Lab 2: Implementing Decision Instructions in RISC V Assembly Language

Task 1:

Type the codes in Listings 3 and 4 in the assembler editor and then in the simulator look at their binary:

- The beq and bne instructions are stored in the SB format with their 32-bit binary. Look it up and try to find out what values their different fields contain.
- Looking at the binary formats, find out how do the labels are stored in the 32-bit instruction. Compare the label values stored inside the instruction to the code and try to guess how the assembler stores the jump addresses inside the instruction. (Hint: it uses a PC-relative scheme) Refer to following figures (from book page 119) to see encoding of instructions.

  Note: To test listing 4, ld will be replaced by lw and for computation of address, offset should be obtained by i*4 i.e., shift left by 2 instead of

For Listing 3

```
if (i == j)
  f = g + h;
else
  f = g - h;
// code after if/else goes here



//assuming that variables f to j are in registers x19-x23
bne x22, x23, Else
add x19, x20, x21
beq x0, x0, Exit  //un conditional jump
Else: sub x19, x20, x21

Exit: # the code after if/else goes here
```

Listing 3: If statement

BEQ

0000000 | 0 | 0 | 000 | 10000 | 1100011

BNE

0000000 | 10111 | 1011 | 001 | 11000 | 1100011

For listing 4

```
while (save[i] == k)
  i += 1;


// assuming i and k in x22 and x24, and the base address of Save in x25
Loop: slli x10, x22, 3 // Temp reg x10 = i * 8
    add x10, x10, x25 // x10 = address of save[i]
    ld x9, 0(x10) // Temp reg x9 = save[i]
    bne x9, x24, Exit // go to Exit if save[i] != k
    addi x22, x22, 1 // i = i + 1
    beq x0, x0, Loop // go to Loop
Exit:

```

Listing 4: while loop

BEQ

1111110 | 0 | 0 | 000 | 11001 | 1100011

BNE

0000000 | 24 | 9 | 001 | 11000 | 1100011

Task 2:

The switch statement is similar to if/else statements. Write the equivalent RISC-V assembly code for Listing 5.

Assume that the variables x, a, b & c are signed integers and stored in x20, x21, x22, x23 respectively. Make sure to first assign suitable values for b and c first.

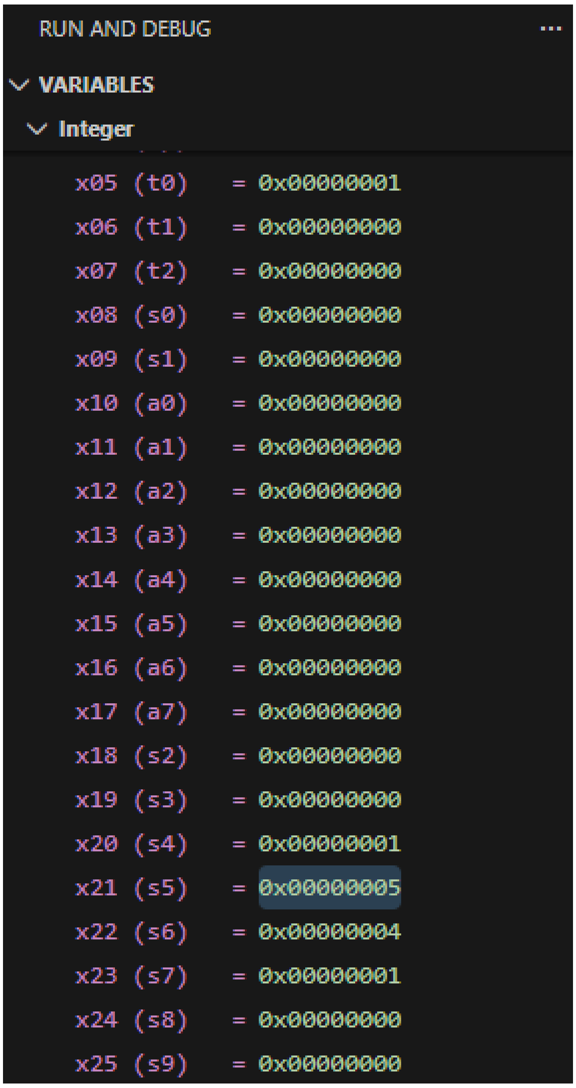
```c
switch (x) {
  case 1:
    a = b+c;
    break;
  case 2:
    a = b-c;
    break;
  case 3:
    a = b * 2;
    break;
  case 4:
    a = b / 2;
    break;
  default:
    a = 0;
}
```

Listing 5: task 1b.c

Code:

```
main:
    li x20, 1           #x=1
    li x22, 4          #b=4
    li x23, 1           #c=1

    li t0, 1
    beq x20, t0, Case1  #case1
    li t0, 2
    beq x20, t0, Case2  #case2
    li t0, 3
    beq x20, t0, Case3  #case3
    li t0, 4
    beq x20, t0, Case4  #case4
    beq x0, x0, Default

    Case1:
        add x21, x22, x23   #a=b+c
        beq x0, x0, Exit
    Case2:
        sub x21, x22, x23   #a=b-c
        beq x0, x0, Exit
    Case3:
        slli x21, x22, 1    #a=b*2(shift left by 1)
        beq x0, x0, Exit
    Case4:
        srai x21, x22, 1    #a=b/2(shift right by 1)
        beq x0, x0, Exit
    Default:
        li x21, 0           #a=0

Exit:
```

Output:


RUN AND DEBUG
VARIABLES
Integer
x05 (t0) = 0x00000001
x06 (t1) = 0x00000000
x07 (t2) = 0x00000000
x08 (s0) = 0x00000000
x09 (s1) = 0x00000000
x10 (a0) = 0x00000000
x11 (a1) = 0x00000000
x12 (a2) = 0x00000000
x13 (a3) = 0x00000000
x14 (a4) = 0x00000000
x15 (a5) = 0x00000000
x16 (a6) = 0x00000000
x17 (a7) = 0x00000000
x18 (s2) = 0x00000000
x19 (s3) = 0x00000000
x20 (s4) = 0x00000001
x21 (s5) = 0x00000005
x22 (s6) = 0x00000004
x23 (s7) = 0x00000001
x24 (s8) = 0x00000000
x25 (s9) = 0x00000000

Task 3:

Write the equivalent RISC-V assembly code for Listing 6. Assume that the variables i and sum are in x22 and x23, while the array a located at address 0x200 is of 4-byte integers.

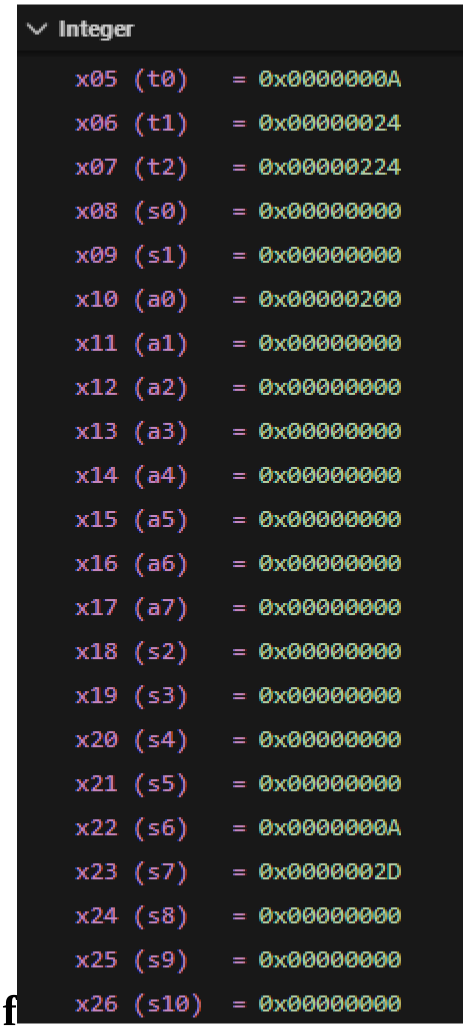
```
for(int i=0; i<10; i++)
  a[i] = i;

for(int i=0; i<10; i++)
  sum = sum+a[i];
```

Listing 6: Reduction Sum

Code:

```
.text
.globl main
main:
    li x10, 0x200       #base address
    li x22, 0           #i=0
    li x23, 0           #sum=0

    Loop1:
        li t0, 10
        bge x22, t0, Reset  #if i>=10 then next loop
        slli t1, x22, 2     #t1=i*4
        add t2, x10, t1     #t2=a[i]
        sw x22, 0(t2)       #store into x22 the value in 0(t2)
        addi x22, x22, 1    #increment x22
        beq x0, x0, Loop1

    Reset:
        li x22, 0           #i=0


    Loop2:
        li t0, 10
        bge x22, t0, Exit
        slli t1, x22, 2
        add t2, x10, t1
        lw t3, 0(t2)
        add x23, x23, t3    # sum = sum + a[i]
        addi x22, x22, 1
        beq x0, x0, Loop2

Exit:
```

Output:



Integer

x05 (t0)	= 0x0000000A
x06 (t1)	= 0x00000024
x07 (t2)	= 0x00000224
x08 (s0)	= 0x00000000
x09 (s1)	= 0x00000000
x10 (a0)	= 0x00000200
x11 (a1)	= 0x00000000
x12 (a2)	= 0x00000000
x13 (a3)	= 0x00000000
x14 (a4)	= 0x00000000
x15 (a5)	= 0x00000000
x16 (a6)	= 0x00000000
x17 (a7)	= 0x00000000
x18 (s2)	= 0x00000000
x19 (s3)	= 0x00000000
x20 (s4)	= 0x00000000
x21 (s5)	= 0x00000000
x22 (s6)	= 0x0000000A
x23 (s7)	= 0x0000002D
x24 (s8)	= 0x00000000
x25 (s9)	= 0x00000000
x26 (s10)	= 0x00000000

Address	+0	+1	+2	+3
0x00000230	00	00	00	00
0x0000022C	00	00	00	00
0x00000228	00	00	00	00
0x00000224	09	00	00	00
0x00000220	08	00	00	00
0x0000021C	07	00	00	00
0x00000218	06	00	00	00
0x00000214	05	00	00	00
0x00000210	04	00	00	00
0x0000020C	03	00	00	00
0x00000208	02	00	00	00
0x00000204	01	00	00	00
0x00000200	00	00	00	00

f

Task 4 (Challenge):

Translate the following C code to RISC-V assembly code. Use a minimum number of instructions. Assume that the values of a, b, i, and j are in registers x5, x6, x7, and x29, respectively. Also, assume that register x10 holds the base address of the array D

```

for(i=0; i<a; i++)
  for(j=0; j<b; j++)
    D[4*j] = i + j;
```

Listing 7: Nested Loops

Code:

```
.text
.globl main
main:
    li x7, 0    #i=0
    li x10, 0x200  #x10=base address of array
    li x5, 3       #a=3
    li x6, 4      #b=4

loop1:
    bge x7, x5, Exit   #if i>=a then exit
    li x29, 0       #j = 0

loop2:
    bge x29, x6, nexti   #if j>=b then end the inner loop and go to next i in outerloop(loop1)
    slli t3, x29, 4     #t3=j*16
    add t5, x10, t3    #t5=array[j]
    add t6, x7, x29     #t6=i+j
    sw t6, 0(t5)       #store t6 into memory
    addi x29, x29, 1   #j++
    beq x0, x0, loop2

nexti:
    addi x7, x7, 1    #i++
    beq x0, x0, loop1

Exit:
```

Output:

Integer

```
x00 (zero) = 0x00000000
x01 (ra)   = 0x0000003C
x02 (sp)   = 0x7FFFFFF0
x03 (gp)   = 0x10000000
x04 (tp)   = 0x00000000
x05 (t0)   = 0x00000003
x06 (t1)   = 0x00000004
x07 (t2)   = 0x00000003
x08 (s0)   = 0x00000000
x09 (s1)   = 0x00000000
x10 (a0)   = 0x00000200
x11 (a1)   = 0x00000000
x12 (a2)   = 0x00000000
x13 (a3)   = 0x00000000
x14 (a4)   = 0x00000000
x15 (a5)   = 0x00000000
x16 (a6)   = 0x00000000
x17 (a7)   = 0x00000000
x18 (s2)   = 0x00000000
x19 (s3)   = 0x00000000
x20 (s4)   = 0x00000000
x21 (s5)   = 0x00000000
x22 (s6)   = 0x00000000
x23 (s7)   = 0x00000000
x24 (s8)   = 0x00000000
x25 (s9)   = 0x00000000
x26 (s10)  = 0x00000000
x27 (s11)  = 0x00000000
x28 (t3)   = 0x00000030
x29 (t4)   = 0x00000004
x30 (t5)   = 0x00000230
```

Address	+0	+1	+2	+3
0x00000230	05	00	00	00
0x0000022C	00	00	00	00
0x00000228	00	00	00	00
0x00000224	00	00	00	00
0x00000220	04	00	00	00
0x0000021C	00	00	00	00
0x00000218	00	00	00	00
0x00000214	00	00	00	00
0x00000210	03	00	00	00
0x0000020C	00	00	00	00
0x00000208	00	00	00	00
0x00000204	00	00	00	00
0x00000200	02	00	00	00

Address: Up Down

Jump to: -- choose --

Display Format: Hex

Bytes per Row: 4

Assessment Rubric

Lab 2: Implementing Decision Instructions in RISC V Assembly Language

Name: Arqish Zaria, Samee Saqib	Student ID: az09714, ss10258	Section: T3

Points Distribution

	Task No.	LR 2 Code	LR 5 Results
In - Lab	Task 1	/0	/5
	Task 2	/10	/10
	Task 3	/15	/10
	Task 4	/20	/10
Total Points: 100		/45	/35
CLO Mapped		CLO2	

Affective Domain Rubric		Points	CLO Mapped
AR7	Report Submission	/20	CLO 2

CLO	Total Points	Points Obtained
2	100	
Total	**100**	

For description of different levels of the mapped rubrics, please refer to the Lab Evaluation Assessment Rubrics and Affective Domain Assessment Rubrics provided here.