Lecture 22 (Wed 11/19/2008)

- Lab #4 Software Simulation – Due Fri Nov 21 at 5pm
- HW #3 Cache Simulator & code optimization – Due Mon Nov 24 at 5pm

- More Virtual Memory

1

Virtual Memory Review

2

Paging system summary (so far)

- Addresses generated by the CPU are virtual addresses
- In order to access the memory hierarchy, addresses must be translated into physical addresses
- That translation is done on a program per program basis. Each program must have its own page table
  - All of the address you use in assembly programming are virtual addresses
  - The virtual address of program A and the same virtual address in program B will, in general, map to two different physical addresses

3

Page faults

- When a virtual address has no corresponding physical address mapping (valid bit is off in the PTE) we have a *page fault*
- On a page fault (a page fault is an exception)
  - the faulting page must be fetched from disk (takes milliseconds)
  - the whole page (e.g., 4 or 8KB) must be fetched (amortize the cost of disk access)
  - because the program is going to be idle during that page fetch, the CPU better be used by another program. On a page fault, the state of the faulting program is saved and the O.S. takes over. This is *context-switching*

4

Top level *questions* for paging systems

- When do we bring a page into main memory?
- Where do we put it?
- How do we know it's there?
- What happens if main memory is full?

Top level *answers* for paging systems

- When do we bring a page into main memory?
  - When there is a page fault for that page, i.e., on demand
- Where do we put it?
  - No restriction; mapping is fully-associative
- How do we know it's there?
  - The corresponding PTE entry has its valid bit on
- What happens if main memory is full
  - We have to replace one of the virtual pages currently mapped. Replacement algorithms can be sophisticated (see CSE 451) since we have a context-switch and hence plenty of time

Translation Buffers (TLBs)

- To perform virtual to physical address translation we need to look-up a page table entry
- Since the page table is in memory, need to access memory

P_addr = MEM[Pg_Tab_Base + (V_addr[31:12]<<2)] + V_addr[11:0]

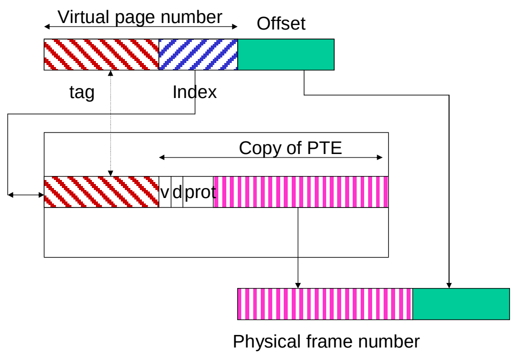
- Too time consuming! 50+ cycles per memory reference!

- Hence we need to cache the page tables
- For that purpose special caches named *translation buffers* are part of the memory system
  - Also named Translation Lookaside Buffers (TLBs)

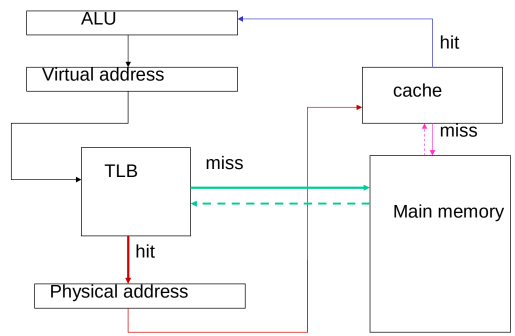
TLB Organization

- TLB organized as caches
- For each entry in the TLB we'll have
  - a tag to check that it is the right entry
  - data which instead of being the contents of memory locations, like in a cache, will be a page table entry (PTE)
- TLB's are smaller than memory caches
  - 32 to 128 entries
  - from fully associative to direct-mapped
  - there can be an instruction TLB, a data TLB and also distinct TLB's for user and system address spaces

TLB organization



Virtual Address to Physical Address (Revisited)



Address Translation

- At each memory reference the hardware searches the TLB for the translation
  - TLB hit and valid PTE the physical address is passed to the cache
  - TLB miss, either hardware or software (depends on implementation) searches page table in memory
    - If PTE is valid, contents of the PTE loaded in the TLB and back to step above
- In hardware the TLB miss takes 10-100 cycles
- In software takes up to 100 -1000 cycles
- In either case, no context-switch
  - Context-switch takes more cycles than a TLB miss
- If PTE is invalid, we have a page fault (even on a TLB hit)

TLB Management

TLBs are caches

- If small (e.g. 32 entries), can be fully associative
- Current trend: larger (about 128 entries); separate TLB's for instruction and data; Some part of the TLB reserved for system
- TLBs are write-back. The only thing that can change is dirty bit + any other information needed for page replacement algorithm (see CSE 451)

MIPS 3000 TLB (old)

- 64 entries: fully associative. "Random" replacement; 8 entries used by system
- On TLB miss, we have a *trap;* software takes over but no context-switch

TLB Management (continued)

- At context-switch, the virtual page translations in the TLB are not valid for the new task
  - Invalidate the TLB (set all valid bits to 0)
  - Or append a Process ID (PID) number to the tag in the TLB. When a new task takes over, the O.S. creates a new PID.
  - PID are recycled and entries corresponding to "old PID" are invalidated

Paging systems -- Hardware/software interactions

- Page tables
  - Managed by the O.S.
  - Address of the start of the page table for a given process is found in a special register which is part of the *state* of the process
  - The O.S. has its own page table
  - The O.S. knows where the pages are stored on disk
- Page fault
  - When a program attempts to access a location which is part of a page that is not in main memory, we have a *page fault*

Page fault detection (simplified)

- Page fault is an *exception*
- Detected by the hardware (invalid bit in PTE either in TLB or page table)
- To resolve a page fault takes millions of cycles (disk I/O)
  - The program that has a page fault must be interrupted
- A page fault occurs in the middle of an instruction
  - In order to restart the program later, the state of the program must be saved and instructions must be restartable (precise exceptions)
- State consists of all registers, including PC and special registers (such as the one giving the start of the page table address)

Page fault handler (simplified)

- Page fault exceptions are cleared by an O.S. routine called the page fault handler which will
  - Grab a physical frame from a free list maintained by the O.S.
  - Find out where the faulting page resides on disk
  - Initiate a read for that page
  - Choose a frame to free (if needed), i.e., run a replacement algorithm
  - If the replaced frame is dirty, initiate a write of that frame to disk
  - Context-switch, i.e., give the CPU to a task ready to proceed

Completion of page fault

- When the faulting page has been read from disk (a few ms later)
  - The disk controller will raise an *interrupt* (another form of exception)
  - The O.S. will take over (context-switch) and modify the PTE (in particular, make it valid)
  - The program that had the page fault is put on the queue of tasks ready to be run
  - Context-switch to the program that was running before the interrupt occurred

17

Two Extremes in Memory Hierarchy

PARAMETER	L1	PAGING SYSTEM
block (page) size	16-64 bytes	4K-8K (also 64K)
miss (fault) time	10-100 cycles (20-1000 ns)	Millions of cycles (3-20 ms)
miss (fault) rate	1-10%	0.00001-0.001%
memory size	4K-64K Bytes (impl. depend.)	Gigabytes (depends on ISA)

18

Other extreme differences

- Mapping: Restricted (L1) vs. General (Paging)
  - Hardware assist for virtual address translation (TLB)
- Miss handler
  - Hardware only for caches
  - Software only for paging system (context-switch)
  - Hardware and/or software for TLB
- Replacement algorithm
  - Not that important for caches
  - Very important for paging system
- Write policy
  - Always write back for paging systems

19