Assembly Time

Assembly is one of the closest languages to the actual machine code that the computer gets.

Instruction Set Architecture (ISA): Basic instructions and variations of them

A single line of code in a normal language will be multiple instructions.

**General structure of syntax:**

*Keyword*: defines which operation

*Destination*: result of operation goes here

*Inputs*: what influences the result of the operation

**Syntax:**

*add:* add a, b, c

*addi:* addi a, b, 3

*sub*: sub a, b, c

There are a fixed number of operands (one output, two inputs)

Some sequences are better than others.

There is no *subi,* so use *addi* and a negative number

Instructions need to operate on registers, and there are a fixed, limited number of registers

MIPS ISA has 32 registers, x86 has only 8

Registers are usually 32-bit wide, but some modern architectures have 64-bit

A word: a 32-bit register

Registers are partitioned for code readability:

$s0 - $s7 (variables) instead of made up names

$t0 - $t9 (temp variables) if you need those

Loading word:

lw $t0, memory address (hex value)

Storing word:

sw $t0, memory address

You *can* put things into memory with MIPS, but he didn’t explain them well

An instruction may require a constant as its input

Immediate instruction: an instruction that uses a constant number as one of its inputs

Addi $s0, $zero, 10 to make $s0 = 10