C335 Homework #4 - Solution

Part I (4 points)

Given the following two 32 bit binary numbers:
1000 1101 0010 1000 0000 0000 0000 0000
0010 1000 1000 1000 0000 0000 0000 0001
Find the MIPS assembly instruction represented by each number.

```
lw      $t0, 0($t1)
slti    $t0, $a0, 1
```

Part II (6 points)

Implement the following C code in MIPS assembly language, assuming that ***compare*** is the first function called:

```
int compare (int a, int b) {
        if (sub(a, b) >= 0)
                return 1;
        else
                return 0;
}

int sub (int a, int b) {
        Return a-b;
}
```

Version 1

```
compare:    addi    $sp, $sp, -4          # move stack pointer
            sw      $ra, 0($sp)           # save return address

            jal     sub                   # can jump directly to sub
            slt     $t0, $v0, $zero
            add     $v0, $zero, $zero
            bne     $t0, $zero, Finish
            addi    $v0, $v0,1
Finish:     lw      $ra, 0($sp)           # restore return address
            addi    $sp, $sp, 4           # restore stack pointer
            jr      $ra                   # return

sub:        sub     $v0, $a0, $a1         # return a-b
            jr $ra                        # return
```

Version 2

```
compare:    addi   $sp, $sp, -4          # move stack pointer
            sw     $ra, 0($sp)           # save return address

            jal    sub                   # can jump directly to sub
            slt    $v0, $v0, $zero       # if sub(a,b) >= 0, return 1
            slti   $v0, $v0, 1
            lw     $ra, 0($sp)           # restore return address
            addi   $sp, $sp, 4           # restore stack pointer
            jr $ra                       # return

sub:        sub $v0, $a0, $a1            # return a-b
            jr $ra                       # return
```

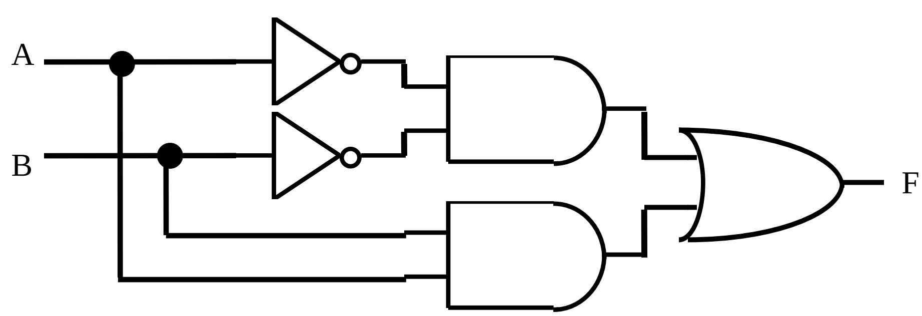
Part III (8 points)

Draw the truth table and the logic circuit for the following function

F = A’• B’ + A•B

(Note, for the logic circuit part, you could draw it by hand)

A	B	A’• B’ + A•B
0	0	1
0	1	0
1	0	0
1	1	1



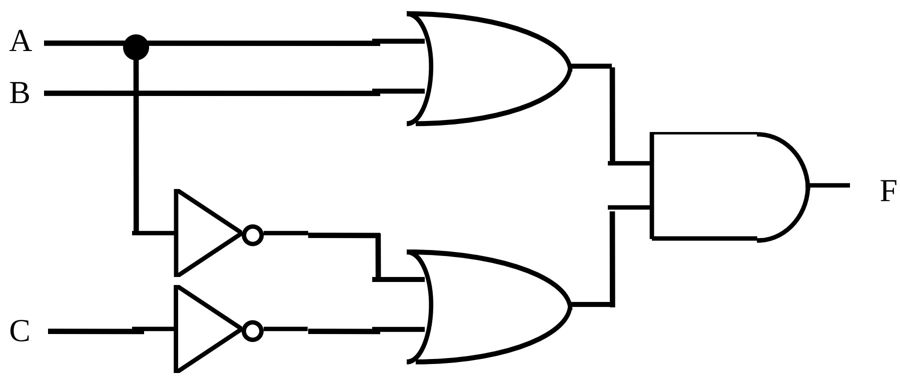
Part IV (8 points)

Draw the truth table and the logic circuit for the following function

F = (A + B) • (A'+ C')

(Note, for the logic circuit part, you could draw it by hand)

A	B	C	A + B	A' + C'	(A+B) • (A'+C')
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	0



Part VI (6 points)

Use perfect induction to prove $x \cdot (x' + y) = x \cdot y$

x	y	x'	x' + y	$x \cdot (x' + y)$	$x \cdot y$
0	0	1	1	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	1	0	1	1	1

Equvialent

Part VII (8 points)

We discussed the logical equations in **minterm form** for CarryOut and Sum of the 1-bit full adder. Can you write down the logical equations for these two output signals in **maxterm form**?

CarryOut = (A + B + CarryIn) • (A + B + CarryIn’) • (A + B’ + CarryIn) • (A’ + B + CarryIn)

Sum = (A + B + CarryIn) • (A + B’ + CarryIn’) • (A’ + B + CarryIn’) • (A’ + B’ + CarryIn)

Part VII (Bonus 5 points)

The following is a C code segment doing bubble sorting on an integer array:

```
for (i = 0; i < n - 1 ; i ++)
{
        for (j = 0; j < n – i -1; j ++)
        {
                If (array[j] > array[j+1]
                {
                        temp = array[j];
                        array[j] = array[j+1];
                        array[j+1] = temp;
                }
        }
}
```

Compile this code segment into MIPS assembly language, assume integer variable ***n*** is in \$s0, integer variable ***i*** is in \$s1, integer variable ***j*** is in \$s2,the base address of integer array ***array*** is in \$s3.

```
#--------------------------------------------bubble sort the array

            add    $s1, $0, $0           # i = 0
            addi   $t0, $s0, -1          # n-1 --> $t0
OUTERLP:    beq    $s1, $t0, EXITOUTER           # if i == n-1, goto EXITOUTER
            add    $s2, $0, $0           # j = 0
            sub    $t1, $t0, $s1         # n-i-1 --> $t1
INNERLP:    beq    $s2, $t1, EXITINNER   # if j == n-i-1 goto EXITINNER
            sll    $t3, $s2, 2           # j*4 --> $t3
            add    $t3, $t3, $s3         # address of array[j] --> $t3
            lw     $t4, 0($t3)           # array[j] --> $t4
            lw     $t5, 4($t3)           # array[j+1] ==> $t5
            slt    $t6, $t5, $t4
            beq    $t6, $0, SKIP         # if !(array[j] > array[j+1]) goto SKIP
            add    $t6, $t4, $0          # array[j] --> temp
            sw     $t5, 0($t3)           # array[j+1] --> array[j]
            sw     $t6, 4($t3)           # temp --> array[j+1]
SKIP:       addi   $s2, $s2, 1           # j++
            j INNERLP
EXITINNER:  addi   $s1, $s1, 1           # i++
            j OUTERLP
EXITOUTER:
```