ECE 2300
Digital Logic & Computer Organization
Spring 2018

Exceptions
Input/Output
Announcements

• **HW8 due tomorrow**

• **Lab 6 on Monday/Tuesday**
  – Please return your FPGA boards

• **Final Exam: Tuesday May 22, 2pm, Hollister B14**
  – Covers the entire course
  – An old exam will be posted over the weekend
  – More details to follow next Tuesday
    • Review session, office hours, etc.
  – **Inform instructor about schedule conflict ASAP**
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>
</tbody>
</table>
Virtual Memory Write Policy

• Disk writes can take millions of cycles

• Write back policy is used (instead of write through)
  – *Dirty* bit in PTE is set on a write to main memory
  – Page with set Dirty bit is written to disk if replaced
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)
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

- Exception handler
- Control Signals
- Exception PC
- Exception Cause
- Sign bit
- ALU
- MUX
- PC
- L1 Inst Cache
- SE
- Decoder
- RF
- LD, SA, SB, DR, D_in
- Adder
- MentorWare
- D_IN
- MUX
- MD
- L1 Data Cache
- Data TLB
- EX/MEM
- MEM/WB

PC of the exception handler
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]

LW R4,0(R1)

ADD R1,R2,R3

<not in main memory>
Instruction Page Fault

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

LW R4,0(R1)

ADD R1,R2,R3

Lecture 24: 14
Instruction Page Fault

- Exception PC
- Exception Cause
- Control Signals
- Sign bit
- Adder
- RF
- LD, SA, SB, DR
- D_in
- Exception
- MW
- L1 Data Cache
- EL
- F_m … F_0
- ALU
- MB
- ID/EX
- EX/MEM
- MEM/WB
- [SUB R5,R5,R7]
- <stall>
- <access page table>
- PC of the exception handler
- Inst TLB
- Decoder
- IF/ID
- MUX
- PCJ
- PCL
- L1 Inst Cache
- IF/ID
- <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

PC of the exception handler

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

1st instruction in exception handler
Handling a Page Fault

- **Instruction page boundary**
- **ADD R1,R2,R3**
- **LW R4,0(R1)**
- **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**

Lecture 24: 21
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
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**
Example Server System with I/O

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)

Main memory DIMMs

FB DDR2 667 (5.3 GB/sec)

Disk

Serial ATA (300 MB/sec)

Disk

LPC (1 MB/sec)

Keyboard, mouse, ...

USB 2.0 (60 MB/sec)

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

PCIe x4 (1 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
Accessing I/O Devices

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

• Dedicate 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
Reading Data from Disk

Intel Xeon 5300 processor

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

Main memory DIMMs

FB DDR2 667 (5.3 GB/sec)

Serial ATA (300 MB/sec)

LPC (1 MB/sec)

USB 2.0 (60 MB/sec)

Keyboard, mouse, ...

Disk

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

Memory controller

FSB interface

NB-SB interface
Reading Data from Disk

“read N bytes from disk starting at addr X to memory starting at addr Y”
DMA Transfer from Disk Controller

- Data written to memory location Y
- Data returned to SATA controller

Flowchart Diagram:
- Intel Xeon 5300 processor
- Front Side Bus (1333 MHz, 10.5 GB/sec)
- FB DDR2 667 (5.3 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

Connections:
- SATA controller
- NB-SB interface
- Memory controller
- FSB interface
- Main memory DIMMs
- Disk
- Serial ATA (300 MB/sec)
- LPC (1 MB/sec)
- USB 2.0 (60 MB/sec)
- Keyboard, mouse, ...

Flow of Data:
- Data transferred to memory controller
- Data returned to SATA controller
- Data written to memory location Y

Lecture 24: 30
Informing the Processor

• Need 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
Informing Using an Interrupt

Interrupt
“I/O complete”

Intel Xeon 5300 processor

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

FB DDR2 667 (5.3 GB/sec)

Main memory DIMMs

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

Serial ATA (300 MB/sec)

PCIe

PCle x4 (1 GB/sec)

Disk

PCIe x4 (1 GB/sec)

LPC (1 MB/sec)

PCI-X bus (1 GB/sec)

Keyboard, mouse, ...

USB 2.0 (60 MB/sec)

Parallel ATA (100 MB/sec)

CD/DVD

Serial ATA controller

Serial ATA controller

PCIe controller

FSB interface

NB-SB interface

memory controller
Let’s Pull Some Pieces Together
Data Page Fault Occurs in Task A

- Exception signal triggers a control signal to the CU.
- The PC of LW (Load Word) is marked as "page fault".
- Exception PC and Exception Cause are indicated.
- The sign bit is checked.
- The PC of the exception handler is determined.

- Instruction TLB miss and page fault are indicated.
- LW R1,0(R1) operation with TLB miss and page fault.

- Control Signals flow through the CU, ALU, and other components.
- IF/ID, ID/EX, EX/MEM, MEM/WB stages are shown.
- MF, MUX, Adder, RF, LD, SA, SB, DR, EL, MB, Data TLB, Data Cache, MW, MD are connected.

Lecture 24: 34
Exception Handler Gets Loaded

1st instruction in exception handler

PC of the exception handler

PCJ

PC

PCL

Inst TLB

L1 Cache

IF/ID

Decoder

+2

MUX

PC

MUX

PCJ

L1 Inst Cache

SE

MUX

SE

MUX

RF

D_in

LD
SA
SB
DR

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

Exception PC

Exception Cause

Control Signals

sign bit

Exception Handler Gets Loaded

1st instruction in exception handler

Exception PC

EL
F_m...F_0

D_IN

ALU

MB

L1 Data Cache

Data TLB

MW

MD

NOP

NOP

NOP

NOP
Exception Handler Takes Action

• Saves task 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

Intel Xeon 5300 processor

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

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

Storage Devices

Disk (300 MB/sec)

Serial ATA (5.3 GB/sec)

Main memory DIMMs

Main memory (5.3 GB/sec)

“read 4K bytes from disk starting at addr X to memory starting at addr Y”

FSB interface

memory controller

NB-SB interface

SATA controller

CD/DVD

Keyboard, mouse, ...

LPC (1 MB/sec)

USB 2.0 (60 MB/sec)
OS Schedules Another Task

- While waiting for the disk read to complete, the OS scheduler runs a different task (task B)

- It loads the processor with the program state of task B
OS Switches from Task A to Task B

The program state of Task A needs to be saved
OS Switches from Task A to Task B

1. What do we do about the PC?

2. What do we do about the registers in RF?

3. What do we do about the TLBs?

4. What do we do about the caches?  
   (Hint: Our cache uses physical addresses)

   Save, Flush, or Do Nothing?
Data are Read from Disk

(now running task B)

Intel Xeon 5300 processor

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

Main memory DIMMs

FB DDR2 667 (5.3 GB/sec)

Serial ATA (300 MB/sec)

Disk

PCIe x8 (4 GB/sec)

PCIe x16 (or 2 PCIe x8)

"read 4K bytes starting at addr X"

Keyboard, mouse, ...

USB 2.0 (60 MB/sec)

LPC (1 MB/sec)

Parallel ATA (100 MB/sec)

PCI-X bus (1 GB/sec)

PCI-X bus (1 GB/sec)

PCIe x4 (1 GB/sec)

PCIe x4 (1 GB/sec)

CD/DVD

Lecture 24: 41
Data are Read from Disk

(now running task B)

- Intel Xeon 5300 processor
- Front Side Bus (1333 MHz, 10.5 GB/sec)

Main memory DIMMS

- FB DDR2 667 (5.3 GB/sec)
- PCIe x16 (or 2 PCIe x8) (4 GB/sec)

Disk

- Serial ATA (300 MB/sec)
- PCIe

- PCIe x4 (1 GB/sec)
- PCIe x4 (1 GB/sec)
- PCI-X bus (1 GB/sec)
- PCI-X bus (1 GB/sec)
- Parallel ATA (100 MB/sec)

Keyboard, mouse, ...

USB 2.0 (60 MB/sec)

CD/DVD

(data returned to disk controller)

FSB interface

memory controller

NB-SB interface

SATA controller
DMA Transfer to Memory

(now running task B)

Intel Xeon 5300 processor

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

FB DDR2 667 (5.3 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

Intel Xeon 5300 processor

memory controller

NB-SB interface

SATA controller

FSB interface

data transferred to memory controller

data written to memory location Y

Main memory DIMMs

Serial ATA (300 MB/sec)

Disk

Disk

LPC (1 MB/sec)

USB 2.0 (60 MB/sec)

Keyboard, mouse, ...

Lecture 24: 43
Device Interrupts Processor

(now running task B)

interrupt
“I/O complete”

FSB (1333 MHz, 10.5 GB/sec)

Intel Xeon 5300 processor

Main memory DIMMs

Disk

Serial ATA (300 MB/sec)

LPC (1 MB/sec)

USB 2.0 (60 MB/sec)

Keyboard, mouse, ...

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

interface

Intel Xeon 5300 processor

Front Side Bus
Task B is Interrupted

PC of the interrupt handler

ADD R1,R2,R3

PCJ

MUX

PCL

IF/ID

Decoder

L1 Instruction Cache

LD,SA,SB,DR

D_in

RF

MUX

Adder

ALU

MUX

ALU

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

MUX

M
Interrupt Handler Gets Loaded

1st instruction in interrupt handler
Task A Can Now Run Again

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

- If OS scheduler chooses to run task 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
Next Time

More on Final Exam
Advanced Topics