CSE 210: Computer Organization
Lecture 2: Assembly Language

Stephen Checkoway

Oberlin College

Oct 6th, 2021

Slides by Cynthia Taylor

Announcements

- Problem set 0 due Friday, Oct. 8 at 23:59
  - Access it via gradescope (link on course website)

How to Speak Computer?

10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100

2

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

1

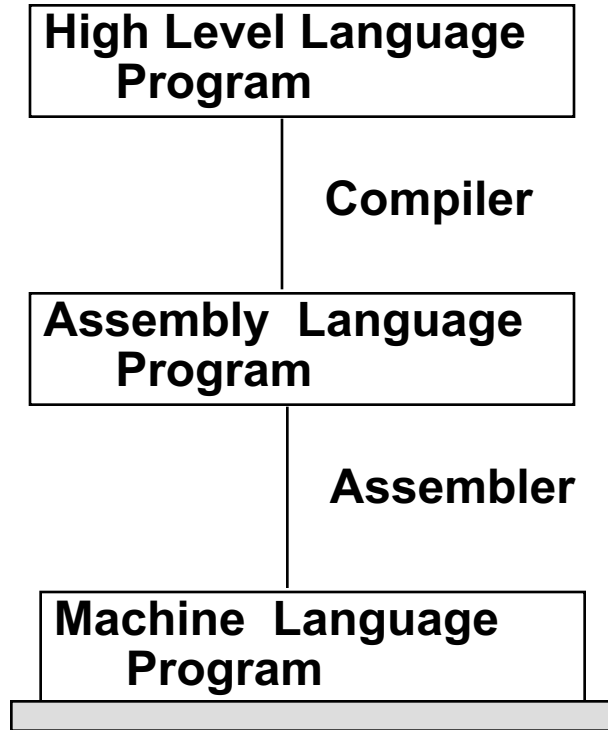temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

3

Selection	High Level Language	Assembly	Machine Language
A	3	2	1
B	3	1	2
C	2	1	2
D	1	2	2
E	None of the above		

What Your CPU Understands

- Electricity

- Ones and zeros

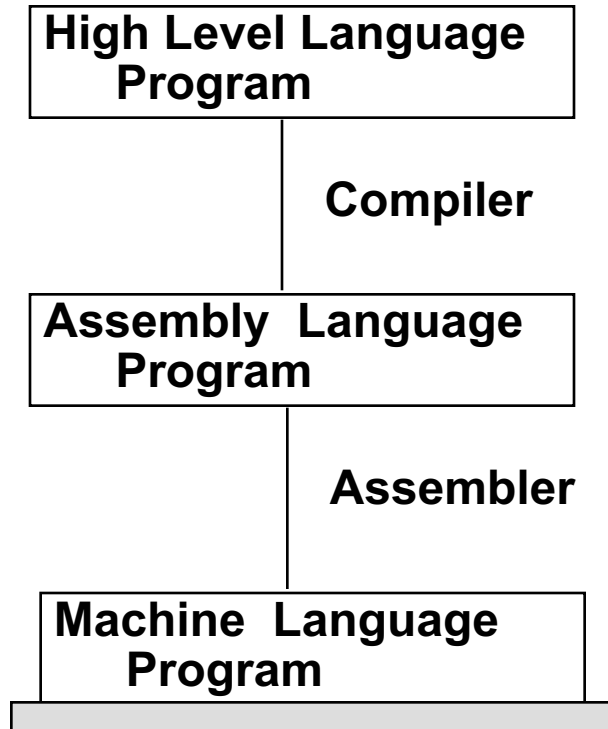- Problem: People don't like writing programs in ones and zeros

How to Speak Computer

High Level Language Program

Compiler

Assembly Language Program

Assembler

Machine Language Program

Machine Interpretation

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

How to Speak Computer

High Level Language Program

Compiler

Assembly Language Program

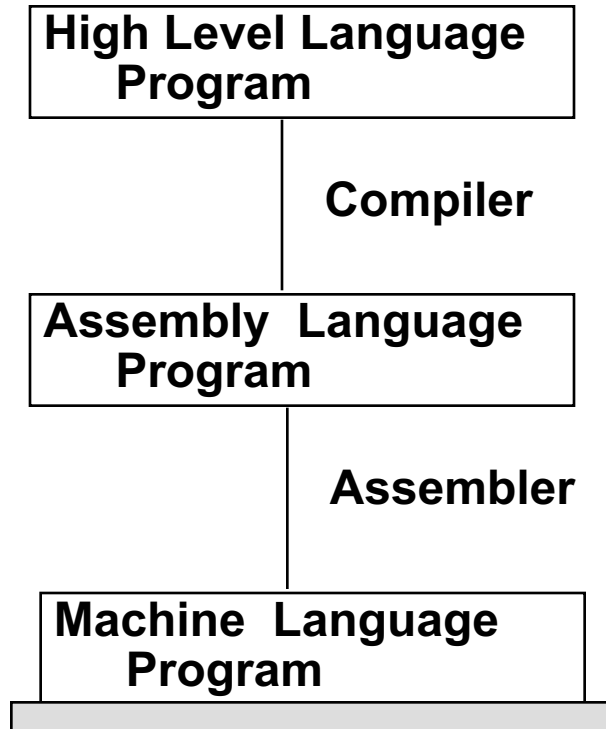**Assembler**

Machine Language Program

Machine Interpretation

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)

How to Speak Computer

High Level Language Program

Compiler

Assembly Language Program

Assembler

Machine Language Program

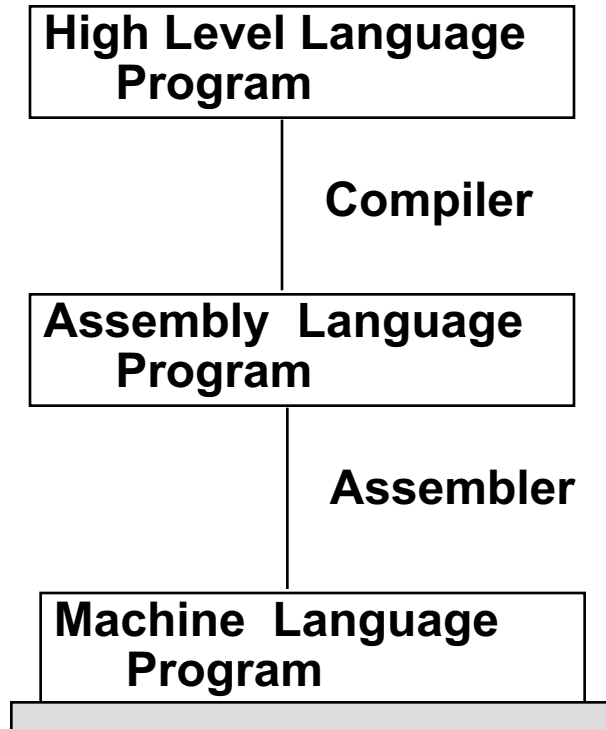**Machine Interpretation**

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```
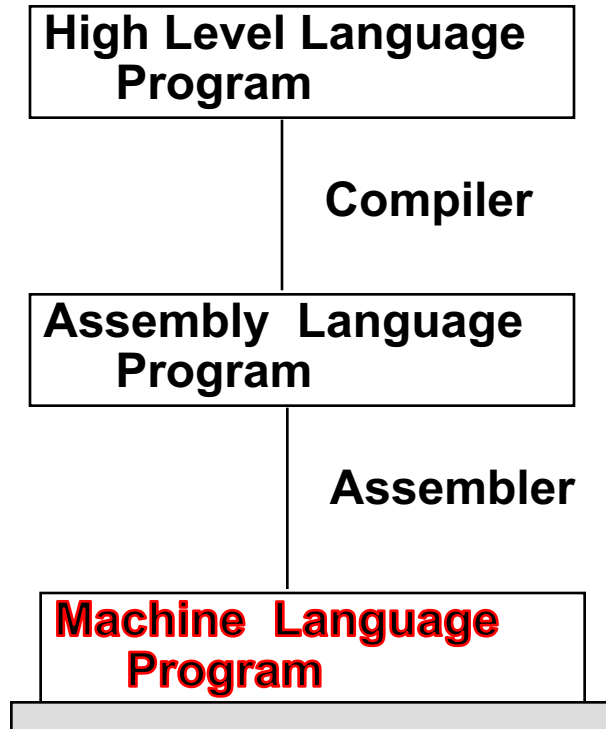
```
lw  $15,    0($2)
lw  $16,    4($2)
sw  $16,    0($2)
sw  $15,    4($2)
```

```
100011000110001000000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
```

How to Speak Computer

High Level Language Program

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

Compiler

Assembly Language Program

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

Assembler

Machine Language Program

10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100

Machine Interpretation

Machine does something!

How to Speak Computer

High Level Language Program

Compiler

Assembly Language Program

Assembler

Machine Language Program

Machine Interpretation

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)

1000110001100010000000000000000
1000110011110010000000000000100
1010110011110010000000000000000
1010110001100010000000000000100

Machine does something!

Machine Language

- Actual operations built into hardware.
  - Directly translated to electrical impulses
  - 1 = electricity > .5 V, 0 = electricity < .5V

- Provides direct access to CPU components.

CPU

- Central Processing Unit, or "chip"
- Actually performs instructions

- Contains
  - Mechanism to perform arithmetic instructions
  - Small amount of memory to hold inputs and outputs for these instructions
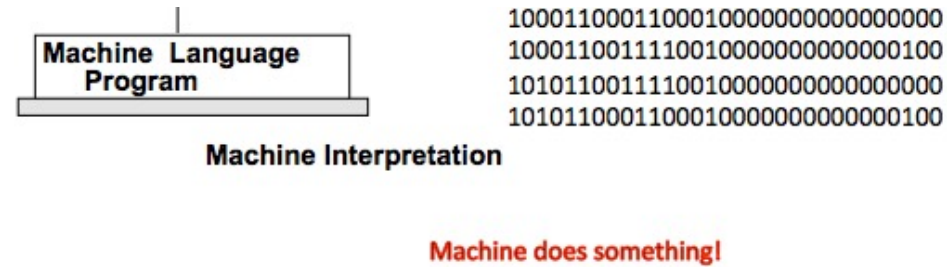
Registers

- (Very) Small amount of memory inside the CPU

- Data is put into a register before it is used in an instruction

- Manipulated data is then stored back in main memory.

Typical Machine Language Operations

- Load a word from memory into a register

- Store the contents of a register into a memory word

- Compute the sum (or difference) of two registers, store the result in a register

- Change which instruction runs next

- Change which instruction runs next based on a register value

Instruction Set Architecture (ISA)

Machine Interpretation

Machine does something!

- Abstracts from hardware (voltages) to machine language (1s & 0s)

- Encompasses all the information necessary to write a machine language program, including instructions, registers, memory access, …

- The definition (specification) of the machine language for a particular CPU
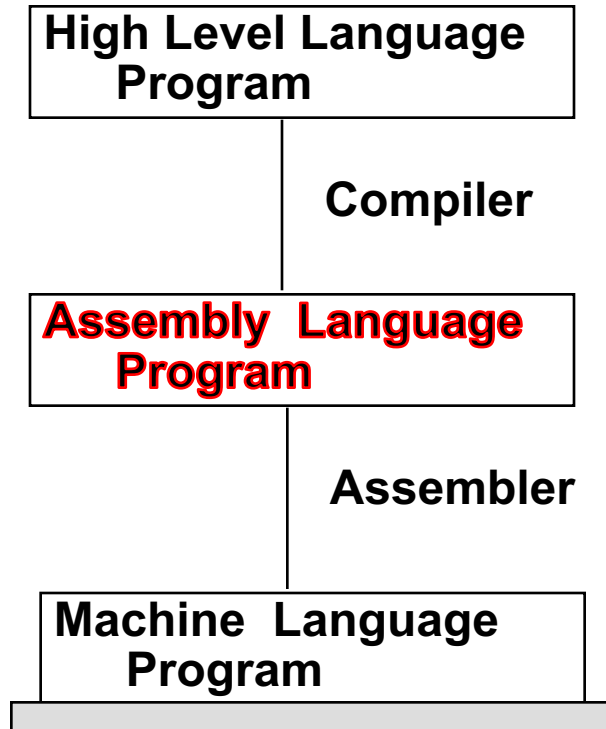
Examples of ISAs

- Intel x86, x86_64

- MIPS

- ARM: A32 (32-bit ARM), A64 (64-bit ARM), T32 (Thumb)

- Power ISA (PowerPC)

Which of the following statement is generally true about ISAs?

Select	Statement
A	Many models of processors support exactly one ISA.
B	An ISA is unique to one model of processor.
C	Every processor supports multiple ISAs.
D	Each processor manufacturer has its own unique ISA.
E	None of the above

How to Speak Computer

High Level Language Program

Compiler

Assembly Language Program

Assembler

Machine Language Program

Machine Interpretation

```
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 does something!

Assembly Language

- Abstraction of machine language
  - From 1s & 0s to symbolic names

- Allows direct access to architectural features (registers, memory)

- Symbolic names are used for
  - operations (mnemonics)
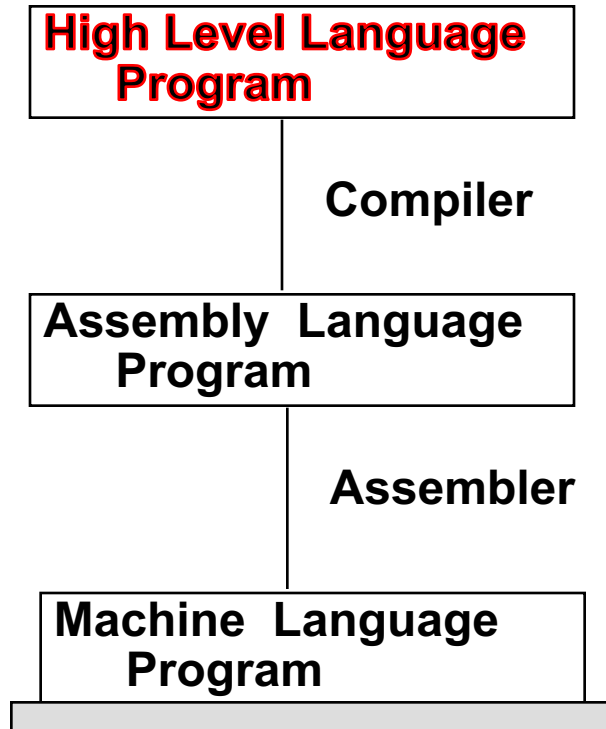 - memory locations (variables, branch labels)

C code

```
x = 4;
y = 5;
x = x + y;
```

MIPS code

```
li  $t0, 4  #load into reg
li  $t1, 5  #load into reg
add $t0, $t0, $t1 #add
```

How to Speak Computer

High Level Language Program

Compiler

Assembly Language Program

Assembler

Machine Language Program

Machine Interpretation

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 does something!

What are some advantages to a high-level language?

Rear Admiral Grace Hopper

- Invented the compiler
- Conceptualized machine-independent programming languages.
- Popularized term "debugging".

A single program written in a high level language can be compiled into _____ assembly language programs

A. Exactly one

B. Multiple

C. At most three

A single program written in assembly can be assembled into _____ machine language programs

A. Exactly one

B. Multiple

C. At most two

- High-level language program (in C)

```
swap (int v[], int k)
(int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
)
```

one-to-many

C compiler

- Assembly language program (for MIPS)

```
swap:  sll $2, $5, 2
       add $2, $4, $2
       lw  $15, 0($2)
       lw  $16, 4($2)
       sw  $16, 0($2)
       sw  $15, 4($2)
       jr  $31
```

one-to-one

assembler

- Machine (object, binary) code (for MIPS)

```
000000 00000 00101 0001000010000000
000000 00100 00010 0001000000100000
 . . .
```

Reading

- Next lecture: Hardware!
  - Sections 1.5

- Problem set 0 due Friday, Oct. 8 at 23:59