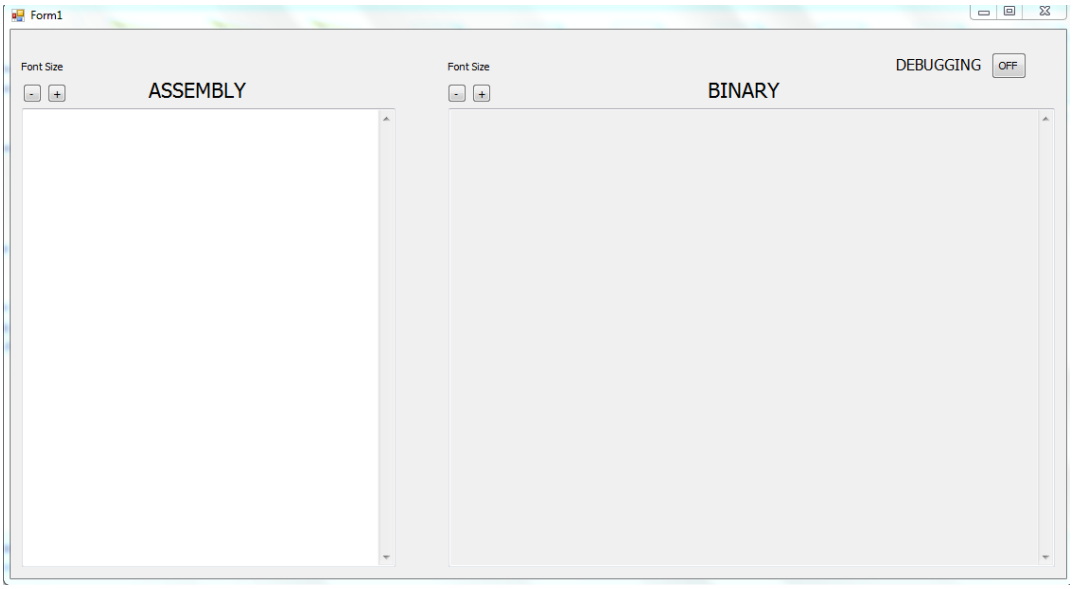
Ain Shams University

Faculty of engineering

Computer and System Engineering Department

Computer Organisation II

Report on: assembler GUI



Team members:

Eslam Alaa Zaki	section 1
Fady Faragallah Khalifa	section 2
Mark Remon Ebrahim	section 2
Kirollos Sherif Henry	section 2
Bishoy George Michael	section 1

- **Overview:**

this is a GUI for assembling the mips instructions into a machine code

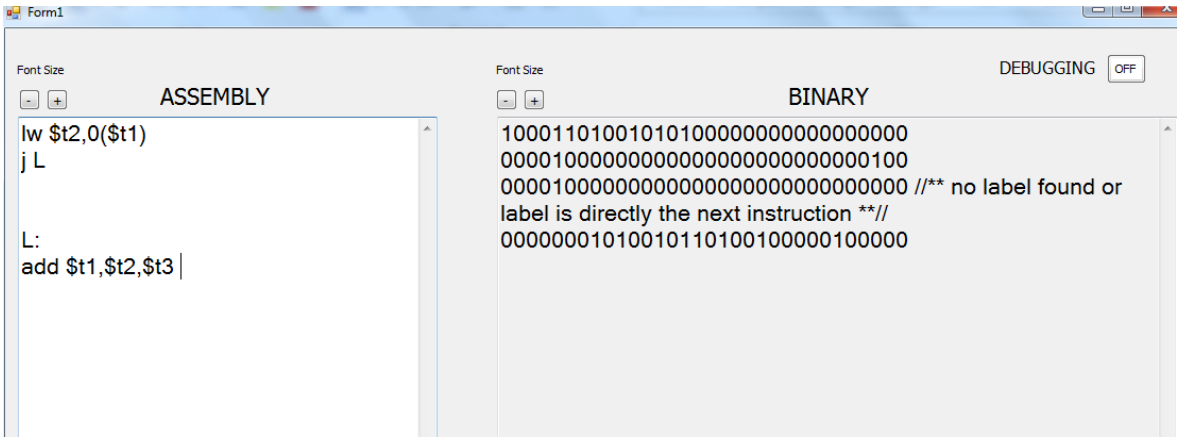
- the assembler supports:

add	sub	and	or	nor	sll	srl	sra
slt	addi	andi	ori	slti	lui	beq	bne
lw	sw	j	jr	jal			

- **how to use the GUI:**

  the GUI immediately assembles the instructions into a machine code that is displayed on the right also stores it in a text file named ”inst_mem.txt”

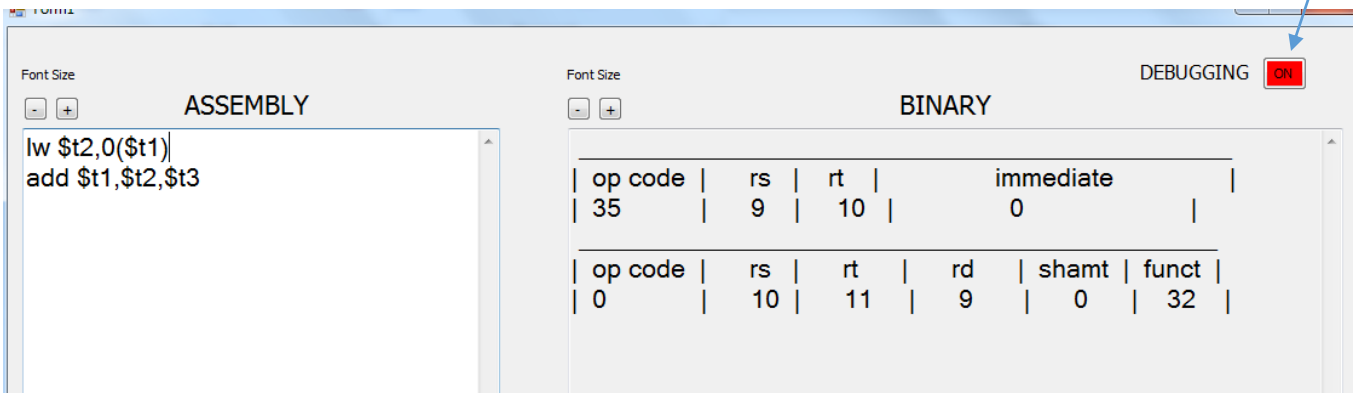
- you must put it in the same directory as the inst_mem you are going to use.
- if you want to use a label you need to put the instruction in the same line. don’t leave the label line empty it won’t work



this is wrong

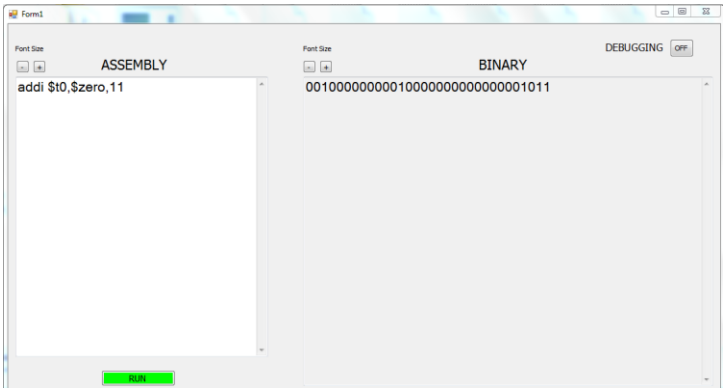
write the instruction in the same line besides the label

- also don’t leave empty lines
- there’s a debugging button (OFF by default) click on it to turn it ON to see the instruction fields in decimal numbers



- don’t use nop as it isn’t supported instead use sll $zero,$zero,0

another feature is added is the ability to see the output but you must close modelsim and the file should be cpu.v in the path as assembler.exe and "inst_mem.txt" "reg_f.txt" "data_mem.txt" should be in the same path.



after filling the the assembly code just click run and it will show the results

to edit again click edit.

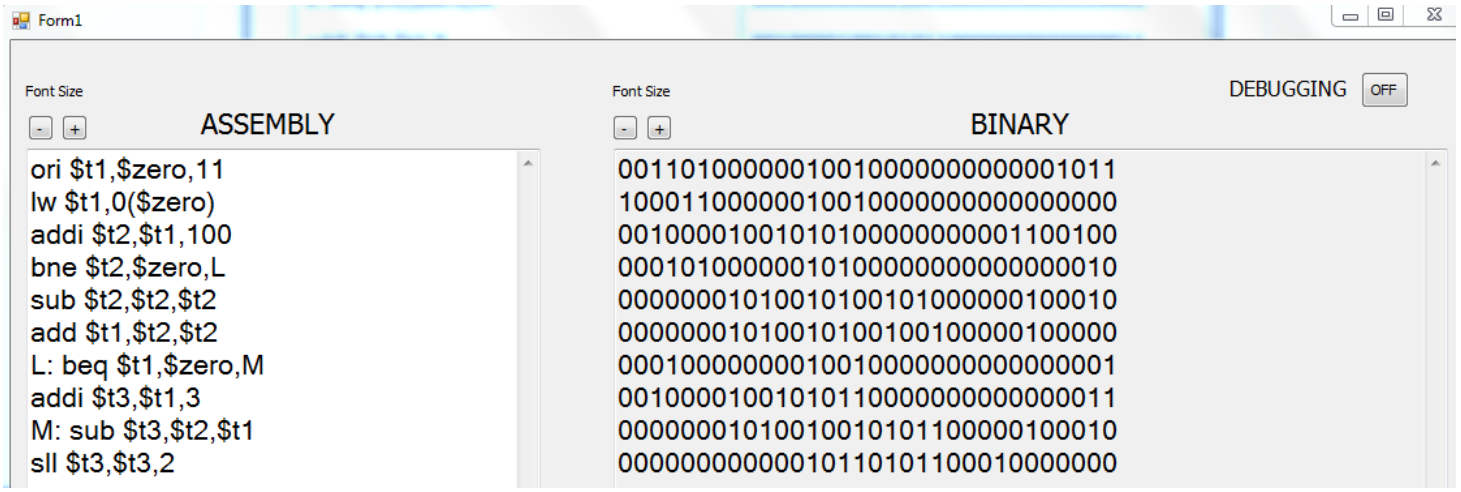
- **Take care: the gui set simulation time = 300 ns if you want to run instructions take more than 300 ns, you should run and see the result from modelsim.**

- **test cases:**

  test case(1):

assembly	expected output
ori $t1,$zero,11	00110100000010010000000000001011
lw $t1,0($zero)	10001100000010010000000000000000
addi $t2,$t1,100	00100001001010100000000001100100
bne $t2,$zero,L	00010100000010100000000000000010
sub $t2,$t2,$t2	00000001010010100101000000100010
add $t1,$t2,$t2	00000001010010100100100000100000
L: beq $t1,$zero,M	00010000000010010000000000000001
addi $t3,$t1,3	00100001001010110000000000000011
M: sub $t3,$t2,$t1	00000001010010010101100000100010
sll $t3,$t3,2	00000000000010110101100010000000

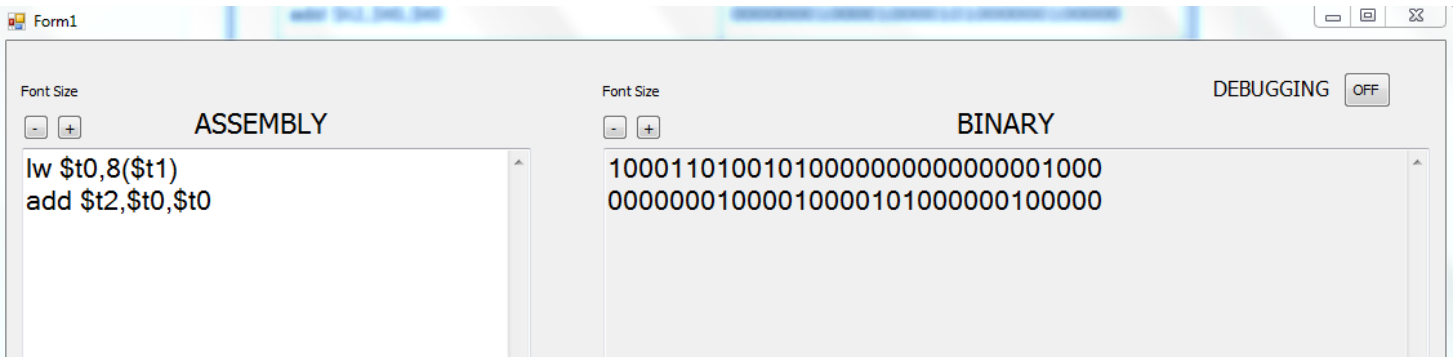
GUI output:



test case(2):

assembly	expected output
lw $t0,8($t1)	10001101001010000000000000001000
add $t2,$t0,$t0	00000001000010000101000000100000

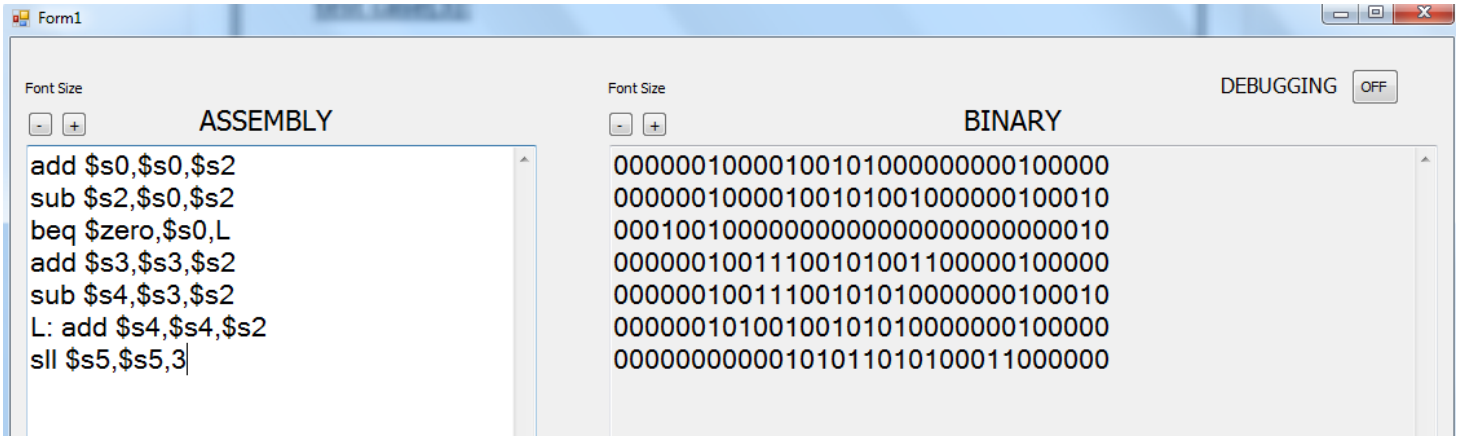
GUI output:



test case(3):

assembly	expected output
add $s0,$s0,$s2	00000010000100101000000000100000
sub $s2,$s0,$s2	00000010000100101001000000100010
beq $zero,$s0,L	00010010000000000000000000000010
add $s3,$s3,$s2	00000010011100101001100000100000
sub $s4,$s3,$s2	00000010011100101010000000100010
L: add $s4,$s4,$s2	00000010100100101010000000100000
sll $s5,$s5,3	00000000000101011010100011000000

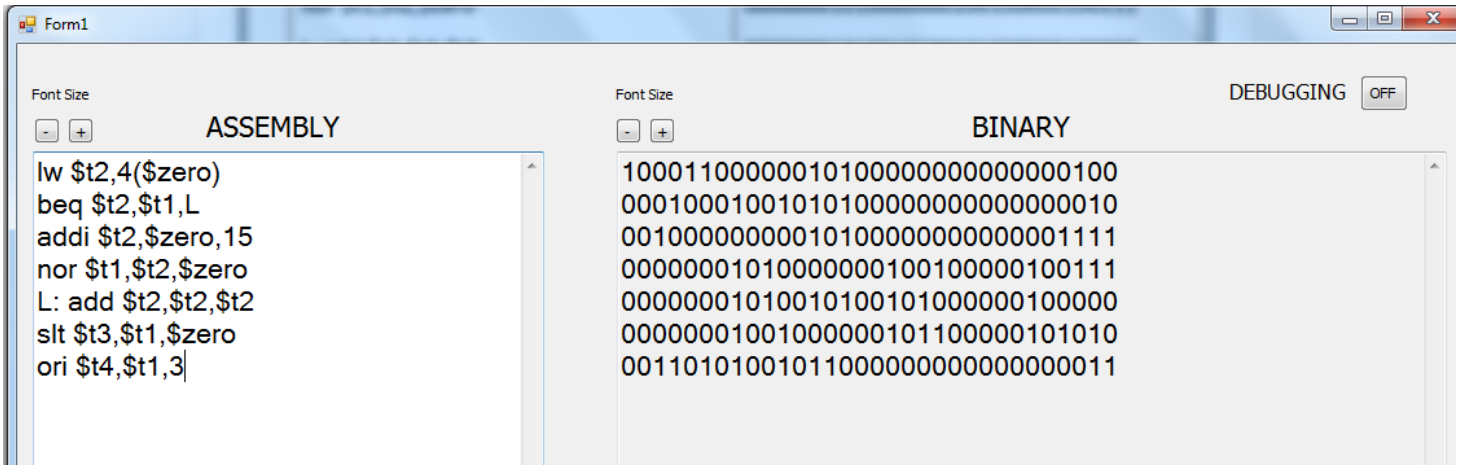
GUI output:



test case(4):

assembly	expected output
lw $t2,4($zero)	10001100000010100000000000000100
beq $t2,$t1,L	00010001010010010000000000000010
addi $t2,$zero,15	00100000000010100000000000001111
nor $t1,$t2,$zero	00000001010000000100100000100111
L: add $t2,$t2,$t2	00000001010010100101000000100000
slt $t3,$t1,$zero	00000001001000000101100000101010
ori $t4,$t1,3	00110101001011000000000000000011

GUI output:

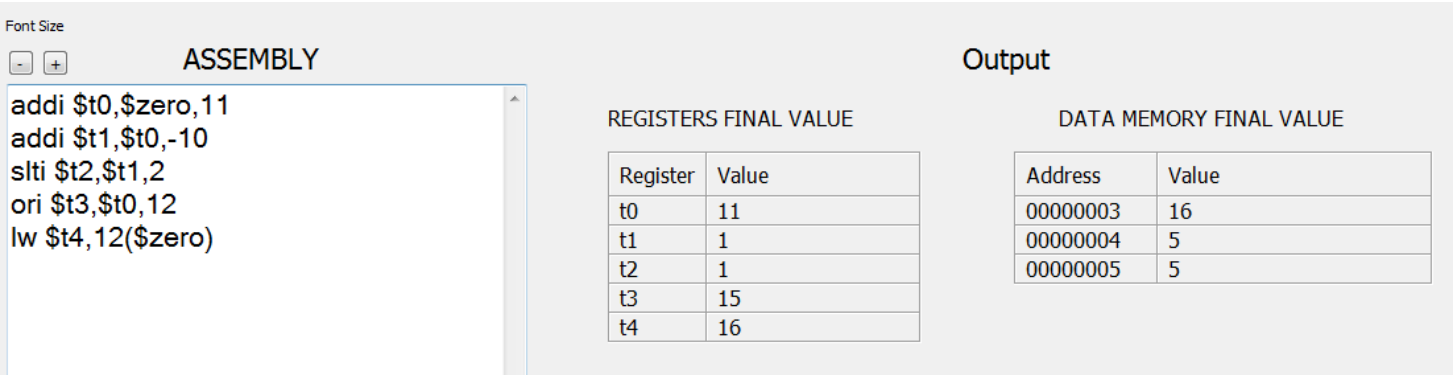


- **test case for the added feature:**

assembly	machine code
addi $t0,$zero,11	00100000000010000000000000001011
addi $t1,$t0,-10	00100001000010011111111111110110
slti $t2,$t1,2	00101001001010100000000000000010
ori $t3,$t0,12	00110101000010110000000000001100
lw $t4,12($zero)	10001100000011000000000000001100

initial (decimal values)	expected output (decimal values)
all register are 00000000 data memory[3]=16 data memory[4]=5 data memory[5]=5	$t0=11 $t1=1 $t2=1 $t3=15 $t4=16 no change in data memory

GUI output:


Font Size
ASSEMBLY
addi $t0,$zero,11
addi $t1,$t0,-10
slti $t2,$t1,2
ori $t3,$t0,12
lw $t4,12($zero)
Output
REGISTERS FINAL VALUE
Register Value
t0 11
t1 1
t2 1
t3 15
t4 16
DATA MEMORY FINAL VALUE
Address Value
00000003 16
00000004 5
00000005 5