Sijo Rajigeorge
11708233

Assignment -1

Q.19) As the processor has an instruction length of 14 bit and 64 bit general purpose register, the address field will be of size 6 bit. so the opcode will be the remaining bit.

for two address Instruction
Address bit required = 2 x 6 = 12 bit
opcode bit remain = 14 - 12 = 2 bit
possible two addr opcode = $2^2$ = 4

for one address Instruction
Address bit required = 1 x 6 = 6 bit
opcode bits remain = 14 - 6 = 8 bit
possible one addr opcode = $2^8$ = 256

So for 3 two bit address instruction
$3 \times 2^6 \times 2^6 = 12288$
— 63 One address instruction
$63 \times 2^6 = 4032$
45 zero addr instruction
45
Σ total Combined = 16365

As it is less than given instruction length i.e $2^{14} = 16384$

It is possible to have all these instruction

15b for 3 - two address instruction

$= 3 \times 2^6 \times 2^6 = 12288$

— 63 - one address instruction

$= 63 \times 2^6 = 4160$

51 zero address instruction

$= 51$

So total combined instruction = 16483

as it is greater than the instruction length $2^{14} = 16384$

It is not possible to have this type of instructions together.

16) for this as there are already 3 - two address instruction and 24 zero address instruction, the remaining address instruction that can be of the maximum no of one instruction are

Total possible instruction $2^4 = 16384$

$16384 = x + (3 \times 2^6 \times 2^6) + (24 \times 2^6)$

$16384 = x + 3 \times 2^6 \times 2^6 + 24$

$16384 = x + 12288 + 24$

$x = 4072$

So 1 address Instruction = $\frac{4076}{26} = \frac{4072}{64}$

$= 63$

So we can have 63 one address instruction.

(Q21)

```
Sub $t0, $t0, $t1     // Sub b from a. store in $t0
addi $t2, $t1, 1      // Add 1 to b store in $t2
Sw $t2, 0($s0)        // store t2 in c[0]
lw $t3, 0($s0)        // load c[0]

add $t4, $t1, 1       // add 1 to b store in $t4
sll $t4, $t4, 2       // mul t4 by 4 to get offset
add $t4, $t4, $s0     // add base address
lw $t5, 0($t4)        // load c[b+1]

Sub $t6, $t3, $t5     // Sub c[b+1] from c[0]
add $t5, $t0, 2       // add 2 to a. store $t5
sll $t5, $t5, 2       // mul t7 by 4 to offset
add $t5, $t5, $s0     // add base address
Sw $t6, 0($t5)        // store c[a+2]
```

Q2.2 a)

```
ble $t1, $t2 if
sll $t3, $t1, 2
add $t3, $t3, $s0

sll $t4, $t1, 1
add $t5, $t0, $t4
sw $t5, 0($t3)
j exit

if : sll $t3 $t1, 2
     add $t3, $t3, $s0
     srl $t4, $t0, 2
     sw $t4, 0($t3)

exit:
```

Q2.2 b)

```
li $t2, 0
sll $t4, $t1, 2
add $t4, $t4, $s0
lw. $t5, 0($t4)
condition : blt $t3, $t0 loop
            j exit

loop : sll $t6, $t4, 1
       Sub $t0, $t5, $t6
```

```
        sll $t7, $t3, 2
        add $t7, $t7, $s0
        sw $t6, 0($t7)
add:    $t3, $t3, 1
        j condition
exit:
```

(22c)

```
        i: $t3, 0
loop:
        sll $t4, $t3, 2
        add $t5, $s0, $t4
        lw  $t6, 0($t5)
        ble $t6, $t3, end
        sll $t7, $t0, 2
        add $t8, [illegible] $s0, $t7
        lw  $t9, 0($t8)
        add $t10, $t9, $t4
        sw  $t10, 0($t5)
add:    $t3, $t3, 1
        j loop
end:
```

Q3.

2.3)
```
main: lw $a0, n
      jal fib
      move $t0, $v0
```

```
fib:  addi $sp $sp -8
      sw, $ra, 4($sp)
      sw $a0 ,0($sp)
      li $t1, 1
      ble $a0, $t1, fib_base_case
      addi $a0, $a0, -1
      jal fib
      move $t2, $v0
      lw $a0, 0($sp)
      addi $a0, $a0, -2
      jal fib
      move $t3, $v0
      mul $v0, $t2, $t3
      lw $ra, 4($sp)
      lw $a0, 0($sp)
      addi $sp, $sp, 8
      jr $ra
```

```
fib_base_case:
      move $v0, $a0
      lw $ra, 4($sp)
      lw $a0, 0($sp)
      addi $sp, $sp, 8
      jr $ra
```

Q8. Convert hex into binary

0010 0000 0000 1000 0000 0000 0000 0000
0001 0001 0000 1001 0000 0000 0000 0110
0000 0001 0000 0000 0101 0000 0100 0010
0000 0001 0000 0000 0101 1000 1000 0000
0000 0001 0111 0000 0101 1000 0010 0000
0010 1101 0110 1010 0000 0000 0000 0000
0010 0001 0000 1000 0000 0000 0000 0001
0000 1000 0000 0000 0001 0000 0000 0001

Identify the type of instruction from opcode

001000 — opcode 8.
000100 — opcode 4.
000000 — opcode 0
000000 — opcode 0.
000000 — opcode 0
101011 — opcode 43
001000 — opcode 8.
000010 — opcode 2

Expected based on opcode format

I 8 0 8 0
I 4 8 9 +6
R 0 8 0 10 1 2

R 0 8 0 11 2 0
R 0 11 16 11 0 32
I 43 11 10 +0
I 8 8 8 +1
J 2 4097

- Mips assembly code

0x00004000 addi $0 $8, 0
0x00004004 beq $8, $9, Exit
0x00004008 srl $10, $8, 1
0x0000400c sll $11, $8, 2
0x00004010 add $11, $11, $16
0x00004014 sw $10, 0($11)
0x00004018 addi $8, $8, 1
0x0000401c j 0x001001

- Mips Assembly

addi $0 $t0, 0
loop beq $t0, $t1, Exit
srl $t2, $t0, 1
sll $t3, $t0, 2
add $t2, $t3, $s0
sw $t2, 0($t3)
addi $t0, $t0, 1

j loop
Exit

— possible code

```
int i = 0;
while (i != n) {
        c[i] = i/2;
        i++;
}
```

or

```
int i = 0
while (i < n) {
        c[i] = (i < n) ? i : 0;
        i++;
}
```