AIN SHAMS UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING AND SOFTWARE SYSTEMS PROGRAM

CSE116: COMPUTER ARCHITECTURE – FALL 2018

Submitted to:

Dr. Cherif Salama

Eng. Ahmed Fathy

Submitted by:

Moataz Khaled Zakaria	16p8244
Mohammed Ehab ElSaeed	16p8160
Karim Walid ElHammady	16p3090
Youssef Assem Mohammed	16p6064
Ahmed Sameh Shahin	16p6063

Contents

1. Description 2
2. Critically Important Notes 2
3. Simple Use Instructions 3
4. Implemented Bonus Features 7
5. Datapath 8
6. Assumptions 8
7. Truth table 9
8. Test Cases 9
9. Roles 11

List of Figures

Figure 1 Shows Adress Setting Dialog 3
Figure 2 Preloading Data Dialog 3
Figure 3 Shows Memory Tab 4
Figure 4 Sample Assembly code Written in Assembly Text Area 5
Figure 5 Run and Step by Step Buttons 5
Figure 6 Registers Tab 6
Figure 7 X-ray (Educational) MIPS Diagram 7
Figure 8 MIPS standard Data Path 8

List of Tables

Table 1 Truth Table of the Control Unit 9

1. Description

This project is a GUI MIPS simulator developed using Java Swing. It contains an Educational “X-Ray” of the MIPS Datapath which showcases the data through each wire in the data path. This Project was Implemented by imagining that each class represents an actual MIPS Datapath component, however, selected data was shared across classes, when needed, in disregard to actual physical data path implementation. This was needed to avoid creation of a whole class, for example, a shifting unit; instead, we implemented it inside the ALU with a message that was passed from the Control Class.

2. Critically Important Notes

1. Register zero is strictly written as “$00”.
2. Only a space after the instruction name is allowed ex. (“addi $t0,$t0,1”).
3. When writing a label there **should be** a space between the label’s name and the colon
   ex (“Label1 :”)
4. A label **should be on a single line** alone without any other instructions in the same line.
5. You can not store/load a word/byte to/from a memory location reserved for an instruction.
6. If you use Windows 10 or any other OS which scales apps on your screen, right click on your desktop, select Display settings and set the “scale and Layout” to 100%. The resolution of the screen should be 1920x1080 or higher for all components to show.
7. Please note that if a store word instruction which doesn't override all the 4 bytes it won't change the value of the first bit. For example,
   - addi $t0,$t0,55
   - addi $t1,$t1,55
   - sw $t1,0($00)
   - sb $t0,0($00)

3. Simple Use Instructions

1. Please specify the Memory's starting address from the "Set Starting Address" Button **before anything** (Default address = 0).

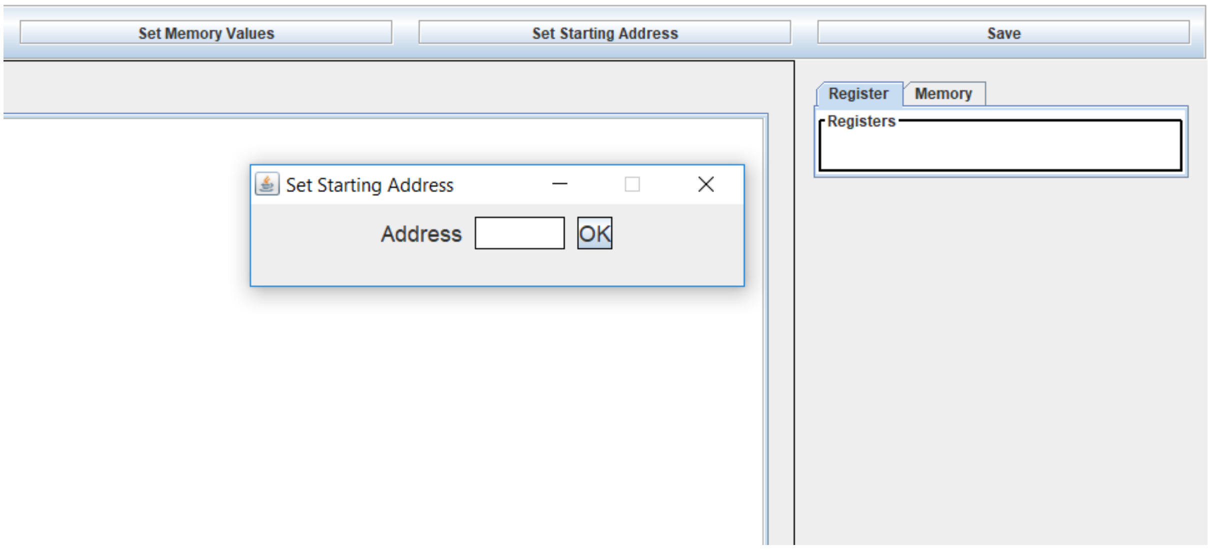


Figure 1 Shows Address Setting Dialog

2. To specify any memory values that should be preloaded before running, use the "Set Memory Values" Button.

Please Note that

- Data is loaded byte by byte.
- Please use the numbers displayed in the Memory tab.
- When you press OK the data is loaded; the frame doesn't close unless you press the frame's close button.

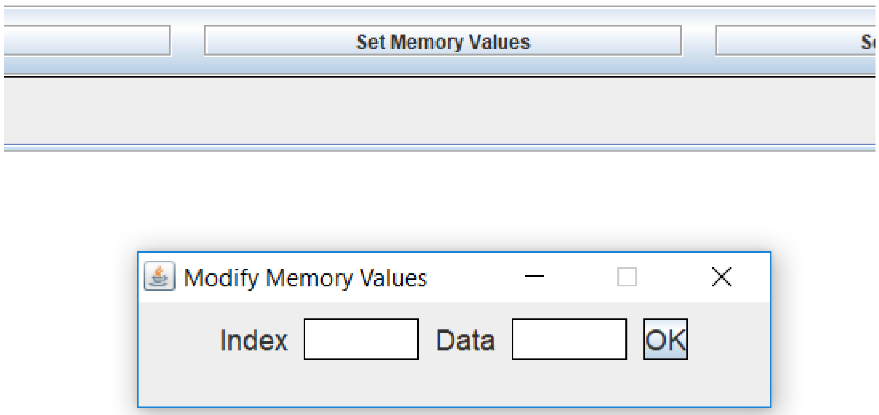
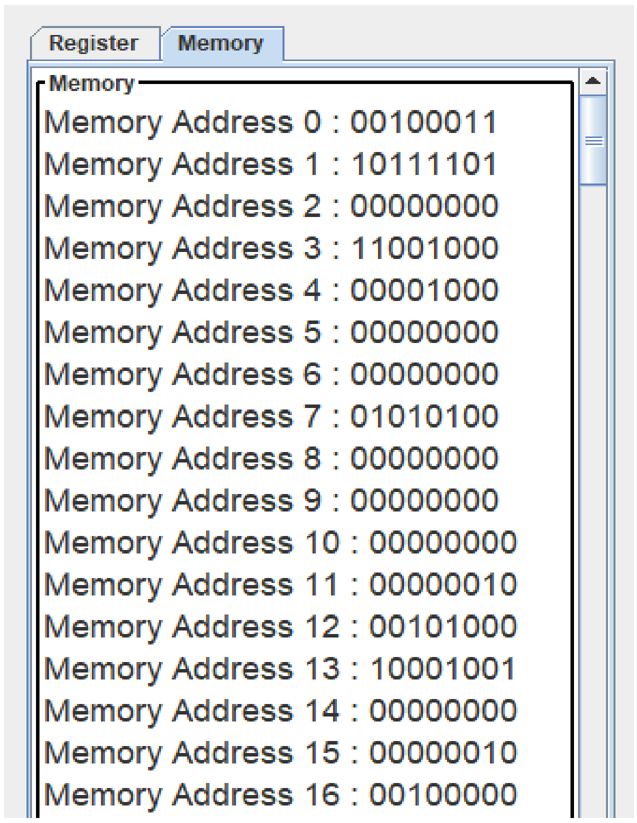


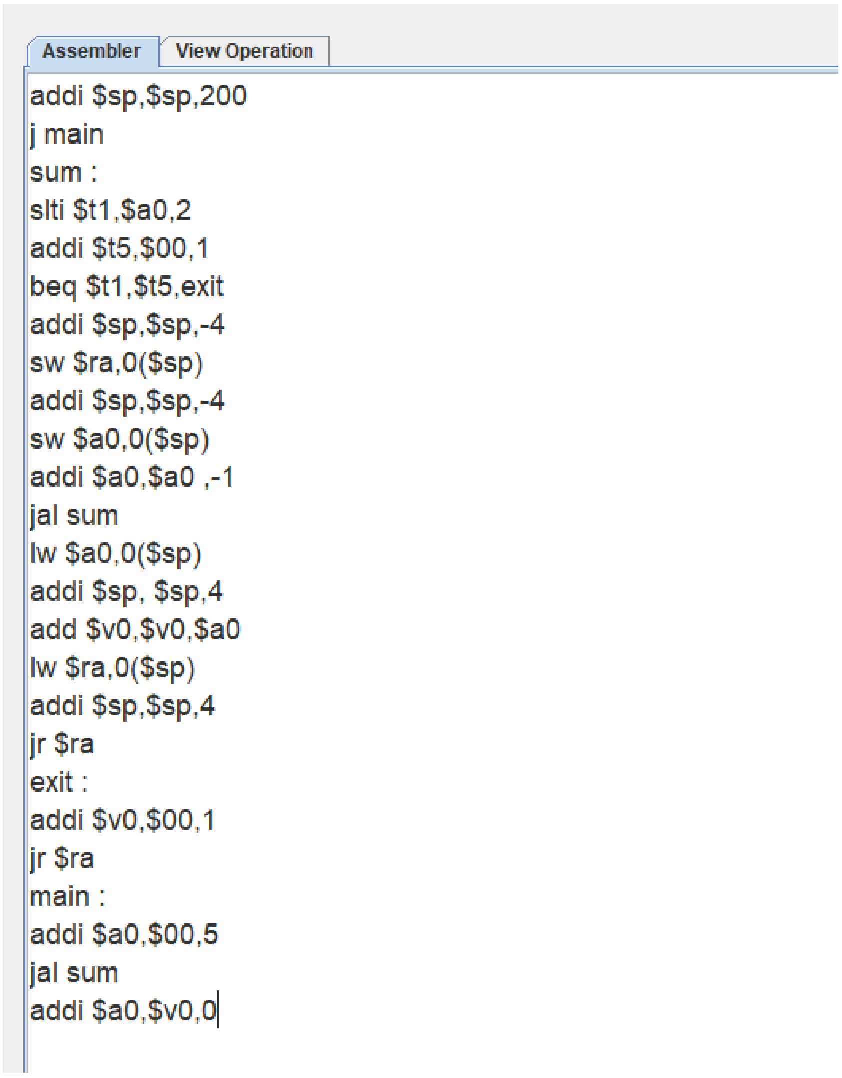
Figure 2 Preloading Data Dialog



```
Memory Address 0 : 00100011
Memory Address 1 : 10111101
Memory Address 2 : 00000000
Memory Address 3 : 11001000
Memory Address 4 : 00001000
Memory Address 5 : 00000000
Memory Address 6 : 00000000
Memory Address 7 : 01010100
Memory Address 8 : 00000000
Memory Address 9 : 00000000
Memory Address 10 : 00000000
Memory Address 11 : 00000010
Memory Address 12 : 00101000
Memory Address 13 : 10001001
Memory Address 14 : 00000000
Memory Address 15 : 00000010
Memory Address 16 : 00100000
```

Figure 3 Shows Memory Tab

3. Write your assembly code in the specified area then press "Save".



```
addi $sp,$sp,200
j main
sum :
slti $t1,$a0,2
addi $t5,$00,1
beq $t1,$t5,exit
addi $sp,$sp,-4
sw $ra,0($sp)
addi $sp,$sp,-4
sw $a0,0($sp)
addi $a0,$a0 ,-1
jal sum
lw $a0,0($sp)
addi $sp, $sp,4
add $v0,$v0,$a0
lw $ra,0($sp)
addi $sp,$sp,4
jr $ra
exit :
addi $v0,$00,1
jr $ra
main :
addi $a0,$00,5
jal sum
addi $a0,$v0,0
```

Figure 4 Sample Assembly code Written in Assembly Text Area

4. Press either step by step or Run Buttons.

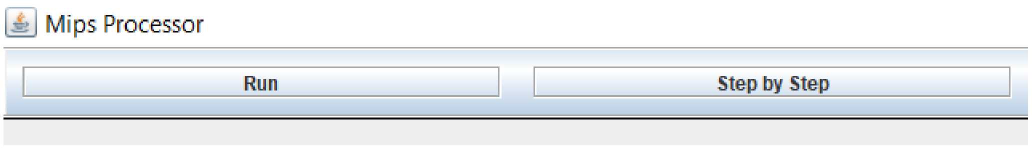
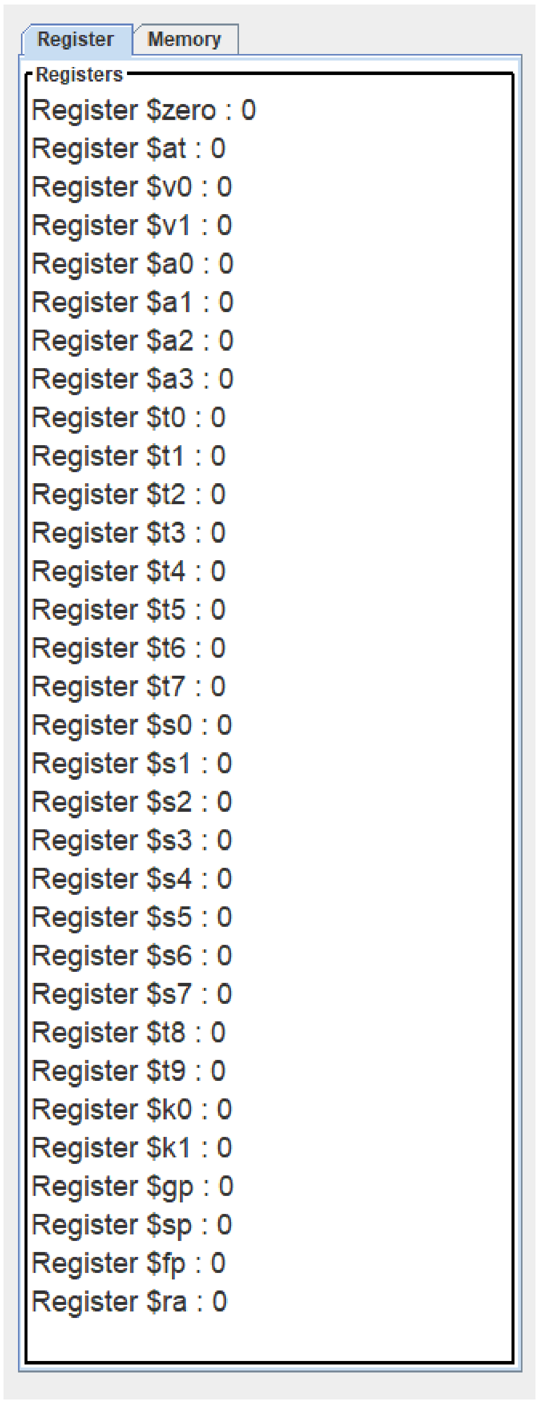


Figure 5 Run and Step by Step Buttons

5. Select the register tab to view the data stored in the registers



```
Register $zero : 0
Register $at : 0
Register $v0 : 0
Register $v1 : 0
Register $a0 : 0
Register $a1 : 0
Register $a2 : 0
Register $a3 : 0
Register $t0 : 0
Register $t1 : 0
Register $t2 : 0
Register $t3 : 0
Register $t4 : 0
Register $t5 : 0
Register $t6 : 0
Register $t7 : 0
Register $s0 : 0
Register $s1 : 0
Register $s2 : 0
Register $s3 : 0
Register $s4 : 0
Register $s5 : 0
Register $s6 : 0
Register $s7 : 0
Register $t8 : 0
Register $t9 : 0
Register $k0 : 0
Register $k1 : 0
Register $gp : 0
Register $sp : 0
Register $fp : 0
Register $ra : 0
```

Figure 6 Registers Tab

6. To view an "X-Ray" of the MIPS data path select the "View Operation".

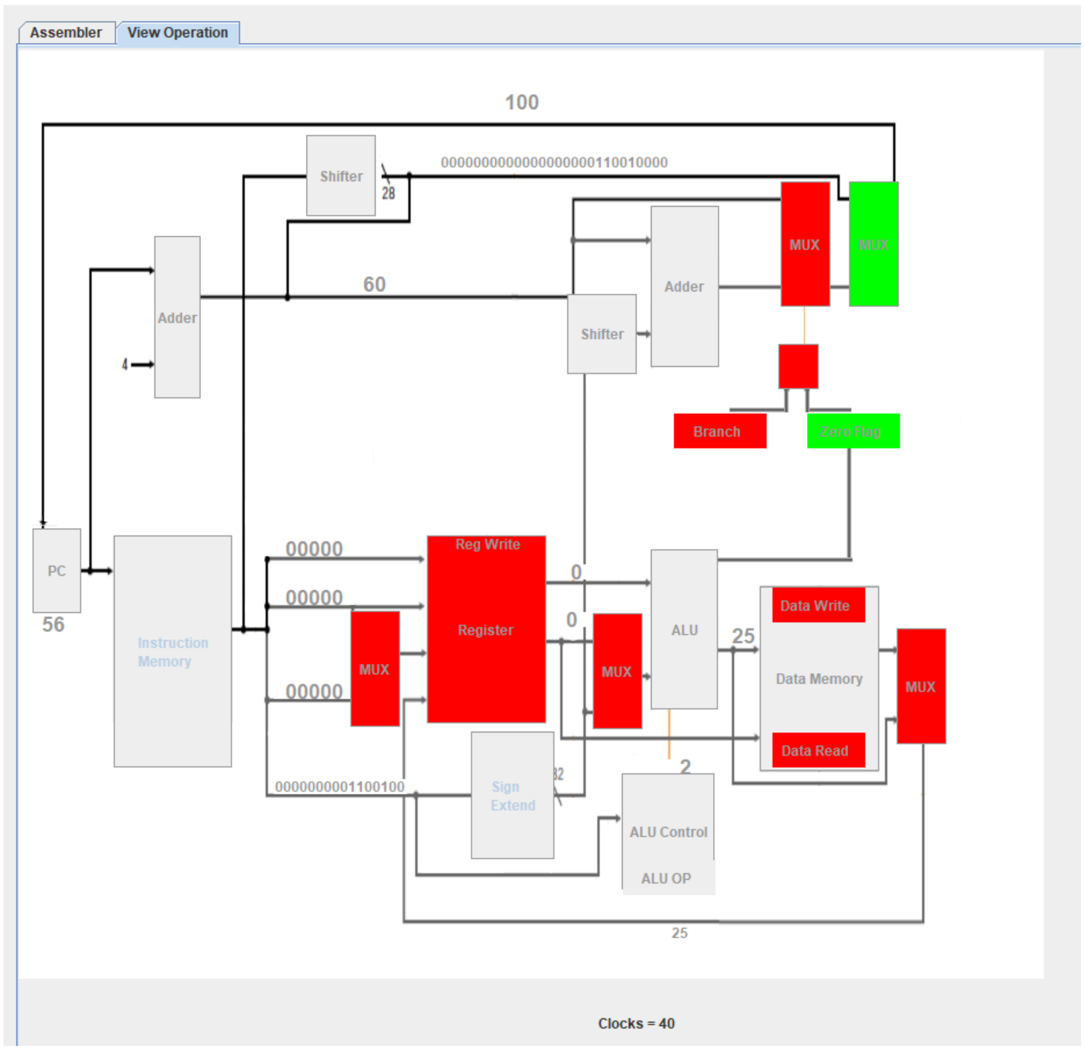


Figure 7 X-ray (Educational) MIPS Diagram

To test new code repeat steps 1 to 6.

4. Implemented Bonus Features

1. Project implemented as GUI.
2. Educational GUI.

5. Datapath

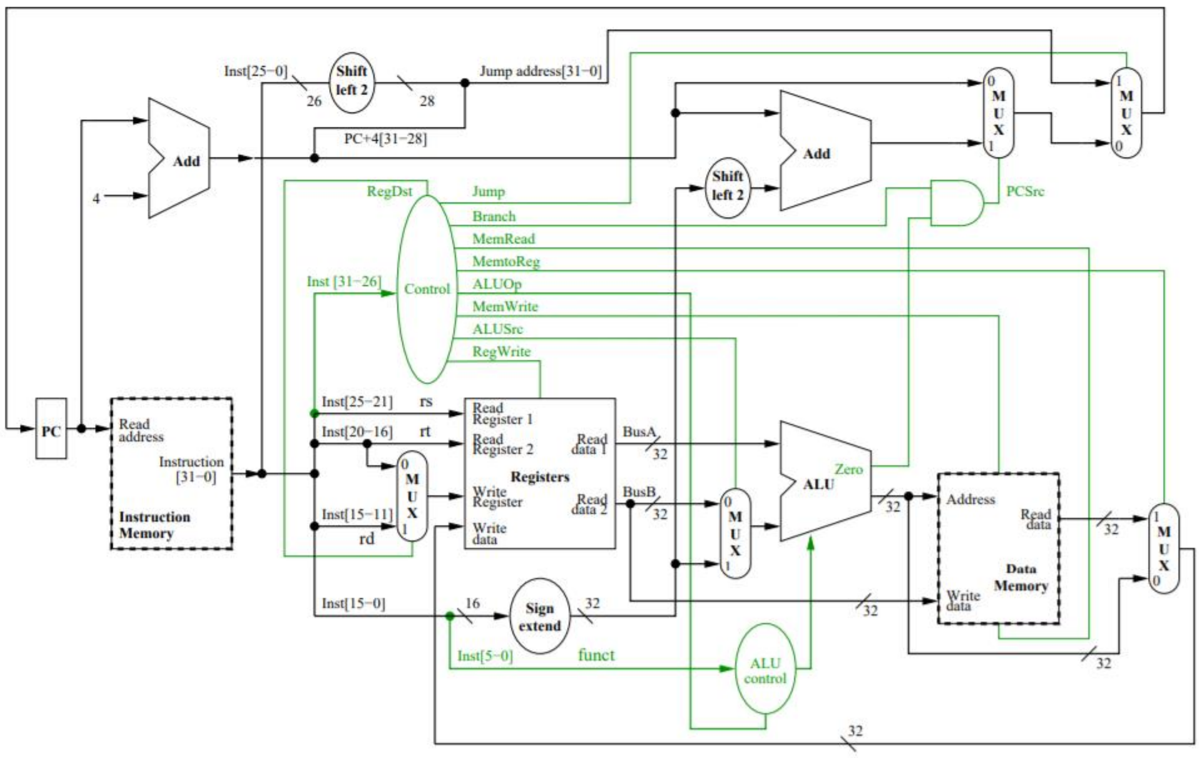


Figure 8 MIPS standard Data Path

6. Assumptions

We assumed that the logical operations like sll and nor are implemented in the ALU; furthermore, we assumed that the load/store operations like lb, lbu and sb are implemented in the data memory to provide more space for the Datapath wires information to be displayed and decrease complexity.

7. Truth table

	regDst	jump	branch	memRead	memToReg	ALUOp	memWrite	aluSrc	regWrite
add 32	1	0	0	0	0	10	0	0	1
addi 8	0	0	0	0	0	00	0	1	1
lw 35	0	0	0	1	1	00	0	1	1
lb 32	0	0	0	1	1	00	0	1	1
lbu 36	0	0	0	1	1	00	0	1	1
sw 43	x	0	0	0	x	00	1	1	0
sb 40	x	0	0	0	x	00	1	1	0
beq 4	x	0	1	0	x	01	0	0	0
j 2	x	1	x	0	x	xx	0	x	0
jal 3	x	1	x	0	x	xx	0	x	0
jr 8	x	1	x	0	x	xx	0	x	0
sll 0	1	0	0	0	0	10	0	0	1
nor 39	1	0	0	0	0	10	0	0	1
slt 42	1	0	0	0	0	10	0	0	1
slti 10	0	0	0	0	0	11	0	1	1

Table 1 Truth Table of the Control Unit

8. Test Cases

1)

```
addi $t6,$t6,5
sll $t7,$t6,1
nor $t8,$t6,$t7
addi $s0,$s0,-4
addi $s1,$s1,-2
addi $s5,$s5,5
jal loopfn
addi $sp,$sp,304
add $sp,$sp,$s0
sw $s0,0($sp)
sb $s1,-8($sp)
lw $s2,0($sp)
lb $s3,-8($sp)
```

```
    lbu $s4,-8($sp)
    j exit
    loopfn :
    slti $t0,$s5,3
    addi $t0,$t0,-1
    beq $t0,$00,return
    addi $s5,$s5,-1
    addi $t5,$00,3
    slt $t5,$s5,$t5
    j loopfn
    return :
    jr $ra
    exit :
```

2)

```
    addi $t0,$00,1
    addi $t1,$00,101
    loop :
    slt $t2,$t0,$t1
    beq $t2,$00,return
    add $t3,$t3,$t0
    addi $t0,$t0,1
    j loop
    return :
    sw $t3,44($s0)
    exit :
```

9. Roles

Classes were distributed equally at the beginning to ensure that everyone works and benefits from the project.

Mohammed Ehab ElSaeed: Assembler, InstructionMemory
Moataz Khaled Zakaria: ALU, Control
Youssef Assem Mohammed: MainGUI, Registers
Karim Walid ElHammady: DataMemory, OutputtingToGUI
Ahmed Sameh Shahin: XRayPanel