CI583: Data Structures and Operating Systems
Memory management part 1

Outline

History of memory management

Memory management is of central importance in operating system design, and the different strategies that an OS might adopt have a crucial effect on its performance.

The operating system's address space is its addressable memory.

History of memory management

This may be less than the total memory available, since a memory location has to be stored in a number.

Taking the biggest 32-bit number as the limit, the largest possible address space that can be accessed directly on a 32-bit system is 4GB.

History of memory management

We also use the term "address space" to refer to the memory allocated to a particular process when it is executed and loaded into memory.

In early systems, the entire code and data were loaded into memory.

History of memory management

The main concern of memory management in the early era was to protect the OS from user code.

The simplest way of doing this was to establish a memory fence – an address in memory below which no user process can access any location.

This protects the OS from user processes but not one user process from another.

History of memory management

A solution which addresses both issues of protection is the use of base and bounds registers.

When the address space is created for a new process, it is assigned an upper and lower limit of addressable memory.

This has the added benefit of making it simple to translate logical memory addresses to real ones – the process is loaded into what appears to be location 0, then all addresses are relative to the base register.

History of memory management

As well the need to protect address spaces, there is the problem of fitting a program into memory.

In the early days, programmers needed to use manual memory allocation and deallocation very carefully to make sure that their process would fit in the available space.

History of memory management

To write programs larger than the (tiny) space available, large portions of a program would need to be dedicated to moving routines and data in and out of memory (a technique called overlaying), with no help from the OS.

This was difficult and highly error-prone.

History of memory management

Virtual memory was invented at Manchester University in the early 60s to solve this problem.

Virtual memory allows the OS to address a larger amount of memory than the available primary storage.

The address space of a process is composed of virtual locations some of which map to addresses within pages of code or data in primary storage, and some of which map to pages in secondary storage.

History of memory management

If a process refers to an address which is not currently part of a page held in memory, a page fault occurs, raising an interrupt which causes the page to be loaded.

This might mean moving a page which is currently in memory back out.

Virtual memory is used in every modern OS, other than those intended for small embedded applications.

Page tables

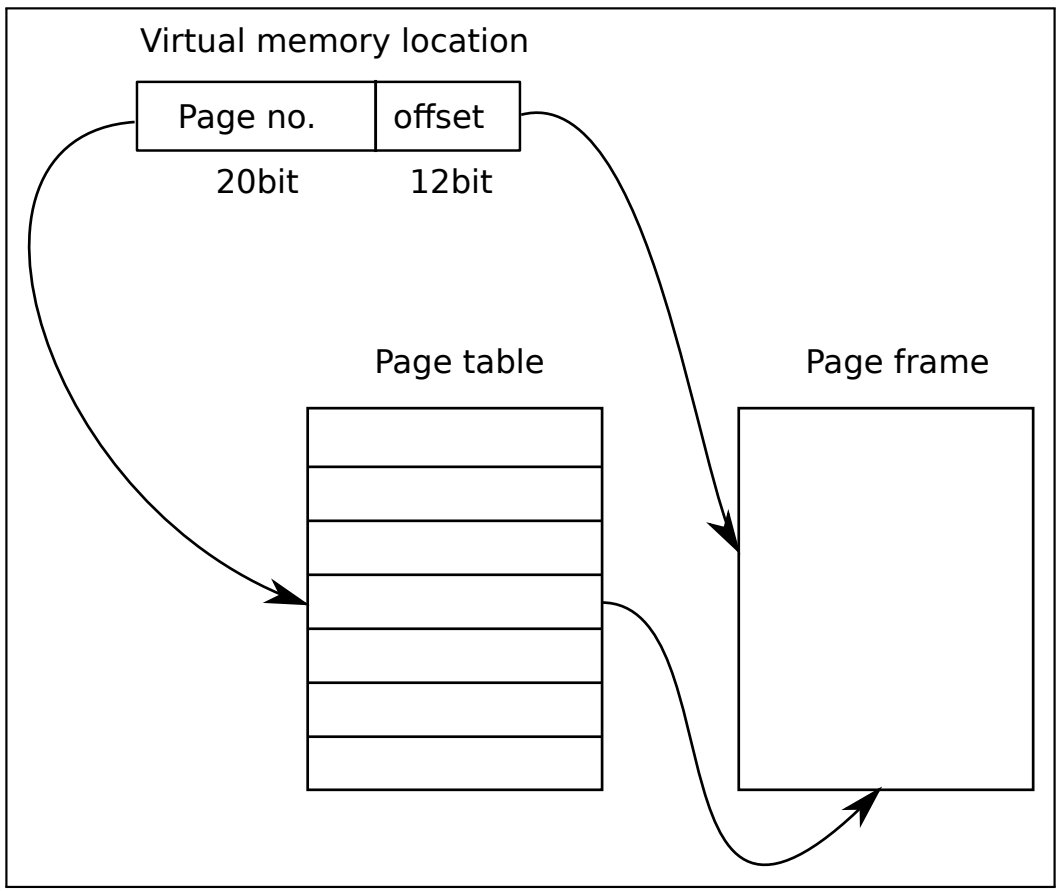
Although there have been approaches to virtual memory that use variable-sized segments, the dominant and most successful approach uses fixed-size pages.

Virtual memory locations are mapped to real locations using a page table.

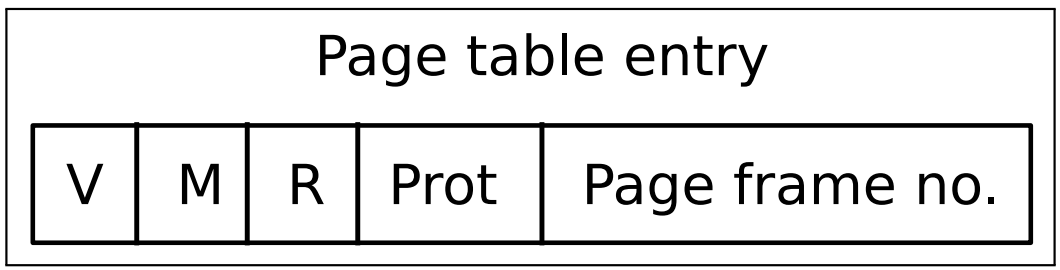
This is supported with special instructions at the hardware level.

Page tables

A virtual memory location consists of a page number, used to index the page table, and an offset, used to specify a particular location within the page frame, which is a section of primary storage.



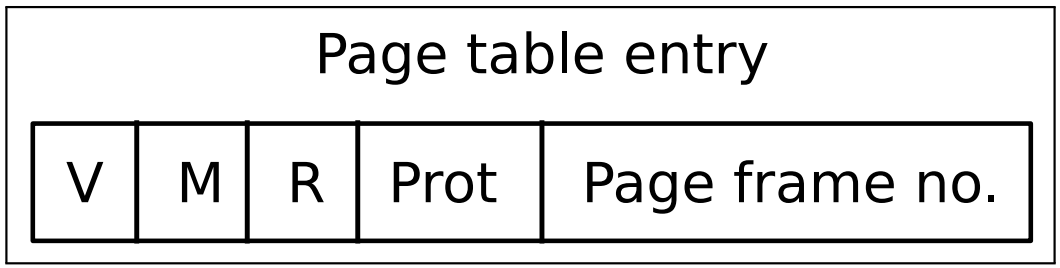
Page tables



Each page table entry is one word long.

If the validity bit is set to 1, this means that the page is already in memory, and the page frame number is returned.

Page tables



If `V` is set to zero, the page needs to be loaded.

A page fault occurs and the hardware address-translation facility allows the OS to take over, loading the page then informing the hardware address-translation module of its page frame number.

Page fault

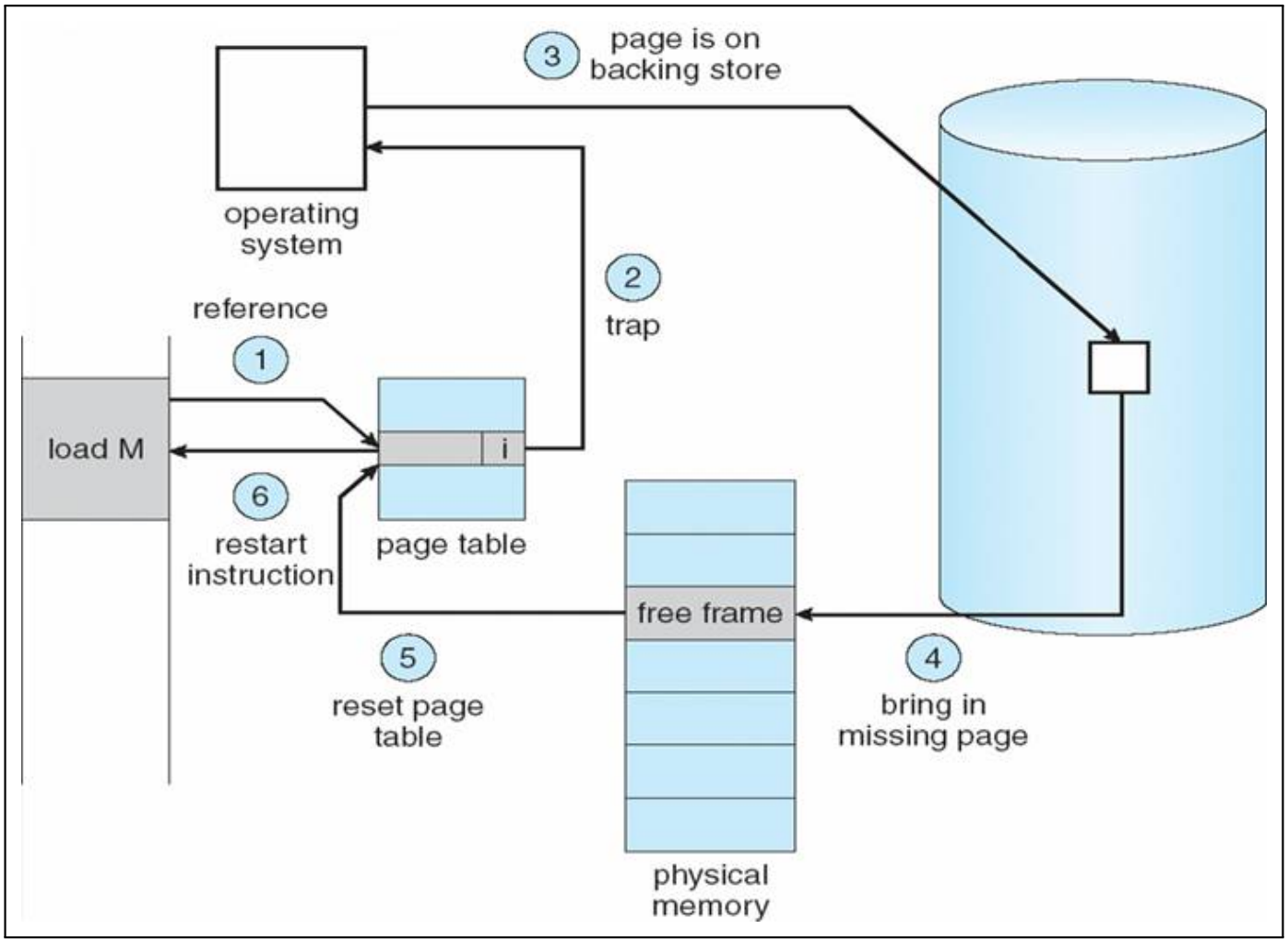
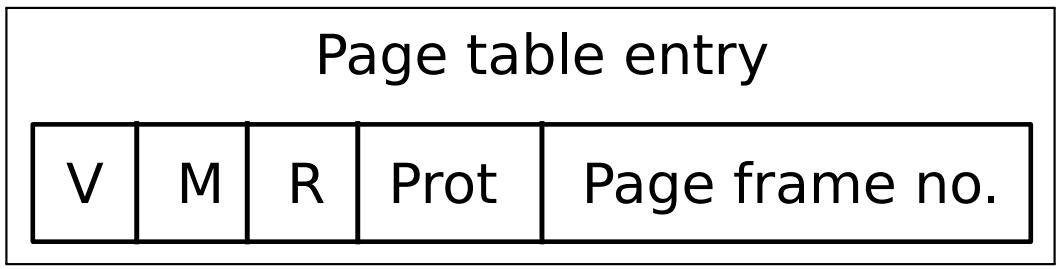


Image ©http://www.cs.odu.edu/~cs471w

Page tables



The page table also holds other information:

- A modified bit: has this page been written to since it was loaded?
- A reference bit: has this page been referenced by a runnable thread?
- Page-protection bits: who can access this page?

Page tables

Making these extra lookups, even with hardware support, is extremely expensive, and so the most recently accessed parts of the page table are copied into a cache called the translation lookaside buffer (TLB).

The various means we describe to translate a virtual memory location into a real one are only necessary when we have a "miss" in the TLB.

Page tables

As new addresses are added to the TLB, old ones are removed on either a least-recently-used or a FIFO basis.

Also, the TLB is emptied altogether when the OS switches context, so misses will be the norm when a thread begins to run.