Virtual Memory Organization

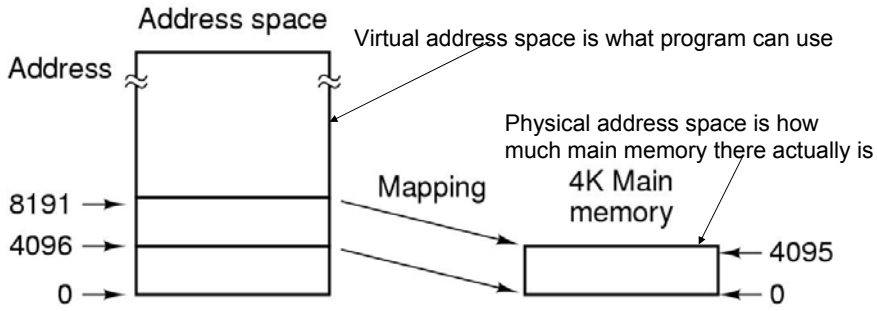
Virtual Memory

- Operating system level adds new instructions called system calls
- Virtual memory: not enough space in memory to hold entire program so pretend some of hard drive is memory and swap part into memory from hard drive when needed
  - Paging
  - Segmentation

Paging

- Paging concept
  - Separate number of real memory locations (smaller) and address space (larger)
  - Address space (virtual address space) limited by number of bits in an address. 16-bit address can access 65536 words (regardless of how much actual memory there is)
  - Real memory locations (real address space) is limited by amount of physical main memory in the system

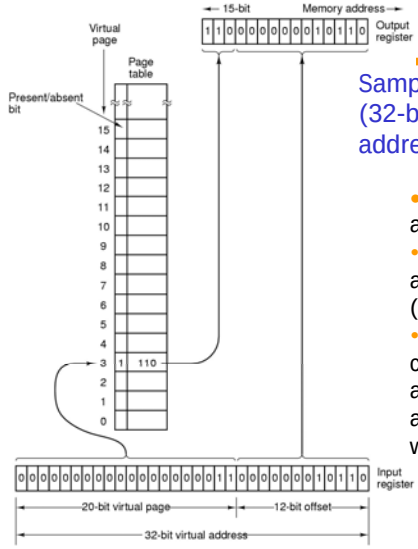
Paging & Physical Memory



Contents of main memory saved on disk.
When address space 8192 to 12287 (4 KB) accessed, loaded into memory at main memory addresses 0 – 4095.
Addresses 8192 – 12287 mapped to 0 – 4095 and execution continues.
This automatic overlaying/mapping called paging.

Page Tracking

- Address space broken up into equal-sized pages (usually 512 bytes to 64 KB each)
- "Pages" in main memory called page frames
- MMU (Memory Management Unit): device that does the mapping between page frames in main memory and pages in virtual memory (on disk). Keeps a page table that has a present/absent bit for each page (is it in memory or not)



Sample MMU that maps 4GB virtual memory (32-bit address) to 32KB main memory (15-bit address) with 8 4KB-pages.

- MMU checks page table to see if page is in a page frame in memory (1 = yes)
- Take page frame value from page table and put it in upper 3 bits of memory address (tells which of 8 page frames it's in)
- Lower 12-order bits from virtual address copied to lower 12-order bits of virtual address to indicate at which memory address in that page frame the requested word is located

Byte-addressable memory

Virtual address made up of 20-bit virtual page number and 12-bit address in the page (2^{12} = 4K).

Paging Overview

- Page fault: when a page is referenced and it is not in main memory so operating system has to read it from disk into a memory page frame and repeat the instruction that caused the fault.
- Working set model: assume program references cluster around a few pages (working set – the ones most recently used) and keep them in memory
- A program that generates many page faults is thrashing

Page Replacement

- Page replacement policies: which page to overwrite when a new one comes in?
  - Least Recently Used (LRU)
  - First-In-First-Out (FIFO)
- What if a page in memory has been altered and needs to be written to memory? The MMU keeps a dirty bit in page table to know.

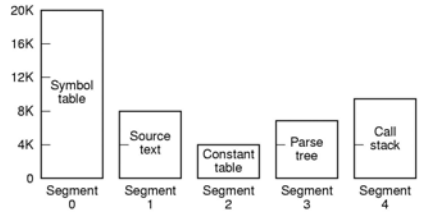
Page Fragmentation

- Internal fragmentation: when a program does not fill an integral number of pages there is wasted space in some page frames in memory.
- If a page size is *n* bytes, the average amount of wasted space in the last page of the program is *n/2,* which seems to say use small page sizes
- But small pages mean big page table, many separate reads from disk and waste disk bandwidth (since it's not the transfer rate that slow but the seek/rotate)

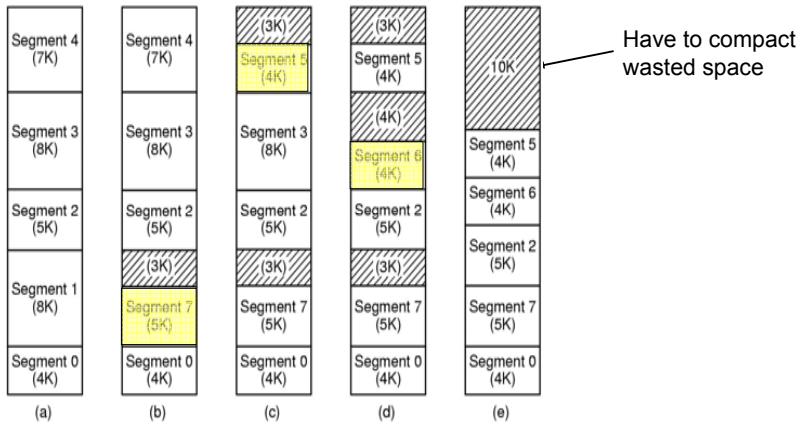
Segmentation

- Segmentation: have many separate virtual address spaces called segments (each segment is addressed from 0 but segment sizes not all the same and size can change dynamically)
- To find word, specify segment number and address in that segment.

Separate segments for different part of program



Segment Fragmentation



Segmentation doesn't have internal fragmentation because each segment is as big as it needs to be but it develops external fragmentation – space between segments that aren't big enough to be used by other segments