Chapter 2
Instructions:
Language of the Computer

Chapter 2_4 ppt

Target Addressing example

Should be 2

Loop: sll $t1, $s3, 2	80000	0	0	19	9	4	0
add $t1, $t1, $s6	80004	0	9	22	9	0	32

Loop: sll $t1, $s3, 2   80000

add $t1, $t1, $s6   80004

lw $t0, 0($t1)   80008

bne $t0, $s5, Exit   80012

addi $s3, $s3, 1   80016

j   Loop   80020

Exit: …   80024

0	0	19	9	4	0
0	9	22	9	0	32
35	9	8	0		
5	8	21	2		
8	19	19	1		
2	20000				

Branching Far Away

- If branch target is too far to encode with 16-bit offset, assembler rewrites the code

- Example

```
beq $s0,$s1, L1
        ↓
bne $s0,$s1, L2
j L1
L2:    …
```

RISC Vs. CISC

- Complex instruction set makes implementation difficult
  - Hardware translates instructions to simpler microoperations
    - Simple instructions: 1–1
    - Complex instructions: 1–many
  - Microengine similar to RISC
  - Market share makes this economically viable
- Comparable performance to RISC
  - Compilers avoid complex instructions

Fallacies

- Powerful instruction $\Rightarrow$ higher performance
  - Fewer instructions required
  - But complex instructions are hard to implement
    - May slow down all instructions, including simple ones
  - Compilers are good at making fast code from simple instructions

- Use assembly code for high performance
  - But modern compilers are better at dealing with modern processors
  - More lines of code $\Rightarrow$ more errors and less productivity

Fallacies
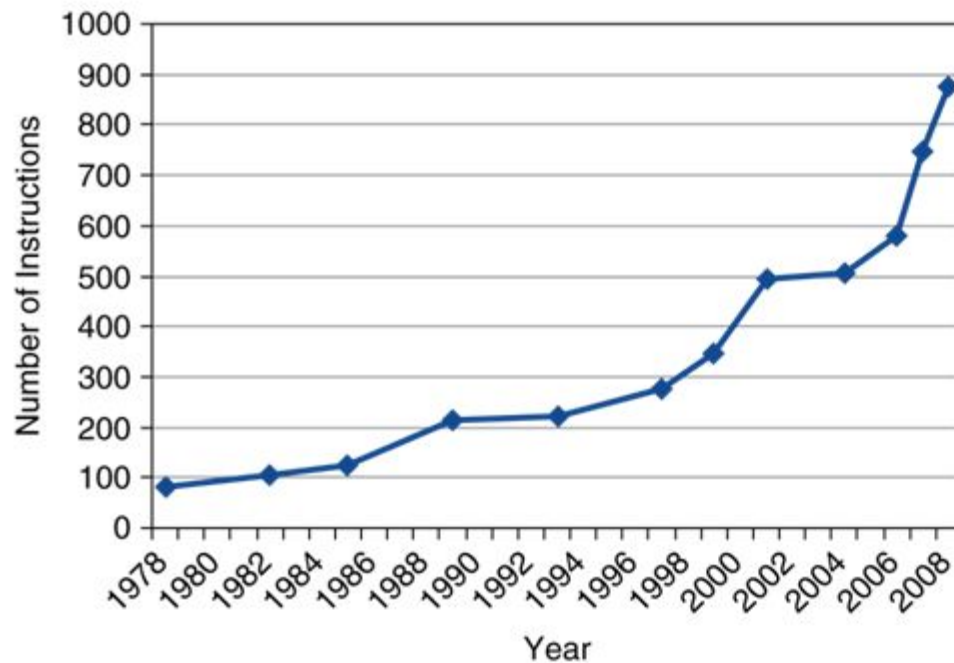
Remarks

- Design principles
  1. Simplicity favors regularity
  2. Smaller is faster
  3. Make the common case fast
  4. Good design demands good compromises
- Layers of software/hardware
  – Compiler, assembler, hardware