Assembly Language Structure

1. Instruction Set

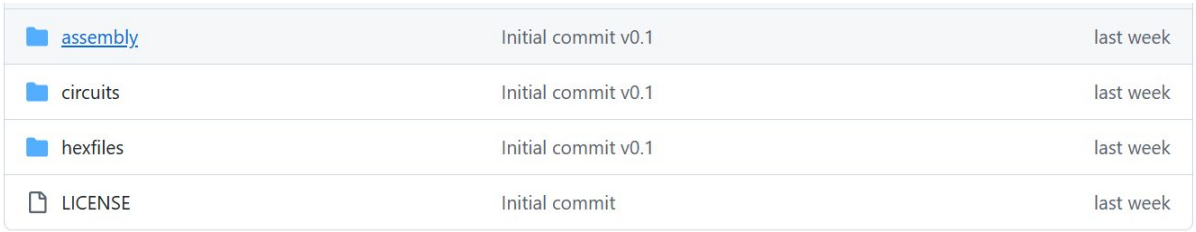
Opcode	Operand	Cycles	Description
LDA	$A	2	Loads data from address $A to AX (accumulator).
STA	$A	2	Stores data to address $A from AX.
MOV	$A,$B	4	Moves data at address $A to address $B.
JMP	$Q	2	Jumps to next instruction at address $Q.
JCB	$Q	2	Jumps to $Q if “Carry” flag is high.
JAL	$Q	2	Jumps to $Q if “A Larger” flag is high.
JEQ	$Q	2	Jumps to $Q if “Equal” flag is high.
JZR	$Q	2	Jumps to $Q if “Zero” flag is high.
PHS	$A	4	Push data at address $A to stack.
PLS	$A	4	Pull data to address $A from stack.
NOP		8	No operation (useful for sleep).
HLT		1	Halt until reset.
ADD	$A,$B	4	Add the data contained within addresses $A and $B.
SUB	$A,$B	4	Subtract the data contained within addresses $A and $B.
MUL	$A,$B	4	Multiply the data contained within addresses $A and $B.
CMP	$A,$B	4	Bitwise compare data at addresses $A against $B.
ANA	$A	3	Logical AND data in AX with data at address $A.
ORA	$A	3	Logical OR data in AX with data at address $A.
XRA	$A	3	Logical XOR data in AX with data at address $A.
NOT	$A	3	Logical NOT data at address $A.

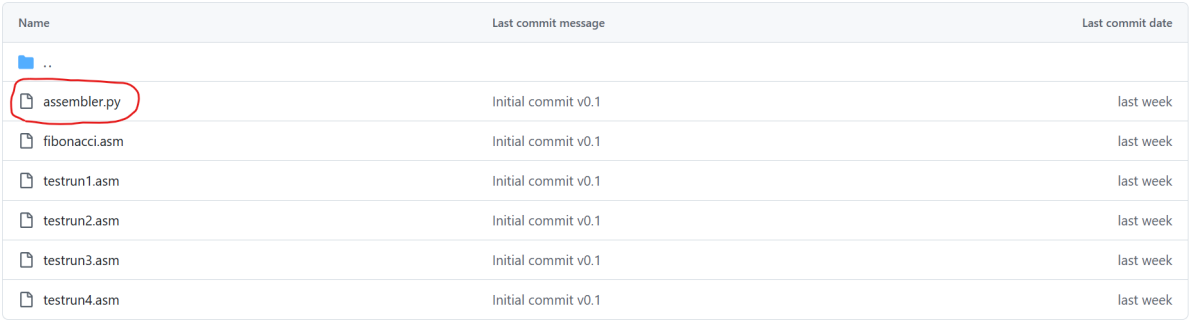
2. Sourcefile Structure

```
; ****************** ABACUS-8 ASSEMBLY GUIDE ******************
; Version 0.1 (26/01/2026)
; (c) 2026 KemalAbizar
;
; This is how comments are written. Current version of Abacus-8'
; assembler does not support multi-line comments, such as ones
; in Python utilizing triple quote or double-quote symbols. Hence,
; one must manually write new lines and add semicolon symbol, only
; then can "multi-line" comments be made.
;
; NOTE: be sure to only have maximum of one semicolons per line!


; Program segment. This is where one writes the program he/she
; desires the computer to execute. Any other instruction lines, or
; instruction loops written outside of this scope, will be ignored.
.prog:
count:          ; Loop descriptor / label.
      add x,y
      sta z
      mov y,x
      mov z,y
      cmp x,l
      jal terminate
      jmp count
terminate:      ; Another loop descriptor / label.
      lda mssg
      hlt

; Data segment. All variables necessary are declared here.
; NOTE: variables must always be declared with ':=' symbols.
.data:
x := $00
y := $01
z := $00
l := $5f
mssg := $e2
```

3. How to Write, Assemble and Run the Program

Writing the assembly sourcefile for Abacus-8 won't require anything more than basic text editors (Notepad/++, MS Word etc.) or program editors (MS Visual Studio Code). After one finishes writing the program, it must be saved in the project folder titled "assembly," which also contains a Python program titled "assembler.py," as shown in figures below.

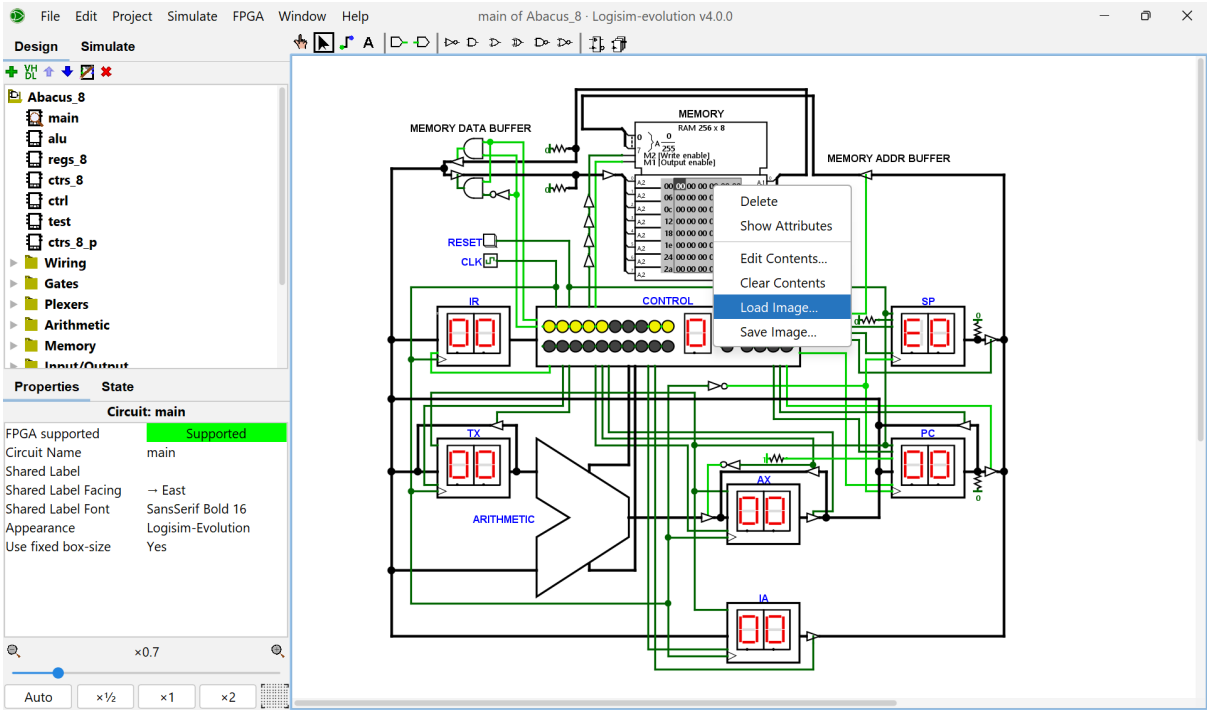


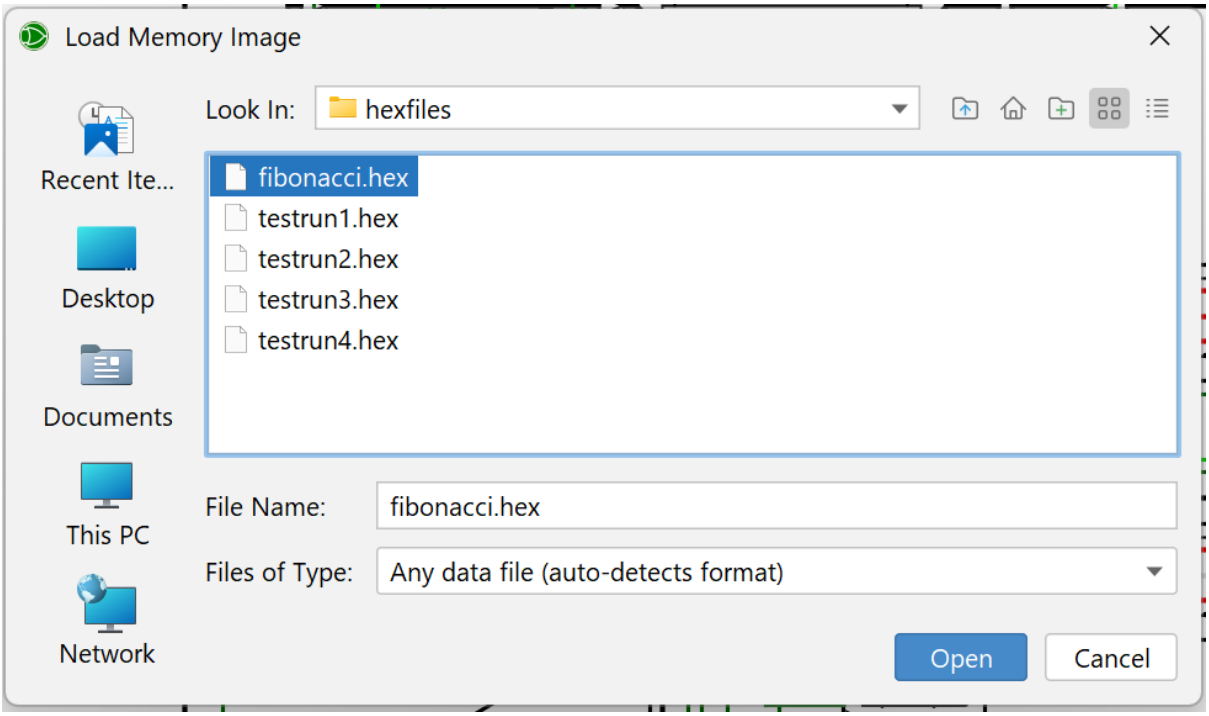


Open a Command Prompt (Windows) or Terminal (Linux), then change the directory to access the "assembly" folder. To compile (technically it's called assemble) the sourcefile that one just wrote, type the following command shown in green box, then press Enter. The output file, highlighted by yellow box, contains the generated hexadecimal data and their corresponding memory locations. Open another empty text/program editor window, then copy all generated memory contents into it; save it with identical name as the assembly sourcefile, but this time with .hex extension. Save the .hex file into the "hexfiles" folder.

```
C:\Users\M. Kemal Abizar\Documents\Projek\Computer\assembly> python assembler.py fibonacci.asm
v3.0 hex words addressed
00: 00 80
02: 80 80 81
05: 10 82
07: 20 81 80
0a: 20 82 81
0d: b0 82 83
10: 3a 14
12: 30 00
14: 00 80
16: 70
80: 00
81: 01
82: 00
83: 1f

C:\Users\M. Kemal Abizar\Documents\Projek\Computer\assembly>
```

The final step is to open the circuit file (ones with .circ extension), then right-click the RAM or "Memory" module, which would show a dropdown menu named "Load Image." Click this, then double-click on your recently compiled hexfile; this way, one have successfully loaded your program into the Abacus-8 computer. All one needs to do, is to enable auto-clock by pressing Ctrl+K, and observe the computer coming to life.


File Edit Project Simulate FPGA Window Help
main of Abacus_8 · Logisim-evolution v4.0.0
Design Simulate
Abacus_8
main
alu
regs_8
ctrs_8
ctrl
test
ctrs_8_p
Wiring
Gates
Plexers
Arithmetic
Memory
Properties State
Circuit: main
FPGA supported Supported
Circuit Name main
Shared Label
Shared Label Facing → East
Shared Label Font SansSerif Bold 16
Appearance Logisim-Evolution
Use fixed box-size Yes
×0.7
Auto ×½ ×1 ×2
MEMORY DATA BUFFER
MEMORY
RAM 256 x 8
MEMORY ADDR BUFFER
Delete
Show Attributes
Edit Contents...
Clear Contents
Load Image...
Save Image...
RESET
CLK
IR
CONTROL
SP
TX
PC
AX
ARITHMETIC
IA


Load Memory Image
Look In: hexfiles
Recent Ite...
Desktop
Documents
This PC
Network
fibonacci.hex
testrun1.hex
testrun2.hex
testrun3.hex
testrun4.hex
File Name: fibonacci.hex
Files of Type: Any data file (auto-detects format)
Open
Cancel