Lab 2 - Implementing Decision Instructions in RISC V Assembly Language

Name: Aqsa Muneer, Maleeha Khan	Student ID: am10527, mk09991

Task 1:

Code:

```
bne x22, x23, else
add x19,x20,x21
beq x0, x0, exit

else:
sub x19, x20, x21

exit:
```

Results:

```
0x00000000      0x017B1663      bne x22 x23 12
0x00000004      0x015A09B3      add x19 x20 x21
0x00000008      0x00000463      beq x0 x0 8
0x0000000C      0x415A09B3      sub x19 x20 x21
```

For bne:

Machine code: 0000 0001 0111 1011 0001 0110 0110 0011

Opcode: 1100011, funct3: 001, rs1: 10110, rs2: 11011, imm: 0 0 00000 01100

For beq:

Machine Code: 0000 0000 0000 0000 0000 0100 0110 0011

Opcode: 1100011, funct3: 000, rs1 : 00000, rs2 : 00000, imm: 0 0 000000 01000

The opcode refers to the broad category of instruction, aka the conditional branch instructions. The funct3 code tells us the exact operation being performed. Rs1 and rs2 are the source registers and imm refers to the number of instructions that are jumped to reach the label that it refers if the conditional branch is true.

```
#x=x20, a=x21, b=x22, c=x23

li x20, 3
li x21, 6
li x22, 7
li x23, 8

li x1, 1
li x2, 2
li x3, 3
li x4, 4

beq x20, x1, case1
beq x20, x2, case2
beq x20, x3 case3
beq x20, x4, case4

li x21, 0

case1:
add x21, x22, x23
beq x0, x0, exit

case2:
sub x21, x22, x23
beq x0, x0, exit

case3:
slli, x21, x22, 1
beq x0, x0, exit
```

```
case2:
sub x21, x22, x23
beq x0, x0, exit

case3:
slli, x21, x22, 1
beq x0, x0, exit

case4:
srli, x21, x22, 1
beq x0, x0, exit

exit:
```

Code:

Results:

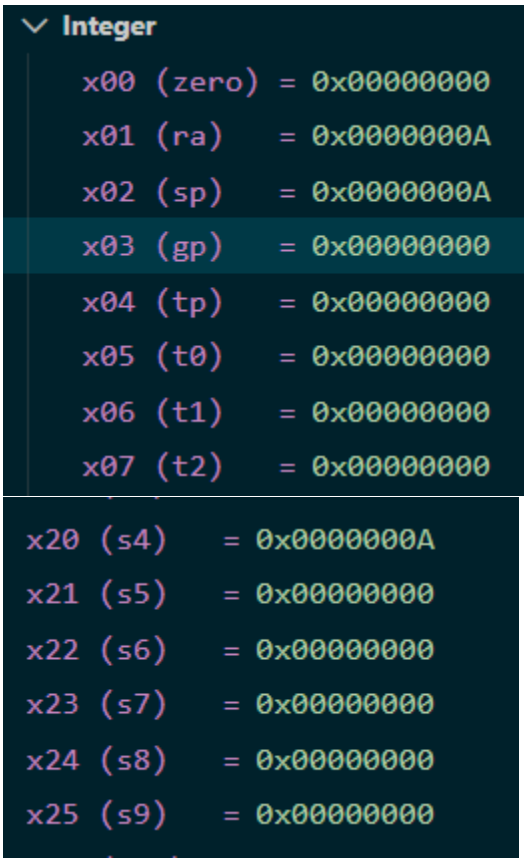
```
x01 (ra)    = 0x00000001
x02 (sp)    = 0x00000002
x03 (gp)    = 0x00000003
x04 (tp)    = 0x00000004
x19 (s3)    = 0x00000000
x20 (s4)    = 0x00000003
x21 (s5)    = 0x0000000E
x22 (s6)    = 0x00000007
x23 (s7)    = 0x00000008
x24 (s8)    = 0x00000000
x25 (s9)    = 0x00000000
```

Task 3:

Code:

```
li x1, 10 # comparison point

li x2, 0 # i

li x3, 0 # sum

li x10, 0x200 #base address

bne x2, x1, loop1 #if i !=10, go to loop1

loop1:
bge x2, x1, loop2
sw x2, 0(x10) #stored value in array
addi x10, x10, 4 #increased memory by 4 to go to next array index
addi x2, x2, 1 #i++
j loop1

li x20, 0
loop2:
bge x20, x1, exit
lw x4, 0(x10)
add x3, x3, x4
addi x10, x10, 4 #increased memory by 4 to go to next array index
addi x20, x20, 1 #i++
j loop2

exit:
```

Results:


Integer
x00 (zero) = 0x00000000
x01 (ra) = 0x0000000A
x02 (sp) = 0x0000000A
x03 (gp) = 0x00000000
x04 (tp) = 0x00000000
x05 (t0) = 0x00000000
x06 (t1) = 0x00000000
x07 (t2) = 0x00000000
x20 (s4) = 0x0000000A
x21 (s5) = 0x00000000
x22 (s6) = 0x00000000
x23 (s7) = 0x00000000
x24 (s8) = 0x00000000
x25 (s9) = 0x00000000

Task 4:

Code:

```
li   x7, 0 #i=0

outer_loop:

bge x7, x5, exit
li x29, 0

inner_loop:
bge  x29, x6, outer_inc
add  x5, x7, x29
slli x6, x5, 2
add  x6, x10, x6
sw   x5, 0(x6)
addi x29, x29, 1
j inner_loop

outer_inc:
addi x7, x7, 1
j outer_loop

exit:
```

Results:

```
x05 (t0)   = 0x00000003
x06 (t1)   = 0x00000002
x07 (t2)   = 0x00000000
x08 (s0)   = 0x00000000
x09 (s1)   = 0x00000000
x10 (a0)   = 0x10000000
x11 (a1)   = 0x00000000
x12 (a2)   = 0x00000000
x13 (a3)   = 0x00000000
x14 (a4)   = 0x00000000
```

```
x00 (zero) = 0x00000000
x01 (ra)   = 0x00000003
x02 (sp)   = 0x1000000C
x03 (gp)   = 0x10000000
x04 (tp)   = 0x00000000
x05 (t0)   = 0x00000003
x06 (t1)   = 0x00000002
x07 (t2)   = 0x00000003
x08 (s0)   = 0x00000000
x09 (s1)   = 0x00000000
```

```
x28 (t3)   = 0x00000000
x29 (t4)   = 0x00000002
```

Assessment Rubric
Lab 2 - Implementing Decision Instructions in RISC V Assembly Language

Name	Student ID:	Section:

Points Distribution:

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

Affective Domain Rubric		Points	CLO Mapped
AR7	**Report Submission & Git Upload**	/10 & /10	**CLO 2**

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