme, the organization of the flow of information between main and secondary memory is a major system concern
- The responsibility for this flow could be assigned to the individual programmer, but this is impractical and undesirable for two reasons:
 - The main memory available for a program plus its data may be insufficient. In that case, the programmer must engage in a practice known as overlaying that wastes programmer's time
 - In a multiprogramming environment, the programmer does not know at the time of coding how much space will be available or where that space will be.
- So the task of moving information between the two levels of memory should be a system responsibility.

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. Some nodes are highlighted with blue circles, and others with blue dots. The lines are thin and gray, creating a mesh-like structure.

MEMORY PARTITIONING

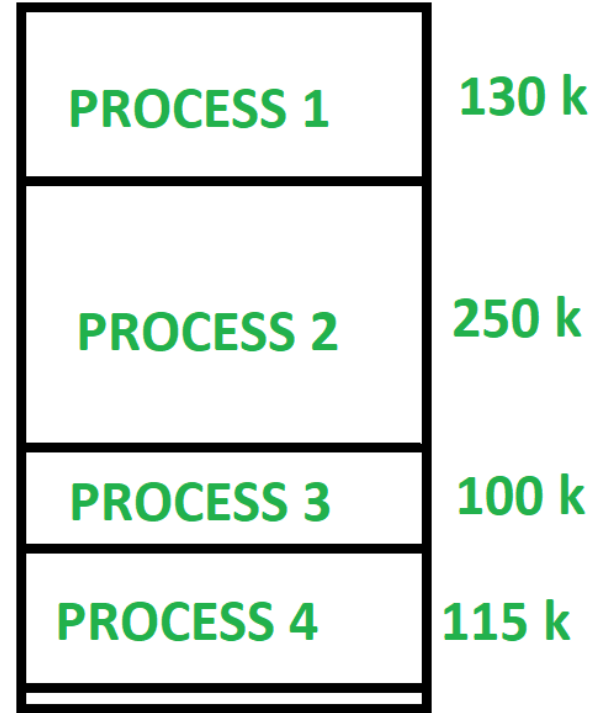
A decorative network diagram in the bottom-right corner, similar to the one in the top-left. It shows a network of nodes and lines, with some nodes highlighted by blue circles and others by blue dots.

Memory Partitioning

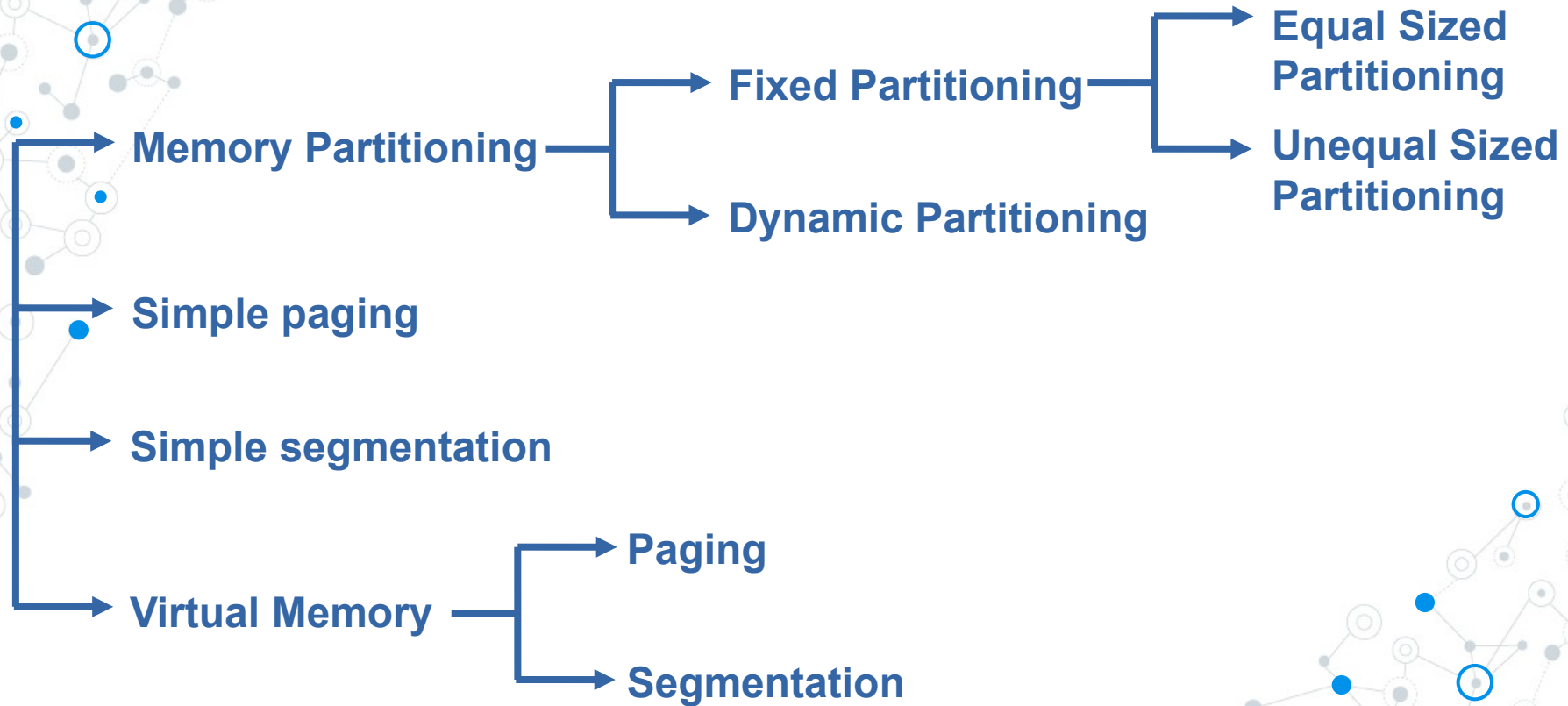
- The principal operation of memory management is to bring processes into main memory for execution by the processor.
- In almost all modern multiprogramming systems, this involves a sophisticated scheme known as virtual memory.
- Before we can look at the virtual memory techniques, we must prepare the ground by looking at simpler techniques
- One of these techniques, partitioning, has been used in several variations in some now-obsolete operating systems
- Other two techniques we will discuss are simple paging and simple segmentation which are not used as such, but will lay down the background clear for virtual memory

Memory Partitioning

In memory partitioning scheme, the main memory is partitioned and one process is accommodated in each partition



Memory Management Schemes

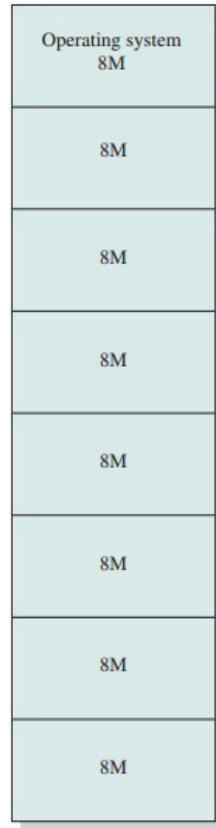




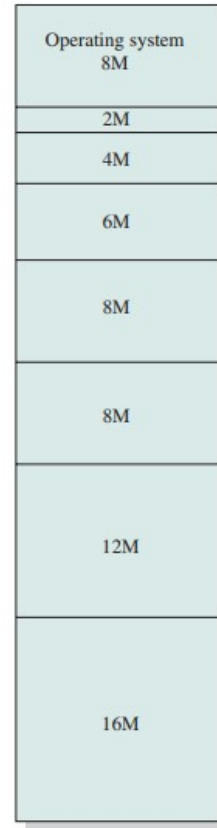
• **Fixed Partitioning**

- In most schemes for memory management, we can assume that the OS occupies some fixed portion of main memory and that the rest of main memory is available for use by multiple processes.
- The simplest scheme for managing this available memory is to partition it into regions with fixed boundaries.
- Two alternatives for fixed partitioning are to make use of equal-size partitions or unequal-size partitions.
- If any process whose size is less than or equal to the partition size can be loaded into any available partition.

Fixed Partitioning



(a) Equal-size partitions



(b) Unequal-size partitions



• **Fixed Partitioning**

- There are two difficulties with the use of equal-size fixed partitions:
- A program may be too big to fit into a partition. In this case, the programmer must design the program with the use of overlays so that only a portion of the program need be in main memory at any one time.
- Main memory utilization is extremely inefficient. Any program, no matter how small, occupies an entire partition.



• **Fixed Partitioning**

- This phenomenon, in which there is wasted space internal to a partition due to the fact that the block of data loaded is smaller than the partition, is referred to as **internal fragmentation**.
- Both of these problems can be lessened, though not solved, by using unequal size partitions.

Fixed Partitioning

- With unequal-size partitions, there are two possible ways to assign processes to partitions.
- The simplest way is to assign each process to the smallest partition within which it will fit, called best fit
- The advantage of this approach is that processes are always assigned in such a way as to minimize wasted memory within a partition (internal fragmentation)
- The other approach known as worst fit assigns each process to the largest partition
- First fit will choose the first free partition whereas another scheme known as next fit will assign a process to the next free partition



• **Fixed Partitioning**

- The use of unequal-size partitions provides a degree of flexibility to fixed partitioning. But still there are disadvantages
 - The number of partitions specified at system generation time limits the number of active (not suspended) processes in the system.
 - Because partition sizes are preset at system generation time, small jobs will not utilize partition space efficiently.



• **Fixed Partitioning**

- This scheme is not used by any operating system now.
- One example of a successful operating system that did use this technique was an early IBM mainframe operating system, OS/MFT (Multiprogramming with a Fixed Number of Tasks).

Dynamic Partitioning

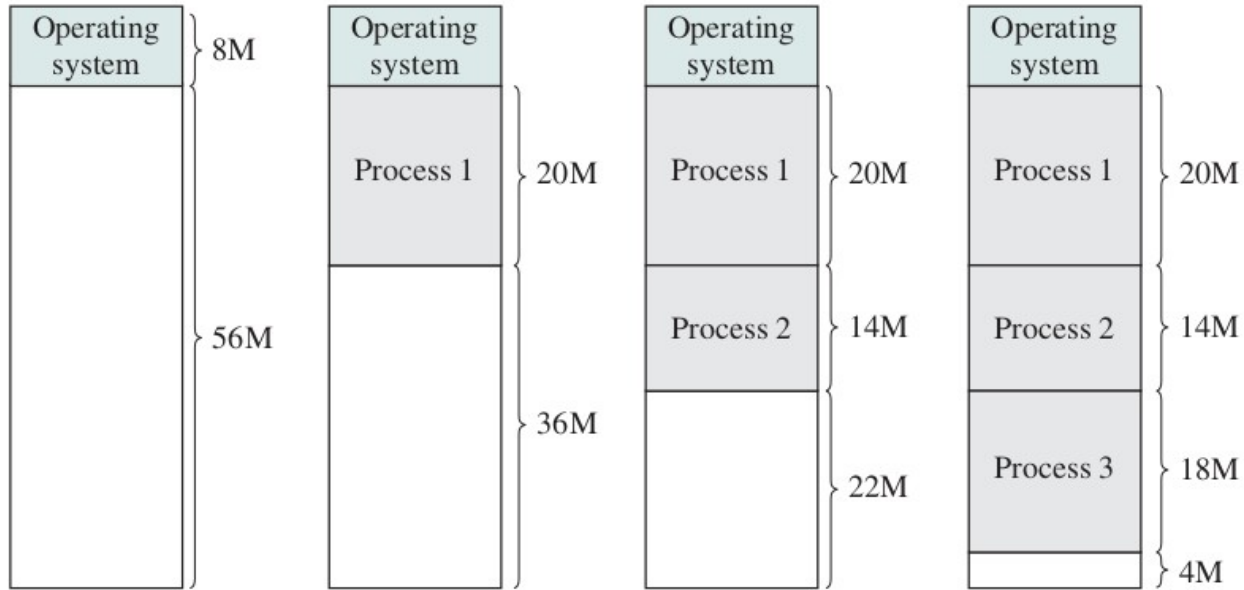
- To overcome some of the difficulties with fixed partitioning, an approach known as dynamic partitioning was developed.
- Again, this approach has been replaced by more sophisticated memory management techniques.
- An important operating system that used this technique was IBM's mainframe operating system, OS/MT (Multiprogramming with a Variable Number of Tasks).



Dynamic Partitioning

- With dynamic partitioning, the partitions are of variable length and number.
- When a process is brought into main memory, it is allocated exactly as much memory as it requires and no more.

Dynamic Partitioning





Dynamic Partitioning

- This method starts out well, but eventually it leads to a situation in which there are a lot of small holes in memory.
- As time goes on, memory becomes more and more fragmented, and memory utilization declines.
- This phenomenon is referred to as external fragmentation, indicating that the memory that is external to all partitions becomes increasingly fragmented.
- This is in contrast to internal fragmentation, referred to earlier.


Dynamic Partitioning

- One technique for overcoming external fragmentation is **compaction** : From time to time, the OS shifts the processes so that they are contiguous and so that all of the free memory is together in one block.
- The difficulty with compaction is that it is a time-consuming procedure and wasteful of processor time.
- Placement algorithms: worst fit, best fit, first fit, next fit




Limitations Of Static and Dynamic Partitioning

- A fixed partitioning scheme limits the number of active processes and may use space inefficiently if there is a poor match between available partition sizes and process sizes.
- A dynamic partitioning scheme is more complex to maintain and includes the overhead of compaction.



Technique	Description	Strengths	Weaknesses
Fixed Partitioning	Main memory is divided into a number of static partitions at system generation time. A process may be loaded into a partition of equal or greater size.	Simple to implement; little operating system overhead.	Inefficient use of memory due to internal fragmentation; maximum number of active processes is fixed.
Dynamic Partitioning	Partitions are created dynamically, so that each process is loaded into a partition of exactly the same size as that process.	No internal fragmentation; more efficient use of main memory.	Inefficient use of processor due to the need for compaction to counter external fragmentation.



BUDDY SYSTEM

- Here, the entire space available for allocation is treated as a single block of size 2^U .
- If a request of size s such that $2^{U-1} < s \leq 2^U$ is made, then the entire block is allocated.
- Otherwise, the block is split into two equal buddies of size 2^{U-1} .
 - If $2^{U-2} < s \leq 2^{U-1}$, then the request is allocated to one of the two buddies.
 - Otherwise, one of the buddies is split in half again.
 - This process continues until the smallest block greater than or equal to s is generated and allocated to the request.





• BUDDY SYSTEM

- A modified form of the buddy system is used for**
- UNIX kernel memory**
- allocation**

Relocation: Fixed and Variable

- When the **fixed partition** scheme is used, we can expect that a process will always be assigned to the same partition.
- That is, whichever partition is selected when a new process is loaded will always be used to swap that process back into memory after it has been swapped out.
- In this scheme, a simple relocating loader can be used.
- When the process is first loaded, all relative memory references in the code are replaced by absolute main memory addresses, determined by the base address of the loaded process.



Relocation: Fixed and Variable

- In variable partition scheme, a process may occupy different partitions during the course of its life.
- To facilitate it, a distinction is made among several types of addresses.
- A logical address is a reference to a memory location independent of the current assignment of data to memory
- A relative address is a particular example of logical address, in which the address is expressed as a location relative to some known point, usually a value in a processor register.
- A physical address , or absolute address, is an actual location in main memory.



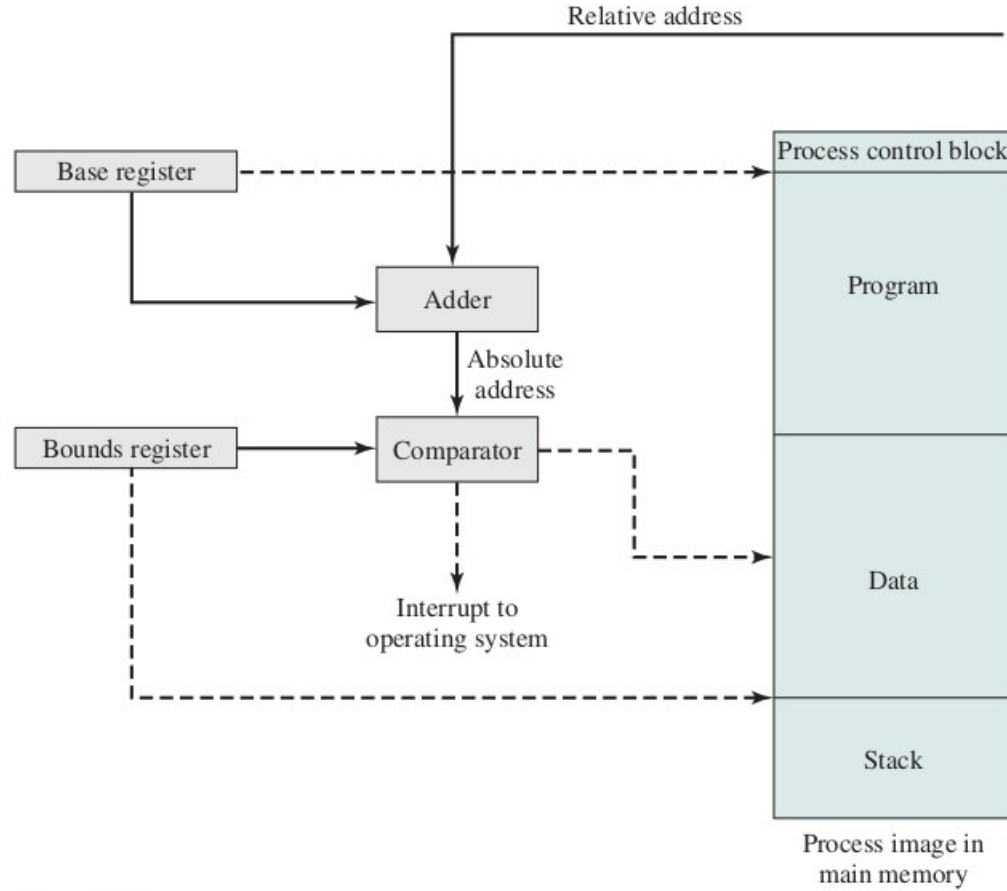
Relocation: Fixed and Variable

- In a typical scheme, when a process is assigned to the Running state, a special processor register, sometimes called the base register, is loaded with the starting address in main memory of the program.
- There is also a “bounds” register that indicates the ending location of the program
- These values must be set when the program is loaded into memory or when the process image is swapped in.

Relocation: Fixed and Variable

- Each relative address goes through two steps of manipulation by the processor.
 - First, the value in the base register is added to the relative address to produce an absolute address.
 - Second, the resulting address is compared to the value in the bounds register.
- If the address is within bounds, then the instruction execution may proceed.
- Otherwise, an interrupt is generated to the operating system, which must respond to the error in some fashion.

Relocation: Fixed and Variable





Paging

- Both fixed and dynamic partitioning schemes are inefficient in the use of memory
- The former results in internal fragmentation, the latter in external fragmentation
- Here, main memory is partitioned into equal fixed-size chunks that are relatively small and called frames
- Each process is also divided into small fixed-size chunks of the same size called pages
- The scheme suffers from internal fragmentation (small). There is no external fragmentation.

Paging

- At a given point in time, some of the frames in memory are in use and some are free.
- A list of free frames is maintained by the OS.
- Let us consider processes A, B, C and D with following number of pages
 - A: 4
 - B: 3
 - C: 4
 - D: 5

Paging

Frame number	Main memory
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

(a) Fifteen available frames

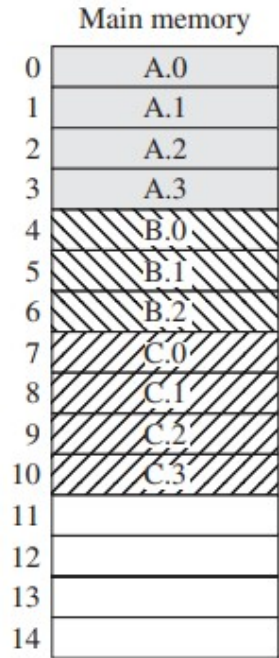
	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	

(b) Load process A

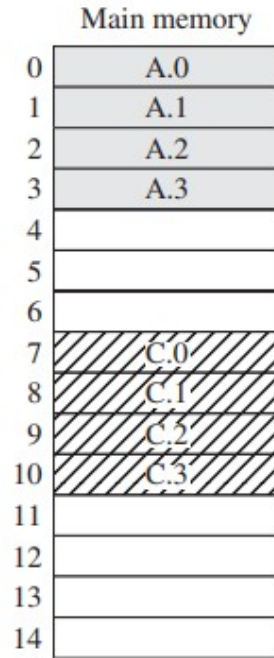
	Main memory
0	A.0
1	A.1
2	A.2
3	A.3
4	B.0
5	B.1
6	B.2
7	
8	
9	
10	
11	
12	
13	
14	

(c) Load process B

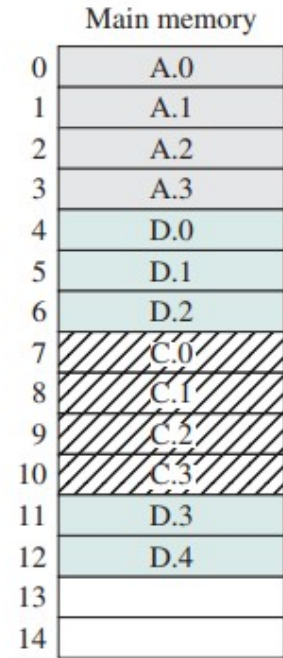
Paging



(d) Load process C



(e) Swap out B



(f) Load process D



Paging

- Does this prevent the operating system from loading D?
 - The answer is no.
 - A simple base address register will no longer suffice.
 - Rather, the operating system maintains a page table for each process.
 - The page table shows the frame location for each page of the process.
 - Within the program, each logical address consists of a page number and an offset within the page.

Paging

- Recall that in the case of simple partition, a logical address is the location of a word relative to the beginning of the program; the processor translates that into a physical address.
- With paging, the logical-to-physical address translation is still done by processor hardware.
- Presented with a logical address (page number, offset), the processor uses the page table to produce a physical address (frame number, offset)

Paging

0	0
1	1
2	2
3	3

Process A
page table

0	—
1	—
2	—

Process B
page table

0	7
1	8
2	9
3	10

Process C
page table

0	4
1	5
2	6
3	11
4	12

Process D
page table

13
14

Free frame
list

Main memory	
0	A.0
1	A.1
2	A.2
3	A.3
4	D.0
5	D.1
6	D.2
7	C.0
8	C.1
9	C.2
10	C.3
11	D.3
12	D.4
13	
14	



Paging

- Thus we see that simple paging, as described here, is similar to fixed partitioning.
- The differences are that, with paging, the partitions are rather small;
- A program may occupy more than one partition; and these partitions need not be contiguous.

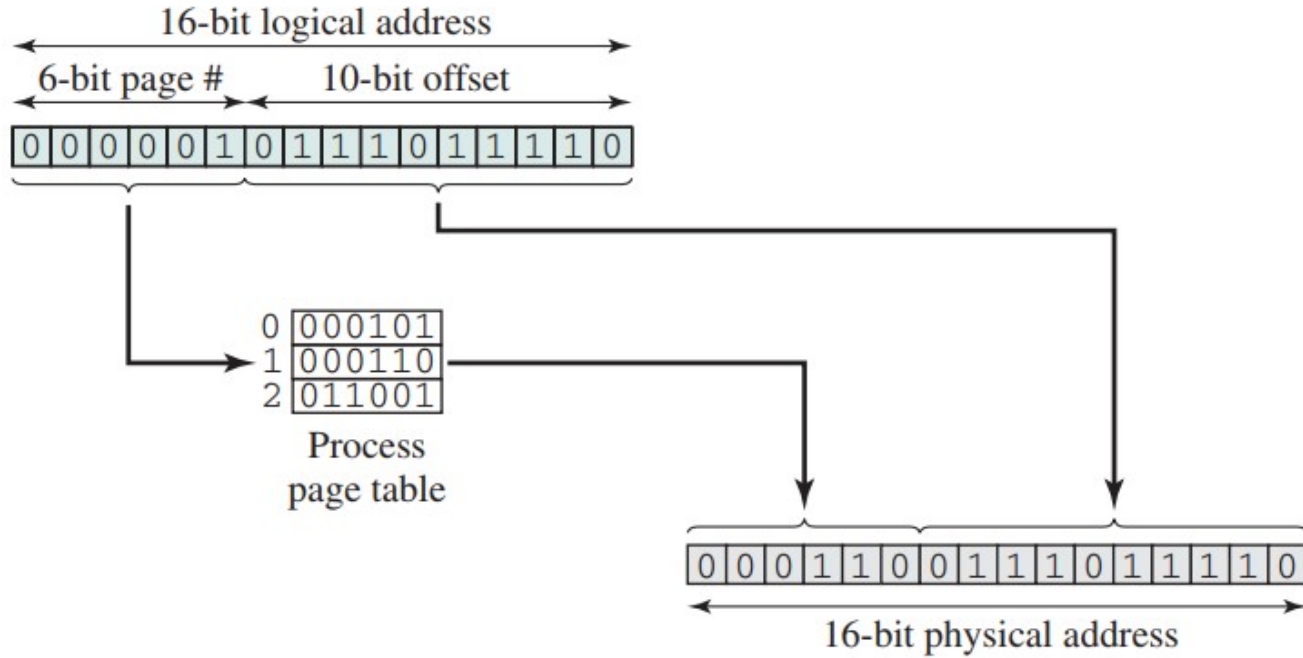
Paging

• Consider an address of $n + m$ bits, where the leftmost n bits are the page number and the rightmost m bits are the offset.

• The following steps are needed for address translation:

- Extract the page number as the leftmost n bits of the logical address.
- Use the page number as an index into the process page table to find the frame number, k .
- The physical address is constructed by appending the frame number to the offset.

Paging



Segmentation

- In segmentation, a program and data are divided into a number of segments .
- It is not required that all segments of all programs be of the same length, although there is a maximum segment length.
- As with paging, a logical address using segmentation consists of two parts, in this case a segment number and an offset.
- The difference, compared to dynamic partitioning, is that with segmentation a program may occupy more than one partition, and these partitions need not be contiguous.



Segmentation

- Segmentation eliminates internal fragmentation but, like dynamic partitioning, it suffers from external fragmentation.
- However, because a process is broken up into a number of smaller pieces, the external fragmentation's effect is less.
- Analogous to paging, a simple segmentation scheme would make use of a segment table for each process and a list of free blocks of main memory.



Segmentation

- Each segment table entry would have to give the starting address in main memory of the corresponding segment.
- The entry should also provide the length of the segment, to assure that invalid addresses are not used.
- When a process enters the Running state, the address of its segment table is loaded into a special register used by the memory management hardware.



Segmentation

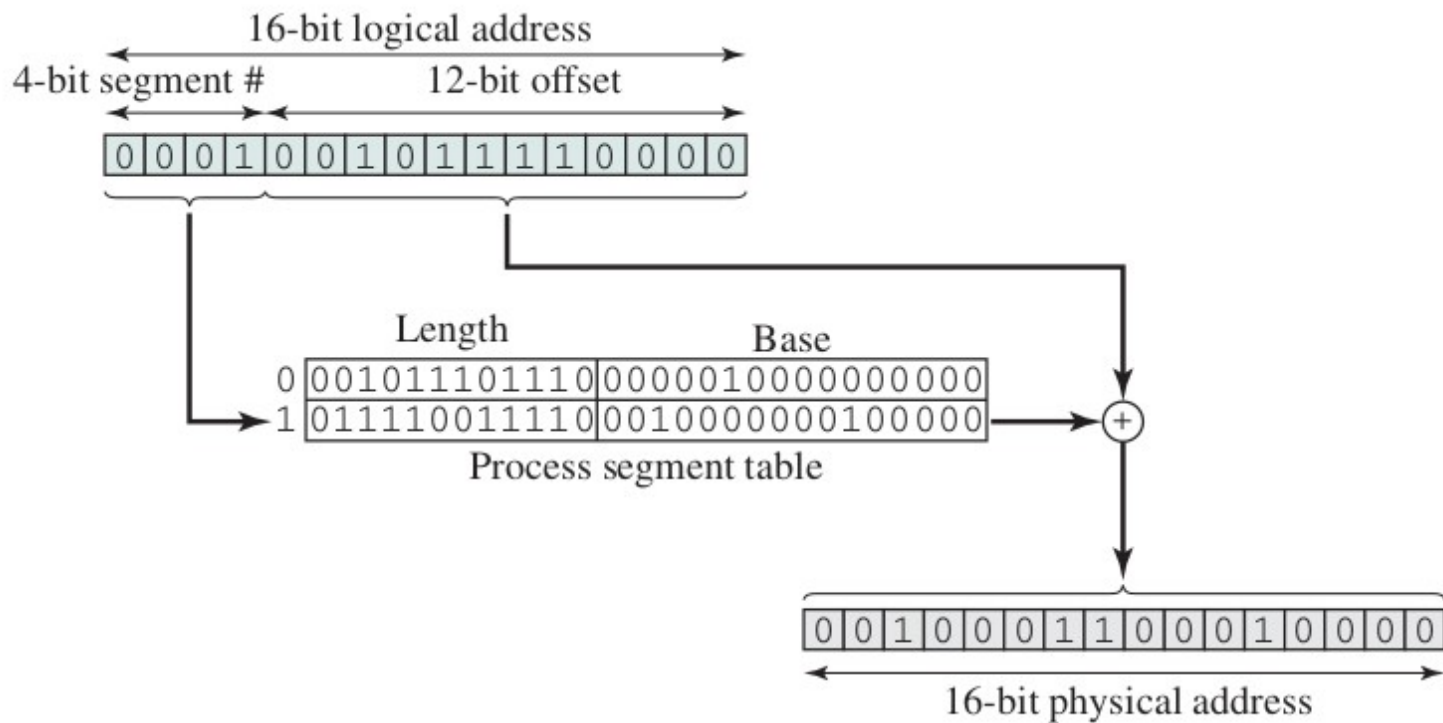
- Consider an address of $n + m$ bits, where the leftmost n bits are the segment number and the rightmost m bits are the offset.
- In such case maximum number of segments will be 2^n

Segmentation

⦿ The following steps are needed for address translation:

- Extract the segment number as the leftmost n bits of the logical address.
- Use the segment number as an index into the process segment table to find the starting physical address of the segment.
- Compare the offset, expressed in the rightmost m bits, to the length of the segment. If the offset is greater than or equal to the length, the address is invalid.
- The desired physical address is the sum of the starting physical address of the segment plus the offset.

Segmentation





Virtual Memory

- A storage allocation scheme in which secondary memory can be addressed as though it were part of main memory.
- In paging and segmentation, it is necessary that all of the pages or all of the segments of a process be in main memory during execution
- In virtual memory, it is not

Virtual Memory

- The portion of a process that is actually in main memory at any time is called the resident set of the process.
- As long as all memory references are in the resident set, using the segment or page table, the processor accesses it.
- If the processor encounters a logical address that is not in main memory, it generates an interrupt indicating a memory access fault.
- The OS puts the interrupted process in a blocking state.
- For the execution of this process to proceed later, the OS must bring into main memory the piece of the process that contains the logical address that caused the access fault.



Virtual Memory


Why are we accepting this loss

- **More processes may be maintained in main memory:** This leads to more efficient utilization of the processor because it is more likely that at least one of the more numerous processes will be in a Ready state at any particular time.
- **A process may be larger than all of main memory:** If the program being written is too large, the programmer must devise ways to structure the program into pieces that can be loaded separately in some sort of overlay strategy. With virtual memory based on paging or segmentation, that job is left to the OS and the hardware.



Virtual Memory


Simple Paging	Virtual Memory Paging	Simple Segmentation	Virtual Memory Segmentation
Main memory partitioned into small fixed-size chunks called frames	Main memory partitioned into small fixed-size chunks called frames	Main memory not partitioned	Main memory not partitioned
Program broken into pages by the compiler or memory management system	Program broken into pages by the compiler or memory management system	Program segments specified by the programmer to the compiler (i.e., the decision is made by the programmer)	Program segments specified by the programmer to the compiler (i.e., the decision is made by the programmer)
Internal fragmentation within frames	Internal fragmentation within frames	No internal fragmentation	No internal fragmentation





Virtual Memory


No external fragmentation	No external fragmentation	External fragmentation	External fragmentation
Operating system must maintain a page table for each process showing which frame each page occupies	Operating system must maintain a page table for each process showing which frame each page occupies	Operating system must maintain a segment table for each process showing the load address and length of each segment	Operating system must maintain a segment table for each process showing the load address and length of each segment
Operating system must maintain a free frame list	Operating system must maintain a free frame list	Operating system must maintain a list of free holes in main memory	Operating system must maintain a list of free holes in main memory
Processor uses page number, offset to calculate absolute address	Processor uses page number, offset to calculate absolute address	Processor uses segment number, offset to calculate absolute address	Processor uses segment number, offset to calculate absolute address





Virtual Memory

All the pages of a process must be in main memory for process to run, unless overlays are used	Not all pages of a process need be in main memory frames for the process to run. Pages may be read in as needed	All the segments of a process must be in main memory for process to run, unless overlays are used	Not all segments of a process need be in main memory for the process to run. Segments may be read in as needed
	Reading a page into main memory may require writing a page out to disk		Reading a segment into main memory may require writing one or more segments out to disk





Virtual Memory

- The benefits of virtual memory are attractive, but is the scheme practical?
- At one time, there was considerable debate on this point, but experience with numerous operating systems has demonstrated beyond doubt that virtual memory does work.
- Accordingly, virtual memory, based on either paging or paging plus segmentation, has become an essential component of contemporary operating systems.

Virtual Memory

- The OS must be clever about how it manages this scheme.
- In the steady state, practically all of main memory will be occupied with process pieces, so that the processor and OS have direct access to as many processes as possible.
- Thus, when the OS brings one piece in, it must throw another out. If it throws out a piece just before it is used, then it will just have to go get that piece again almost immediately.
- Too much of this leads to a condition known as thrashing.
- The system spends most of its time swapping pieces rather than executing instructions.

Virtual Memory

- ❖ In essence, the OS tries to guess, based on recent history, which pieces are least likely to be used in the near future.
- ❖ This reasoning is based on belief in the principle of locality of reference
- ❖ The principle of locality states that program and data references within a process tend to cluster.
- ❖ Hence, the assumption that only a few pieces of a process will be needed over a short period of time is valid.

Virtual Paging

- The term virtual memory is usually associated with systems that employ paging, although virtual memory based on segmentation is also used and is discussed next.
- In the discussion of simple paging, we indicated that each process has its own page table, and when all of its pages are loaded into main memory, the page table for a process is created and loaded into main memory.
- Each page table entry (PTE) contains the frame number of the corresponding page in main memory.
- A page table is also needed for a virtual memory scheme based on paging.

Virtual Paging

- Again, it is typical to associate a unique page table with each process.
- In this case, however, the page table entries become more complex
- Because only some of the pages of a process may be in main memory, a bit is needed in each page table entry to indicate whether the corresponding page is present (P) in main memory or not.
- If the bit indicates that the page is in memory, then the entry also includes the frame number of that page.

Virtual Paging

Virtual address

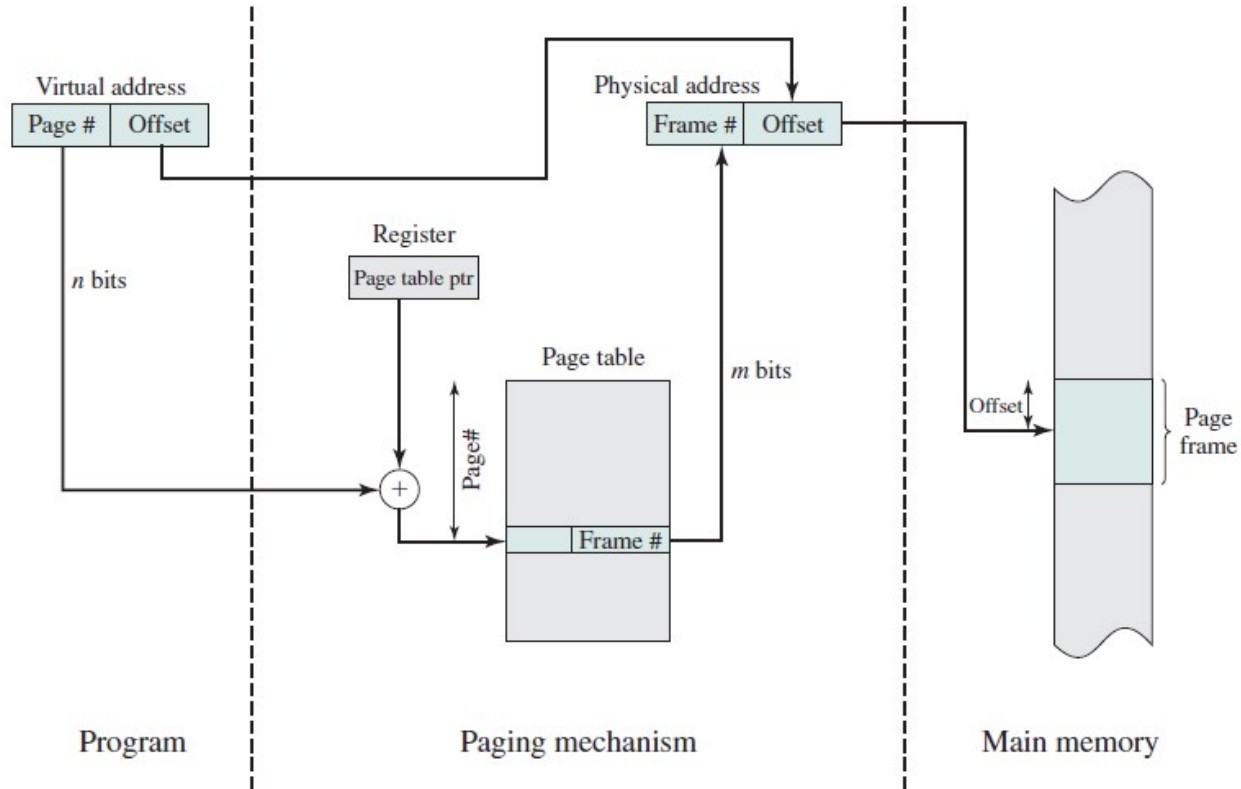
Page number	Offset
-------------	--------

Page table entry

P	M	Other control bits	Frame number
---	---	--------------------	--------------

- The page table entry includes a modify (M) bit, indicating whether the contents of the corresponding page have been altered since the page was last loaded into main memory.
- If there has been no change, then it is not necessary to write the page out when time comes to replace the page in the frame that it currently occupies.
- Other control bits may also be present. For example, if protection or sharing is managed at the page level, then bits for that purpose will be required.

• Address Translation in Virtual Paging





Inverted Page Table

- A drawback of the type of page tables that we have been discussing is that their size is proportional to that of the virtual address space.
- An alternative approach to the use of one or multiple-level page tables is the use of an inverted page table structure.
- Variations on this approach are used on the PowerPC, UltraSPARC, and the IA-64 architecture.



Inverted Page Table

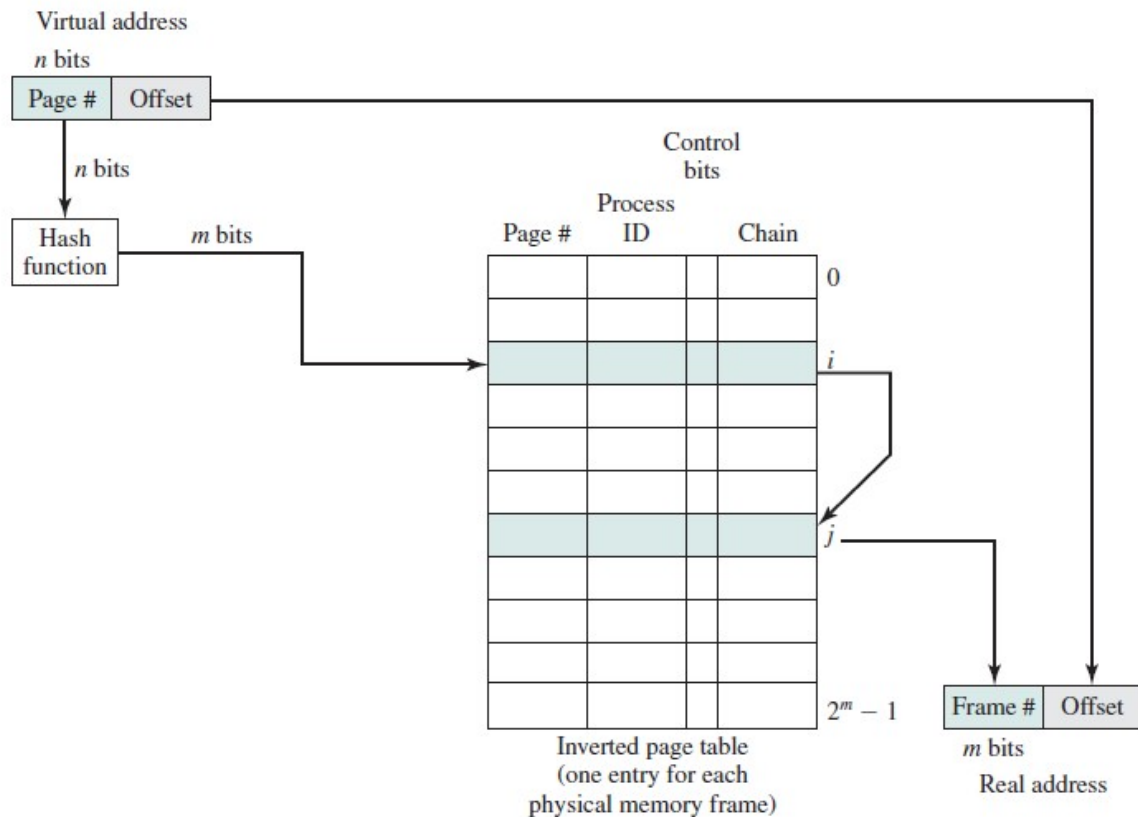
- In this approach, the page number portion of a virtual address is mapped into a hash value using a simple hashing function.
- The hash value is a pointer to the inverted page table, which contains the page table entries.
- There is one entry in the inverted page table for each real memory frame rather than one per virtual page.
- Thus, a fixed proportion of real memory is required for the tables regardless of the number of processes or virtual pages supported.



Inverted Page Table

- **Because more than one virtual address may map into the same hash table entry, a chaining technique is used for managing the overflow.**
- **The hashing technique results in chains that are typically short—between one and two entries.**
- **The page table's structure is called inverted because it indexes page table entries by frame number rather than by virtual page number.**

Inverted Page Table



Translation Lookaside Buffer

- In principle, every virtual memory reference can cause two physical memory accesses: one to fetch the appropriate page table entry and one to fetch the desired data.
- To overcome this problem, most virtual memory schemes make use of a special high-speed cache for page table entries, usually called a translation lookaside buffer (TLB).
- This cache contains those page table entries that have been most recently used.



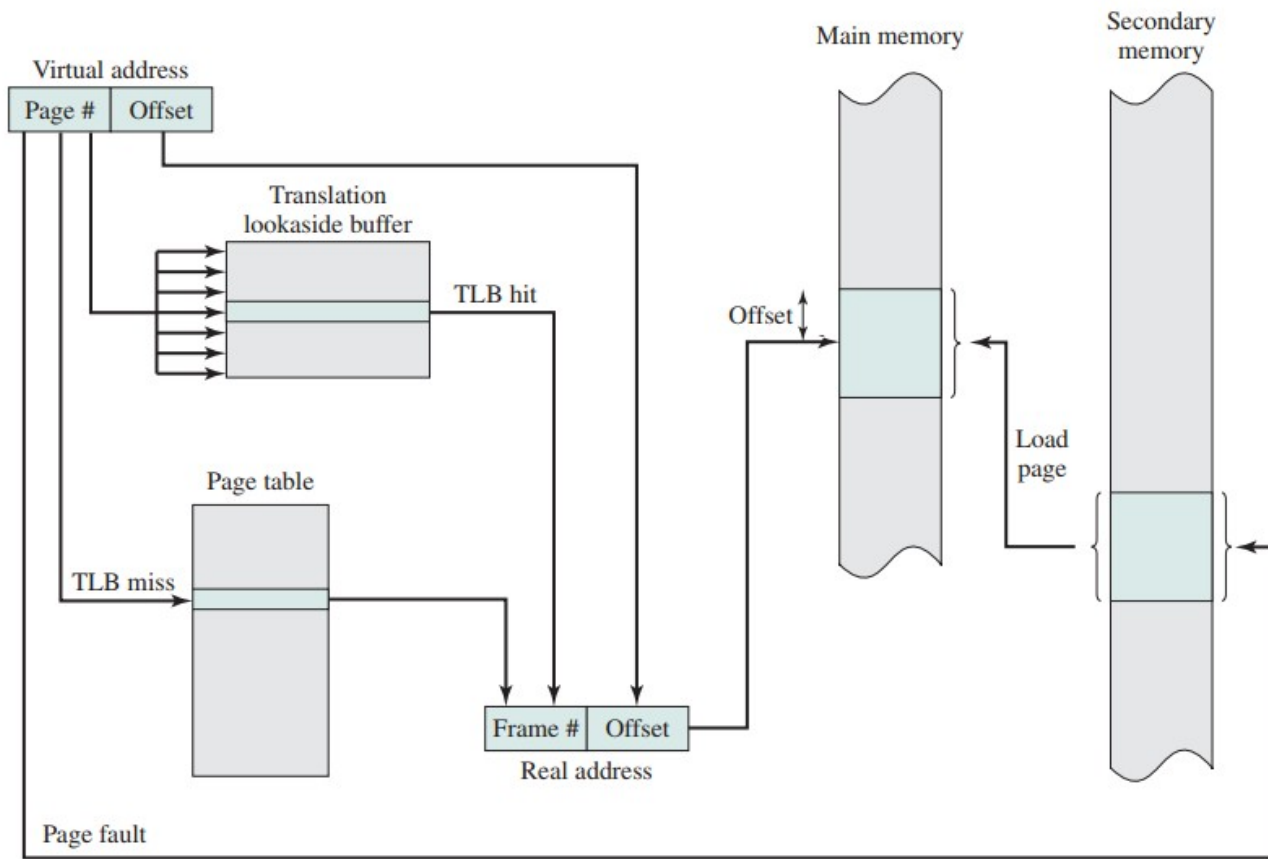
Translation Lookaside Buffer

- Given a virtual address, the processor will first examine the TLB.
- If the desired page table entry is present (TLB hit), then the frame number is retrieved and the real address is formed.
- If the desired page table entry is not found (TLB miss), then the processor uses the page number to index the process page table and examine the corresponding page table entry.

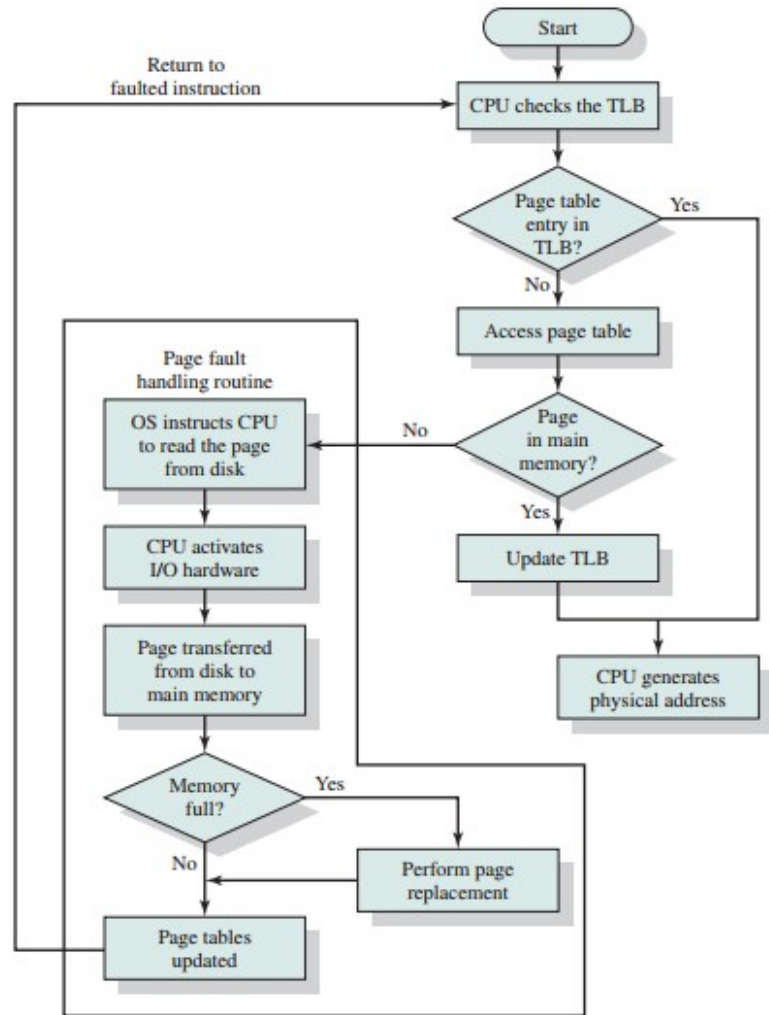
Translation Lookaside Buffer

- If the “present bit” is set, then the page is in main memory, and the processor can retrieve the frame number from the page table entry to form the real address.
- The processor also updates the TLB to include this new page table entry.
- Finally, if the present bit is not set, then the desired page is not in main memory and a memory access fault, called a page fault, is issued.
- At this point, we leave the realm of hardware and invoke the OS, which loads the needed page and updates the page table

Translation Lookaside Buffer



Translation Lookaside Buffer



Virtual Segmentation

- Segmentation allows the programmer to view memory as consisting of multiple address spaces or segments.
- Segments may be of unequal, indeed dynamic, size.
- Memory references consist of a (segment number, offset) form of address.
- Each process has its own segment table, and when all of its segments are loaded into main memory, the segment table for a process is created and loaded into main memory.
- Each segment table entry contains the starting address of the corresponding segment in main memory, as well as the length of the segment.

Virtual Segmentation

- A segment table is needed when we consider a virtual memory scheme based on segmentation.
- Again, it is typical to associate a unique segment table with each process.
- In this case, however, the segment table entries become more complex
 - A bit is needed in each segment table entry to indicate whether the corresponding segment is present in main memory or not.
 - Another control bit in the segmentation table entry is a modify bit, indicating whether the contents of the corresponding segment have been altered since the segment was last loaded into main memory.

Virtual Segmentation

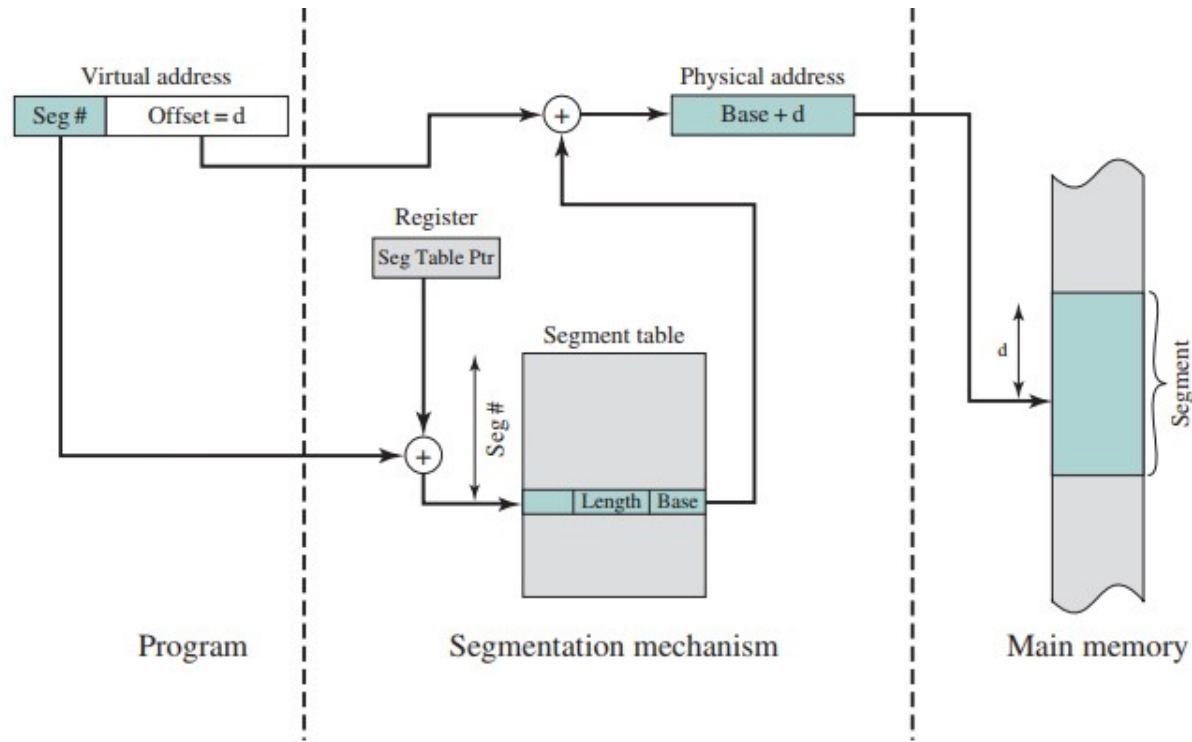
Virtual address

Segment number	Offset
----------------	--------

Segment table entry

P	M	Other control bits	Length	Segment base
---	---	--------------------	--------	--------------

Virtual Segmentation





Segmented Paging

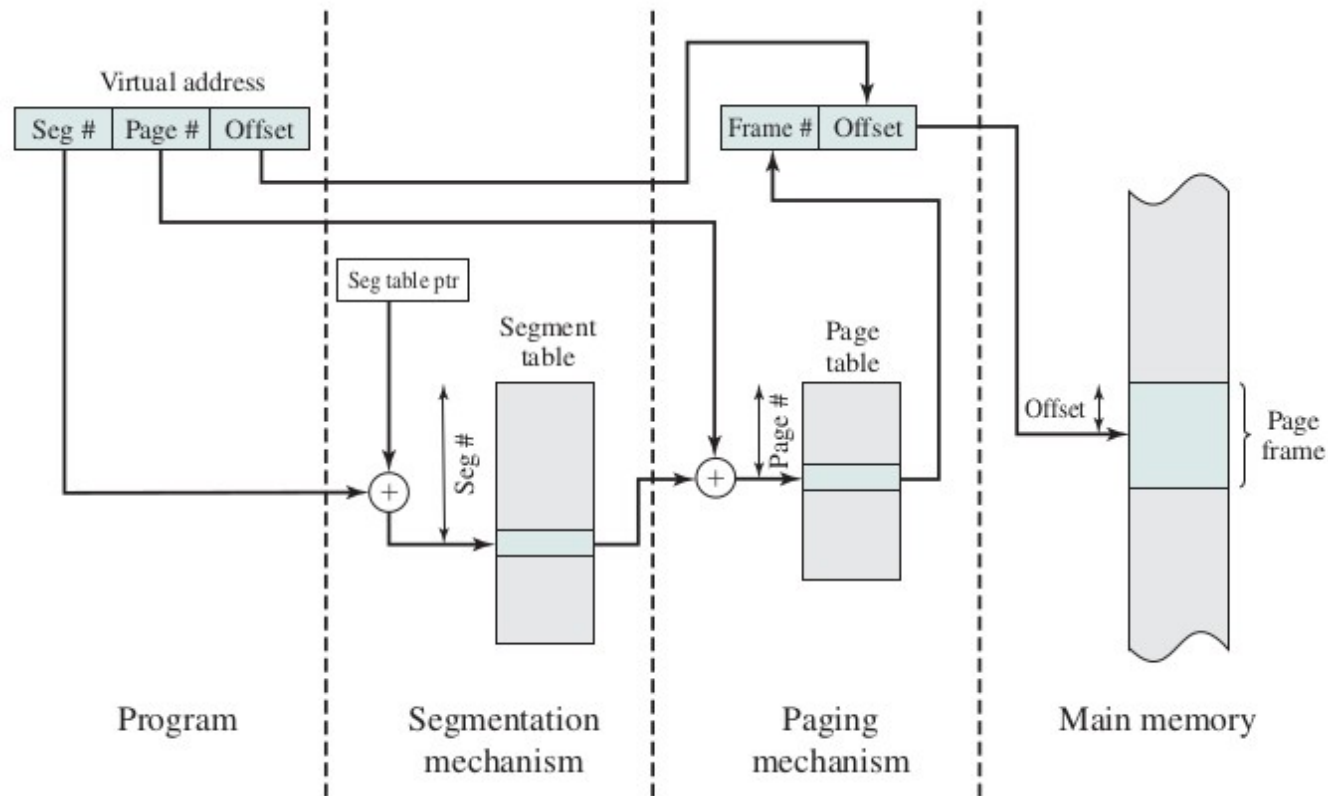
- Both paging and segmentation have their strengths. Paging, which is transparent to the programmer, eliminates external fragmentation and thus provides efficient use of main memory.
- In a combined paging/segmentation system, a user's address space is broken up into a number of segments, at the discretion of the programmer.
- Each segment is, in turn, broken up into a number of fixed-size pages, which are equal in length to a main memory frame.



Segmented Paging

- If a segment has length less than that of a page, the segment occupies just one page.
- From the programmer's point of view, a logical address still consists of a segment number and a segment offset.
- From the system's point of view, the segment offset is viewed as a page number and page offset for a page within the specified segment.

Segmented Paging



Segmented Paging

- Associated with each process is a segment table and a number of page tables, one per process segment.
- When a particular process is running, a register holds the starting address of the segment table for that process.
- Presented with a virtual address, the processor uses the segment number portion to index into the process segment table to find the page table for that segment.
- Then the page number portion of the virtual address is used to index the page table and look up the corresponding frame number.
- This is combined with the offset portion of the virtual address to produce the desired real address.

Segmented Paging

Virtual address

Segment number	Page number	Offset
----------------	-------------	--------

Segment table entry

Control bits	Length	Segment base
--------------	--------	--------------

Page table entry

P	M	Other control bits	Frame number
---	---	--------------------	--------------

P = present bit
M = modified bit

(c) Combined segmentation and paging



Segmented Paging

- As before, the segment table entry contains the length of the segment.
- It also contains a base field, which now refers to a page table.
- The present and modified bits are not needed because these matters are handled at the page level.
- Other control bits may be used, for purposes of sharing and protection.
- The page table entry is essentially the same as is used in a pure paging system.



thank you

The image features a light blue background with a repeating pattern of interconnected circles and lines, resembling a molecular or network structure. The text "thank you" is written in a black, cursive script, flanked by horizontal lines that extend to the left and right edges of the frame.

???