Topics for today:

More on interrupts

Alternative architecture:
The RISC philosophy

Recall: Interrupts

An *interrupt* occurs when normal execution is halted temporarily in order for the CPU to attend to some more pressing matter.

Question

Can you interrupt an interrupt service routine?

Yes, sometimes: *maskable* vs. *nonmaskable* interrupts.

Note

Interrupts generated by software are also known as *exceptions* or *traps*.

The RISC design philosophy

What is RISC?

RISC = *Ridiculously Simple Computer*

Or

RISC = *Reduced Instruction Set Computers*

As instruction sets became increasingly complex, a movement developed to simplify instruction sets and the machines that run them. It became known as the *Reduced Instruction Set Computer* (RISC) approach.

Its goal could be summarized as:

Increase cycle speed at all costs.

Cray-1 memory board

- It requires its own electrical sub-station to provide with power. The unit electrical bill was $30,000-$35,000 per month. $20,000 for the air-condition room and about $10,000-$15,000 for the machine. Cost per board was about $50,000 to replace with the $1,000,000 per year maintenance contract.

Benefits of RISC

- RISC processor will cost less to design -- since a significant cost of the chip can be the actual R&D costs to create it, this can be substantial on its own
- easier to design (and fewer bugs) means that the processor will have a faster time to market
- faster time to market means the processor can use newer processes

Elements of the RISC philosophy

1) Small, simple instruction sets
2) Many general-purpose registers
3) Instructions execute in one clock cycle whenever possible
4) No microprogram
5) Intensive pipelining

1. Small, simple instruction sets

- Studies show that most instructions *executed* are of very basic types
- (Not the same as the proportion of instructions *written*)
- Examples: assign, loop, if, call

2. Large number of registers

- Studies show that compiled code has
  - lots of copy statements
  - lots of operand accesses
- Two possible approaches:
  - Software: Compiler optimizes register use
  - Hardware: Register windows

Register windows

- Small sets of registers
- Subroutine calls switch windows instead of pushing parameters/local variables/ register contents onto stack
- Examples: The Pyramid (first commercial RISC machine) had 16 windows with 32 registers each

Registers vs. cache

Large register sets and cache memory are two approaches that use proximity as a way to speed up access time.

- How are they different?
- What are the advantages and disadvantages of each?

3. One instruction per cycle

- 1 RISC instruction =
  1 CISC microinstruction
- Instructions are hard-wired – decoder circuitry
- "Fetch/decode/execute" becomes "fetch/execute"

3. One instruction per cycle

- Instructions are register-to-register whenever possible
- Simpler addressing modes
- In fact: no expanding opcodes, either
- Therefore: what use is a microprogram?

4. No microprogram

- If the other principles have been implemented, there is no need/use for a microprogram

5. Pipelining

- Note: load and store necessarily take longer, because of memory access
- Start them in one cycle, let them finish later; meanwhile, go ahead and execute next instruction(s)
- What if next instructions depend on the results of load/store?
- Arrange it so this doesn't happen!

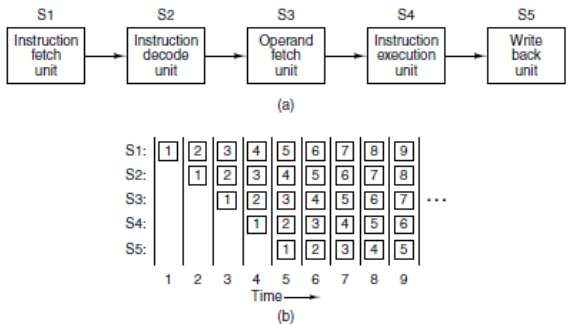

Figure 2-4. (a) A five-stage pipeline. (b) The state of each stage as a function of time. Nine clock cycles are illustrated.

Note

At one time, this discussion was cast as the RISC *vs.* CISC debate.

In practice, RISC principles have now largely been adopted into processor design.

Why RISC?

Simple instructions are preferred

- Complex instructions are mostly ignored by compilers
  - Due to semantic gap

Simple data structures

- Complex data structures are used relatively infrequently
- Better to support a few simple data types efficiently
  - Synthesize complex ones

Simple addressing modes

- Complex addressing modes lead to variable length instructions
  - Lead to inefficient instruction decoding and scheduling