Review

• I/O gives computers their 5 senses + long term memory
• I/O speed range is 7 Orders of Magnitude (or more!)
• Processor speed means must synchronize with I/O devices before use
• Polling works, but expensive
  — processor repeatedly queries devices
• Interrupts work, but more complex
  — we’ll talk about these today

Exceptions and Interrupts

• “Unexpected” events requiring change in flow of control
  — Different ISAs use the terms differently
• Exception
  — Arises within the CPU
    • e.g., Undefined opcode, overflow, syscall, TLB Miss,…
• Interrupt
  — From an external I/O controller
• Dealing with them without sacrificing performance is difficult

Handling Exceptions

• In MIPS, exceptions managed by a System Control Coprocessor (CP0)
• Save PC of offending (or interrupted) instruction
  — In MIPS: save in special register called Exception Program Counter (EPC)
• Save indication of the problem
  — In MIPS: saved in special register called Cause register
  — We’ll assume 1-bit
    • 0 for undefined opcode, 1 for overflow
• Jump to exception handler code at address 8000 0180 \text{hex}

Exception Properties

• Restartable exceptions
  — Pipeline can flush the instruction
  — Handler executes, then returns to the instruction
    • Refetched and executed from scratch
• PC saved in EPC register
  — Identifies causing instruction
  — Actually PC + 4 is saved because of pipelined implementation
    • Handler must adjust PC to get right address

Handler Actions

• Read Cause register, and transfer to relevant handler
• Determine action required
• If restartable exception
  — Take corrective action
  — use EPC to return to program
• Otherwise
  — Terminate program
  — Report error using EPC, cause, …
Exceptions in a Pipeline

- Another kind of control hazard
- Consider overflow on add in EX stage
  - Prevent $1 from being clobbered
  - Complete previous instructions
  - Flush add and subsequent instructions
  - Set Cause and EPC register values
  - Transfer control to handler
- Similar to mispredicted branch
  - Use much of the same hardware

Exception Example

- Pipelining overlaps multiple instructions
  - Could have multiple exceptions at once
  - E.g., Page fault in LW same clock cycle as Overflow of following instruction ADD
- Simple approach: deal with exception from earliest instruction, e.g., LW exception serviced 1st
  - Flush subsequent instructions
- Called Precise exceptions
  - In complex pipelines:
    - Multiple instructions issued per cycle
    - Out-of-order completion
    - Maintaining precise exceptions is difficult!

Imprecise Exceptions

- Just stop pipeline and save state
  - Including exception cause(s)
- Let the software handler work out
  - Which instruction(s) had exceptions
  - Which to complete or flush
    - May require “manual” completion
- Simplifies hardware, but more complex handler software
- Not feasible for complex multiple-issue out-of-order pipelines to always get exact instruction
- All computers today offer precise exceptions—affects performance though

I/O Interrupt

- An I/O interrupt is like an exception except:
  - An I/O interrupt is “asynchronous”
  - More information needs to be conveyed
- An I/O interrupt is asynchronous with respect to instruction execution:
  - I/O interrupt is not associated with any instruction, but it can happen in the middle of any given instruction
  - I/O interrupt does not prevent any instruction from completion
Interrupt-Driven Data Transfer

1. I/O interrupt
2. Save PC
3. Jump to interrupt service routine
4. Perform transfer
5. Jump to interrupt service routine

• Find the % of processor consumed if the hard disk is only active 5% of the time. Assuming 500 clock cycle overhead for each transfer, including interrupt:
  - Disk Interrupts/s = 5% * 16 [MB/s] / 16 [B/interrupt] = 50,000 [interrupts/s]
  - Disk Interrupts [clocks/s] = 50,000 [interrupts/s] * 500 [clocks/interrupt] = 25,000,000 [clocks/s]
  - % Processor for during transfer: 2.5*10^7 [clocks/s] / 1*10^9 [clocks/s] = 2.5% Busy
  - DMA (Direct Memory Access) even better – only one interrupt for an entire page!

Benefit of Interrupt-Driven I/O

Administrivia

• Project 3 part 2 grades will be released around next Monday.

• Project 4 is due May 5th, but you should get started ASAP
  - Next week’s lab will be free section working on project 4
  - We will be holding normal discussions next week

Networks: Talking to the Outside World

• Originally sharing I/O devices between computers
  - E.g., printers

• Then communicating between computers
  - E.g., file transfer protocol

• Then communicating between people
  - E.g., e-mail

• Then communicating between networks of computers
  - E.g., file sharing, www, ...

The Internet (1962)

- JCR Licklider, as head of ARPA, writes on “intergalactic network”
- 1963 : ASCII becomes first universal computer standard
- 1969 : Defense Advanced Research Projects Agency (DARPA) deploys 4 “nodes” @ UCLA, SRI, Utah, & UCSB
- 1973 Robert Kahn & Vint Cerf invent TCP, now part of the Internet Protocol Suite

- Internet growth rates
  - Exponential since start!


- “System of interlinked hypertext documents on the Internet”
- History
  - 1945: Vannevar Bush describes hypertext system called “memex” in article
  - 1989: Tim Berners-Lee proposes, gets system up ’90
  - 1990 Dot-com entrepreneurs rushed in, 2001 bubble burst

- Wayback Machine
  - Snapshots of web over time

- Today : Access anywhere!
Shared vs. Switched Based Networks

- **Shared vs. Switched:**
  - **Shared:** 1 at a time (CSMA/CD)
  - **Switched:** pairs (“point-to-point” connections) communicate at same time
- Aggregate bandwidth (BW) in switched network is many times shared:
  - point-to-point faster since no arbitration, simpler interface

What makes networks work?

- links connecting switches and/or routers to each other and to computers or devices
- ability to name the components and to route packets of information - messages - from a source to a destination
- Layering, redundancy, protocols, and encapsulation as means of abstraction (61C big idea)

Software Protocol to Send and Receive

- SW Send steps
  1: Application copies data to OS buffer
  2: OS calculates checksum, starts timer
  3: OS sends data to network interface HW and says start
- SW Receive steps
  3: OS copies data from network interface HW to OS buffer
  2: OS calculates checksum, if OK, send ACK; if not, delete message (sender resends when timer expires)
  1: If OK, OS copies data to user address space & signals application to continue

Protocol for Networks of Networks?

- Abstraction to cope with complexity of communication
- Networks are like onions
  - Hierarchy of layers:
    - Application (chat client, game, etc.)
    - Transport (TCP, UDP)
    - Network (IP)
    - Physical Link (wired, wireless, etc.)

Protocol Family Concept

- Key to protocol families is that communication occurs logically at the same level of the protocol, called peer-to-peer...
  ...but is implemented via services at the next lower level
- Encapsulation: carry higher level information within lower level “envelope”
- Fragmentation: break packet into multiple smaller packets and reassemble
Protocol for Network of Networks

- **Transmission Control Protocol/Internet Protocol (TCP/IP)**
  (TCP :: a Transport Layer)
  - This protocol family is the **basis of the Internet**, a WAN protocol
  - IP makes best effort to deliver
    - Packets can be lost, corrupted
  - TCP guarantees delivery
  - TCP/IP so popular it is used even when communicating locally: even across homogeneous LAN

TCP/IP packet, Ethernet packet, protocols

- Application sends message
- TCP breaks into 64KiB segments, adds 20B header
- IP adds 20B header, sends to network
- If Ethernet, broken into 1500B packets with headers, trailers (24B)

And in conclusion...

- Exceptions are “Unexpected” events
- Interrupts are asynchronous
  - can be used for interacting with I/O devices
- Need to handle in presence of pipelining, etc.
- Networks are another form of I/O
- Protocol suites allow networking of heterogeneous components
  - Another form of principle of abstraction
- Interested in Networking?
  - EE122 (CS-based in Fall, EE –based in Spring)