RISC & CISC

Kevin Walsh
CS 3410, Spring 2010
Computer Science
Cornell University

ISA defines the permissible instructions

- MIPS: load/store, arithmetic, control flow, …
- ARM: similar to MIPS, but more shift, memory, & conditional ops
- VAX: arithmetic on memory or registers, strings, polynomial evaluation, stacks/queues, …
- Cray: vector operations, …
- x86: a little of everything

Toy example: subleq a, b, target

```
Mem[b] = Mem[b] – Mem[a]
then if (Mem[b] <= 0) goto target
else continue with next instruction
```

```
clear a == subleq a, a, pc+4
jmp c == subleq Z, Z, c
add a, b == subleq a, Z, pc+4;
            subleq Z, b, pc+4;
            subleq Z, Z, pc+4
```

Not-a-toy example: PDP-8

One register: AC

Eight basic instructions:

```
AND a       # AC = AC & MEM[a]
TAD a       # AC = AC + MEM[a]
ISZ a       # if (!++MEM[a]) skip next
DCA a       # MEM[a] = AC; AC = 0
JMS a       # jump to subroutine (e.g. jump and link)
JMP a       # jump to MEM[a]
IOT x       # input/output transfer
OPR x       # misc operations on AC
```

Stack machine

- data *stack* in memory, *stack pointer* register
- Operands popped/pushed as needed
  add

[Java Bytecode, PostScript, odd CPUs, some x86]

Tradeoffs:

Accumulator machine

- Results usually put in dedicated accumulator register

  add b

  store b

[Some x86]

Tradeoffs:

Load/store (register-register) architecture

- computation only between registers

[MIPS, some x86]

Tradeoffs:

Axes:

- Arguments: stack-based, accumulator, 2-arg, 3-arg
- Operand types: load-store, memory, mixed, stacks, …
- Complexity: CISC, RISC

MIPS = Reduced Instruction Set Computer (RISC)

- ≈ 200 instructions, 32 bits each, 3 formats
- all operands in registers
  - almost all are 32 bits each
- ≈ 1 addressing mode: Mem[reg + imm]

x86 = Complex Instruction Set Computer (CISC)

- > 1000 instructions, 1 to 15 bytes each
- operands in dedicated registers, general purpose registers, memory, on stack, …
  - can be 1, 2, 4, 8 bytes, signed or unsigned
- 10s of addressing modes
  - e.g. Mem[segment + reg + reg*scale + offset]

RISC Philosophy

Regularity & simplicity

Leaner means faster

Optimize the common case

CISC Rebuttal

Compilers can be smart

Transistors are plentiful

Legacy is important

Code size counts

Micro-code!