Virtual Memory

4 classes to go!

Last problem set online.

Today: Virtual Memory
Main memory is a CACHE for disk

Advantages:
- illusion of having more physical memory
- program relocation
- protection
Pages: Virtual Memory Blocks

Page faults: the data is not in memory, retrieve it from disk

huge miss penalty (remember 10 months at human scale)

Pages should be fairly large (e.g., 4KB)

Find something else to do while waiting

reducing page faults is important (LRU is worth the price)

can handle the faults in software instead of hardware

using write-through is too expensive so we use writeback
One page table per process!
Where are the page tables?

Page tables are potentially BIG

4kB page, 4MB program, 1k page table entries per program!

Powerpoint 18MB
GBMail 32MB
SpamFilter 48MB
mySQL 40MB
iCalMinder 5MB
iCal 9MB
Explorer 20MB
40 More Processes!

Page the page tables!
Have to look up EVERY address!
Making Address Translation Fast

TLB = Translation Lookaside Buffer
a CACHE for address translation
What is in the page table?

Address = upper bits of physical memory address OR disk address of page if not in memory
Valid = bit, set if page is in memory
Use = bit, set when page is accessed
Protection = bit (or bits) to specify access permissions
Dirty = bit, set if page has been written
Integrating TLB and Cache

- **Virtual address**
  - TLB access
    - TLB hit? (Yes → Physical address)
    - TLB miss exception (No → Cache miss stall)
    - No → Write?
      - Write? (Yes → Write access bit on?)
        - Write access bit on? (Yes → Write data into cache, update the tag, and put the data and the address into the write buffer)
        - Write access bit on? (No → Write protection exception)
      - Write? (No → Try to read data from cache)
        - Cache hit? (Yes → Deliver data to the CPU)
        - Cache hit? (No → Cache miss stall)
- Yes → Physical address
Program Relocation?

We want to run multiple programs on our computer “simultaneously”

To start a new program

Without Virtual Memory:

We have to modify all the address references to correspond to the range chosen. This is “relocation”.

With Virtual Memory:

EVERY program can pretend that it has ALL of memory. TEXT segment always starts at 0, STACK always resides a some huge high address (Oxffffffff0)
Protection?

We’d like to protect one program from the errors of another

Without Virtual Memory (old Macs, win3-)
   One program goes bad (or the programmer makes a mistake) and kills another program or the whole system!

With Virtual Memory (new Macs, win95+)
   Every program is isolated from every other. You can’t even NAME the addresses in another program.
   Each page can have read, write, and execute permissions
Some Issues

Processor speeds continue to increase
— much faster than either DRAM or disk access times

Design challenge: dealing with this growing disparity

Trends:
  synchronous SRAMs (provide a burst of data)
  redesign DRAM chips to provide higher bandwidth or processing
  restructure code to increase locality
  use prefetching (make cache visible to ISA)
What cache and VM have in common

Question 1: Where can a block be placed?

Question 2: How is a block found?

Question 3: Which block should be replaced on a cache miss?

Question 4: What happens on a write?
Where can a block be placed?

Direct Mapped Cache: only 1 place for any block
(many blocks map to the same place)
2-Way Set Associative: 2 places for any block
4-Way Set Associative: 4 places for any block
Fully Associative: anywhere
Virtual Memory: anywhere
How is a block found?

Direct mapped cache: compute the index

Set associative cache: compute the index, then search

Fully associative cache: search all cache entries

Virtual memory: separate lookup table
Which block is replaced on miss?

Direct mapped cache have no choice

Others can use:

Random replacement

Least Recently Used (LRU)

Other schemes
What happens on write?

**Write-through:** write to both the cache and the next lower level

**Write-back:** write only to the cache, remember that we have to write to the lower level on replacement
Misses

Compulsory misses
Capacity misses
Conflict misses