Computer Architecture Project

Submitted to:

Dr. Cherif Salama Ramzi

Submitted by:

NAME:	***ID:***
Mayar Wessam Hassan	16P3008
Rowan Hazem Wagieh	16P3023
Engy Samy Salah	16P3004
Gina Emil Attia	16P3022

Implementation Description:

- **Project Language:** We have implemented our project using JAVA language.
- **Implementing:** We have simulated all the components (Registers , Alus , Memories , Multiplexers , Sign Extenders , Control Unit , Pc & Instruction Memory) as separated classes. Each class contains a method or more.
  Class Instruct_mem: contains a method that stores the instructions which the user has entered in array.
  Class PC: contains a static variable that has the address which the user has entered. Also it contains a method that adds 4 on the address.
  Class Reg_file: contains 3 arrays for registers. One for the registers' addresses, and one for the initial values (zero), and the last one is for the registers' names. It also contains 2 methods, one for writing in a specific register, and the other one for reading the value in a specific register.
  Class Instruction: contains a method that takes the instruction from the user (in 32-bit binary form). If it is R-type, it will be divided into 6 bits for the opcode, 5 bits for first source register, 5 bits for second source register, 5 bits for destination register, 5 bits for shift amount, and 6 bits for function code.
  If it is I-type, it will be divided into 6 bits for opcode, 5 bits for source register, 5 bits for destination register

(rt in this condition), and 16 bits for a constant or address.
If it is J-type, it will be divided into 6 bits for opcode and 26 bits for address.

Class Control_Unit: contains a method that takes the opcode from the instruction class's method and generate the control unit signals for each opcode.

Class Sign_Extend: contains a method that takes the constant (or the address) from the instruction class's method and extends it from 16 bits to 32 bits.

Class ALU_Control: contains a method that takes the opcode and the function code (if it is R-type) from the instruction class's method and create the needed function code.

Class ALU: contains a method that takes the function code from the ALU_control class's method and controls the ALU for which operation it will perform. Also it contains a variable (zeroflag) which its value is always (0), its value might change to (1) in case of (branch instruction) only according to the value of ALU.

Class Memory: contains a method that support SW, SB, LW, LB, LBU operations. In SW operation it takes the needs address from ALU and read the needed value from (rt) which is the word (32-bits).It divides this word into 4 bytes. For example, the needed address to store in is 200, So first byte will be stored in 200, the second in 201, the third in 202 and the last one in 203. In SB operation it takes the byte and store

it in the address needed. In LW operation it takes the needed address from ALU. For example, the address is 200, it goes to 200 and load the first byte then to 201 and load the second byte then to 202 and load the third byte then to 203 and load the last byte, Finally, it concatenates the 4 bytes. In LB it takes the needed address from the ALU to load from it. And in LBU operation the same thing happens as in LB operation, but it converts the loaded bytes into a positive number.

200	**201**	**202**	**203**
204	205	206	207
208	209	210	211
212	213	214	215

Class Muxes: contains a method that takes RegWrite, Memtoreg, and Regdst signals from Control_Unit class's method to know where to write the output value from the ALU class's method.
Class ALU_Branch: contains a method that takes the constant (the address) from the Sign_Extend class's method and multiply it by 4.
Class MUX_Branch: contains a method that controls if the new address which will be entered the pc will be the value from the (Pc class)'s method or

the value from (ALU_Branch class)’s method added to the value from the (Pc class)’s method.

Class Project1: contains 2 methods, one for calling all the previous classes, and the other one is for printing all the signal wires values. Also it contains the main of the program.

Datapath:

All the instructions are saved in the instruction memory, then each clock cycle a single instruction is loaded from it according to the pc address.

Firstly the instruction goes to class (Instruction) which divides the instruction (32-bit) into specific number of bits according to its format.

In addition we have supported an extension which is a wire for the shift-amount 5-bits that goes directly to class (ALU).

Secondly we use the Opcode in class (Contol_Unit) to specify the right control signals for this instruction.

Also we have supported an extension which is a signal wire (1-bit) for the jump operation.

Thirdly the constant which is from Instruction[15-0] bit goes to class (Sign_Extend) to extend it from 16 bit to 32 bit.

Fourthly the ALUOp from class (Control_Unit) and the function code from class (Instruction) goes to class (ALU_Control) to determine the needed ALU- function code for class (ALU).

After that we call the class (reg_file) to determine from it the needed value of (RS) and read it.

Then It goes to class (ALU) to do the specific operation.

Moreover, it goes to class (Memory) to know if we need to read from the memory or write in it. Otherwise it writes the output value in (rd register) by using class (Muxes).

Then detecting the address which passes to PC in class (Mux_branch).

Finally The signal wires, register file and memory are printed.

Truth table:

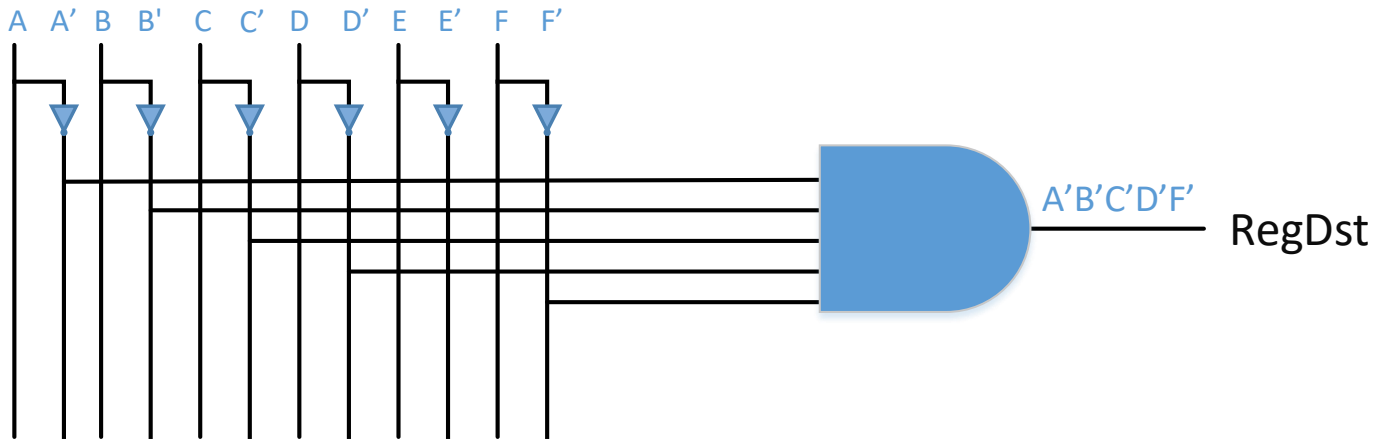
OP-CODE						Control Signals									
						RegDst	Branch	Mem-Read	Mem-toReg	ALUOp		Mem-Write	ALU-Src	Reg-Write	Jump
0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	X	1	0	X	0	1	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0
0	0	1	0	1	0	0	0	0	0	1	1	0	1	1	0
0	0	0	0	1	0	X	0	0	X	x	x	0	x	0	1
0	0	0	0	1	1	X	0	0	X	x	x	0	x	1	1
1	0	0	0	1	1	0	0	1	1	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0
1	0	0	1	0	0	0	0	1	1	0	0	0	1	1	0
1	0	1	0	1	1	X	0	0	X	0	0	1	1	0	0
1	0	1	0	0	0	X	0	0	X	0	0	1	1	0	0

K-maps:

Case 1: (RegDst)

ABC / DEF	000	001	011	010	100	101	111	110
000	1	0			0	X		
001								
011	X				0	X		
010	X	0						
100	X				0			
101								
111								
110								

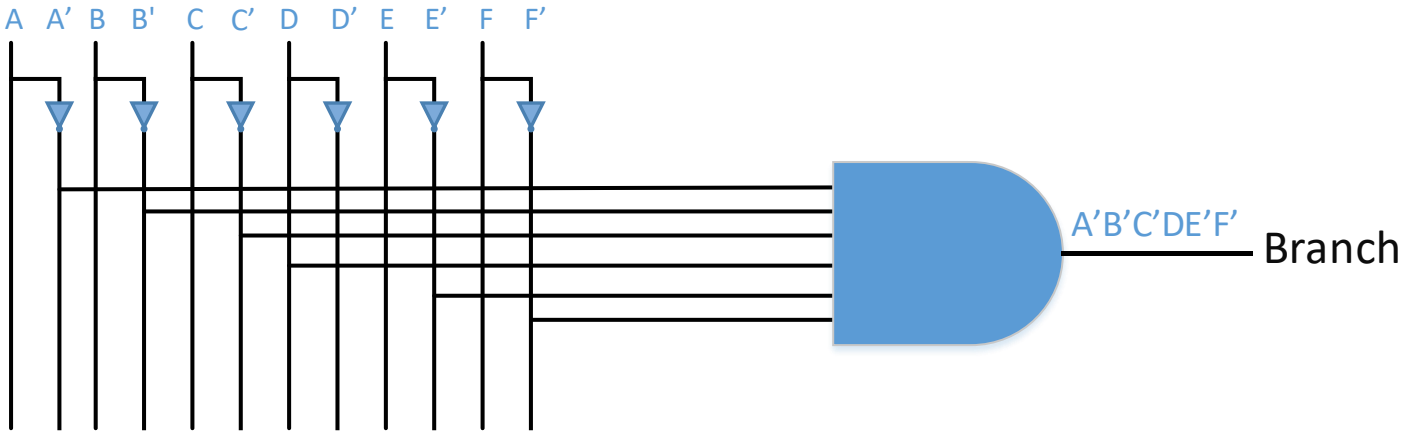
RegDst = A’B’C’D’F’



Case 2: (Branch)

ABC / DEF	000	001	011	010	100	101	111	110
000	0	0			0	0		
001								
011	0				0	0		
010	0	0						
100	1				0			
101								
111								
110								

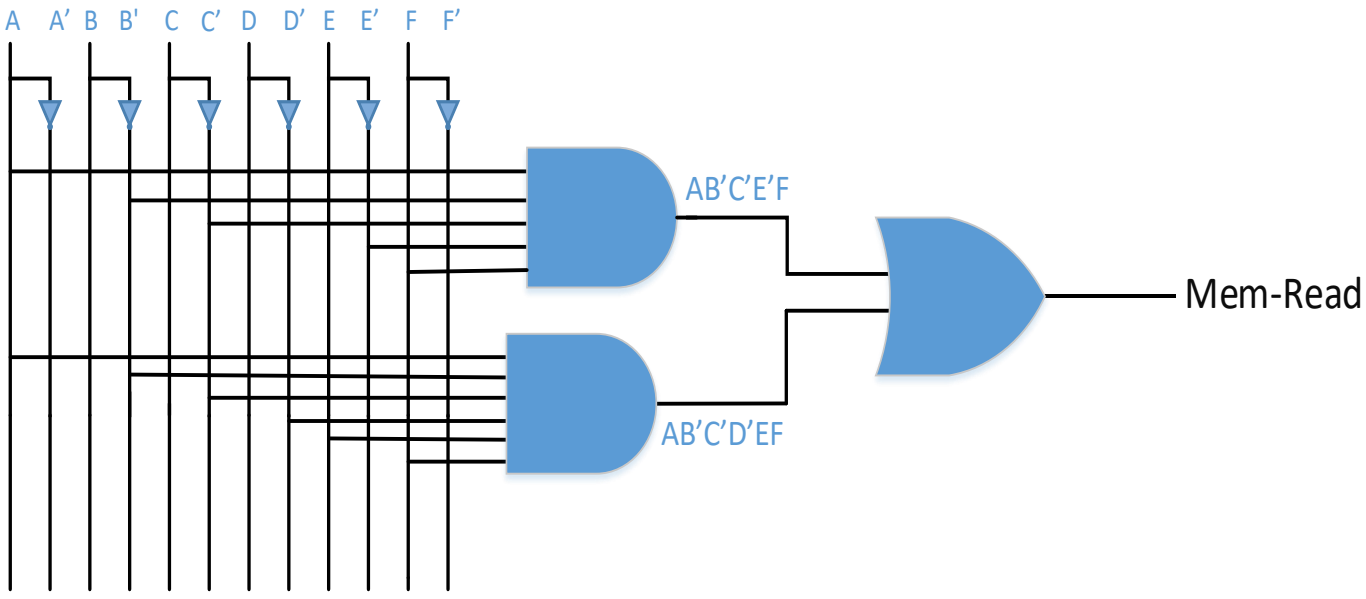
Branch =A’B’C’DE’F’



Case 3: (MemRead)

ABC / DEF	000	001	011	010	100	101	111	110
000	0	0			1	0		
001								
011	0				1	0		
010	0	0						
100	0				1			
101								
111								
110								

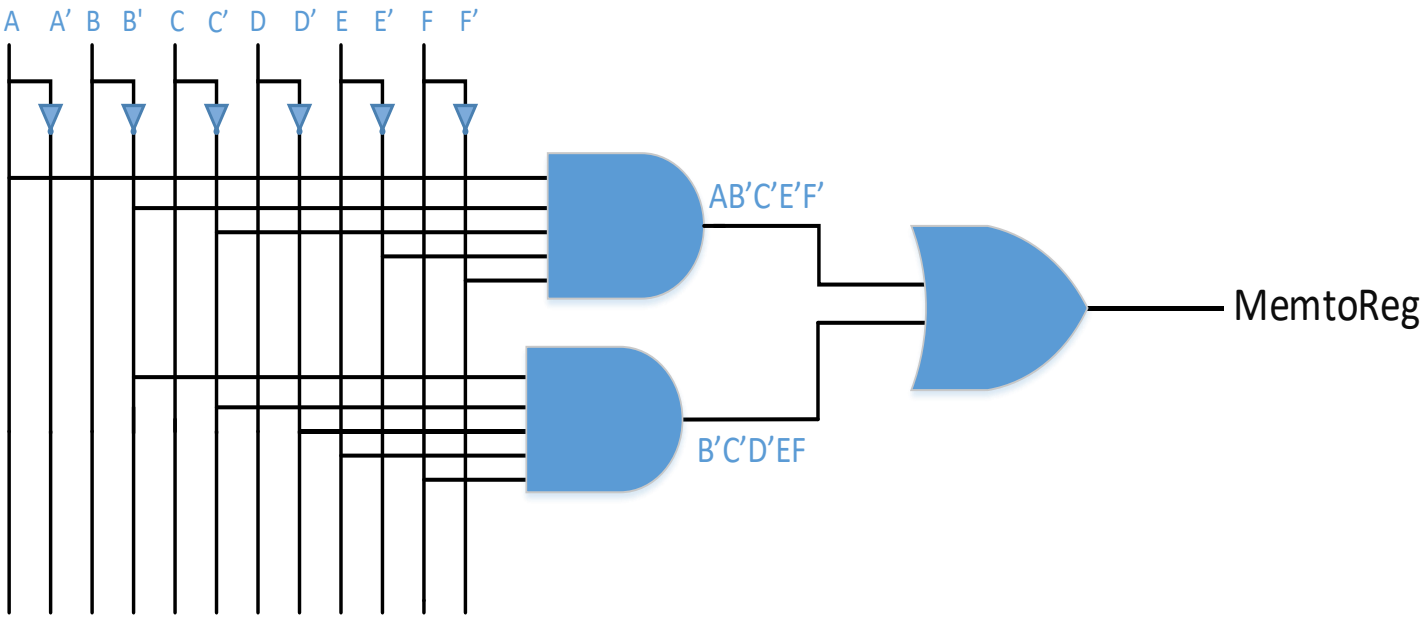
MemRead =AB’C’E’F + AB’C’D’EF



Case 4: (MemtoReg)

DEF \ ABC	000	001	011	010	100	101	111	110
000	0	0			1	X		
001								
011	X				1	X		
010	X	0						
100	X				1			
101								
111								
110								

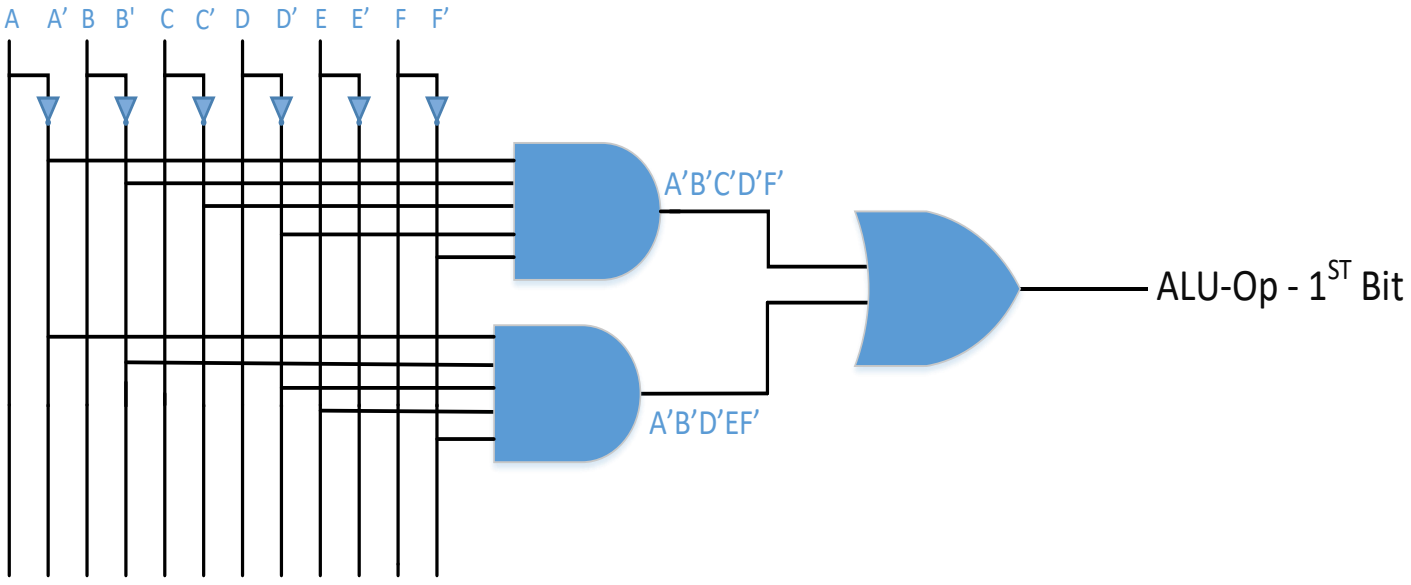
MemtoReg = AB'C'E'F' + B'C'D'EF



Case 5: (ALU-Op “First bit ”)

ABC / DEF	000	001	011	010	100	101	111	110
000	1	0			0	0		
001								
011	X				0	0		
010	X	1						
100	0				0			
101								
111								
110								

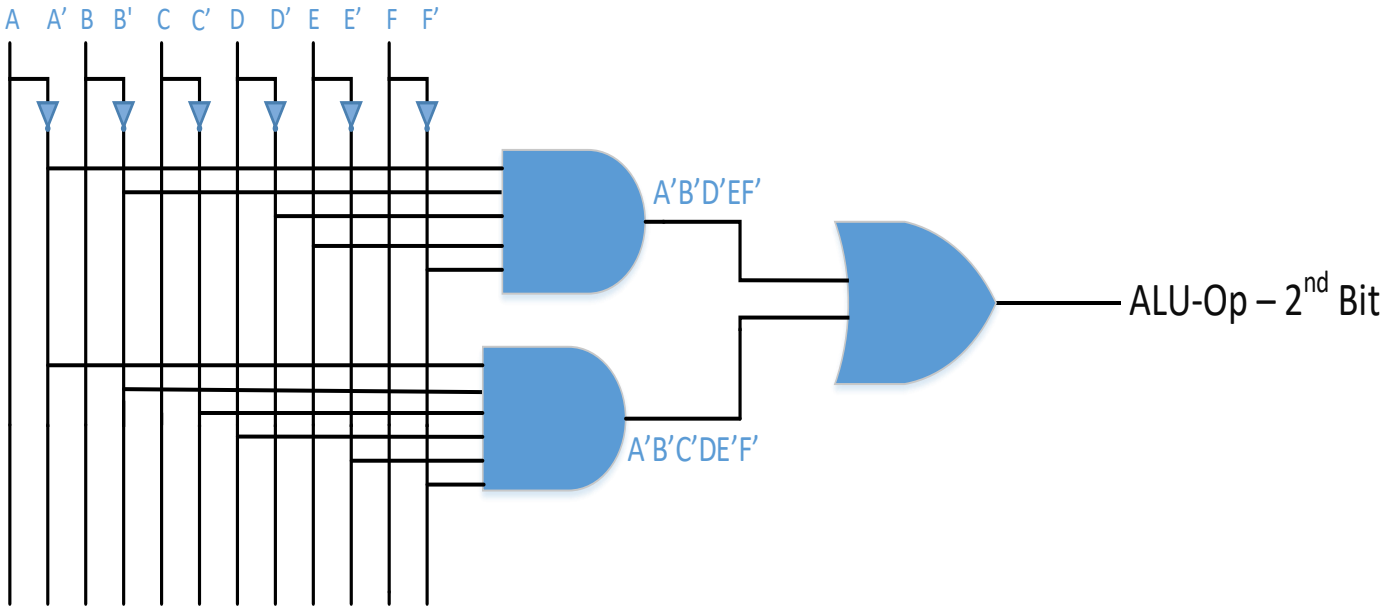
ALU-Op = A’B’C’D’F’ + A’B’D’EF’



Case 6: (ALU-Op “Second bit ”)

ABC / DEF	000	001	011	010	100	101	111	110
000	0	0			0	0		
001								
011	X				0	0		
010	X	1						
100	1				0			
101								
111								
110								

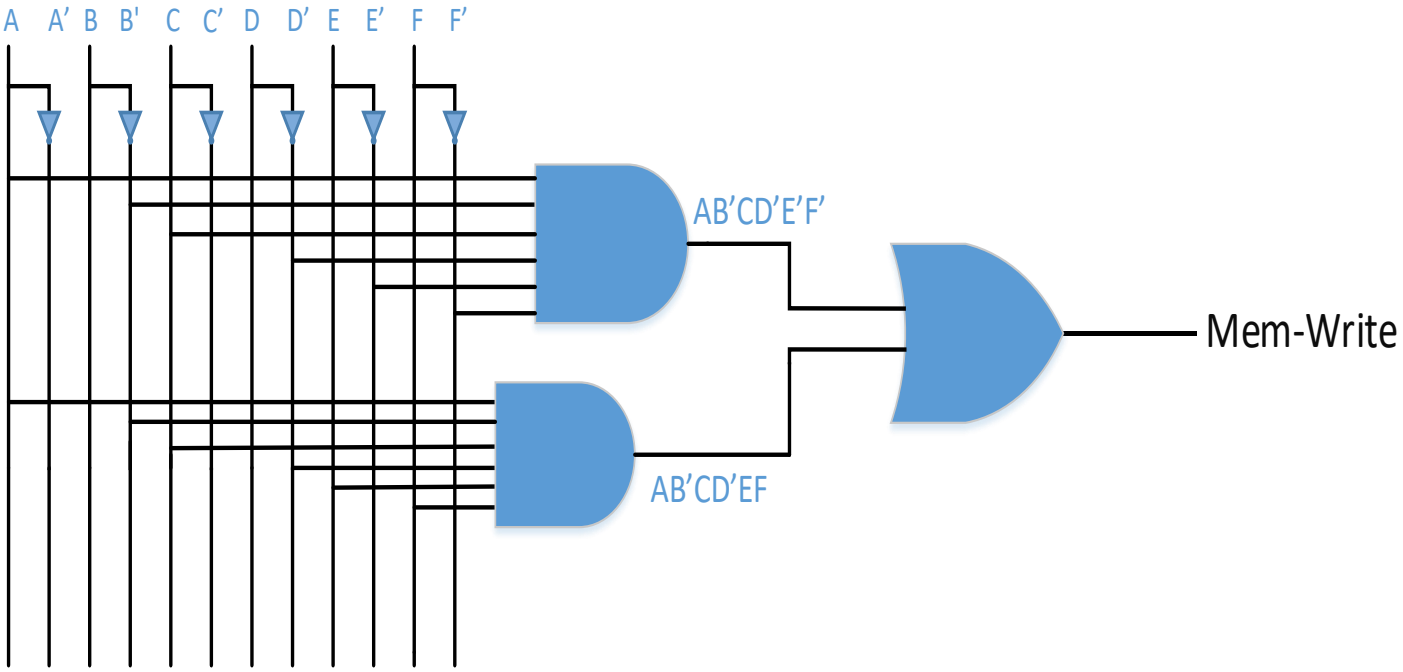
ALU-Op = A’B’D’EF’ + A’B’C’DE’F’



Case 7: (MemWrite)

DEF \ ABC	000	001	011	010	100	101	111	110
000	0	0			0	1		
001								
011	0				0	1		
010	0	0						
100	0				0			
101								
111								
110								

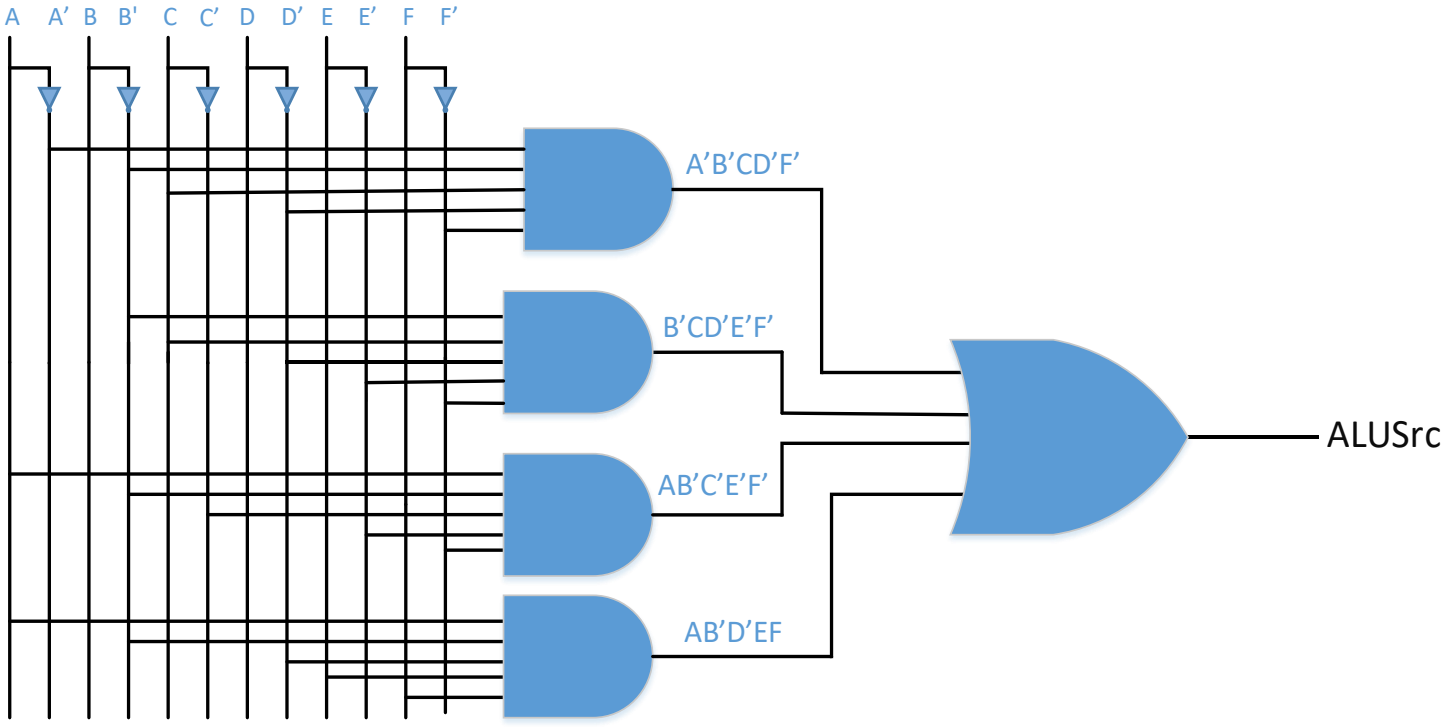
MemWrite = AB'CD'E'F' + AB'CD'EF



Case 8: (ALUSrc)

ABC / DEF	000	001	011	010	100	101	111	110
000	0	1			1	1		
001								
011	X				1	1		
010	X	1						
100	0				1			
101								
111								
110								

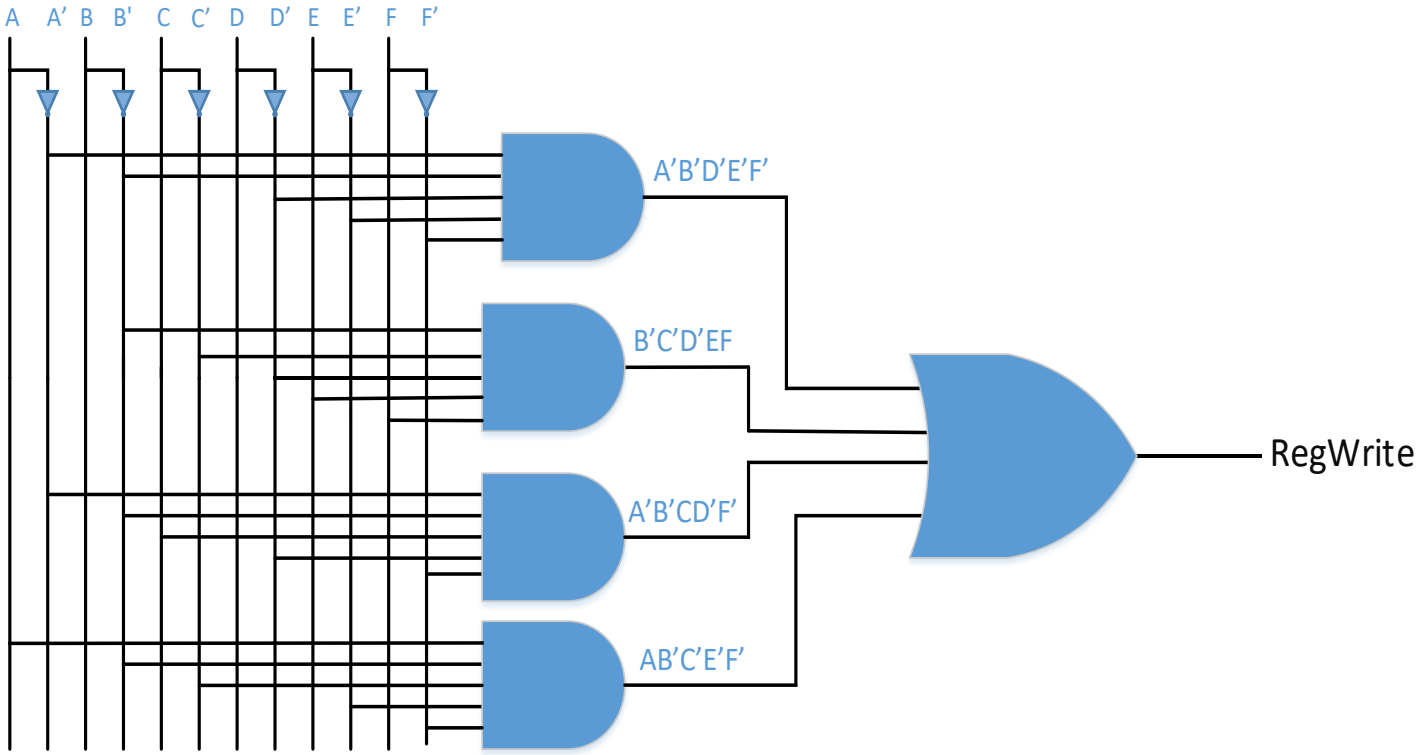
ALUSrc = A’B’CD’F’ + B’CD’E’F’ + AB’C’E’F’ + AB’D’EF



Case 9: (RegWrite)

ABC / DEF	000	001	011	010	100	101	111	110
000	1	1			1	0		
001								
011	1				1	0		
010	0	1						
100	0				1			
101								
111								
110								

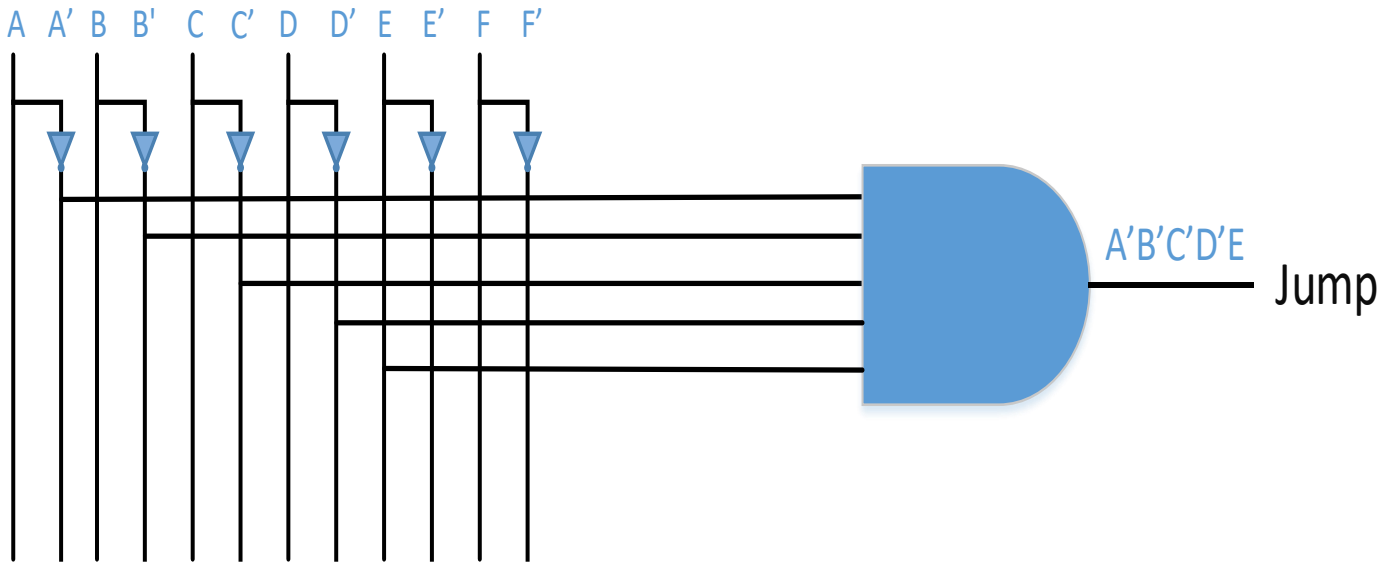
RegWrite=A’B’D’E’F’+B’C’D’EF+ A’B’CD’F’ + AB’C’E’F’



Case 10: (Jump)

DEF \ ABC	000	001	011	010	100	101	111	110
000	0	0			0	0		
001								
011	1				0	0		
010	1	0						
100	0				0			
101								
111								
110								

Jump = A’B’C’D’E



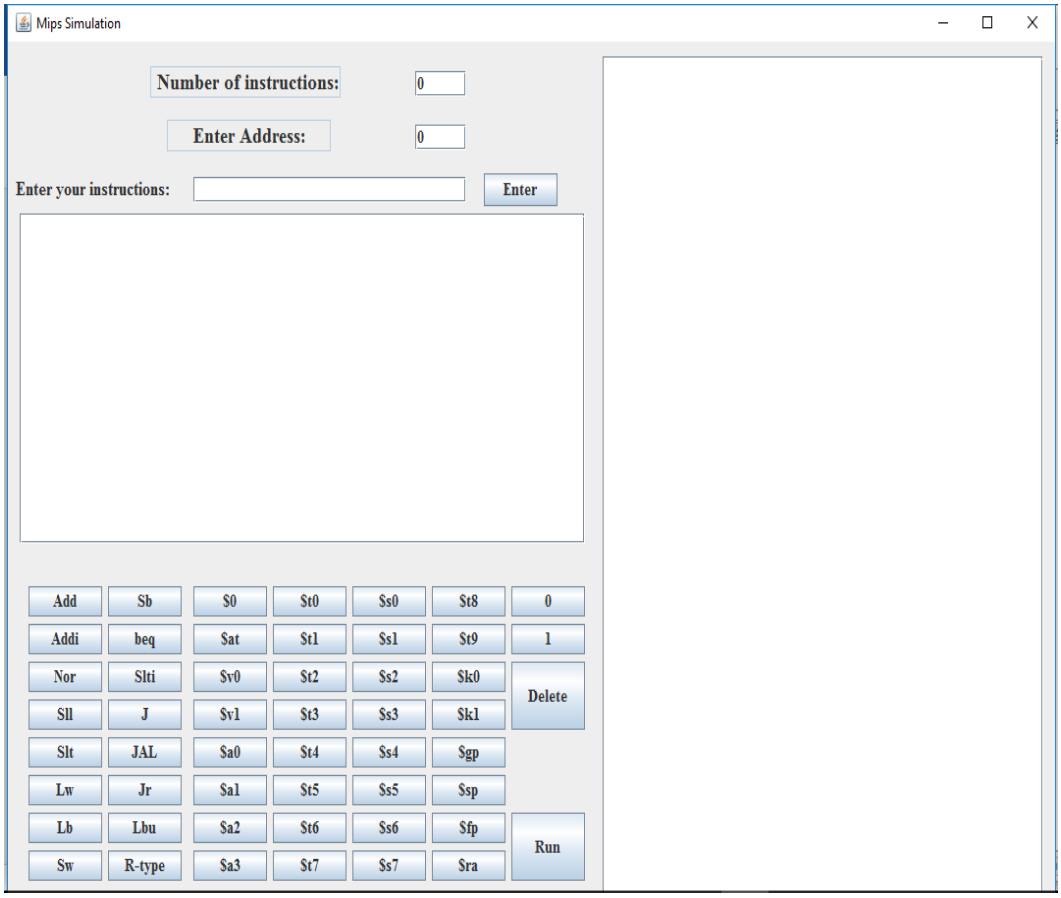
Assumptions:

We assumed:

The ALU control code of :

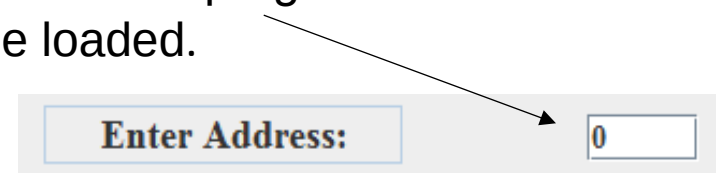
1) “NOR” = 0011
2) “SLL” = 0101
3) “Jump” & “Jal” = 0000

User Guide:

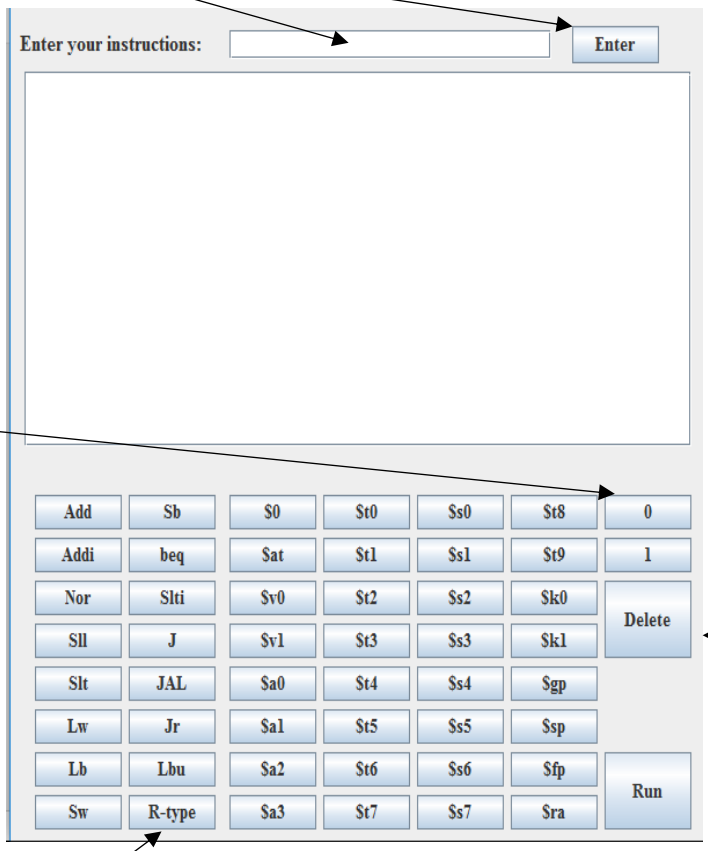


- Enter the number of instructions of the program which you want to build.

- Enter the address where the program's first instruction should be loaded.



- Enter your instructions one by one either by clicking on the buttons (Not as assembly language, the button will only write for you the code in binary) or writing in the text field using **binary language**, after each instruction press enter.

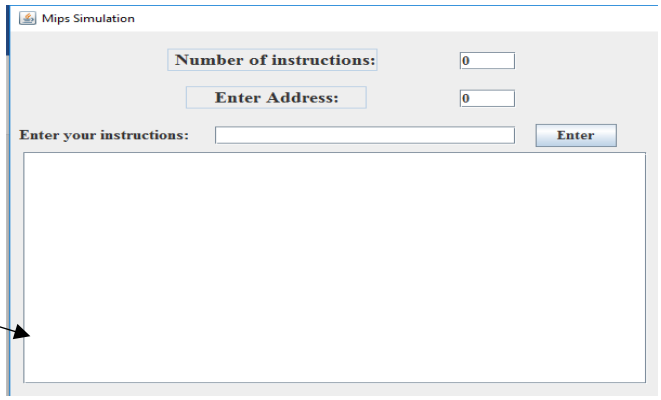


- For entering a constant number or address in your instruction, use 0 and 1 buttons

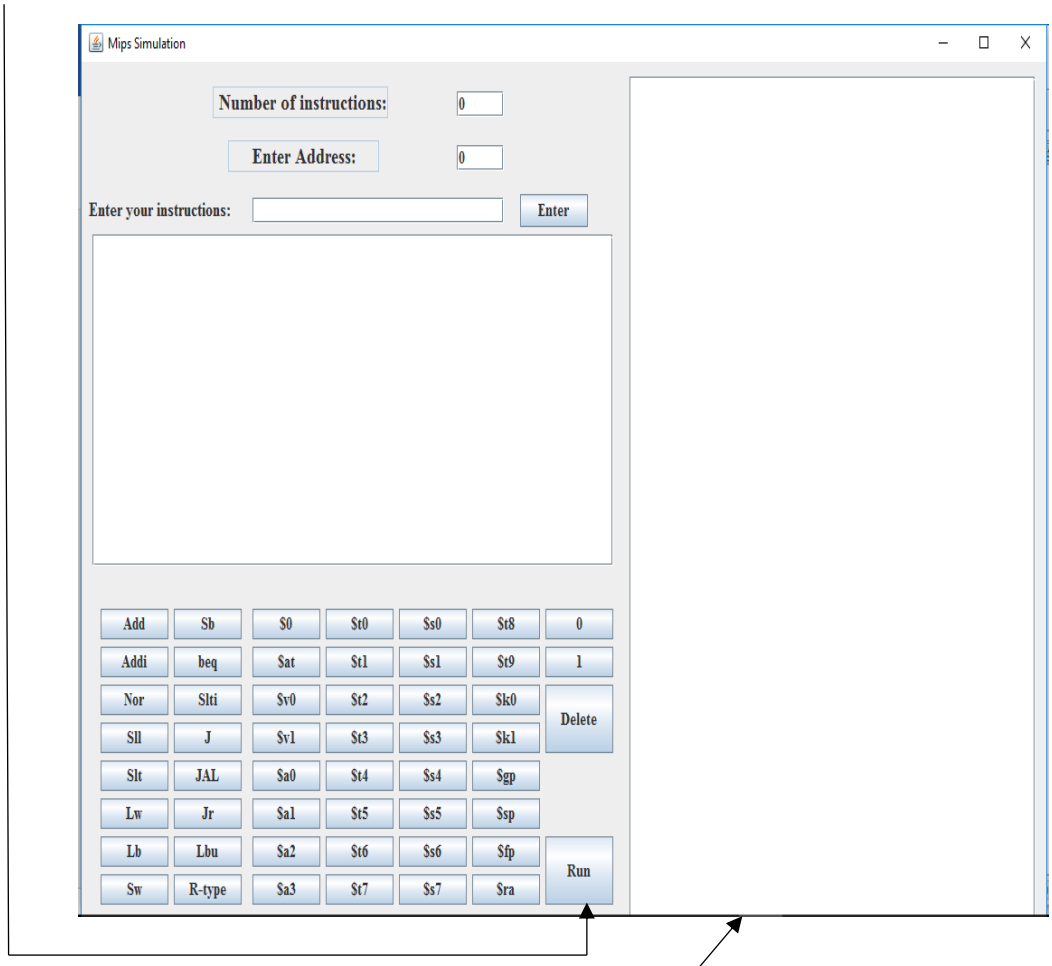
- Only for R-type instructions, **press on R-type button** first then enter your instruction.

- If you want to delete what has been written in the text field, press delete.

- All your instructions that you have entered will appear here.



- Press run button to execute your program.



- Here will appear the output.

List Of Programs

Program 1:

```
addi $t1,$0,1
loop1:
beq $t0,$t1,finish
sll $t3,$t0,2
add $s1,$s0,$t0
sb $s1,0($t3)
lb $t5 ,0($t3)
lbu $t5 ,0($t3)
addi $t0,$t0,1
  j loop1
finish:
add $s1,$s0,$t0
```

```
Clock Cycle 1 is :
Wires:
---------------------
PC output: 0
PC+4 adder output : 4
Instruction Memory Output: 00100000000010010000000000000001
Opcode field of instruction: 8
Rs field of instruction: 0
Rt field of instruction: 9
Rd field of instruction: 0
RegDestination Mux output: 9
Offset field of instruction: 0000000000000001
Sign-extender Output: 00000000000000000000000000000001
Shift-2 output: 00000000000000000000000000000100
Target address adder: 4
Function code of instruction: 1
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 1.0
ALU output: 1.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 1.0
PC input: 4
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 0
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 00
MemWrite: 0
ALUSrc: 1
RegWrite: 1
ALU control output: 0010
-----------------------------------------------------
```

```
-----------------------------------------------------
Clock Cycle 2 is :
Wires:
--------------------
PC output: 4
PC+4 adder output : 8
Instruction Memory Output: 00010001000010010000000000000111
Opcode field of instruction: 4
Rs field of instruction: 8
Rt field of instruction: 9
Rd field of instruction: 0
RegDestination Mux output: 0
Offset field of instruction: 0000000000000111
Sign-extender Output: 00000000000000000000000000000111
Shift-2 output: 00000000000000000000000000011100
Target address adder: 8
Function code of instruction: 7
Read data 1: 0.0
Read data 2: 1.0
ALU second input: 1.0
ALU output: -1.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: -1.0
PC input: 8
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
--------------------
RegDest: x
Branch: 1
MemRead: 0
MemtoReg: x
ALUOp: 01
MemWrite: 0
ALUSrc: 0
RegWrite: 0
ALU control output: 0110
-----------------------------------------------------
```

```
Clock Cycle 3 is :
Wires:
---------------------
PC output: 8
PC+4 adder output : 12
Instruction Memory Output: 00000001000000000101100010000000
Opcode field of instruction: 0
Rs field of instruction: 8
Rt field of instruction: 0
Rd field of instruction: 11
RegDestination Mux output: 11
Offset field of instruction: 0101100010000000
Sign-extender Output: 00000000000000000101100010000000
Shift-2 output: 00000000000000010110001000000000
Target address adder: 12
Function code of instruction: 0
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 0.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 0.0
PC input: 12
AND gate output: 0
-------------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 1
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 10
MemWrite: 0
ALUSrc: 0
RegWrite: 1
ALU control output: 0101
-------------------------------------------------------
```

```
Clock Cycle 4 is :
Wires:
--------------------
PC output: 12
PC+4 adder output : 16
Instruction Memory Output: 00000010000010001000100000100000
Opcode field of instruction: 0
Rs field of instruction: 16
Rt field of instruction: 8
Rd field of instruction: 17
RegDestination Mux output: 17
Offset field of instruction: 1000100000100000
Sign-extender Output: 11111111111111111000100000100000
Shift-2 output: 11111111111111100010000010000000
Target address adder: 16
Function code of instruction: 32
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 0.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 0.0
PC input: 16
AND gate output: 0
--------------------------------------------------
Control Unit Signals:
--------------------
RegDest: 1
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 10
MemWrite: 0
ALUSrc: 0
RegWrite: 1
ALU control output: 0010
--------------------------------------------------
```

```
Clock Cycle 5 is :
Wires:
---------------------
PC output: 16
PC+4 adder output : 20
Instruction Memory Output: 10100010001010010000000000000000
Opcode field of instruction: 40
Rs field of instruction: 17
Rt field of instruction: 9
Rd field of instruction: 0
RegDestination Mux output: 0
Offset field of instruction: 0000000000000000
Sign-extender Output: 00000000000000000000000000000000
Shift-2 output: 00000000000000000000000000000000
Target address adder: 20
Function code of instruction: 0
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 0.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 0.0
PC input: 20
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
---------------------
RegDest: x
Branch: 0
MemRead: 0
MemtoReg: x
ALUOp: 00
MemWrite: 1
ALUSrc: 1
RegWrite: 0
ALU control output: 0010
-----------------------------------------------------

Memory:
----------------
Address: 0
Value: 1
-----------------------------------------------------
```

```
Clock Cycle 6 is :
Value stored 00000001
Wires:
---------------------
PC output: 20
PC+4 adder output : 24
Instruction Memory Output: 10000010001011010000000000000000
Opcode field of instruction: 32
Rs field of instruction: 17
Rt field of instruction: 13
Rd field of instruction: 0
RegDestination Mux output: 13
Offset field of instruction: 0000000000000000
Sign-extender Output: 00000000000000000000000000000000
Shift-2 output: 00000000000000000000000000000000
Target address adder: 24
Function code of instruction: 0
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 0.0
Zero flag:0.0
Data memory output: 1.0
MemtoReg Mux output: 1.0
PC input: 24
AND gate output: 0
-------------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 0
Branch: 0
MemRead: 1
MemtoReg: 1
ALUOp: 00
MemWrite: 0
ALUSrc: 1
RegWrite: 1
ALU control output: 0010
-------------------------------------------------------
```

```
Clock Cycle 7 is :
Wires:
---------------------
PC output: 24
PC+4 adder output : 28
Instruction Memory Output: 10010010001011010000000000000000
Opcode field of instruction: 36
Rs field of instruction: 17
Rt field of instruction: 13
Rd field of instruction: 0
RegDestination Mux output: 13
Offset field of instruction: 0000000000000000
Sign-extender Output: 00000000000000000000000000000000
Shift-2 output: 00000000000000000000000000000000
Target address adder: 28
Function code of instruction: 0
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 0.0
Zero flag:0.0
Data memory output: 1.0
MemtoReg Mux output: 1.0
PC input: 28
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 0
Branch: 0
MemRead: 1
MemtoReg: 1
ALUOp: 00
MemWrite: 0
ALUSrc: 1
RegWrite: 1
ALU control output: 0010
-----------------------------------------------------
```

```
Clock Cycle 8 is :
Wires:
---------------------
PC output: 28
PC+4 adder output : 32
Instruction Memory Output: 00100001000010000000000000000001
Opcode field of instruction: 8
Rs field of instruction: 8
Rt field of instruction: 8
Rd field of instruction: 0
RegDestination Mux output: 8
Offset field of instruction: 0000000000000001
Sign-extender Output: 00000000000000000000000000000001
Shift-2 output: 00000000000000000000000000000100
Target address adder: 32
Function code of instruction: 1
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 1.0
ALU output: 1.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 1.0
PC input: 32
AND gate output: 0
-------------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 0
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 00
MemWrite: 0
ALUSrc: 1
RegWrite: 1
ALU control output: 0010
-------------------------------------------------------
```

```
Clock Cycle 9 is :
Wires:
---------------------
PC output: 32
PC+4 adder output : 36
Instruction Memory Output: 00001000000000000000000000000001
Opcode field of instruction: 2
Rs field of instruction: 0
Rt field of instruction: 0
Rd field of instruction: 0
RegDestination Mux output: 0
Offset field of instruction: 0000000000000001
Sign-extender Output: 00000000000000000000000000000001
Shift-2 output: 00000000000000000000000000000100
Target address adder: 4
Function code of instruction: 1
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 1.0
ALU output: 1.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 1.0
PC input: 4
AND gate output: 0
-------------------------------------------------------
Control Unit Signals:
---------------------
RegDest: x
Branch: 0
MemRead: 0
MemtoReg: x
ALUOp: xx
MemWrite: 0
ALUSrc: x
RegWrite: 0
ALU control output: 0000
-------------------------------------------------------
```

```
Clock Cycle 10 is :
Wires:
---------------------
PC output: 4
PC+4 adder output : 8
Instruction Memory Output: 00010001000010010000000000000111
Opcode field of instruction: 4
Rs field of instruction: 8
Rt field of instruction: 9
Rd field of instruction: 0
RegDestination Mux output: 0
Offset field of instruction: 0000000000000111
Sign-extender Output: 00000000000000000000000000000111
Shift-2 output: 00000000000000000000000000011100
Target address adder: 36
Function code of instruction: 7
Read data 1: 1.0
Read data 2: 1.0
ALU second input: 1.0
ALU output: 0.0
Zero flag:1.0
Data memory output: UNKNOWN
MemtoReg Mux output: 0.0
PC input: 36
AND gate output: 1
-------------------------------------------------------
Control Unit Signals:
---------------------
RegDest: x
Branch: 1
MemRead: 0
MemtoReg: x
ALUOp: 01
MemWrite: 0
ALUSrc: 0
RegWrite: 0
ALU control output: 0110
-------------------------------------------------------
```

```
Clock Cycle 11 is :
Wires:
---------------------
PC output: 36
PC+4 adder output : 40
Instruction Memory Output: 00000010000010001000100000100000
Opcode field of instruction: 0
Rs field of instruction: 16
Rt field of instruction: 8
Rd field of instruction: 17
RegDestination Mux output: 17
Offset field of instruction: 1000100000100000
Sign-extender Output: 11111111111111111000100000100000
Shift-2 output: 11111111111111100010000010000000
Target address adder: 40
Function code of instruction: 32
Read data 1: 0.0
Read data 2: 1.0
ALU second input: 1.0
ALU output: 1.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 1.0
PC input: 40
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 1
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 10
MemWrite: 0
ALUSrc: 0
RegWrite: 1
ALU control output: 0010
-----------------------------------------------------
Register File:
---------------------
$0 : 0.0
$at : 0.0
$v0 : 0.0
$v1 : 0.0
$a0 : 0.0
$a1 : 0.0
$a2 : 0.0
$a3 : 0.0
$t0 : 1.0
$t1 : 1.0
$t2 : 0.0
$t3 : 0.0
$t4 : 0.0
$t5 : 1.0
$t6 : 0.0
$t7 : 0.0
$s0 : 0.0
$s1 : 1.0
$s2 : 0.0
$s3 : 0.0
$s4 : 0.0
$s5 : 0.0
$s6 : 0.0
$s7 : 0.0
$t8 : 0.0
$t9 : 0.0
$k0 : 0.0
$k1 : 0.0
$gp : 0.0
$sp : 1000.0
$fp : 0.0
$ra : 0.0
-----------------------------------------------------
```

program 2:

```
funct:
 sw $s0,0($t1)
 lw $t5,0($t1)
 slti $s1,$s0,3
 beq $s2,$0,Exit
 addi $s1,$s1,-1
Exit:
  jr $ra
Main:
addi $s0,$0,2
  jal funct
add $t5,$s0,$t0
```

```
Clock Cycle 1 is :
Wires:
---------------------
PC output: 24
PC+4 adder output : 28
Instruction Memory Output: 00100010000100000000000000000010
Opcode field of instruction: 8
Rs field of instruction: 16
Rt field of instruction: 16
Rd field of instruction: 0
RegDestination Mux output: 16
Offset field of instruction: 0000000000000010
Sign-extender Output: 00000000000000000000000000000010
Shift-2 output: 00000000000000000000000000001000
Target address adder: 28
Function code of instruction: 2
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 2.0
ALU output: 2.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 2.0
PC input: 28
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 0
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 00
MemWrite: 0
ALUSrc: 1
RegWrite: 1
ALU control output: 0010
-----------------------------------------------------
```

```
Clock Cycle 2 is :
Wires:
---------------------
PC output: 28
PC+4 adder output : 32
Instruction Memory Output: 00001100000000000000000000000000
Opcode field of instruction: 3
Rs field of instruction: 0
Rt field of instruction: 0
Rd field of instruction: 0
RegDestination Mux output: 31
Offset field of instruction: 0000000000000000
Sign-extender Output: 00000000000000000000000000000000
Shift-2 output: 00000000000000000000000000000000
Target address adder: 0
Function code of instruction: 0
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 2.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 2.0
PC input: 0
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
---------------------
RegDest: x
Branch: 0
MemRead: 0
MemtoReg: x
ALUOp: xx
MemWrite: 0
ALUSrc: x
RegWrite: 1
ALU control output: 0000
-----------------------------------------------------
```

```
Clock Cycle 3 is :
Wires:
--------------------
PC output: 0
PC+4 adder output : 4
Instruction Memory Output: 10101101001100000000000000000000
Opcode field of instruction: 43
Rs field of instruction: 9
Rt field of instruction: 16
Rd field of instruction: 0
RegDestination Mux output: 0
Offset field of instruction: 0000000000000000
Sign-extender Output: 00000000000000000000000000000000
Shift-2 output: 00000000000000000000000000000000
Target address adder: 4
Function code of instruction: 0
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 0.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 2.0
PC input: 4
AND gate output: 0
----------------------------------------------------
Control Unit Signals:
--------------------
RegDest: x
Branch: 0
MemRead: 0
MemtoReg: x
ALUOp: 00
MemWrite: 1
ALUSrc: 1
RegWrite: 0
ALU control output: 0010
----------------------------------------------------
Memory:
---------------
Address: 0
Value: 0
Address: 1
Value: 0
Address: 2
Value: 0
Address: 3
Value: 2
----------------------------------------------------
```

```
Clock Cycle 4 is :
Wires:
---------------------
PC output: 4
PC+4 adder output : 8
Instruction Memory Output: 10001101001011010000000000000000
Opcode field of instruction: 35
Rs field of instruction: 9
Rt field of instruction: 13
Rd field of instruction: 0
RegDestination Mux output: 13
Offset field of instruction: 0000000000000000
Sign-extender Output: 00000000000000000000000000000000
Shift-2 output: 00000000000000000000000000000000
Target address adder: 8
Function code of instruction: 0
Read data 1: 0.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 0.0
Zero flag:0.0
Data memory output: 2.0
MemtoReg Mux output: 2.0
PC input: 8
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 0
Branch: 0
MemRead: 1
MemtoReg: 1
ALUOp: 00
MemWrite: 0
ALUSrc: 1
RegWrite: 1
ALU control output: 0010
-----------------------------------------------------
```

```
Clock Cycle 5 is :
Wires:
--------------------
PC output: 8
PC+4 adder output : 12
Instruction Memory Output: 00101010000100010000000000000011
Opcode field of instruction: 10
Rs field of instruction: 16
Rt field of instruction: 17
Rd field of instruction: 0
RegDestination Mux output: 17
Offset field of instruction: 0000000000000011
Sign-extender Output: 00000000000000000000000000000011
Shift-2 output: 00000000000000000000000000001100
Target address adder: 12
Function code of instruction: 3
Read data 1: 2.0
Read data 2: 0.0
ALU second input: 3.0
ALU output: 1.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 1.0
PC input: 12
AND gate output: 0
------------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 0
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 11
MemWrite: 0
ALUSrc: 1
RegWrite: 1
ALU control output: 0111
------------------------------------------------------
```

```
PC output: 12
PC+4 adder output : 16
Instruction Memory Output: 00010010001000000000000000000001
Opcode field of instruction: 4
Rs field of instruction: 17
Rt field of instruction: 0
Rd field of instruction: 0
RegDestination Mux output: 0
Offset field of instruction: 0000000000000001
Sign-extender Output: 00000000000000000000000000000001
Shift-2 output: 00000000000000000000000000000100
Target address adder: 16
Function code of instruction: 1
Read data 1: 1.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 1.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 1.0
PC input: 16
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
---------------------
RegDest: x
Branch: 1
MemRead: 0
MemtoReg: x
ALUOp: 01
MemWrite: 0
ALUSrc: 0
RegWrite: 0
ALU control output: 0110
-----------------------------------------------------
```

```
Clock Cycle 7 is :
Wires:
--------------------
PC output: 16
PC+4 adder output : 20
Instruction Memory Output: 00100010001100011111111111111111
Opcode field of instruction: 8
Rs field of instruction: 17
Rt field of instruction: 17
Rd field of instruction: 31
RegDestination Mux output: 17
Offset field of instruction: 1111111111111111
Sign-extender Output: 11111111111111111111111111111111
Shift-2 output: 11111111111111111111111111111100
Target address adder: 20
Function code of instruction: 63
Read data 1: 1.0
Read data 2: 0.0
ALU second input: -1.0
ALU output: 0.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 0.0
PC input: 20
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
---------------------
RegDest: 0
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 00
MemWrite: 0
ALUSrc: 1
RegWrite: 1
ALU control output: 0010
-----------------------------------------------------
```

```
Clock Cycle 8 is :
Wires:
--------------------
PC output: 20
PC+4 adder output : 24
Instruction Memory Output: 00000011111000000000000000001000
Opcode field of instruction: 0
Rs field of instruction: 31
Rt field of instruction: 0
Rd field of instruction: 0
RegDestination Mux output: 0
Offset field of instruction: 0000000000001000
Sign-extender Output: 00000000000000000000000000001000
Shift-2 output: 00000000000000000000000000100000
Target address adder: 32
Function code of instruction: 8
Read data 1: 28.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 28.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 28.0
PC input: 32
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
--------------------
RegDest: 1
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 10
MemWrite: 0
ALUSrc: 0
RegWrite: 1
ALU control output: 0010
-----------------------------------------------------
```

```
Clock Cycle 9 is :
Wires:
----------------------
PC output: 32
PC+4 adder output : 36
Instruction Memory Output: 00000010000010001000100000100000
Opcode field of instruction: 0
Rs field of instruction: 16
Rt field of instruction: 8
Rd field of instruction: 17
RegDestination Mux output: 17
Offset field of instruction: 1000100000100000
Sign-extender Output: 11111111111111111000100000100000
Shift-2 output: 11111111111111100010000010000000
Target address adder: 36
Function code of instruction: 32
Read data 1: 2.0
Read data 2: 0.0
ALU second input: 0.0
ALU output: 2.0
Zero flag:0.0
Data memory output: UNKNOWN
MemtoReg Mux output: 2.0
PC input: 36
AND gate output: 0
-----------------------------------------------------
Control Unit Signals:
----------------------
RegDest: 1
Branch: 0
MemRead: 0
MemtoReg: 0
ALUOp: 10
MemWrite: 0
ALUSrc: 0
RegWrite: 1
ALU control output: 0010
-----------------------------------------------------
Register File:
----------------------
$0 : 0.0
$at : 0.0
$v0 : 0.0
$v1 : 0.0
$a0 : 0.0
$a1 : 0.0
$a2 : 0.0
$a3 : 0.0
$t0 : 0.0
$t1 : 0.0
$t2 : 0.0
$t3 : 0.0
$t4 : 0.0
$t5 : 2.0
$t6 : 0.0
$t7 : 0.0
$s0 : 2.0
$s1 : 2.0
$s2 : 0.0
$s3 : 0.0
$s4 : 0.0
$s5 : 0.0
$s6 : 0.0
$s7 : 0.0
$t8 : 0.0
$t9 : 0.0
$k0 : 0.0
$k1 : 0.0
$gp : 0.0
$sp : 1000.0
$fp : 0.0
$ra : 28.0
-----------------------------------------------------
```

Teamwork

The whole team would sit together and brainstorm ideas about each part of the code and about how we are going to implement such ideas. We divided the data-path into classes. We worked together class by class, each class we would think about how we will implement it, then after creating all classes we brainstormed how we will integrate them all together. Finally, when we connected all the classes together the code worked perfectly but there wasn't any bonus features. Then we thought about adding the GUI to make the program more user-friendly.