Video 1

Week of Feb 21st

Human readable   Machine language

Computers uses 1 and 0

Humans letters

Assembler translates 0 and 1 into symbols.

ISA - is ~~the~~ assembler

ISA has opcodes that is letters and labels for memory locations.

Ex: multiply by 6.

```
        .ORIG   x3050   (start address)
        LD      R1, Six
        LD      R2, Number
        AND     R3, R3, #0
inner loop
Again   ADD     R3, R3, R2
        ADD     R1, R1, #-1   ; R1 keeps track of the iteration ;
        BrP     Again

    Halt;

Number  .BLKW   1
Six     .Fill   x0006
        .END        ← Should end with END
```

Instruction has the following format:

LABEL OPCODE OPERANDS ; comments

mandatory

optional

OPCODES - same as on LC3

ex: ADD, AND, NOT, ...

OPERANDS:

Registers - writes as Rn, n-is a number

Numbers - indicated by # decimal or # hex

label - symbolic name of memory location

↓

usually placed in the beginning of the line

Assembler Directives

Pseudo - Operations

	Opcode	Operand	Meaning
	.ORIG	address	Starting address of program
	.END		end of program
will asside memory location	.BLKW	n	allocate n words of storage
Just allocate 1 word	.FILL	n	allocate one word, initilize with value n
allow to put characters in memory	.STRINGZ	n-Char. strings	allocate n+1 locations, initialize w/ characters and null terminator

Style Guidelines

- Headers
- Comments
- Use symbolic names
- Try fit line on the page.

Video 3 Assembler first Pass && Second Pass

Machine code = 01

Assembly language (asm) = more time to create file to understand. Using 2 passes for it.

1) look for all code and looks for symbols (Strings) → creates assemble table.

2) Second Pass → convert instructions to 01

First Pass:

1) Find .ORIG statement (tells us adress of first instruction)

(Uses LC - Location counter to ~~[illegible]~~ keep track of current location)

2) For each non-empty line in program.

repeats til ③ {
a) if line contains a label, add label and LC to symbol table.
b) increment LC
}

3) Stop when .END reached

Label	Location
AGAin	x3053
Number	x3057
six	x3058

Second Pass: Generating Machine language

For each executable assembly language statement, generate machine language.

Problems

- improper number of ~~elements~~ arguments
- argument too large
- address more than 256 bits

Practice

Statement	Machine Language
LD R3, PTR	0010 011 000010000 0
ADD R4, R1, #-4	0001 100 001 1 11100
LDR R1, R3, #0	0110 001 011 000000
BRnp GeTCHAR	00 00 101 0000 000001

Q: why to x3012? (13:04)

Video 4 Assembler and Sample Assebly Language Program

-asm ~~[illegible]~~ generate different output files.
  .obj get loaded in simulator
  .obj contains address and instructions

Can have multiple object files.
object file - not nessecary a finished program
  (more like method)
↳ must include starting address

C and C++ has loading and linking

loading ↓ copying images into memory

linking ↓ process of resolving symbols between independent files.