RISC vs. CISC

Different Kinds of ISAs

We have looked at LC3 ISA, which is a classic example of RISC type ISA

Reduced Instruction Set Architecture (RISC) emerged around early 80s

- Designers re-evaluating the current ISAs of the era
- Found that ISAs had extensive instructions that were complex
  - Complex Instruction Set Architecture (CISC)
- Need only 20% of the instructions that were used most of the time

Complex Instruction Set Computer (CISC)

Memory in those days was expensive

- bigger program->more storage->more money

Hence needed to *reduce* the number of instructions per program

Number of instructions are reduced by having *multiple operations* within a single instruction

Multiple operations lead to many different kinds of instructions that access memory

- In turn making instruction length variable and fetch-decode-execute time unpredictable – making it more complex
- Thus hardware handles the complexity

Example: x86 ISA

Reduced Instruction Set Computer (RISC)

Original idea to reduce the ISA

- Provide ***minimal set of instructions*** that could carry out all essential operations

Instruction complexity is reduced by

1. Having *few simple* instructions that are the *same length*

2. Allowed memory access only with *explicit* load and store instructions

Hence each instruction performs less work but instruction execution time among different instructions is consistent

The complexity that is removed from ISA is moved into the domain of the assembly programmer/compiler

Examples: LC3, MIPS, PowerPC (IBM), SPARC (Sun)

RISC vs. CISC

The difference between CISC and RISC becomes evident through the 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* (i.e. simple instructions take less time to interpret)

CISC systems shorten execution time by reducing the *number of instructions per program*

Example for RISC vs. CISC

Consider the the program fragments:

CISC

```
mov ax, 10
mov bx, 5
mul bx, ax
```

RISC

```
        mov ax, 0
        mov bx, 10
        mov cx, 5
Begin   add ax, bx
        loop Begin
```

The total clock cycles for the CISC version might be:

(2 movs × 1 cycle) + (1 mul × 30 cycles) = 32 cycles

While the clock cycles for the RISC version is:

(3 movs × 1 cycle) + (5 adds × 1 cycle) + (5 loops × 1 cycle) = 13 cycles

Micro-architecture Implementations

The simple instruction set of RISC machines takes less time to interpret plus less hardware

- Enables control unit to be hardwired for maximum speed
- Also allows room for performance enhancement such as pipelining
- Fewer instructions would mean fewer transistors, in turn less manufacturing cost

The more complex and variable instruction set of CISC machines require more translation takes time as well more hardware

- Usually implemented as microprogrammed control to tackle the variable length instructions

Other RISC features

Because of their load-store ISAs, RISC architectures require a large number of CPU registers

These register provide fast access to data during sequential program execution

They can also be employed to reduce the overhead typically caused by passing parameters on the stack

Instead of pulling parameters off of a stack, the subroutine is directed to use a subset of registers

RISC vs. CISC Summary

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 Roadblocks in the 80s

RISC chips took over a decade to gain a foothold in the commercial world

This was largely due to a lack of software support

- Many companies were unwilling to take a chance with the emerging RISC technology
- Without commercial interest, processor developers were unable to manufacture RISC chips in large enough volumes to make their price competitive

Another major setback was the presence of Intel

- Had the resources to plow through development and produce powerful processors