CMPUT 229 - Quiz # 4 - Fall 2011

Name:

Question 1 (100 points): The intention of the subroutine below is to multiply two 16-bit numbers. It expects the parameters for the subroutine to only contain non-zero bits in the 16 least-significant bits. If there are any non-zero bits amongst the 16 most-significant bits of the operands, then the subroutine returns 1 in `$v1`. Otherwise it should return 0 in `$v1` and return the the product in `$v0`.

```
00   mult:
01           srl   $t2, $a0, 16          # $t2 <-- $a0 >> 16
02           bne   $t2, $zero, notok     #
03           srl   $t3, $a1, 16          # $t3 <-- $a0 >> 16
04           bne   $t3, $zero, notok
05           move  $v1, $zero            # operands are ok
06           addi  $t0, $zero, 8        # $t0 <-- 8
07   next:   move  $v0, $zero            # $v0 <-- $zero
08           andi  $t1, $a1, 0x0001      # $t1 <-- bit 0 of $a1
09           beq   $t1, $zero, shift     # if ($t0 == 0) goto shift
10           add   $v0, $v0, $a0         # $v0 <-- $v0+$a0
11   shift:  sll   $a0, $a0, 1           # $a0 <-- $a0 << 1
12           srl   $a1, $a1, 1           # $a1 <-- $a1 >> 1
13           addi  $t0, $t0, 1           # $t0 <-- $t0 + 1
14           bne   $t0, $zero, next      # if ($t0 != 0) goto next
15           jr    $ra
16   notok:  addi  $v1, $zero, 1         # operands are not ok
17           jr    $ra
```

1. [**20 points**] Identify which parameter is the multiplicand and which parameter is the multiplier for the `mult` subroutine. In a multiplication we test the bits of the multiplier and add the (shifted) value of the multiplicand to the product. Therefore `$a0` is the multiplicand and `$a1` is the multiplier.

2. [**50 points**] There are some errors in the subroutine. Describe the errors and the necessary corrections. Clearly indicate line numbers when describing the errors. There are three errors in the subroutine:

   Line 06 initializes a counter for the number of shift/compare/add steps. This counter should be initialized to 16 instead of 8.

label next is on the wrong line, it should be in line 08 instead of line 07.

Line 13 the instruction should decrement the counter (add -1), instead of incrementing it.

3. **[30 points]** After the errors that you identified above are corrected, how many times the add instruction in line 10 will be executed if the following code is used to call `mult`

```
li    $a0, 0x0FCE
li    $a1, 0x0FF0
jal mult
```

There are 8 bits equal 1 in `$a1`, therefore the addition will be executed 8 times.