Computer Systems and Networks

ECPE 170 – Jeff Shafer – University of the Pacific

Processor Architectures

Schedule

- **This Week** – Processor Architectures
- **Quiz 6** – Wednesday, April 11th
  - Input / Output (HW #16)
  - Operating Systems (HW #17)
  - Compilers & Assemblers (HW #17)
  - **Review the lecture notes before the quiz (not just the homework!)**
  - **Bring a Calculator**
- **Friday, April 13th – Pacific Day – No class**

Homework #17

- Review HW #17
  - Real-time OS (RTOS)
  - Assembly vs High-Level Language
  - Mobile OS

Processor Architectures

Processor Architectures

- **Starting Chapter 9**
- More details on RISC versus CISC!
- Leaving the safe, familiar world of the von Neumann processor
- **What is the von Neumann model?**
  - Stored program computer
  - Three systems: CPU, memory, I/O
  - Sequential instruction processing
  - Single data path between CPU and memory – von Neumann bottleneck
- More than one processor!
  - Multiprocessor architectures – different types

RISC vs CISC Machines

- **RISC** systems access memory only with **explicit load and store instructions**
  - Instruction length is fixed
  - Fetch-decode-execute time is constant
- **CISC** systems access memory with **many different types of instructions**
  - Instruction length is variable
  - Fetch-decode-execute time is unpredictable

RISC vs CISC Machines

- Basic computer performance equation:

$$\text{CPU Time} = \frac{\text{seconds}}{\text{program}} = \frac{\text{instructions}}{\text{program}} \times \frac{\text{avg. cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

- RISC systems shorten execution time by reducing the clock cycles per instruction
- CISC systems improve performance by reducing the number of instructions per program

RISC vs CISC Machines

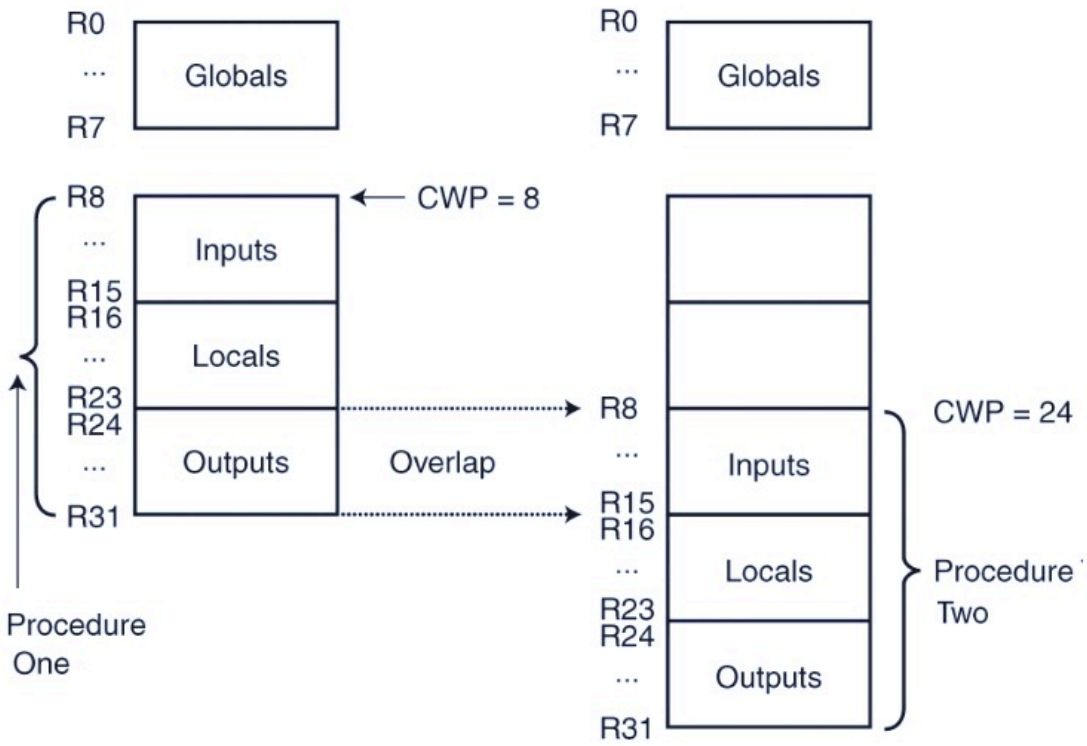
- RISC processors have a simpler instruction set
  - Build a hardwired control unit (faster!)
  - Easier to implement pipelining and speculative execution
- CISC processors have a complex/variable instruction set
  - Build a microcode-based control unit to interpret instructions
  - Microcode processing takes time

RISC vs CISC Machines

- Because of their load-store ISAs, RISC architectures require a large number of CPU registers
  - Register allow fast access to data during sequential program execution – no need to go to memory!
- Registers can also be used to reduce the overhead of calling subroutines
  - Contrast this to MARIE, where you had to store all your arguments in memory before jumping to a subroutine
- Instead of pulling parameters off of a stack, the subroutine is directed to use a subset of registers

Overlapping Registers – "Windows"

- Divide all the registers into "windows"
  - Your subroutine only sees one window
- The current window pointer (CWP) points to the active register window
  - Shift when calling a subroutine
  - Outputs become inputs
- Global registers – shared by all



RISC vs CISC Machines

- It is becoming increasingly difficult to distinguish RISC architectures from CISC architectures.
  - Some RISC systems provide more extravagant instruction sets than some CISC systems
  - Some systems combine both approaches
- Typical differences between the architectures

RISC vs CISC Machines

RISC

- Simple instructions, few in number
- Fixed length instructions
- Complexity in compiler
- Only LOAD/STORE instructions access memory
- Few addressing modes

CISC

- Many complex instructions
- Variable length instructions
- Complexity in microcode
- Many instructions can access memory
- Many addressing modes

RISC vs CISC Machines

RISC

- Multiple register sets
- Three operands per instruction
- Parameter passing through register windows
- Single-cycle instructions
- Highly pipelined

CISC

- Single register set
- One or two register operands per instruction
- Parameter passing through memory
- Multiple cycle instructions
- Less pipelined

Intel

- **So, are Intel x86-32 or x86-64 chips RISC or CISC?**
- **Both!**
- Instruction set is “CISC-like”
  - Many complex instructions
  - Variable length instructions
  - Many instructions can access memory (not just load/store)
  - Multiple cycle instructions
  - etc…

Intel

- But, what happens *internally* is completely different
  - Dedicated hardware unit that decodes the "CISC-like" x86 instructions and replaces them with a sequence of "RISC-like" **micro-ops**
- Intel has been RISC-like internally since the Pentium Pro era (~1996)

Intel versus ARM

Intel

- "CISC-like" ISA
- Dominant market:
  - Desktop PCs
  - Laptop PCs
  - Server PCs
  - Performance critical?
- **Why does Intel dominate these markets?**

ARM

- "RISC-like" ISA
- Dominant market:
  - Mobile devices (cell phones, music players, etc…)
    - Power critical?
  - Embedded devices
- **Why does ARM dominate these markets?**

Intel versus ARM

Intel

- Manufacturing: Every chip made by Intel
- Performance
- Huge amount of "legacy software" for desktops/ servers written for the x86 ISA

ARM

- Manufacturing: None
  - ARM licenses its design to other companies to integrate/build
  - Apple, NVIDIA, IBM, Texas Instruments, Nintendo, Samsung, Freescale, Qualcomm and VIA Technologies, and … Intel!
- Performance **per watt**
- Huge amount of "legacy software" for cell phones written for ARM ISA