0907335 Computer Organization (Fall 2012)
Midterm Exam

الاسم: رقم التسلسل: رقم الشعبة: ١

==

Instructions: Time **50** minutes. Open book and notes exam. No electronics. Please answer all problems in the space provided and limit your answer to the space provided. **No questions are allowed**.

<Good Luck>

Q1. A given application is written in Java runs 10 seconds on a desktop processor. A new Java compiler is released that requires only 0.5 as many instructions as the old compiler. Unfortunately, it increases the CPI by 1.2. What is the percentage performance increase of this application using this new compiler?

<5 marks>

Old Time $= IC_{old} * CPI_{old} * T_{old}$

= 10 seconds

New Time $= 0.5 * IC_{old} * 1.2 * CPI_{old} * T_{old}$

= 0.5 * 1.2 * 10

= 6 seconds **(2 marks)**

Speedup **= 10 / 6 = 1.67** **(2 marks)**

The new compiler is (1.67-1)*100%=67% faster than the old compiler **(1 marks)**

Q2. At the end of executing the following MIPS instruction sequence, specify the contents of the following registers.

<5 marks>

```
        addi $s0, $zero, 6      # s = 6
        addi $v0, $zero, -3     # v0 = -3 = -xfffffffd
        add  $s1, $s0, $v0      # s1 = 6-3 = 3
        beq  $s1, $v0, Skip     # not taken
        sll  $s2, $s1, 1        # s2 = 3 << 1 = 6
Skip
        or   $v0, $v0, $s2      # v0 = 0xfffffffd or 6 = 0xffffffff
                                #    = -1
```

Register $s0 = ____6_____ **(1 mark)**

Register $v0 = ____-1____ **(2 mark)**

Register $s1 = ____3_____ **(1 mark)**

Register $s2 = ____6_____ **(1 mark)**

Q3. Convert the following C function to MIPS leaf procedure. The first argument is the starting address of an array of integers A[] and is in register **`$a0`**. The second argument is the array size and is in register **`$a1`**. The return value is in register **`$v0`**.

<10 marks>

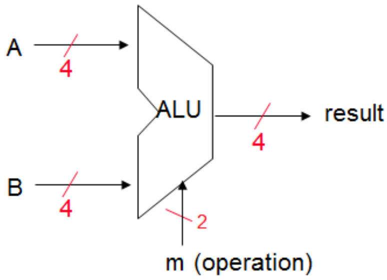
```
int find_array_sum (int* A, int n) {
   int i, sum=0;
   for (i=0; i<n; i=i+1)
      sum = sum + A[i];
   return sum;
}
```

```
find_array_sum:
   add  $t0, $zero, $zero        # i = 0
   add  $t1, $zero, $zero        # sum = 0
Loop:
   slt  $t2, $t0, $a1            # is i < n ?
   beq  $t2, $zero, Exit         # branch if not
   sll  $t3, $t0, 2
   add  $t3, $t3, $a0
   lw   $t4, 0($t3)              # load A[i]
   add  $t1, $t1, $t4            # sum = sum + A[i]
   addi $t0, $t0, 1
   j    Loop
Exit:
   add  $v0, $t1, $zero
   jr   $ra
```

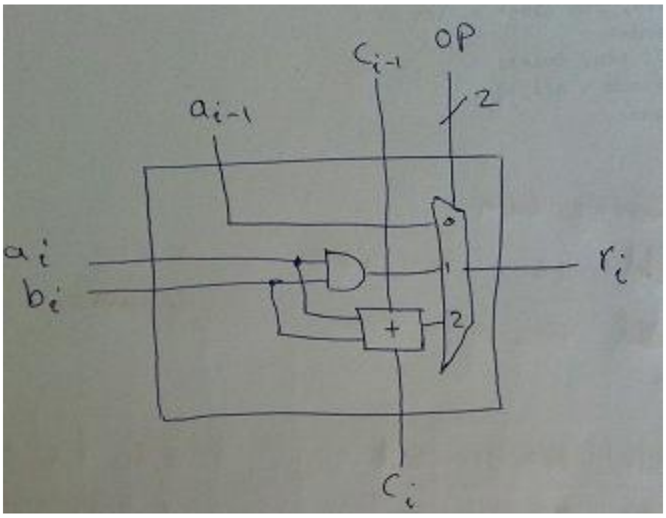
Q4. It is required to design a 4-bit ALU that can perform the operations specified in the following table. This ALU has the interface specified to the right.

<10 marks>

Operation	Function
00	sll
01	and
10	add



a) Design a one-bit ALU slice that performs these three operations. Use full adder, multiplexer, and basic logic gates as your building blocks.



b) Connect four slices to get the required 4-bit ALU.