CSCI 210: Computer Organization
Lecture 2: Assembly Language

Stephen Checkoway
Oberlin College
Slides from Cynthia Taylor

Announcements

- Reading due before class, linked from blackboard
- Problem set 0 due this Friday at 23:59
  - On gradescope, linked from blackboard

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

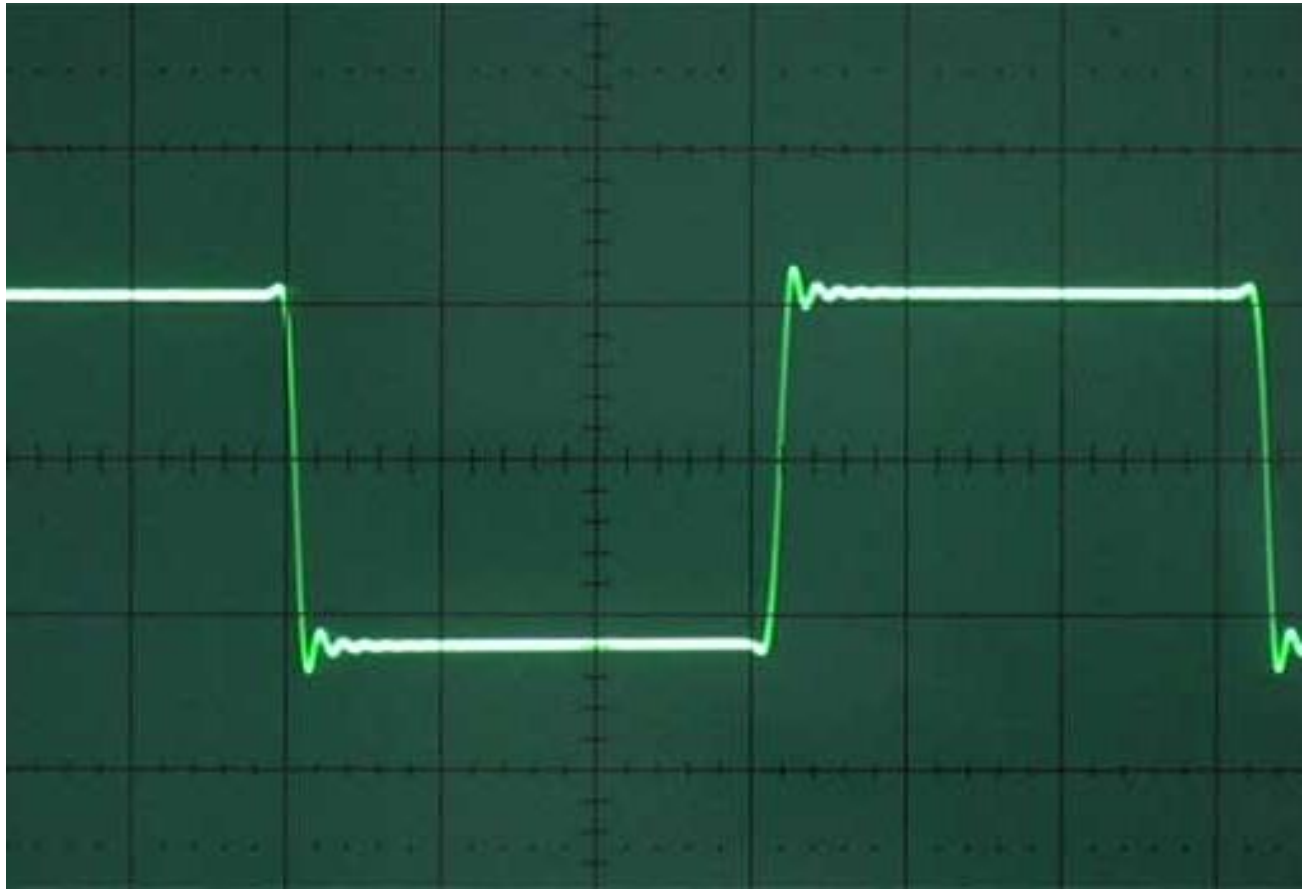
Selection	High Level Language	Assembly	Machine Language
A	3	2	1
B	3	1	2
C	2	1	2
D	1	2	2
E	None of the above		

What Your CPU Understands

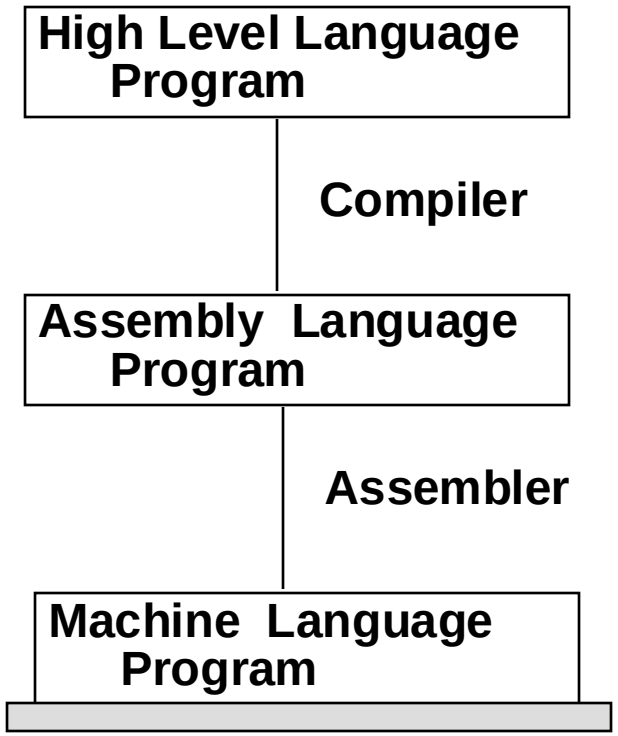
Electricity

Ones and zeros

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



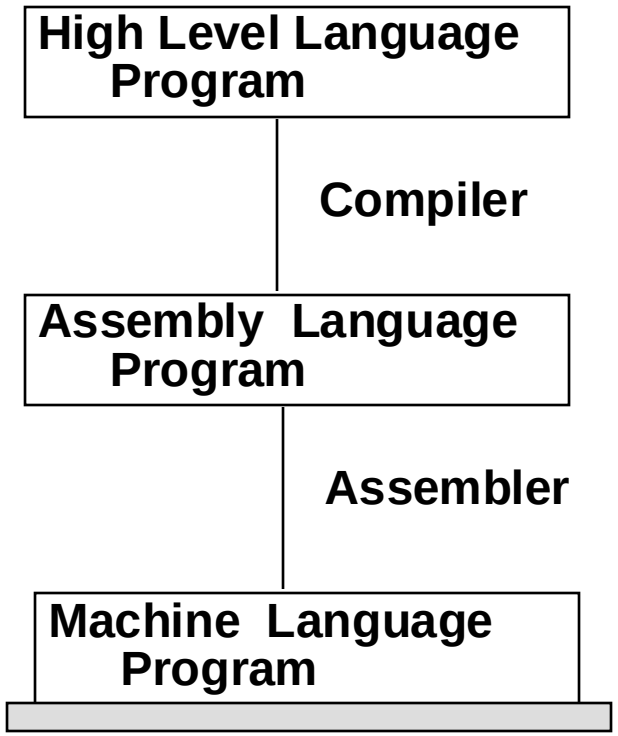
How to Speak Computer




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

Machine Interpretation

How to Speak Computer

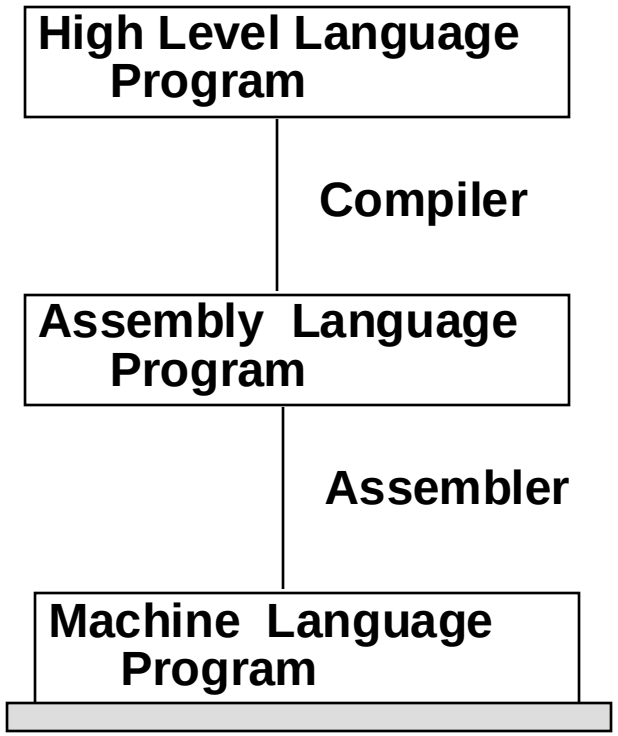



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

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

Machine Interpretation

How to Speak Computer



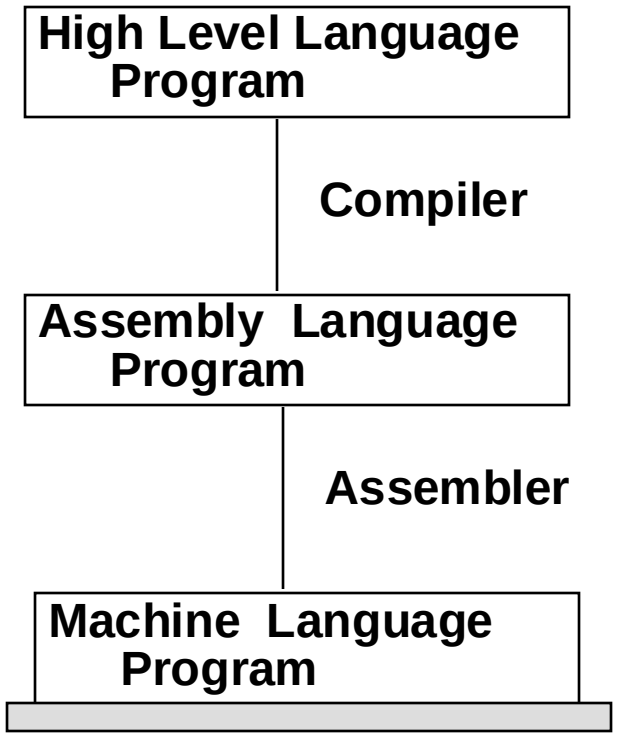

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

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

```
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
```

Machine Interpretation

How to Speak Computer




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

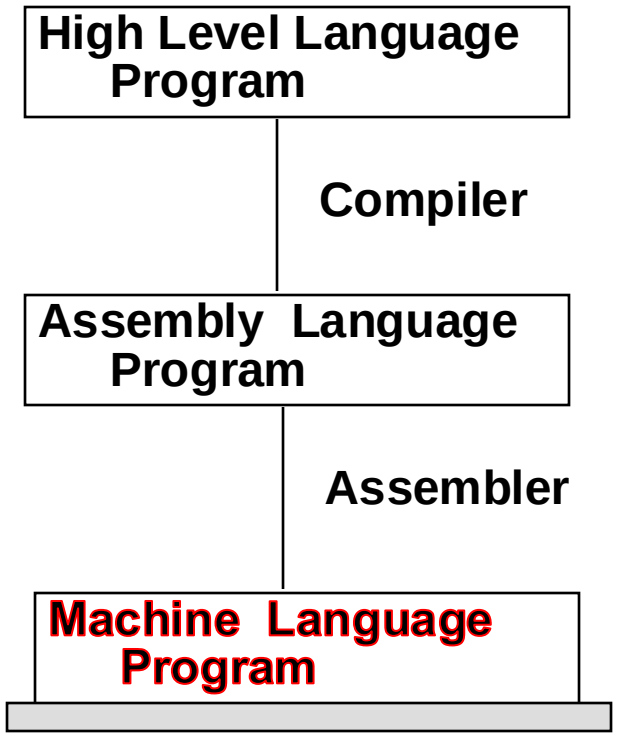
```
lw  $15,   0($2)
lw  $16,   4($2)
sw $16,   0($2)
sw $15,   4($2)
```

```
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
```

Machine Interpretation

Machine does something!

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)
```

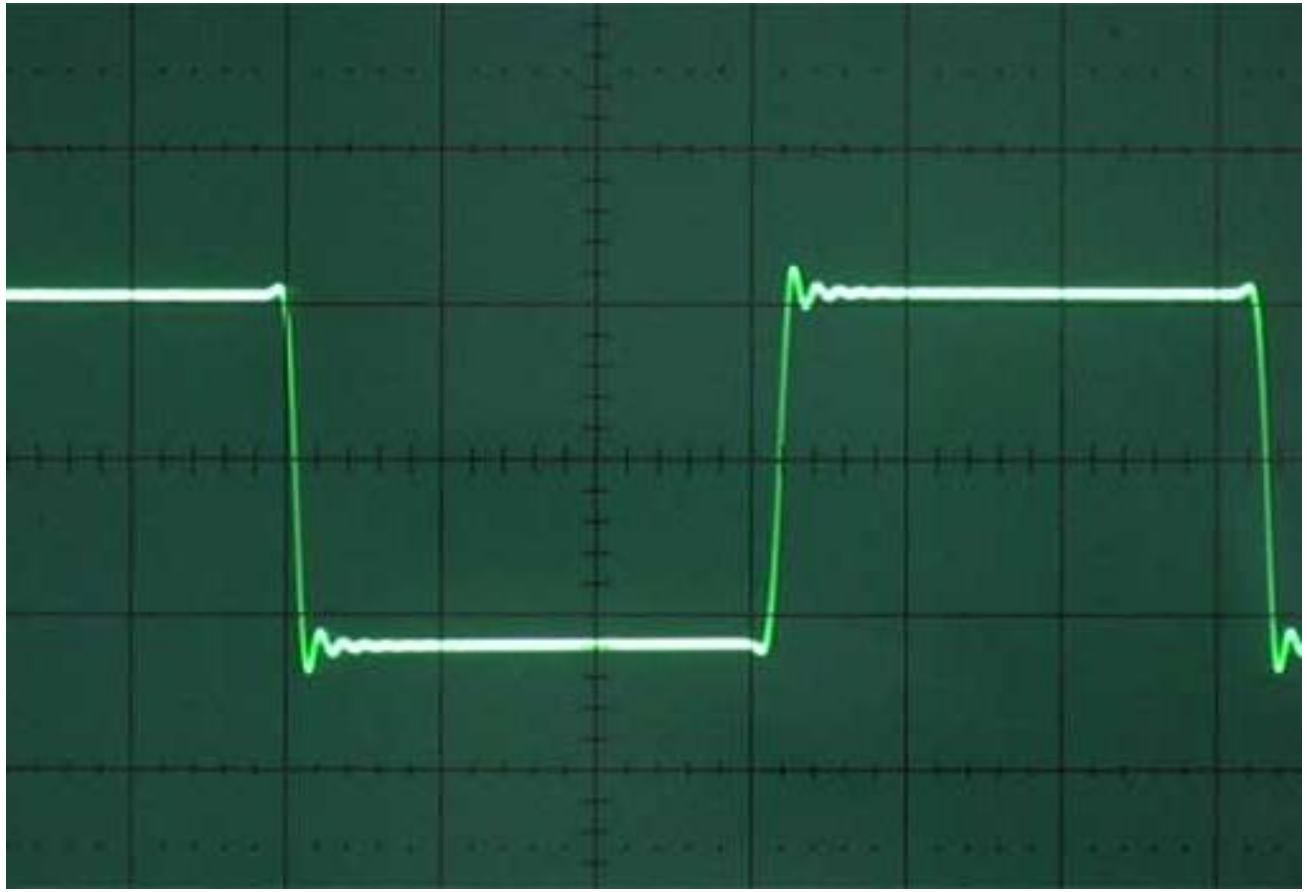
```
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
```

Machine Interpretation

Machine does something!

Machine Language

- Actual operations built into hardware.
  - Translated to electrical impulses
  - 1: voltage > .5 V
    0: voltage < .5V
- Provides direct access to CPU components.



CPU

- Central Processing Unit: contains one or more “cores,” each of which executes instructions independently from the other cores (we’re going to only focus on single-core CPUs but the same ideas apply to multi-core CPUs)
- Performs operations by executing instructions
- Contains
  - Mechanism to perform arithmetic operations
  - Small amount of memory to hold inputs and outputs for these instructions

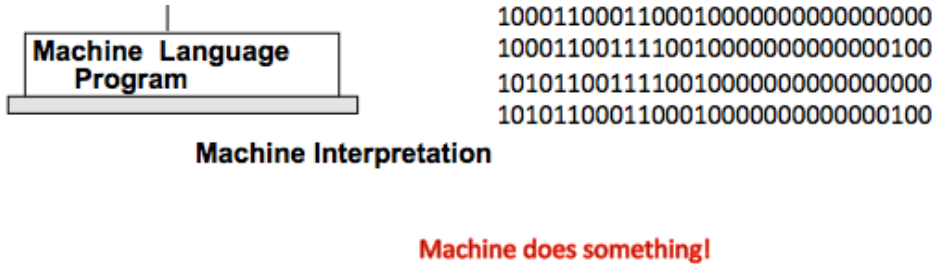
Registers

- (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 (RAM).

Typical Machine Language Operations

- Load data from main memory (RAM) into a register
- Store the contents of a register into main memory
- 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)



- 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

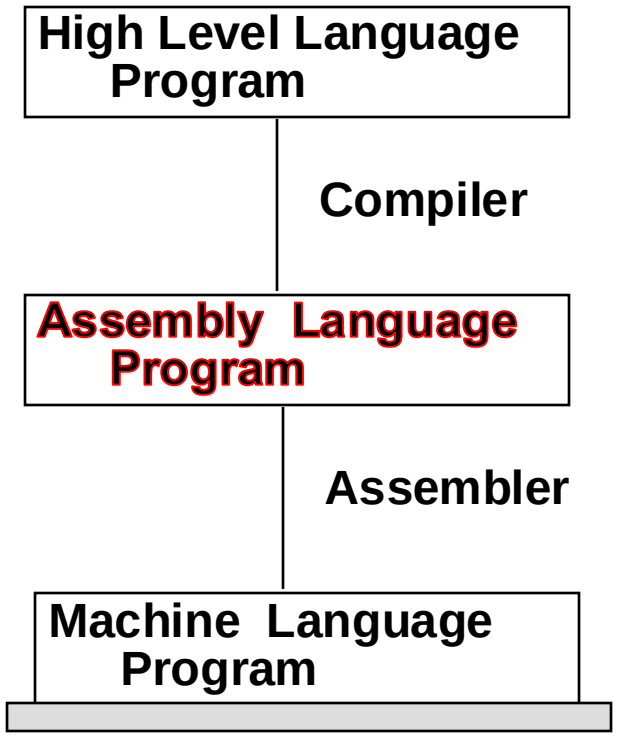
Examples of ISAs

- Intel x86, x86_64
- MIPS32, MIPS64
- ARM: A32 (32-bit ARM), A64 (64-bit ARM), T32 (Thumb), Apple Silicon
- Power ISA (PowerPC)
- Risc-V

Which of the following statement is generally true about ISAs?

Select	Statement
A	Some models of processors support exactly one ISA, others support multiple (usually related) ISAs
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




```
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

Machine does something!

High-level code

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

MIPS code

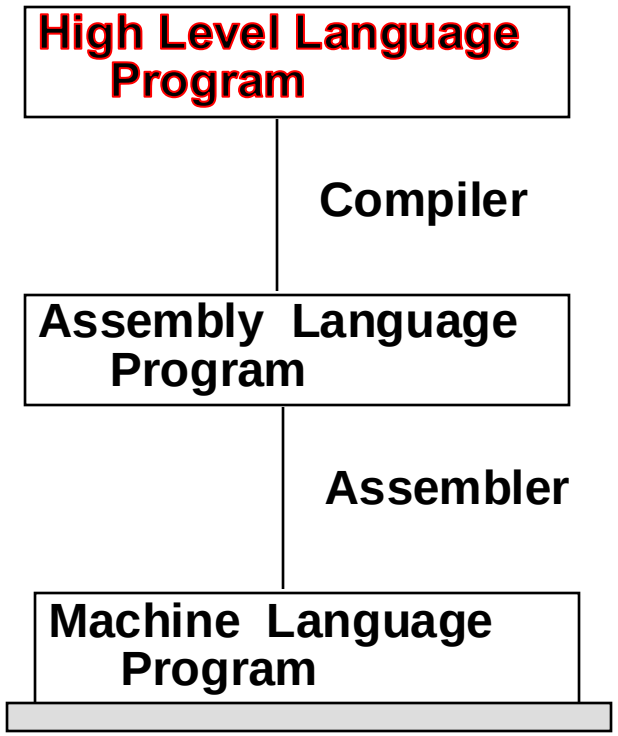
```
addi $t0, $zero, 4   #set $t0 to 4
addi $t1, $zero, 5   #set $t1 to 5
add $t0, $t0, $t1    #perform the add
```

Usually, 1 line of high-level code is translated to multiple assembly instructions; these are very simple

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)

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

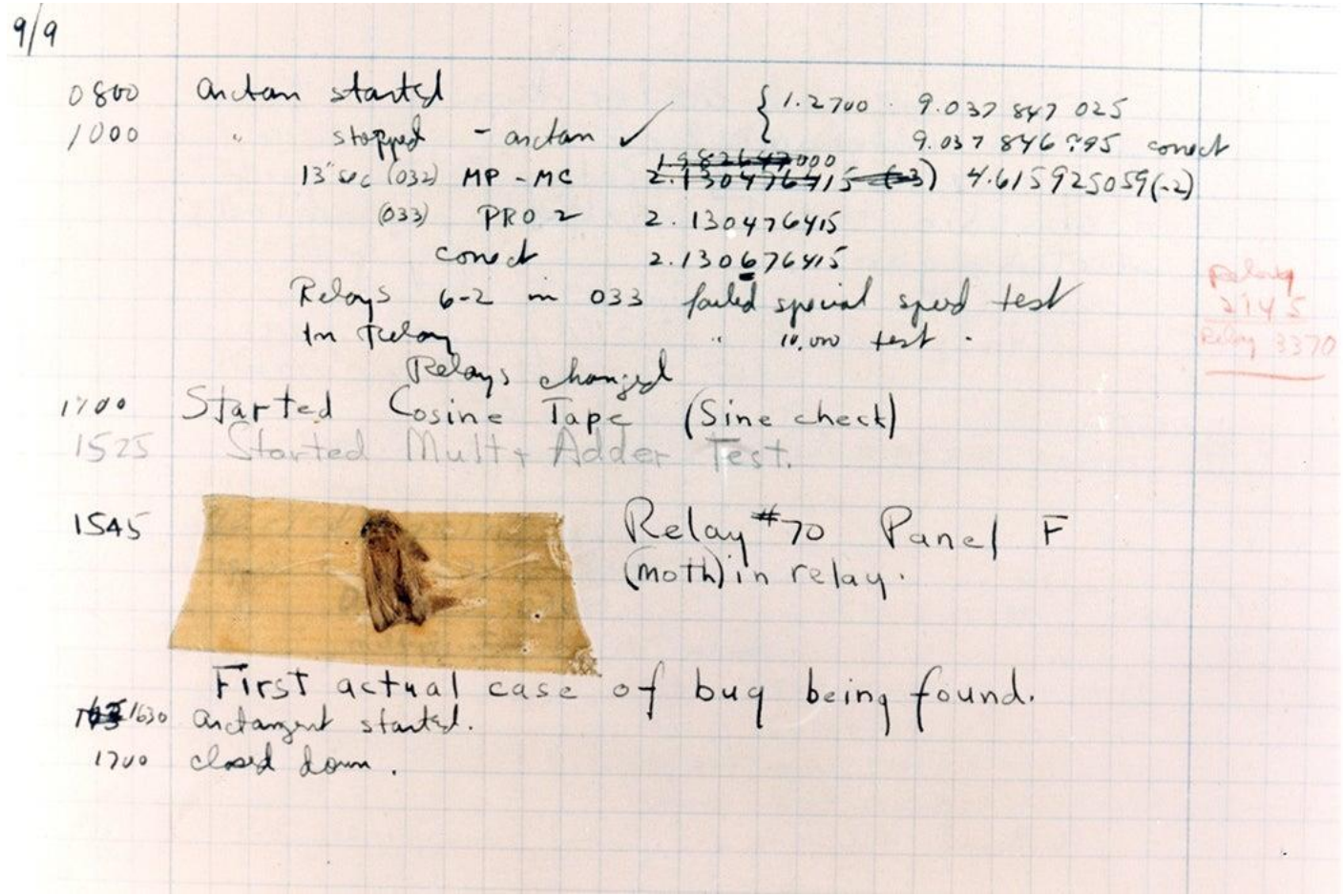
Machine does something!

Group Discussion: What are some advantages to a high-level language over programming in assembly?

CS History: Rear Admiral Grace Hopper

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

Not actually the first use of “bug” but a good story nevertheless



https://daily.jstor.org/the-bug-in-the-computer-bug-story/

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

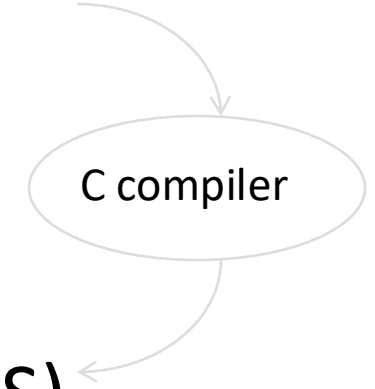
A. Exactly one

B. Multiple

C. At most two

High-level language program (in C)

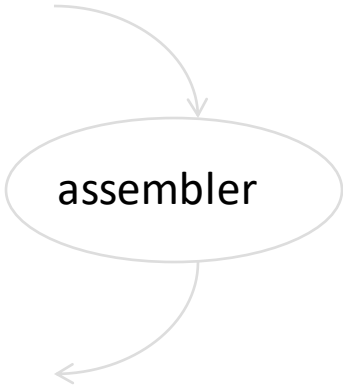
```
void swap (int v[], int k) {
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
}
```



one-to-many

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

Machine (object, binary) code (for MIPS)

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

Reading

- Next lecture: Hardware!
  - Sections 1.4 and 1.5

- Problem set 0 due Friday at 11:59 p.m.