CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Lecture 38: IO and Networking

Instructor:
Dan Garcia
http://inst.eecs.Berkeley.edu/~cs61c

Exceptions and Interrupts

§4.9 Exceptions

- "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 $8000\ 0180_{hex}$

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, …

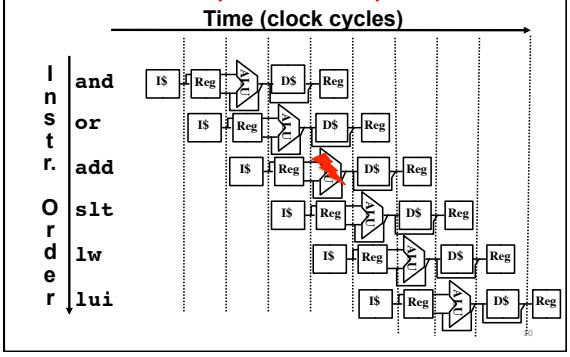
Exceptions in a Pipeline

- Another kind of control hazard
- Consider overflow on add in EX stage

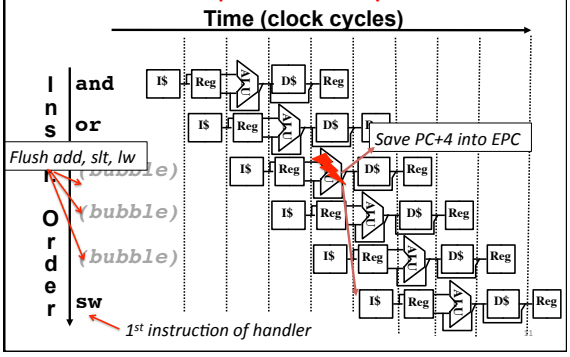
```
add $1, $2, $1
```

  - 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



Exception Example



Multiple Exceptions

- 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!

12

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

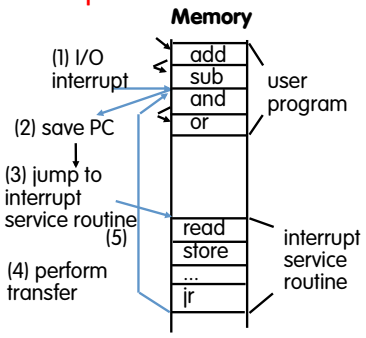
13

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

14

Interrupt-Driven Data Transfer



15

Benefit of Interrupt-Driven I/O

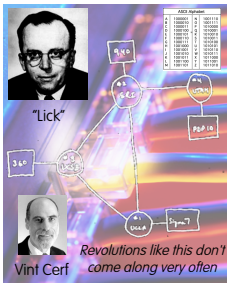
- Find the % of processor consumed if the hard disk is only active 5% of the time. Assuming 500 clock cycle overhead for each transfer, including interrupt:
  - Disk Interrupts/s = 5% * 16 [MB/s] / 16 [B/interrupt] = 50,000 [interrupts/s]
  - Disk Interrupts [clocks/s] = 50,000 [interrupts/s] * 500 [clocks/interrupt] = 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!

16

www.computerhistory.org/internet_history

The Internet (1962)

- Founders
  - 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!



www.greatachievements.org/?id=3736
en.wikipedia.org/wiki/Internet_Protocol_Suite

19

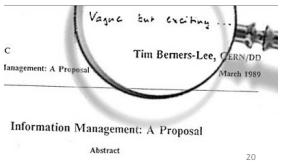
en.wikipedia.org/wiki/History_of_the_World_Wide_Web

The World Wide Web (1989)

- "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
  - ~2000 Dot-com entrepreneurs rushed in, 2001 bubble burst
- Wayback Machine
  - Snapshots of web over time
- Today : Access anywhere!


Tim Berners-Lee

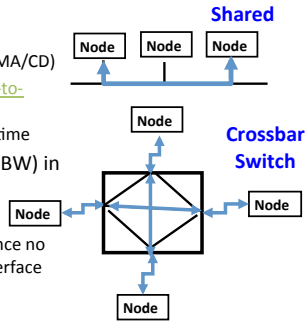

World's First web server in 1990



20

Shared vs. Switched Based Networks

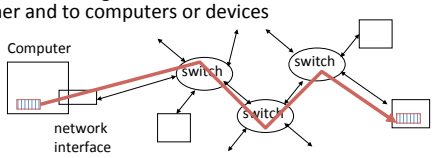
- Shared vs. Switched:
  - Shared: 1 at a time (CSMA/CD)
  - Switched: pairs ("point-to-point" connections) communicate at same time
- Aggregate bandwidth (BW) in switched network is many times shared:
  - point-to-point faster since no arbitration, simpler interface



21

What makes networks work?

- links connecting switches and/or routers to each other and to computers or devices

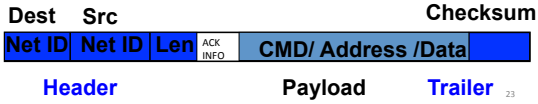


- **ability to name the components and to route packets of information - messages - from a source to a destination**
- **Layering, redundancy, protocols, and encapsulation as means of abstraction (61C big idea)**

22

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



23

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.)**

Networks are like onions.

They stink?

Yes. No!

Oh, they make you cry.

No!... Layers. Onions have layers. Networks have layers.

24

Protocol Family Concept

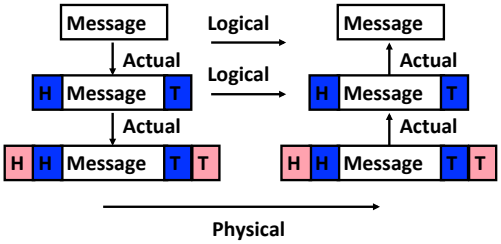
- Key to protocol families is that communication occurs logically at the same level of the protocol, called peer-to-peer…

  …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

25

Protocol Family Concept



26

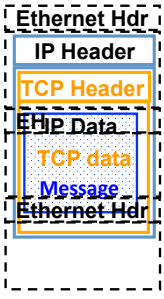
Protocol for Network of Networks

- 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

27

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)**



28

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)

29