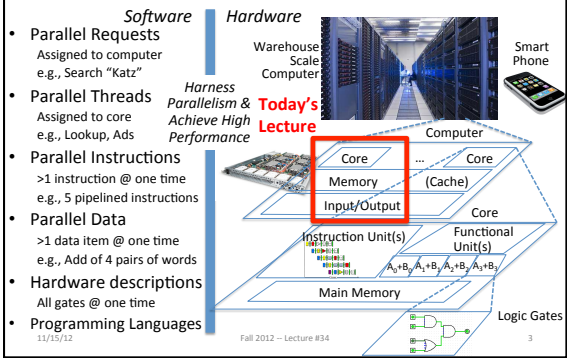
CS 61C:
Great Ideas in Computer Architecture
Exceptions/Traps/Interrupts

Instructors:
Krste Asanovic, Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/fa12

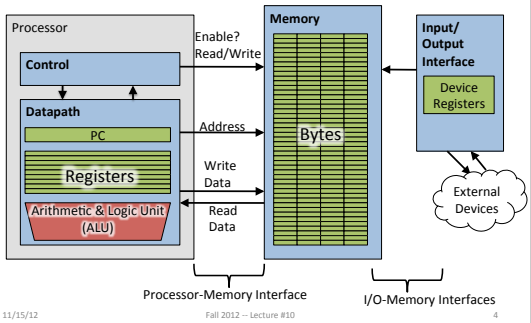
Review

- Great Idea: Redundancy to Get Dependability
  - Spatial (extra hardware) and Temporal (retry if error)
- Reliability: MTTF & Annualized Failure Rate (AFR)
- Availability: % uptime (MTTF-MTTR/MTTF)
- Memory
  - Hamming distance 2: Parity for Single Error Detect
  - Hamming distance 3: Single Error Correction Code + encode bit position of error
  - Hamming distance 4: SEC/Double Error Detection
- CRC for many bit detection, Reed Solomon per disk sector for many bit error detection/correction

You Are Here!



I/O Hardware



Interfacing to I/O Devices

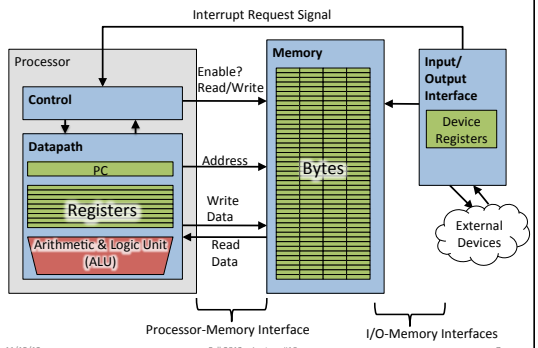
- Programmed I/O
  - Software on CPU reads and writes to I/O device registers to explicitly transfer I/O data
  - Usually "memory-mapped" where device I/O registers are given memory addresses allowing regular load and store instructions to change register contents
    - Also possible to have special I/O instructions (x86, IBM mainframes)
  - Cheap but slow, burdens CPU
- DMA (Direct Memory Access)
  - Extra hardware on device to autonomously transfer blocks of data into and out of memory
  - Expensive but fast, offloads CPU

When to get I/O data?

Most I/O devices are much slower than CPU

- Polling
  - Software occasionally checks I/O devices to see if data ready
  - How often to poll?
    - Too often, waste CPU time
    - Too little, slow I/O
  - What if software forgets?
- Interrupts
  - I/O device requests attention when something to do!

I/O Hardware

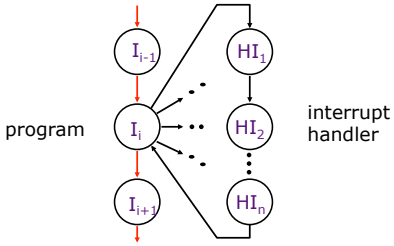


Invoking the interrupt handler

- An I/O device requests attention by asserting one of the prioritized interrupt request signals
  - Signals driven from external hardware
- When the processor decides to process the interrupt
  - Stops the current program at instruction I_i, completing all the instructions up to I_{i-1}
  - Saves the PC of instruction I_i in a special register
    - In MIPS, called EPC (Exception Program Counter)
  - Disables interrupts and transfers control to a designated interrupt handler, switching CPU to kernel mode
    - Most CPUs have a privileged execution mode to support operating systems, allowing software to do more than in regular user mode
- After handling interrupt
  - Interrupt handler restores user program state and jumps back to current program in regular user mode using a special instruction
    - In MIPS, instruction is called RFE (Return From Exception)

Interrupts:

altering the normal flow of control



An *external or internal event* that needs to be processed by another (system) program. The event is usually unexpected or rare from program's point of view.

Types of Interrupts

Interrupt: an *event* that requests the attention of the processor

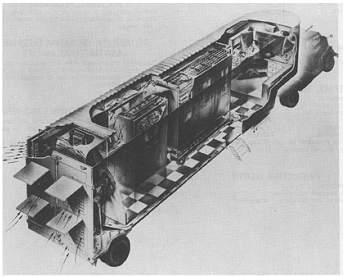
- Asynchronous: an *external event*
  - input/output device service-request
  - timer expiration
  - power disruptions, hardware failure
- Synchronous: an *internal event (a.k.a. traps or exceptions)*
  - undefined opcode, privileged instruction
  - arithmetic overflow, FPU exception
  - misaligned memory access
  - *virtual memory exceptions:* page faults, TLB misses, protection violations
  - system calls, e.g., calling functions in operating system kernel

History of Exception Handling

- First system with exceptions was Univac-I, 1951
  - Arithmetic overflow would either
    - 1. trigger the execution a two-instruction fix-up routine at address 0, or
    - 2. at the programmer's option, cause the computer to stop
  - Later Univac 1103, 1955, modified to add external interrupts
    - Used to gather real-time wind tunnel data
- First system with I/O interrupts was DYSEAC, 1954
  - Had two program counters, and I/O signal caused switch between two PCs
  - Also, first system with DMA (direct memory access by I/O device)

[Courtesy Mark Smotherman]

DYSEAC, first mobile computer!



- Carried in two tractor trailers, 12 tons + 8 tons
- Built for US Army Signal Corps

[Courtesy Mark Smotherman]

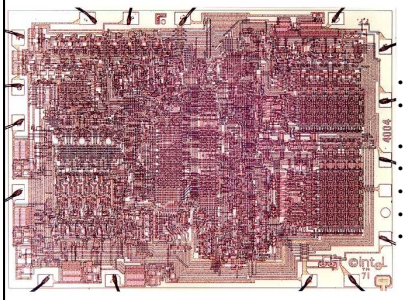
Interrupt Handler (software inside operating system)

- Saves EPC before enabling interrupts to allow nested interrupts
  - need an instruction to move EPC into GPRs
  - need a way to mask further interrupts at least until EPC can be saved
- Needs to read a *status register* that indicates the cause of the interrupt
- Uses a special indirect jump instruction RFE (*return-from-exception*) which
  - enables interrupts
  - restores the processor to the user mode
  - restores hardware status and control state

Administrivia

- Project 3 regrade requests – DON'T EMAIL INSTRUCTORS
  - Email proj3.1regrade@gmail.com or proj3.2regrade@gmail.com
  - See Piazza posting for more details

CS61C in the News: Intel 4004, 41 years old yesterday



- First microprocessor
- 4-bit accumulator architecture
- 8μm pMOS
- 2,300 transistors
- 3 x 4 mm²
- 750kHz clock
- 8-16 cycles/inst.

CS61C In the News: "Texas Instruments Cuts 1,700 Jobs and Winds Down Tablet Chips", NY Times 11/14/2012

"Texas Instruments is eliminating 1,700 jobs, as it winds down its mobile processor business to focus on chips for more profitable markets like cars and home appliances. Texas Instruments said in September it would halt costly investments in the increasingly competitive smartphone and tablet chip business, leading Wall Street to speculate that part of the company's processor unit, called OMAP, could be sold. The layoffs are equivalent to nearly 5 percent of the Austin, Texas-based company's global workforce.

TI has been under pressure in mobile processors, where it has lost ground to rival Qualcomm Inc. Leading smartphone makers Apple Inc and Samsung Electronics Co Ltd have been developing their own chips instead of buying them from suppliers like TI. Instead of competing in phones and tablets, TI wants to sell its OMAP processors in markets that require less investment, like industrial clients like carmakers. TI is expected to continue selling existing tablet and phone processors for products like Amazon.Com Inc's Kindle tablets for as long as demand remains, but stop developing new chips."

[Rumors of Amazon being interested in buying this OMAP unit from TI]

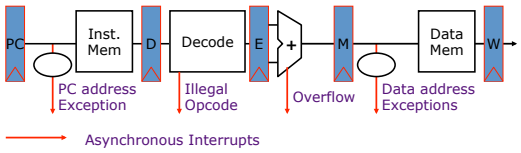
Precise Interrupts

- Interrupt handler's view of machine state is that every instruction prior to the interrupted one has completed, and no instruction after the interrupt has executed.
  - Instruction taking interrupt might have written some special state but can be restarted.
- Implies that handler can return from interrupt by restoring user registers and jumping to EPC
  - Software doesn't need to understand the pipeline of the machine!
- Providing precise interrupts is tricky in a pipelined superscalar out-of-order processor!
  - But handling imprecise interrupts in software is even worse.

Synchronous Trap

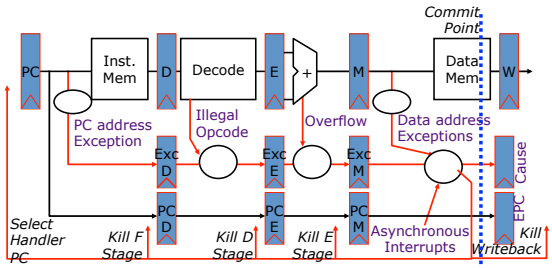
- A synchronous trap is caused by a particular instruction
  - E.g., system call, virtual memory page fault, unimplemented instruction, misaligned memory address
- In general, the instruction cannot be completed and needs to be restarted after the trap has been handled
  - requires undoing the effect of one or more partially executed instructions
- In the case of a system call trap, the instruction is considered to have been completed
  - a special jump instruction involving a change to privileged kernel mode

Exception Handling 5-Stage Pipeline



- How to handle multiple simultaneous exceptions in different pipeline stages?
- How and where to handle external asynchronous interrupts?

Exception Handling 5-Stage Pipeline



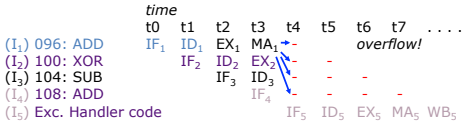
Exception Handling 5-Stage Pipeline

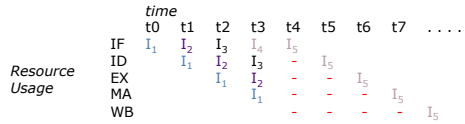
- Hold exception flags in pipeline until commit point (M stage)
- Exceptions in earlier pipe stages override later exceptions *for a given instruction*
- Inject external interrupts at commit point (override others)
- If exception at commit: update Cause and EPC registers, kill all stages, inject handler PC into fetch stage

Speculating on Exceptions to avoid Control Hazard

- Prediction mechanism
  - Exceptions are rare, so simply predicting no exceptions is very accurate!
- Check prediction mechanism
  - Exceptions detected at end of instruction execution pipeline, special hardware for various exception types
- Recovery mechanism
  - Only write architectural state at commit point, so can throw away partially executed instructions after exception
  - Launch exception handler after flushing pipeline
- Bypassing allows use of uncommitted instruction results by following instructions

Exception Pipeline Diagram





Summary

- Asynchronous interrupts versus synchronous traps
- Precise interrupts simplify software view of interrupted state
- Handling exceptions is one of the most difficult parts of processor design, even though they occur infrequently
  - Separate "commit" from execute in design

Acknowledgements

- These slides contain material developed and copyright by:
  - Arvind (MIT)
  - Krste Asanovic (MIT/UCB)
  - James Hoe (CMU)

- MIT material derived from course 6.823
- UCB material derived from course CS252

25