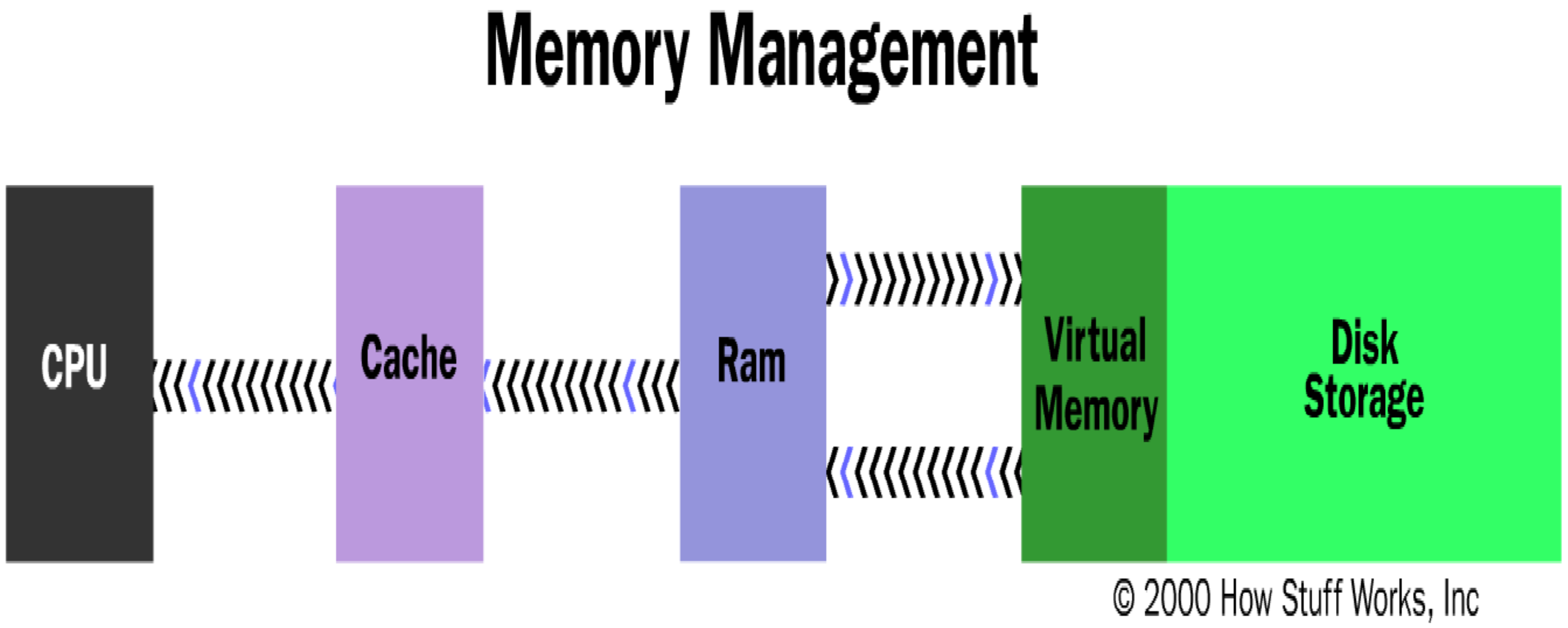
UNIVERSITI TENAGA NASIONAL

CHAPTER 9: Virtual Memory

CGMB143 COMPUTER SYSTEM

Virtual Memory

Memory management



Hardware and Control Structures

- Two characteristics of paging and segmentation that leads to the breakthrough:
  1. Memory references are dynamically translated into physical addresses at run time
  2. A process may be broken up into pieces (pages or segment) that do not need to located contiguously in main memory.

→Hence: all pieces of a process do not need to be loaded in main memory during execution

Execution of a Program

- The OS brings into main memory only a few pieces of the program.
- The portion of process that is in main memory is called “Resident Set”.
- An interrupt is generated when an address is needed that is not in main memory

Execution of a Program

- Piece of process is brought into main memory by:
  - OS issues a disk I/O Read request to bring into main memory the piece referenced
  - Another process is dispatched to run while the disk I/O takes place
  - An interrupt is issued when disk I/O complete
    - this causes the operating system to place the affected process in the Ready state

Types of Memory

- Main Memory / Real memory
  - Main memory where process executes.
- Virtual memory
  - Memory on disk
  - Allows for effective multiprogramming and relieves the user of tight constraints of main memory.

Locality and Virtual Memory

- To accommodate as many processes as possible, only a few pieces of each process is maintained in main memory.
- But main memory may be full: when the OS brings one piece in, it must swap one piece out.
- The OS must not swap out a piece of a process just before that piece is needed
- If it does this too often this leads to trashing:
  - The processor spends most of its time swapping pieces rather than executing user instructions

Principle of Locality

- Principles of locality:

  →Program and data references within a process tend to cluster.
- Hence: Only a few pieces of a process will be needed over a short period of time.
- Possible to make intelligent guesses about which pieces will be needed in the future.
- This suggests that virtual memory may work efficiently.

Support Needed for VM

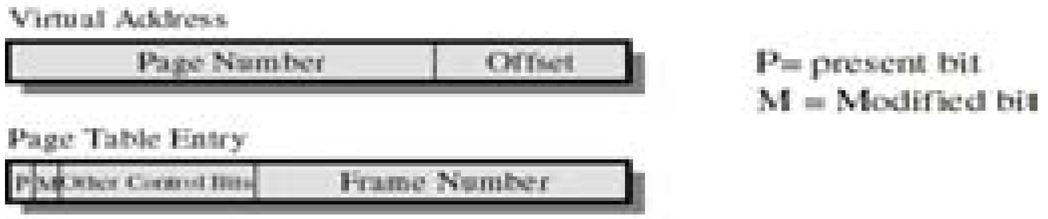
1. Memory management hardware must support paging or segmentation or combination of paging and segmentation and/or segmentation.
2. OS must be able to manage the movement of pages or segments or pages and segments between secondary memory and main memory.

Paging

Paging

- In simple paging each process has its own page table, and when all of its pages are loaded into memory, the page table for a process is created and loaded into memory.
- Each page table entry contains the frame number of the corresponding page in memory.
- With virtual memory scheme page table becomes more complex.

Paging



Each page table entry contains:

1. Present bit (P)

To indicate whether the page is in main memory or not.

- If it is in memory, the entry contains the frame number of the corresponding page in main memory.
- If it is not in memory, the entry may contain the address of that page on disk or the page number may be used to index another table to obtain the address of that page on disk.

Paging

2. A modify bit (M)
   - Indicate if the page has been altered since it was last loaded into main memory
   - If no change has been made, the page does not have to be written to the disk when it needs to be swapped out
3. Other control bits
   - May be present if protection is managed at the page level
     - a read-only/read-write bit
     - protection level bit: kernel page or user page (more bits are used when the processor supports more than 2 protection levels)

Page Table Structure

- The basic mechanism for reading a word from memory involves translation of a virtual or logical address consisting of frame number and offset, using a page table.
- Page tables are variable in length (depends on process size)

Page Table Structure

- Figure 8.3 suggests a hardware implementation.
  - When a particular process is running, register holds the starting address of the page table for the process.
  - The page number of a virtual address is used to index that table to get the frame number.
  - Than combined it with offset of a virtual address to produce the desired physical address.

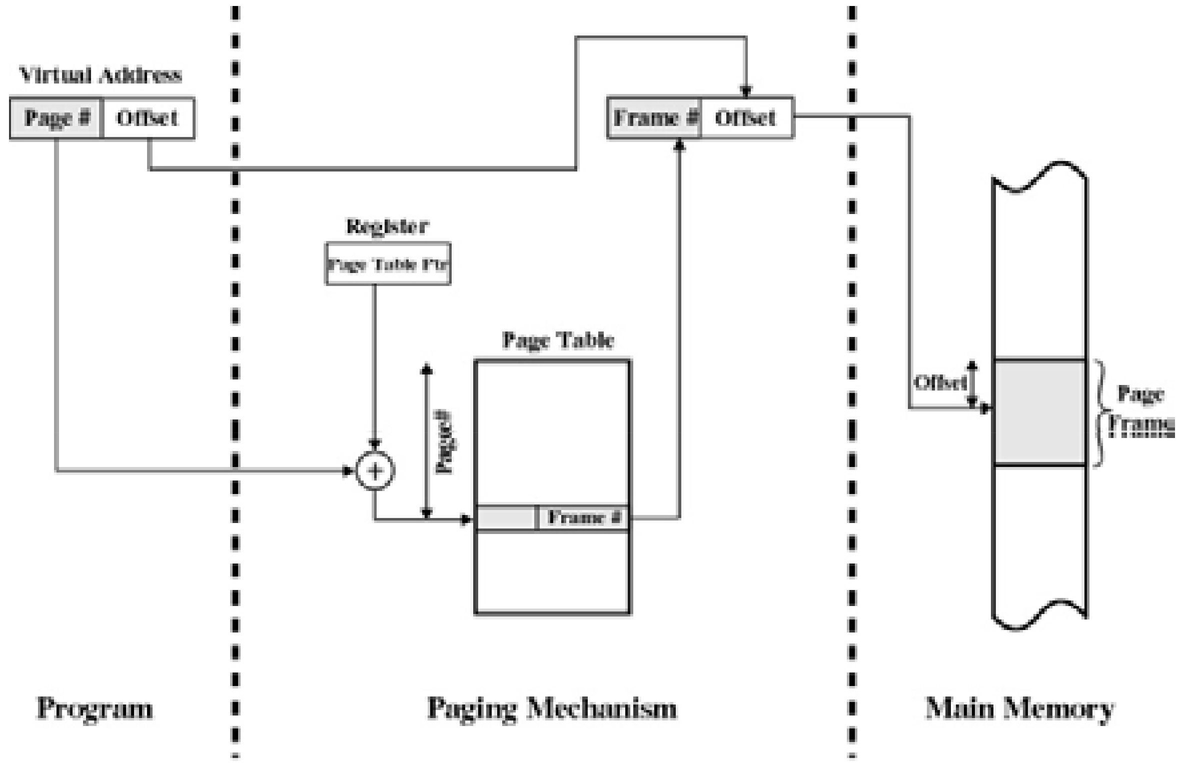


Figure 8.3 Address Translation in a Paging System

Page Table Structure

- There is one page table for each of the process.
- But each of process can occupy huge amounts of virtual memory.
- Clearly the amount of memory devoted to page table alone could be unacceptable.

Page Table Structure

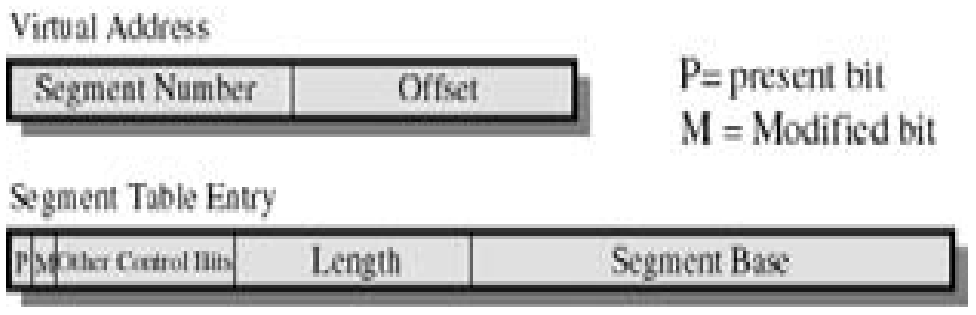
- To overcome this problem, most virtual memory schemes store page tables in virtual memory rather than real memory.
- So, page table is also subject to paging.
- When a process is running at least part of its running, page table must be in memory.
- Approach to organize the page table:
  1. Two level scheme
  2. Inverted Page Table Structure

Segmentation

Segmentation

- Segmentation allow programmer to view memory as consisting of multiple address space or segment
- Segment may be unequal dynamic size.
- Memory references consists of a segment number and offset.
- Each process has its own segment table

Segmentation



- Each segment table entry contains:
  - a present bit (P),
  - modified bit (M)
  - other control bits

Segmentation

- If the segment is in main memory, the entry contains the starting address and the length of that segment.
- Other control bits may be present if protection and sharing is managed at the segment level

Segmentation

- The basic mechanism for reading a word from memory involves the translation of a virtual address consisting of a segment number and offset into physical address using segment table.
- Size of the segment table depends on the size of the process and we can’t expect to hold it in register and it must be in memory to be accessed.

Combined Paging and Segmentation

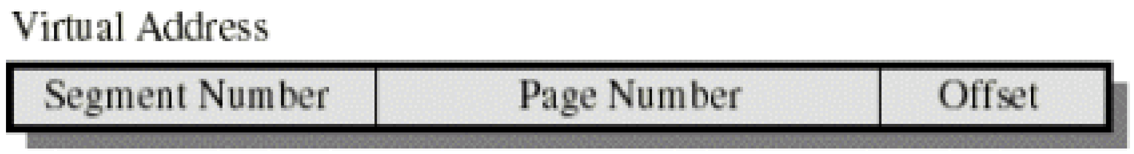
Combined Paging and Segmentation

- To combine their advantages some processors and OS page the segments.
- Several combinations exists. Here is a simple one :
  - User s address space (in memory) will be broken up into a number of segments.
  - Each segment will be broken up into a number of fixed size pages (size must equal with memory frame)
- From programmer’s point of view, a logical address still consists of a segment number and offset.

Combined Paging and Segmentation

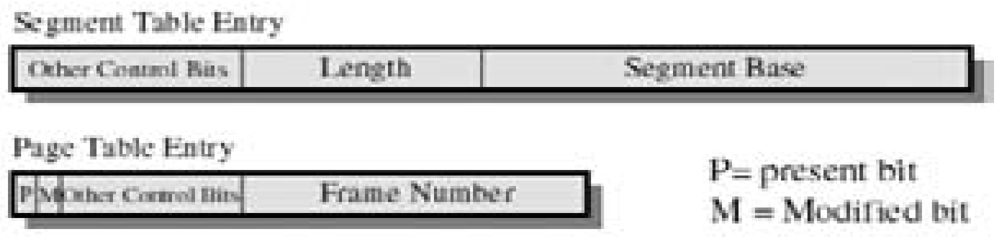
- From system’s point of view the segment offset is viewed as a page number and page offset for a page within the specified segment.
- Each process has:
  - one segment table
  - several page tables: one page table per segment

Combined Paging and Segmentation



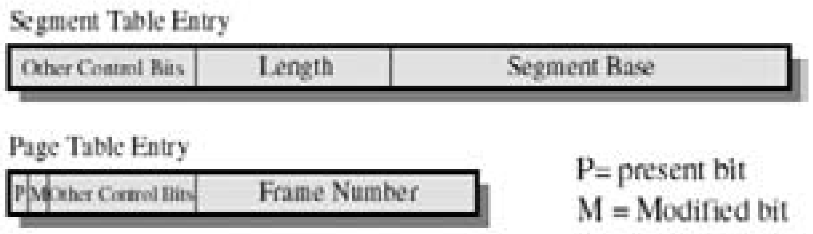
- The virtual address consist of:
  - a *segment number*: used to index the segment table who's entry gives the starting address of the page table for that segment
  - a *page number*: used to index that page table to obtain the corresponding frame number.
  - an *offset*: used to locate the word within the frame

Combined Paging and Segmentation



- Segment table entry:
  - Length field
  - Base field - is the physical address of the page table of that segment.
  - Other control bits is used for sharing and protection purposes.
  - The present and modified bits are not needed because these matters are handled at the page table.

Combined Paging and Segmentation



- Page table entry:
  - Present bit (P)
  - Modify bit (M)
  - Other Control Bits
  - Frame Number.
- Protection and sharing info most naturally resides in segment table entry.
  - Ex: a read-only/read-write bit, a kernel/user bit...

Fetch Policy

Fetch Policy

- Determines when a page should be brought into memory.
- Two common policies:
  1. Demand Paging
  2. Prepaging

Fetch Policy – Demand Paging

- Only brings pages into main memory when a reference is made to a location on the page (paging on demand only).

Fetch Policy – Pre-Paging

- Brings in more pages than needed
- Locality of references suggest that it is more efficient to bring in pages that reside contiguously on the disk
- Efficient not definitely established; the extra pages brought in are “often” not referenced.

Placement and Replacement Policy

Placement Policy

- Determines where in real memory a process piece resides.
- For pure segmentation systems:
  - first-fit, next fit... are possible choices (a real issue)
- For paging (and paged segmentation):
  - the hardware decides where to place the page: the chosen frame location is irrelevant since all memory frames are equivalent (not an issue)

Replacement Policy

- Deals with the selection of a page in main memory to be replaced when a new page is brought in.
- This occurs whenever main memory is full (no free frame available). Occurs often since the OS tries to bring into main memory as many processes as it can to increase the multiprogramming level

Replacement Policy

- Not all pages in main memory can be selected for replacement
- Some frames are locked (cannot be paged out) known as Frame Locking:
  - much of the kernel is held on locked frames as well as key control structures and I/O buffers
- The OS might decide that the set of pages considered for replacement should be:
  - limited to those of the process that has suffered the page fault
  - the set of all pages in unlocked frames

Basic Replacement Algorithms

- 1. Optimal policy
- 2. Least Recently Used
- 3. First In First Out
- 4. Clock policy (not cover in the syllabus)

Basic Replacement Algorithms: Optimal Policy

- Selects for replacement that page for which the time to the next reference is the longest
- Produce the fewest number of page faults.
- Impossible to implement because need to have perfect knowledge of future events.

Basic Replacement Algorithms: Least Recently Used

- Replaces the page that has not been referenced for the longest time.
- By the principle of locality, this should be the page least likely to be referenced in the near future.
- Performs nearly as well as the optimal policy.
- Each page could be tagged with the time of last reference. This would require a great deal of overhead.

Basic Replacement Algorithms: First In First Out

- Treats page frames allocated to a process as a circular buffer.
- Simplest replacement policy to implement.
- Page that has been in memory the longest is replaced.
- These pages may be needed again very soon