About Me

• Christopher Celio
• celio @ eecs
• Office Hours: Monday 1-2pm, 611 Soda
Agenda

• Mystery Processor (A history lesson...)
• 5-stage Pipeline Designs
• Terminology
• CISC/RISC (x86 Processors)
• Iron-Law Questions
Mystery Processor: The Team

http://www.computerhistory.org/revolution/digital-logic/12/286/1593

Courtesy of David Patterson, Computerhistory.org, Circa 1980
RISC I

• 1-1.5 Mhz
• Single-Issue
• 2-stage
• 32-bit micro-processor
• 44,420 transistors
  – Modern Intel is 2,000,000,000 transistors
• 78 registers
• 6% of die for inst decode/ctrl
  – usually 50% at this time!
• 5,000 nm (5um)
  – Modern is 22nm/32nm
• Programs 30% larger (CISC is typically better in terms of code density)

wikipedia gets some of these facts wrong :(

http://www.eecs.berkeley.edu/Pubs/TechRpts/1982/CSD-82-106.pdf

Friday, February 3, 2012
• **TAKE AWAY POINTS**
  
  - used *older* technology
  - fit entire processor on **micro-chip**
  - slightly slower clock frequency
  - true 32-bit
  - large register file kept memory accesses on-chip
  - **2x** THE PERFORMANCE OF THE VAX
  - **4x** FASTER THAN 16-bit z8000 (desktop processor)
  - Patterson claimed C compilers weren’t taking advantage of VAX (CISC too complicated to effectively target!)


Friday, February 3, 2012
RISC II

- 3 Mhz
- Single-Issue
- 3-stage
  - Inst fetch and decode
  - “register read, operate, and temporary latching of result”
  - write back result to register file
  - (stores/loads insts take another cycle)
- 32-bit micro-processor
- 40,760 transistors
- 138 registers
- 3,000 nm (3um)
- 85% to 256% performance of VAX
- 140% to 420% performance of Motorola 68000 (considered best commercial chip)


Friday, February 3, 2012
Figure 4. The Data-Path of the RISC I Chip.

http://www.eecs.berkeley.edu/Pubs/TechRpts/1982/CSD-82-106.pdf
The VAX 11

http://www.old-computers.com/history/detail.asp?n=20&t=3

Friday, February 3, 2012
The VAX 11

http://www.old-computers.com/history/detail.asp?n=20&t=3

Friday, February 3, 2012
RISC-I Performance


Friday, February 3, 2012
Epilogue of RISC-I/II

• Other Berkeley processors have been built...
  – more on those later... (hint: Krste built one too).
  – now we’re on RISC-V!
• Sun SPARC
  – Patterson’s RISC processors are the basis for SPARC
  – created arms race of RISC-like processors
    • DEC Alpha
    • PA-RISC
    • SGI (bought Stanford’s MIPS)
Lab 1

• Any questions?
stall = (rs1_D=ws_E). (opcode_E=LW). (ws_E\neq 0). re1_D + (rs2_D=ws_E). (opcode_E=LW). (ws_E\neq 0). re2_D
Why suboptimal?

\[ \text{stall} = (rs_{1D} = ws_E) \cdot (\text{opcode}_E = \text{LW}_E) \cdot (ws_E \neq 0) \cdot \text{re1}_D + (rs_{2D} = ws_E) \cdot (\text{opcode}_E = \text{LW}_E) \cdot (ws_E \neq 0) \cdot \text{re2}_D \]
Lecture

Original Lab 1

get operand mux off critical path!

(technically this is suboptimal too, but CAD tools can probably figure this one out and re-arrange the muxes for us!)
The Improved Lab 1 5-Stage
The Improved Lab 15-Stage

Critical Path...
The Improved Lab 1 5-Stage

Off the critical path! Yah!
Lab 1

• NO NEED to download updated 5-stage processor
  – (I just want to provide code that matches the diagram)
• A technique for shortening the critical path in a circuit
What is the Instruction Set Architecture?

- A **contract** between the hardware and software
  - software that follows ISA contract will run on (any version of) the hardware
  - hardware can change *(dramatically)* and still maintain compatibility with software

---

**Terminology:**

**ISA**

- Instruction Set Architecture

---

**WALL**

- Hardware Team
- Software Team
- mobile, server, desktop, in-order, out-of-order, pipelined....
- compilers, OS, apps, angry birds
• What is the Micro-architecture?
  – A specific implementation of an ISA
Terminology: ABI

• What is the Application Binary Interface?
  – A **contract** between an application and the OS (or other applications)
    • covers calling conventions
    • register usage (in addition to what the ISA demands)
    • how to make system calls
    • (not important to CS 152, but helpful to know when looking at disassembly)

  – RISC-V ABI (register names and their usages)
    • x0 = always zero
    • x1 = ra (return address, aka link register)
    • x2-x3 = v0-v1 (values for function returns)
    • x4-x11 = a0-a7 (function arguments... extras go on the stack)
    • x12-x19 = t0-t7 (temporaries, must store to stack before calling functions)
    • x20-x29 = s0-s9 (stored temporaries, must save previous value to stack before using)
    • x30 = sp (stack pointer)
    • x31 = tp (thread context pointer, points to important thread state)

  – other ABIs have reserved registers for the OS kernel, the assembler, frame pointer, global pointer, etc.
• A design philosophy based on the insight that simplified instructions enable better machine performance
• Sometimes referred to as load-store architectures
CISC

• A descriptive term of ISAs in which a single macro-instruction can enact many low-level operations (and side-effects)
• instructions are often multi-step operations
• (term coined retroactively)
x86: The 8086

- CISC
- designed in 10 weeks
- initially implemented as a bus-based design

• How do you make x86 run at 8+ Ghz?? (and have excellent performance, i.e., 100s of instructions inflight simultaneously)
  – adopt some of the RISC philosophies!
  – kept the (CISC) ISA untouched... but move to a RISC micro-architecture
  – separate processor into a CISC front-end and a RISC back-end
  – decode CISC macro-instructions into RISC micro-ops
  – now you can heavily pipeline the RISC micro-ops!
x86: Translating from CISC to RISC

- MANY cycles spent fetching and decoding CISC macro-instructions
- many CISC instructions are easy to decode to RISC uops (1:1 correspondence)
- Some CISC instructions are very complex, require **stopping** everything and firing the micro-code engine to spit out uops
- can patch hardware mistakes by changing the programmable decoders

Intel Core 2
Questions?