

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 ~~an~~ 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

BRP Again

track of the
iteration ;

Halt;

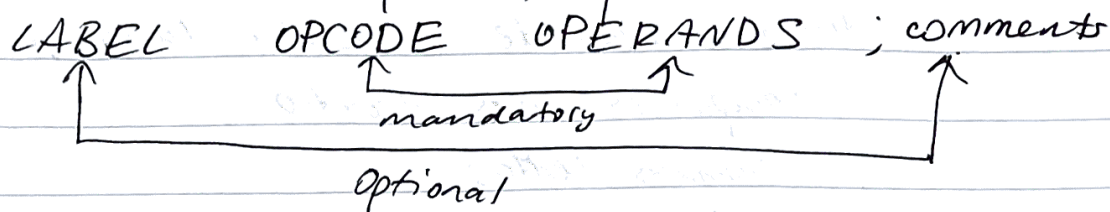
Number .BLKW 1

SIX .Fill x0006

.END

← should end with END

Instruction has the following format:



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
will assign memory location	.ORIG	address	starting address of program
	.END		end of program
	.BLKW	n	allocate n words of storage
Just allocate 1 word	.FILL	n	allocate one word, initialize 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 address of first instruction)

(Uses LC - location counter to ~~the~~ 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 ~~statements~~ arguments
- argument too large
- address more than 256 bits

Practice

Statement	Machine Language
LD R3, PTR	0010 011 000 000 00
ADD R4, R1, #4	0001 100 001 1 111 00
LDR R1, R3, #0	0110 001 011 00 00 00
BRnp GETCHAR	00 00 101 0000 000 001

Q: why to x3012? (13:04)

Video 4 Assembler and Sample Assembly Language Program

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

Can have multiple object files.

object file - not necessarily a finished program
(more like method)

↳ must include starting address

C and C++ has loading and linking

↓
copying images
into memory

↓
process of resolving
symbols between
independent files