CSE451 Virtual Memory Paging
Spring 2001

Gary Kimura
Lecture #14
April 25, 2001

Today

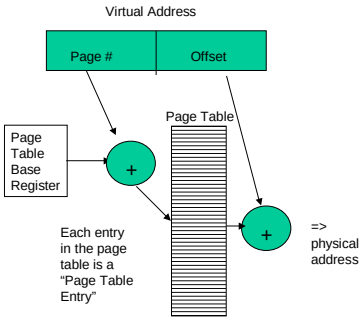
- Fragmentation (how we can't avoid wasting resources)
- Virtual memory and introduce more of the paging concept and hardware support needed for paging.
- And take a look at how we typically divide up virtual memory between the OS (i.e., Kernel) and User Programs

A brief side note on fragmentation

- There are essentially two types of fragmentation
  - Internal Fragmentation: This is where a block of memory is being under utilized. For example a process of size 5000 bytes would need 2 4KB pages of memory. So the system winds up allocating 8192 bytes of memory The program really only uses 5000 bytes so we waste 3092 bytes.
  - External Fragmentation: There is where memory has been broken up into small unallocated pieces whose sum might make a nice usable piece but because the pieces are not adjacent in the Virtual Address Space they cannot be combined.

Paging

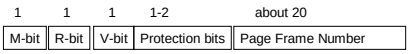
- Goals
  - make allocation and swapping easier
- Make all chunks of memory the same size
  - call each chunk a "PAGE"
  - example page sizes are 512 bytes, 1K, 4K, 8K, etc
  - pages have been getting bigger with time



An Example

- Pages are 4096 bytes long
  - this says that bottom 12 bits of the VA is the offset
- PTBR contains 32768
  - this says that the first page table entry for this process is at physical memory location 32768
- Virtual address is 5000
  - this says "page 2, offset (5000-4096) = 904"
- Physical memory location 32772 contains 8192
  - this says that each PTE is 4 bytes
  - and that the second page of this process's address space can be found at memory location 8192.
- So, we add 904 to 8192 and we get the real data!

What does a PTE contain?



- The Modify bit says whether or not the page has been written.
  - it is updated each time a WRITE to the page occurs.
- The Reference bit says whether or not the page has been touched
  - it is updated each time a READ or a WRITE occurs
- The V bit says whether or not the PTE can be used
  - it is checked each time the virtual address is used
- The Protection bits say what operations are allowed on this page
  - READ, WRITE, EXECUTE
- The Page Frame Number says where in memory is the page

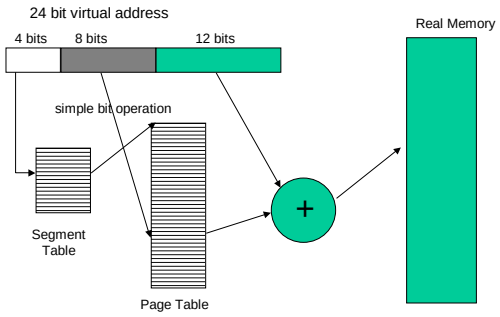
Evaluating Paging

- Easy to allocate memory
  - memory comes from a free list of fixed size chunks.
  - to find a new page, get anything off the free list.
  - external fragmentation not a problem
- easy to swap out pieces of a program
  - since all pieces are the same size.
  - use valid bit to detect references to swapped pages
  - pages are a nice multiple of the disk block size.
- Can still have internal fragmentation
- Table space can become a serious problem
  - especially bad with small pages
    - eg, with a 32bit address space and 4k size pages, that's 2^{20} pages or that many ptes which is a lot!
- Memory reference overhead can be high
  - 2 refs for every one

Segmentation and Paging at the Same Time

- Provide for two levels of mapping
- Use segments to contain logically related things
  - code, data, stack
  - can vary in size but are generally large.
- Use pages to describe components of the segments
  - makes segments easy to manage and can swap memory between segments.
  - need to allocate page table entries only for those pieces of the segments that have themselves been allocated.
- Segments that are shared can be represented with shared page tables for the segments themselves.

An Early Example -- IBM System 370



Lookups

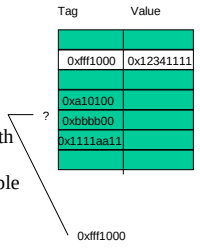
- Each memory reference can be 3
  - assuming no fault
- Can exploit locality to improve lookup strategy
  - a process is likely to use only a few pages at a time
- Use Translation Lookaside buffer to exploit locality
  - a TLB is a fast associative memory that keeps track of recent translations.
- The hardware searches the TLB on a memory reference
- On a TLB miss, either a hardware or software exception can occur
  - older machines reloaded the TLB in hardware
  - newer RISC machines tend to use software loaded TLBs
    - can have any structure you want for the page table
    - fast handler computes and goes. Eg, the MIPS.

Hard versus soft page faults

- Hard page faults are those page faults that require issuing a read from secondary storage.
- Soft page faults are those page faults where the page is already in main memory however the TLB and/or the PTE has marked the page as invalid.
  - Soft faults are used when Hardware support is not available to handle TLB misses
  - Soft faults can also be used in implement certain page replacement algorithms. More to come.

A TLB

- A small fully associative cache
- Each entry contains a tag and a value.
  - tags are virtual page numbers
  - values are physical page table entries.
- Problems include
  - keeping the TLB consistent with the PTE in main memory
    - valid and ref bits, for example
  - keeping TLBs consistent on an MP.
  - quickly loading the TLB on a miss.
- Hit rates are important.



Selecting a page size

- Small pages give you lots of flexibility but at a high cost.
- Big pages are easy to manage, but not very flexible.
- Issues include
  - TLB coverage
    - product of page size and # entries
  - internal fragmentation
    - likely to use less of a big page
  - # page faults and prefetch effect
    - small pages will force you to fault often
  - match to I/O bandwidth
    - want one miss to bring in a lot of data since it will take a long time.

State of maintained by MM

- MM usually maintains a list of physical pages according to the following attributes (various implementations use slightly different lists)
  - Zeroed pages
  - Free pages
  - Standby pages
  - Modified pages
  - Modified No Write pages
  - Bad pages
- MM's goal is to use these pages on these lists to supply memory for both soft and hard page faults
- MM can have a modified page writer process that goes around and flushes out dirty pages.

Address Spaces

- In modern systems the virtual address space is usually divided into two main sections (one for user programs and another for the OS)
- For example in Windows (not the 64bit version) the lower 2GB is used for the user programs and upper 2GB is reserved for the OS
- The OS pages are protected and cannot be read while in user mode
- Each process shares the same upper 2GB of Virtual address, but each also has a different set of pages for its user space
- This design has implications on communication between a user program and the OS, and between user programs

Next Time

- Work through a paging example to understand more of the issues involved