Exceptions
Input/Output
Announcements

• Lab 4C due tomorrow

• Returning FPGA boards
  – tiny.cc/ece2300-fpga-return
  – Fill out this form by Sunday May 16
  – Kit return deadline Friday May 28

• Final Exam: Sunday May 23, 7:30pm, 110mins
  – More details to follow on Thursday
  • Review session, office hours, etc.
  • Sample exam will be posted soon
  – Inform instructor about schedule conflict ASAP
Review: Virtual Address and Page Table

VPN analogous to the index for a cache

Page table stored in MM

Page table register

Virtual address

31 30 29 28 27 15 14 13 12 11 10 9 8 3 2 1 0

Virtual page number

Page offset

Valid

Physical page number

Page table entries (PTEs)

Indicates that a page contains $2^{12} = 4$ KB

If 0 then page is not present in memory

Physical page number

Page offset

Physical address
Example: Page Fault

- Given the following virtual address stream (in decimal), identify the potential page faults
  - 256 (VPN=0), 2049 (VPN=0), 4096 (VPN=1), 1024 (VPN=0)

<table>
<thead>
<tr>
<th>Valid</th>
<th>Physical Page#</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disk</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>0</td>
<td>Disk</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Faster Address Translation

• Have to access the page table before an instruction can be fetched and before data cache/memory can be accessed

• Page table accesses have good locality
  ⇒ Cache the most recent PTEs within the CPU
Translation Lookaside Buffer (TLB)

- Small cache of recently accessed PTE (typically 16-512 entries, fully associative)

![Diagram of TLB and page table]

- Tag: virtual page number
- Data: physical page number
- Physical memory
- Disk storage

accessed only on a TLB miss
TLB Miss Scenarios

• **TLB miss, page table hit**
  – Bring in the PTE information from page table to TLB
  – Retry the access
  – Usually completely performed by hardware

• **TLB miss, page fault**
  – Bring in the page from disk (orchestrated by OS)
  – Load the page table and TLB (orchestrated by OS)
  – Retry the access
    • Cache miss will definitely occur!
Exceptions and Interrupts

• Useful methods for signaling the CPU that some event has occurred that requires action
  – In response, the CPU may *suspend* the running program in order to handle the exception/interrupt

• **Exceptions** are used to handle conditions that arise when executing instructions on the processor
  – Detected by the processor itself

• **Interrupts** are used to handle (asynchronous) events external to the processor
  – I/O device request, external error or malfunction
Why are Exceptions Useful?

• **Allow user programs to get service from the OS**
  – A system call creates an exception that kicks out the user program and transfers control to exception handler, which will invoke a proper OS service routine.

• **Handle unexpected events**
  – Memory protection violation, divide-by-zero, invalid opcode, etc.

• **Handle page faults**
Pipeline with Exception Handling
Pipeline with Exception Handling

• An Exception signal is sent to the CU

• The CU sets the Exception PC (with the address of the faulting instruction) and Cause

• All instructions before the exception complete

• The faulting instruction (that causes the exception), and any behind it in the pipeline, are turned into NOPs

• The PC of the first instruction in the exception handler code is loaded into the PC register
Enabling Program Restart

- After the exception is handled, want to restart the program starting at the faulting instruction

- The program state is saved by the exception handler into memory, and restored when the exception has been handled
Instruction Page Fault

[SUB R5, R5, R7]  
<not in main memory>

LW  R4, 0(R1)  
ADD R1, R2, R3
Instruction Page Fault

```
[SUB R5,R5,R7]  
<TLB miss>

Lecture 24: 14
```
Instruction Page Fault

[SUB R5, R5, R7]
<stall>
<access page table>

<LW and ADD have completed>
Instruction Page Fault

[SUB R5,R5,R7]

<page fault>
Instruction Page Fault

[SUB R5,R5,R7]

<page fault>
Instruction Page Fault

[SUB R5, R5, R7]

<page fault>
Instruction Page Fault

[SUB R5,R5,R7]  
<page fault>
Instruction Page Fault

1st instruction in exception handler
Handling a Page Fault

ADD R1, R2, R3
LW R4, 0(R1)
SUB R5, R5, R7

TLB miss
Page table miss

CPU generates a page fault exception

OS brings in the page, updates page tables and TLB, restores program state, and resumes the program

exception handler saves program state and transfers control to the OS
Computer with Input/Output

- Processor
  - L1 Inst Cache + Inst TLB
  - L1 Data Cache + Data TLB

- L2 Cache

- Main Memory

- Input/Output
  - disk, keyboard, graphics, network, etc

interconnect (e.g., bus)

Input

Output
Input/Output Devices

• I/O devices are the media to allow computer systems to interact with the outside world
Example Server System with I/O

**DIMM** = “Dual Inline Memory Module” (circuit board with DRAMs on both sides)

**Serial ATA** = “Serial Advanced Technology Attachment” (interface standard for storage device)

**DDR** = “Double Data Rate” (memory interface standard)

**PCIe** = “Peripheral Component Interconnect Express” (high-speed bus standard)

**Serial ATA** = “Serial Advanced Technology Attachment” (interface standard for storage device)

**USB 2.0** (60 MB/sec)

**LPC** (1 MB/sec)

**FB DDR2 667** (5.3 GB/sec)

**Main memory DIMMs**

**Intel Xeon 5300 processor**

**Front Side Bus (1333 MHz, 10.5 GB/sec)**

**Memory controller hub (north bridge) 5000P**

**PCIe x16 (or 2 PCIe x8)** (4 GB/sec)

**PCIe x4** (1 GB/sec)

**PCI-X bus** (1 GB/sec)

**Parallel ATA** (100 MB/sec)

**CD/DVD**

**Keyboard, mouse, ...**

**LPC**

**PCI-X bus** (1 GB/sec)

**PCI-X bus** (1 GB/sec)

**Parallel ATA** (100 MB/sec)

**I/O controller hub (south bridge) Enterprise South Bridge 2**

**Intel Xeon 5300 processor**

**PCIe**

**Serial ATA (300 MB/sec)**

**Disk**

**Disk**

**USB 2.0 (60 MB/sec)**

**LPC (1 MB/sec)**

**Parallel ATA (100 MB/sec)**

**CD/DVD**

**I/O controller hub (south bridge) Enterprise South Bridge 2**

**PCIe x16 (or 2 PCIe x8)** (4 GB/sec)

**PCIe x4** (1 GB/sec)

**PCI-X bus** (1 GB/sec)

**Parallel ATA** (100 MB/sec)

**CD/DVD**

**Intel Xeon 5300 processor**

**Front Side Bus (1333 MHz, 10.5 GB/sec)**

**Memory controller hub (north bridge) 5000P**

**PCIe**

**Serial ATA (300 MB/sec)**

**Disk**

**Disk**

**USB 2.0 (60 MB/sec)**

**LPC (1 MB/sec)**

**Parallel ATA (100 MB/sec)**

**CD/DVD**

**Intel Xeon 5300 processor**

**Front Side Bus (1333 MHz, 10.5 GB/sec)**

**Memory controller hub (north bridge) 5000P**

**PCIe**

**Serial ATA (300 MB/sec)**

**Disk**

**Disk**

**USB 2.0 (60 MB/sec)**

**LPC (1 MB/sec)**

**Parallel ATA (100 MB/sec)**

**CD/DVD**

**Intel Xeon 5300 processor**

**Front Side Bus (1333 MHz, 10.5 GB/sec)**

**Memory controller hub (north bridge) 5000P**

**PCIe**

**Serial ATA (300 MB/sec)**

**Disk**

**Disk**

**USB 2.0 (60 MB/sec)**

**LPC (1 MB/sec)**

**Parallel ATA (100 MB/sec)**

**CD/DVD**

**Intel Xeon 5300 processor**

**Front Side Bus (1333 MHz, 10.5 GB/sec)**

**Memory controller hub (north bridge) 5000P**

**PCIe**

**Serial ATA (300 MB/sec)**

**Disk**

**Disk**

**USB 2.0 (60 MB/sec)**

**LPC (1 MB/sec)**

**Parallel ATA (100 MB/sec)**

**CD/DVD**

**Intel Xeon 5300 processor**

**Front Side Bus (1333 MHz, 10.5 GB/sec)**

**Memory controller hub (north bridge) 5000P**

**PCIe**

**Serial ATA (300 MB/sec)**

**Disk**

**Disk**

**USB 2.0 (60 MB/sec)**

**LPC (1 MB/sec)**

**Parallel ATA (100 MB/sec)**

**CD/DVD**
**I/O Controller**

- **An I/O controller manages one or more peripheral devices**
  - Function: coordinates data transfers between the device(s) and the rest of computer system
  
- Interface: contains a set of special registers for communication with the processor
  - **Command registers**
    - Tells the device to do something
    - Written by CPU/OS
  - **Status registers (read by processor/OS)**
    - Indicates the status of the device (ready, busy, error)
    - Read by CPU/OS
  - **Data input/output registers**
Accessing I/O Devices

• How do we get a command/data to the right device?

• **Dedicated I/O instructions**
  – Separate Load/Store instructions to access I/O registers
  – Only the OS can use these instructions

• **Memory-mapped I/O**
  – Portion of the physical address space is assigned to I/O device registers
  – Only the OS can access these addresses
  – Each I/O device register has a unique memory address
Data Transfer Between I/O and Memory

- **Programmed I/O (PIO)**
  - Processor completely arbitrates transfer of data from device to memory
  - Typically much less efficient than DMA

- **Direct Memory Access (DMA)**
  - I/O device transfers data directly to main memory
  - Processor/OS sets up the transfer through I/O commands
  - And then can do something else, like run another program
Informing the Processor

- A device needs to inform the processor when an I/O operation is completed

- **Polling**
  - Processor periodically reads the Status Register, which indicates when an operation is done

- **Interrupt-driven I/O**
  - I/O device sends an interrupt to the processor when the operation is done
  - More efficient than polling
Let’s Pull Some Pieces Together

- Page fault occurs
- Exception handler gets loaded
- Exception handler takes action
- OS sets up disk transfer
- OS schedules another program
- Data is read from disk and transferred to main memory
- Disk controller interrupts processor
- Second program is interrupted
- First program can run again
Data Page Fault Occurs in Program A

INSTR

Lecture 24: 30
Exception Handler Gets Loaded

PC of the exception handler

1st instruction in exception handler
Exception Handler Takes Action

• Saves program A state

• Reads the Cause register and determines that a page fault occurred

• Calls the appropriate part of the OS
OS Sets Up Disk Transfer using DMA

“read 4K bytes from disk starting at addr X to memory starting at addr Y”
OS Schedules Another Program

• While waiting for the disk read to complete for program A, the OS scheduler may run a different task (program B)

• It loads the processor with the state (PC, PTR, and RF) of program B
OS Switches from Program A to B

Page Table Register

Exception PC
Exception Cause

Control Signals

CU

sign bit

=?

Adder

ID/EX

EX/MEM

MEM/WB

IF/ID

LD
SA
SB
DR

D_in

RF

F_m ... F_0

ALU

MB

L1
Data

$D_IN

TLB

Data

TLB

Exception

PC

Exception

Cause

EL

MW

L1
Data

$
Page is Read from Disk

(now running program B)

Intel Xeon 5300 processor

Front Side Bus (1333 MHz, 10.5 GB/sec)

Serial ATA (300 MB/sec)

Disk

Main memory DIMMs

FB DDR2 667 (5.3 GB/sec)

PCIe x16 (or 2 PCIe x8) (4 GB/sec)

Intel Xeon 5300 processor

PCIe

LPC (1 MB/sec)

USB 2.0 (60 MB/sec)

Parallel ATA (100 MB/sec)

PCI-X bus (1 GB/sec)

PCI-X bus (1 GB/sec)

CD / DVD

“read 4K bytes starting at addr X”

Keyboard, mouse, ...

PCIe x4 (1 GB/sec)

PCIe x4 (1 GB/sec)

memory controller

NB-SB interface

SATA controller

interface

interface
Page is Read from Disk

(now running program B)

Intel Xeon 5300 processor

Front Side Bus (1333 MHz, 10.5 GB/sec)

FB DDR2 667 (5.3 GB/sec)

Main memory DIMMs

Serial ATA (300 MB/sec)

Disk

PCIe x16 (or 2 PCIe x8) (4 GB/sec)

Intel Xeon 5300 processor

FSB interface

memory controller

NB-SB interface

SATA controller

LPC (1 MB/sec)

Keyboard, mouse, ...

USB 2.0 (60 MB/sec)

PCI-X bus (1 GB/sec)

PCI-X bus (1 GB/sec)

Parallel ATA (100 MB/sec)

CD / DVD

data returned to disk controller

PCIe

PCIe x4 (1 GB/sec)

PCIe x4 (1 GB/sec)
DMA Transfer of Page to Memory

(now running program B)

data written to memory location Y.

Intel Xeon 5300 processor

Front Side Bus (1333 MHz, 10.5 GB/sec)

PCIe x16 (or 2 PCIe x8) (4 GB/sec)

PCIe x4 (1 GB/sec)

PCI-X bus (1 GB/sec)

Parallel ATA (100 MB/sec)

CD/DVD

SATA controller

NB-SB interface

memory controller

FSB interface

data transferred to memory controller

data transferred to memory controller

LPC (1 MB/sec)

USB 2.0 (60 MB/sec)

Keyboard, mouse, ...

Main memory DIMMs

FB DDR2 667 (5.3 GB/sec)

Disk

Disk

Serial ATA (300 MB/sec)

PCIe

PCle x4 (1 GB/sec)

PCle x4 (1 GB/sec)

PCle x4 (1 GB/sec)

PCI-X bus (1 GB/sec)

PCI-X bus (1 GB/sec)
I/O Controller Interrupts Processor

(now running program B)

interrupt
“I/O complete”

Intel Xeon 5300 processor

Front Side Bus (1333 MHz, 10.5 GB/sec)

FSB interface

memory controller

NB-SB interface

SATA controller

Intel Xeon 5300 processor

Main memory DIMMs

FB DDR2 667 (5.3 GB/sec)

Serial ATA (300 MB/sec)

Disk

LPC (1 MB/sec)

Keyboard, mouse, ...

USB 2.0 (60 MB/sec)

Pcie x16 (or 2 Pcie x8) (4 GB/sec)

Pcie x4 (1 GB/sec)

PCI-X bus (1 GB/sec)

PCI-X bus (1 GB/sec)

Parallel ATA (100 MB/sec)

CD / DVD
Program B is Interrupted

The state of program B is saved by the interrupt handler
Program A Can Now Run Again

- Page fault was handled, so OS marks program A as runnable

- If OS scheduler chooses to run program A, it loads its state (PC of the LW, PTR, and registers)

- Key point: Processor was free to do other work during the long I/O transfer time
  - DMA: Processor did not have to directly handle data transfers from device to memory
  - Interrupt-driven I/O: Processor did not have to poll the device to see when the I/O operation completed
Building a Complete Computer

Diagram showing the components of a computer system, including processor, cache, main memory, I/O controllers, disks, graphics output, and network.
Next Time

More on Final Exam
Advanced Topics