Memory Hierarchy

Lecture 25
CS301
Administrative

- Program #3 due Friday, 12/7 at 4:59pm
- Daily Review due 11/29 at 8am
- Lab #8 due Friday 11/30 at 1:29pm
- HW #9 assigned
  - Due Wednesday, 12/5 at 4:59pm
Virtual Memory
DRAM as cache

- What about programs larger than DRAM?
- When we run multiple programs, all must fit in DRAM!
- Add another larger, slower level to the memory hierarchy – use part of the hard drive.
Virtual Memory

- Memory Size –
- Protection –
Virtual Memory

- Memory Size – allows total memory allocation to exceed DRAM capacity.
- Protection –
Virtual Memory

• Memory Size – allows total memory allocation to exceed DRAM capacity.

• Protection – programs may not access each other’s memory.
Multi-Processing – no VM

- Program A begins
Multi-Processing – no VM

- Program A begins
- Program B begins

What happens if A wants more memory?
Multi-Processing – no VM

- Program A begins
- Program B begins

What happens if A wants more memory?
Out of luck. If we gave A memory after the end of B, then A would be able to access all of B’s memory.
Multi-Processing – no VM

- Program A begins
- Program B begins
- Program A ends
Multi-Processing – no VM

- Program A begins
- Program B begins
- Program A ends
- Program C ready
Multi-Processing – no VM

- Program A begins
- Program B begins
- Program A ends
- Program C ready
  - It can not run even though there is enough free space
  - Fragmentation
Virtual Memory

- Use hard drive for memory that does not fit in DRAM
- Allocate memory in pages
- Provide protection by page
Address Space

- Virtual Address Space
- Physical Address Space
Address Space

- Virtual Address Space
  - Located on Hard Drive (essentially)
  - Starts at 0 for each program
- Physical Address Space
Address Space

- Virtual Address Space
  - Located on Hard Drive (essentially)
  - Starts at 0 for each program
- Physical Address Space
  - Located in DRAM
  - The “cache” for the Disk Drive
Multi-Processing – VM

- Program A begins
- Program B begins
Multi-Processing – VM

- Program A begins
- Program B begins
Multi-Processing – VM

- Program A begins
- Program B begins

What happens if A wants more memory?
Multi-Processing – VM

- Program A begins
- Program B begins

What happens if A wants more memory? Allocate another virtual page.
Multi-Processing – VM

- Program A begins
- Program B begins
- Program A ends
Multi-Processing – VM

- Program A begins
- Program B begins
- Program A ends
- Program C begins
  - Not all placed in DRAM
  - DRAM use need not be contiguous
Virtual Memory is like caching...

- _______ is the cache for the ___________
  - It contains only a subset of the total space

- Given an address, determine whether it is currently in the “cache”

- On a miss, obtain data and place in “cache”
Virtual Memory is like caching...

- **DRAM** is the cache for the **hard drive**
  - It contains only a subset of the total space
- Given an address, determine whether it is currently in the “cache”
- On a miss, obtain data and place in “cache”
Virtual Memory is *not* like caching…

- The miss penalty is orders of magnitude larger than for the cache
- You must know where it resides in DRAM before you can look it up in L1 cache

- This leads to a **much** different implementation
The Search

- Cache – search each block in set for the proper tag
- Virtual Memory – store a table that is a mapping from virtual address to physical location (DRAM location).
Virtual Memory Implementation

- Programs use virtual addresses
- VM Block is called a __________
- A VM DRAM “cache miss” is called a ______________.
- To access data, the address must translate it to a ________________.
- This translation is called ________________ or ________________.
Virtual Memory Implementation

- Programs use **virtual** addresses
- VM Block is called a **page**
- A VM DRAM “cache miss” is called a ________________.
- To access data, the address must translate it to a ________________.
- This translation is called ________________ or ________________.
Virtual Memory Implementation

- Programs use virtual addresses
- VM Block is called a page
- A VM DRAM “cache miss” is called a page fault.
- To access data, the address must translate it to a _______________.
- This translation is called _______________ or _______________.

Virtual Memory Implementation

- Programs use **virtual** addresses
- VM Block is called a **page**
- A VM DRAM “cache miss” is called a **page fault**.
- To access data, the address must translate it to a **physical address**.
- This translation is called ______________ or ________________.
Virtual Memory Implementation

- Programs use **virtual** addresses
- VM Block is called a **page**
- A VM DRAM “cache miss” is called a **page fault**.
- To access data, the address must translate it to a **physical address**.
- This translation is called **memory mapping** or **virtual to physical translation**.
Virtual Memory Implementation

• Translation process:

  Virtual page number  Page offset

  Translation

  Physical page number  Page offset

• Why is Physical address smaller than Virtual?
Translation process:

Why is Physical address smaller than Virtual? DRAM is the cache – should be smaller than the total virtual address space.
Virtual Memory Implementation

Page faults incredibly costly

• DRAM is a cache –
  ✷ Direct-mapped?
  ✷ Set-associative?
  ✷ Fully associative?

• Low associativity
  ✷ _____miss rate,______ search time

• High associativity
  ✷ _____miss rate,______ search time
Virtual Memory Implementation

Page faults incredibly costly

- DRAM is a cache –
  - Direct-mapped?
  - Set-associative?
  - Fully associative?

- Low associativity
  - high miss rate, low search time

- High associativity
  - low miss rate, high search time
Virtual Memory Implementation

Page faults incredibly costly

- DRAM is a cache –
  - Direct-mapped?
  - Set-associative?
  - Fully associative?

- Low associativity
  - high miss rate, low search time

- High associativity
  - low miss rate, high search time

Access time: ~50 cycles + search, miss penalty: hundreds of cycles
Virtual Memory Implementation

Page fault incredibly costly

- DRAM is a cache –
  - Direct-mapped?
  - Set-associative?
  - **Fully associative!!!!**

- Low associativity
  - high miss rate, low search time

- High associativity
  - low miss rate, high search time

Access time: ~50 cycles + search, miss penalty: hundreds of cycles
Virtual Memory Implementation
Page fault incredibly costly

- Fully associative!!!!
- Large block size (4KB–16KB)
- Sophisticated software to implement replacement policy
  - updates are hidden under page fault penalty
Address Space

• Virtual Address Space
  ✷ Located on Hard Drive (essentially)
  ✷ Starts at 0 for *each* program

• Physical Address Space
  ✷ Located in DRAM
  ✷ The “cache” for the Disk Drive
Translation

- Does the same virtual address (in all processes) always translate to the same physical address (at one moment in time)?

- How often does a translation occur?
Translation

• Does the same virtual address always translate to the same physical address?
  - No – each process has same virtual addresses

• How often does a translation occur?
Translation

• Does the same virtual address always translate to the same physical address?
  ✷ No – each process has same virtual addresses
  ✷ Store translations in process page table

• How often does a translation occur?
Translation

• Does the same virtual address always translate to the same physical address?
  ◦ No – each process has same Virtual addresses
  ◦ Store translations in process page table

• How often does a translation occur?
  ◦ At least once per instruction
Translation

• Does the same virtual address always translate to the same physical address?
  - No – each process has same Virtual addresses
  - Store translations in process page table

• How often does a translation occur?
  - At least once per instruction
  - Need to perform translation quickly
  - Cache recent translations!
Translation

• Maintaining per process page tables
  ♦ Process page table maintained by ____ – ________________

• Making translations fast
  ♦ Use a TLB (__________________________) to cache recent translations
  ♦ Fully associative but ____________________
Translation

- Maintaining per process page tables
  - Process page table maintained by OS – some pinned into DRAM
- Making translations fast
  - Use a TLB (__________________________) to cache recent translations
  - Fully associative but ________________
Translation

• Maintaining per process page tables
  ♦ Process page table maintained by OS – some pinned into DRAM

• Making translations fast
  ♦ Use a TLB (Translation Lookaside Buffer) to cache recent translations
  ♦ Fully associative but ______________
Translation

• Maintaining per process page tables
  ✷ Process page table maintained by OS – some pinned into DRAM

• Making translations fast
  ✷ Use a Translation Lookaside Buffer (TLB) to cache recent translations
  ✷ Fully associative but very small – 16–64 entries
Step 1: TLB

- Search all locations of TLB in parallel (fully-associative)
Step 1: TLB

- Search all locations of TLB in parallel
- Hit –
- Miss –
Step 1: TLB

- Search all locations of TLB in parallel
- Hit – return address, proceed with memory access
- Miss –
Step 1: TLB

- Search all locations of TLB in parallel
- Hit – return address, proceed with memory access
- Miss – retrieve translation from process page table –
Step 1: TLB

- Search all locations of TLB in parallel
- Hit – return address, proceed with memory access
- Miss – retrieve translation from process page table
  - Exception/Trap to OS
Step 2: TLB Miss

- Access the process’ page table to retrieve translation
- If valid (DRAM hit)
  - fill in TLB with this entry
  - restart TLB access and translation
- If invalid, page fault (DRAM miss)
Step 3: Page Fault

- Operating System invoked to swap a page in DRAM with the page requested on hard drive.
- Operating system looks up page’s location on hard drive.
- Operating system maintains replacement algorithm.
- OS updates the process’ page tables.
Putting it all together

TLB Access

TLB Hit?

Cache Hit?

Cache read

Stall

Yes

No

TLB Miss
Exception

Write?

Write access bit on?

Write protection
Exception

Write to $(Depends)

Yes

No

No

Yes

Return data

Exception

No

Yes
Why do we have to wait for the TLB access to complete before accessing the cache?

What happens on a write miss to the $?

What is the maximum number of misses encountered on a read request?
Why do we have to wait for the TLB access to complete before accessing the cache?

- Cache is indexed using physical addresses which we don’t have until after the TLB request finishes.

What happens on a write miss to the $?

What is the maximum number of misses encountered on a read request?
Why do we have to wait for the TLB access to complete before accessing the cache?
- Cache is indexed using physical addresses which we don’t have until after the TLB request finishes.

What happens on a write miss to the $?
- Evict line
- Go get your line
- Write data into $ line

What is the maximum number of misses encountered on a read request?
TLB

• Why do we have to wait for the TLB access to complete before accessing the cache?
  - Cache is indexed using physical addresses which we don’t have until after the TLB request finishes.
• What happens on a write miss to the $?
  - Evict line
  - Go get your line
  - Write data into $ line
• What is the maximum number of misses encountered on a read request?
  - TLB miss, process page table miss (may require going all the way to disk, creating its own page fault), page fault, L1 cache miss, L2 cache miss, ..., hits in memory because of page fault but might not be in any caches
Virtual Memory Summary

- VM increases total available memory
- VM provides multi-process protection
- TLB is necessary for fast translations
- OS manages virtual memory
  - Therefore it is slooooooooow