ECE 432 - Assignment 2

Rohan Jhaveri
830962238

Q1 a)

MIPS Code / C-code

```
sll $t0, $s0, 2     # $t0 = f * 4
add $t0, $s6, $t0   # $t0 = &A[f]
sll $t1, $s1, 2     # $t1 = g * 4
add $t1, $s7, $t1   # $t1 = &B[g]
lw $s0, 0($t0)      # f = A[f]
addi $t2, $t0, 4    # $t2 = &A[f+1]
lw $t0, 0($t2)      # $t0 = A[f+1]
add $t0, $t0, $s0   # $t0 = A[f+1] + A[f]
sw $t0, 0($t1)      # B[g] = A[f+1] + A[f]
```

Hence, this MIPS code could be converted into the following C instruction.

B[g] = A[f+1] + A[f] //

b)

MIPS Code / C-code.

```
addi $t0, $s6, 4    # $t0 = &A[1]
add $t1, $s6, $0    # $t1 = &A[0]
sw $t1, 0($t0)      # &A[1] = &A[0]
lw $t0, 0($t0)      # $t0 = &A[0] = &A[0]
add $s0, $t1, $t0   # f = &A[0] + &A[0]
```

Hence, this MIPS code could be converted into the following C instruction.

f = 2 * [&A] //

Q2)

```
srl $t0, $t0, 11          # It makes the bits 31 to 26
                            of register $t0 to 0
                            and moves bits 11 to 16 to the
                            position 0 to 5

sll $t0, $t0, 26          # It shifts the bits 0 to 5
                            of register $t0 to bits
                            positions 26 to 31
ori $t2, $0, 0x03ff       # load $t2 with binary
                            value (0000001111111111)2
sll $t2, $t2, 16          # It shifts $t2 by 16 bits
ori $t2, $t2, 0xffff      # It performs logical OR
                            with register $t2 and immediate
                            constant value.
and $t1, $t1, $t2         # This and operation makes
                            bits 26 to 31 as zeros
                            of $t1 register and rest
                            of the bits as ones
or $t1, $t1, $t0          # This ~~makes~~ extracts the
                            required bits of $t0 in
                            31 to 26 bits of register $t1
                            keeping rest of the bits of $t1
                            the same.
```

Q3)

```
      andi  $t0, 0
loop: blt   $s0, $t0, end
      addi  $t0, $t0, 1
      andi  $t1, 0
loop1: blt  $s1, $t2, loop
      addi  $t2, $t0, -1
      add   $t3, $t2, $t1
      sll   $t4, $t1, 2
      add   $t4, $t4, $s2
      sw    $t3, $t4.
      addi  $t1, $t1, 1
      j     loop1
end:
```

Q 4a)

op	rs	rt	rd	shamt	funct
000000	10001	10000	100000	00000	100000
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

The given binary value represents an <u>R-type</u> instructions.

It's opcode field is 0
It's source register field is 17 i.e. rs is \$s1
It's 2nd source register field is 16 i.e. rt is \$s0
It's destination register field is 16 i.e. rd is \$s0
Its shmt field is 0 i.e. shift offset is 0
Its function field value is 32 i.e. It represent an <u>add</u> instruction

add \$s0, \$s1, \$s0 //

b) sw \$t1, 32(\$t2)

It is an <u>I-type</u> instruction

op	rs	rt	address
101011	01010	01001	0000000000100000
6 bits	5 bits	5 bits	16 bits

Hex: 0x AD490020 //

c) op = 0, rs = 3, rt = 2, rd = 3, shamt = 0, funct = 34

op	rs	rt	rd	shamt	funct
000000	00011	00010	00011	00000	100010
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

It is an R-type instruction with opcode field 0; source register value 3 i.e register \$v1; 2nd source register value is 2 i.e register \$v0; destination register value is 3 i.e register \$v1; it's shamt value is 0 and its function value is 34 i.e. its a subtract (sub) instruction

sub \$v1, \$v1, \$v0

It's binary representation is given below

$(00000000011000100001100000100010)_2$

Q5)
```
      slt  $t2, $0, $t0
      bne  $t2, $0, ELSE
      j    DONE
ELSE: addi $t2, $t2, 2
DONE:
```

We know that

$\$t0 = 0b\,00101000_2$

Now, since \$t0 is greater than \$0
\$t2 will be

$\$t2 = 0b\,00000001_2$

Now, as \$t2 is not equal to \$0, the program flow jumps to ELSE label and 2 is added ~~to~~ to \$t2 and is stored back in \$t2. Hence, value stored in \$t2 will be

$\$t2 = 0b\,00000011_2$ or 3_{10} //