CMPEN331 – Quiz 4

Section: **Major:**

Name: **Email:**

1.

item	This is the output of:	This is the input to:
bne $t0, $s0, done	compiler	assembler
Char*s = "hello world"	programmer	compiler
firefox	linker	loader

2. In

```
loop:
        lw $t1, 0($s0)          # Load A[i]
        lw $t2, 8($s1)          # Load B[i+2]
        mul $t3, $t1, $t2       # A[i] * B[i+2]
        lw $t1, 4($s0)          # Load A[i+1]
        add $t2, $t1, $t3       # A[i+1] + A[i]*B[i+2]
        sw $t2, 0($s2)          # C[i] = A[i+1] + A[i]*B[i+2]
        addi $s0, 4             # Go to A[i+1]
        addi $s1, 4             # Go to B[i+1]
        addi $s2, 4             # Go to C[i+1]
        addi $t0, 1             # Increment index variable
        bne $t0, $t5, loop      # Compare with Loop Bound, one line
```

3.

i.

i-type

ii.

Note that R[rs] is equivalent to $t2 and the instruction, PC = PC + 4 + BranchAddr, simply means jumping the program counter to "BranchAddr" or "loop" (as given in the instruction), relative to the current instruction (where PC is at). Therefore, we would need a branch instruction for jumping.

Now, assuming the semantics of the instruction are:

```
If ($t2 > 0):
    Subtract the value in $t2 by 1
```

Jump to "loop" (relative to PC)

We can achieve this effect using **two** different ways:

1. If we use a branch greater than, **bgt** (pseudo instruction):

   addi $t2, $t2, -1 #can also use subi

   bgt $t2, $0, loop #jump to loop

   addi $t2, $t2, 1   #if we don't jump to loop => condition false => need to "reverse" the subtraction by 1 done earlier (assuming we would jump from loop to the instructions following the last addi)

2. If we don't use bgt, we can achieve the same effect by using **slt** and **bne**:

   slt $t1, $0, $t2 #$t1 = 1, if 0 < $t2, 0 otherwise.

   addi $t2, $t2, -1

   bne $t1, $0, loop #jump to loop if condition is met

   addi $t2, $t2, 1

Note that another valid assumption of the semantics can be:

```
If ($t2 > 0):
    Subtract the value in $t2 by 1
Jump to "loop" (relative to PC)
```

This implies to jump, regardless of the if condition and can be achieved by:

slt $t1, $0, $t2

sub $t1, $0, $t1 #$t1 would now be 0 if condition not met, otherwise it would be -1

add $t2, $t2, $t1 #depending on if the condition is met, $t2 would either get subtracted by 1, or remain as is. Note that we can use this trick in the solution assuming the first semantics as well. If we do so, we don't need the addi $t2, $t2, 1 instruction, after the bgt or bne.

beq $0, $0, loop #0=0 and branch to loop regardless.