

# CSCI 210: Computer Architecture

## Lecture 8: Computer Representation of MIPS Instructions

Stephen Checkoway

Oberlin College

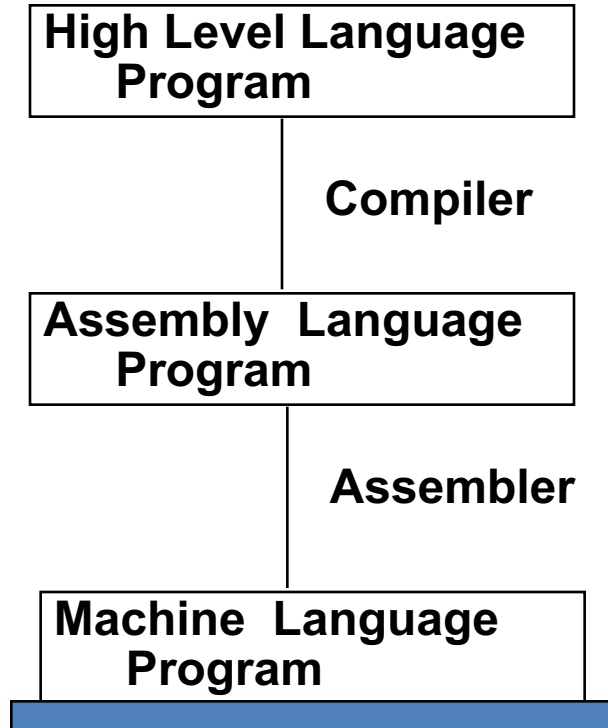
Mar. 7, 2022

Slides from Cynthia Taylor

# Announcements

- Problem Set 2 due Friday
- Lab 1 due Sunday
- No Office Hours on Tuesday

# How to Speak Computer



```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

```
lw $15, 0($2)  
lw $16, 4($2)  
sw $16, 0($2)  
sw $15, 4($2)
```

```
10001100011000100000000000000000  
10001100111100100000000000000100  
10101100111100100000000000000000  
10101100011000100000000000000100
```

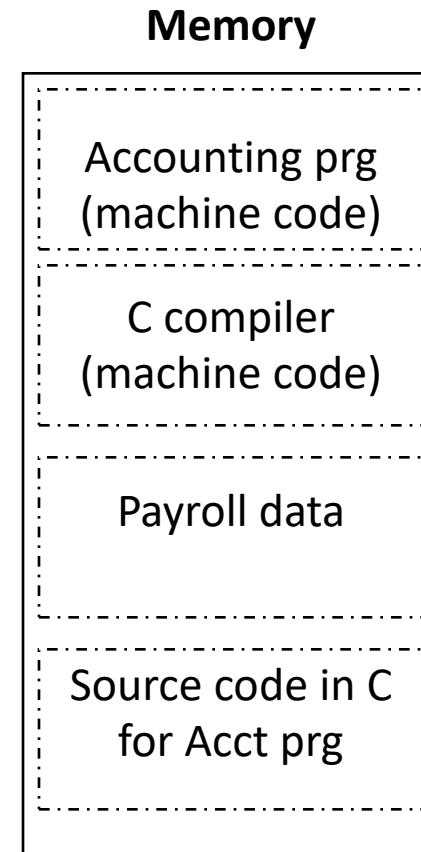
**Machine Interpretation**

# Two Key Principles of Machine Design

1. Instructions are represented as numbers and, as such, are indistinguishable from data
2. Programs are stored in alterable memory (that can be read or written to) just like data

## Stored-program concept

- Programs can be shipped as files of binary numbers – binary compatibility
- Computers can inherit ready-made software provided they are compatible with an existing ISA and OS – leads industry to align around a small number of ISAs



What happens if someone writes new machine code in the memory where your program is stored, overwriting your program?

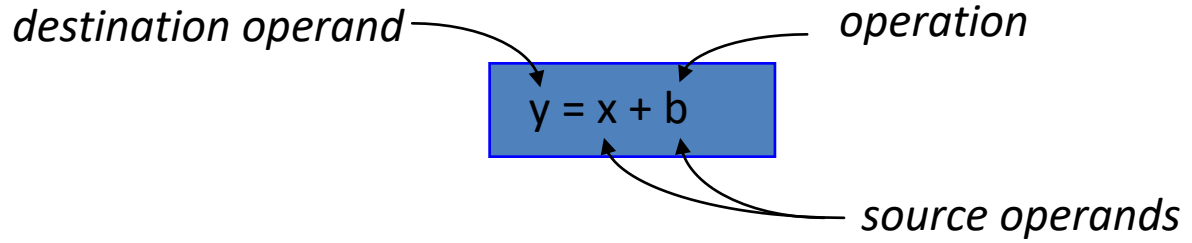
- A. The program will crash.
- B. The old instructions will run.
- C. The new instructions will run.
- D. None of the above

# Recall: Instruction Set Architecture

- Definition of how to access the hardware from software
- Supported instructions, registers, etc . . .

# Key ISA decisions

- operations
  - how many?
  - which ones
- operands
  - how many?
  - location
  - types
- instruction format
  - size
  - how many formats?



add r1, r2, r5

*how does the computer know what  
0001 0100 1101 1111  
means?*

# RISC versus CISC (Historically)

- Complex Instruction Set Computing
  - Larger instruction set
  - More complicated instructions built into hardware
  - Variable number of clock cycles per instruction
- Reduced Instruction Set Computing
  - Small, highly optimized set of instructions
  - Memory accesses are specific instructions
  - One instruction per clock cycle (only the very first RISCs!)



$$A = A * B$$

RISC

```
lw    $t0, 0(A)
lw    $t1, 0(B)
mul   $s1, $t0, $t1
sw    $s1, 0(A)
```

CISC

```
mul   B, A
```

# Which of these is faster?

RISC

```
lw    $t0, 0(A)
lw    $t1, 0(B)
mul   $s1, $t0, $t1
sw    $s1, 0(A)
```

CISC

```
mul   B, A
```

# RISC vs CISC

## RISC

- More work for compiler/assembly programmer
- More RAM used to store instructions
- Less complex hardware

## CISC

- Less work for compiler/assembly programmer
- Fewer instructions to store
- More complex hardware

# So . . . Which System “Won”?

- Most processors are RISC
- BUT the x86 (Intel) is CISC
- x86 breaks down CISC assembly into multiple, RISC-like, machine language instructions
- Distinction between RISC and CISC is less clear
  - Some RISC instruction sets have more instructions than some CISC sets

The computer figures out what format an instruction is from

- A. Codes embedded in the instruction itself.
- B. A special register that is loaded with the instruction.
- C. It tries each format and sees which one forms a valid instruction.
- D. None of the above

# Instruction Formats

## What does each bit mean?

- Having many different instruction formats...
  - complicates decoding
  - uses more instruction bits (to specify the format)

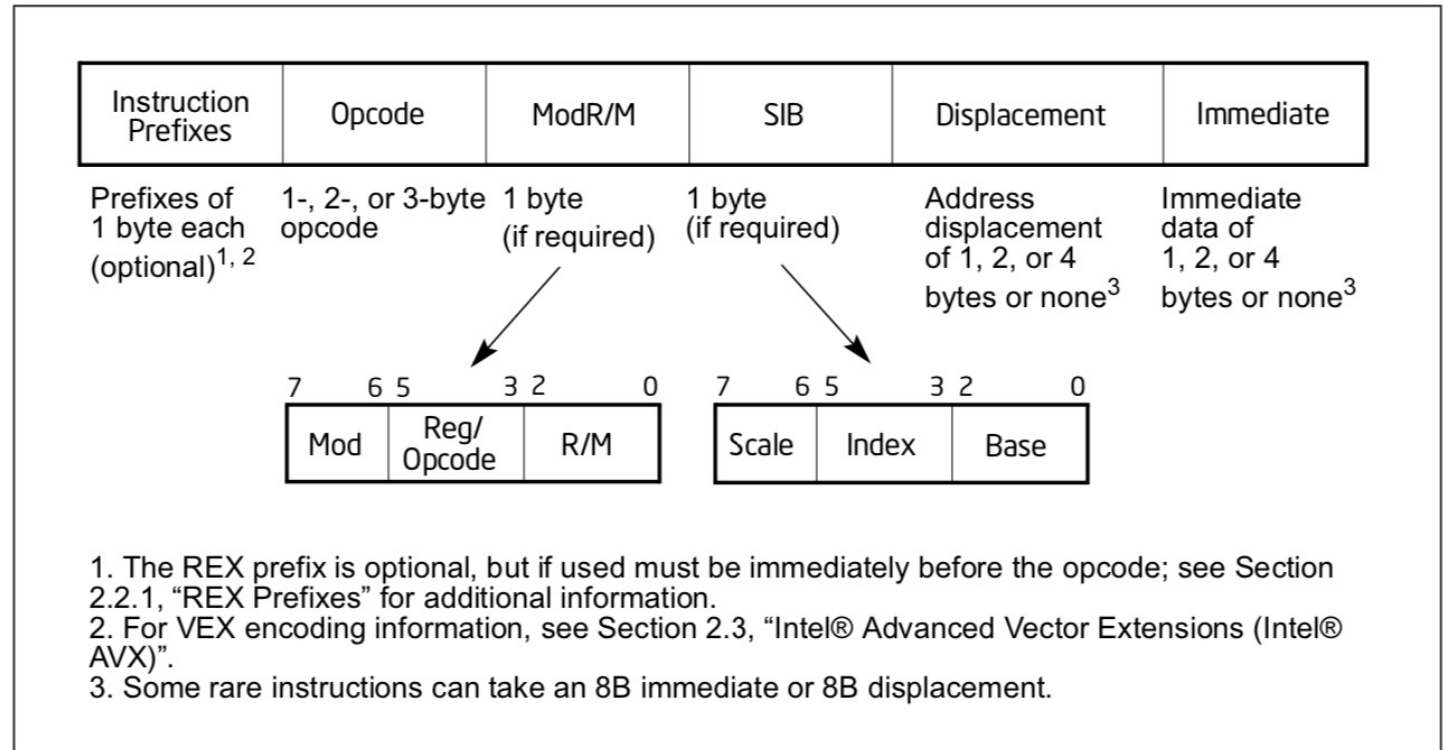


Figure 2-1. Intel 64 and IA-32 Architectures Instruction Format

Your architecture supports 16 instructions and 16 registers (r0–r15). You have fixed width instructions which are 16 bits. How many register operands can you specify (explicitly) in an add instruction?

- A.  $\leq 1$  operand
- B.  $\leq 2$  operands
- C.  $\leq 3$  operands
- D.  $\leq 4$  operands
- E. None of the above

Hint: Remember you need to specify which instruction it is, and all the registers

# Reading

- Next lecture: More on representing instructions
  - Section 2.6
- Problem Set 2 due Friday
- Lab 1 due Sunday