ICS 233: Computer Architecture & Assembly Language
Fall Semester 2021 (211) – Section 01
Quiz 2

Name: *ID#*

Q1. (4 points) Below is a MIPS assembly code. For each instruction, determine the register that is modified, and its new value (in hexadecimal). Indicate if any instruction causes an exception. Assume the initial values stored in **`$t0 = 0xFFFF_FFF0, $t1 = 0x0123_4567, $t2 = 0x0000_0006, $t3 = $t4 = 0x0000 0000, and $t5 = $t6 = 0xFFFF_FFFF`** initially.

Code	Reg.	New Value
`addi $t3, $t2, -8`	`$t3`	`0xFFFF FFFE (-2)`
`subu $t4, $t2, $t0`	`$t4`	`0x0000 0016 (22)`
`and $t5, $t0, $t1`	`$t5`	`0x0123 4560`
`sll $t6, $t1, 1`	`$t6`	`0x0246 8ACE`

Q2. (4 points) Translate the following high-level language expression into the **shortest** sequence of assembly language instructions:

a. `$t0 = 28 * $t1` (you cannot use multiplication instructions)

```
# 28*$t1 = (16 + 8 + 4) *$t1
sll $t2, $t1, 4 # 16*$t1
sll $t3, $t1, 3 # 8*$t1
sll $t4, $t1, 2 # 4*$t1
addu $t0, $t2, $t3 # 24 * $t1
addu $t0, $t0, $t4 # 28 * $t1
```

```
# 28*$t1 = (32 - 4) *$t1
sll $t2, $t1, 5 # 32*$t1
sll $t3, $t1, 2 # 4*$t1
subu $t0, $t3, $t2 # 28 *
$t1
```

b. `$t2 = -$t3`

```
subu $t2, $zero, $t3
```

Q3. (2 points) Rewrite a pseudo-instruction (**`even $t0, $t1`**) that sets $t0 to 1 if $t1 is even number and resets it to 0 otherwise. You may use ONLY the $at register as a temporary register for intermediate results.

```
ori $t0, $zero, 1
nor $t0, $t0, $t0
nor $t0, $t0, $t1
```

```
andi $t0, $t1, 1
xori $t0, $t0, 1
```

```
sll $t0,$t1,31
srl $t0,$t0,31
xori $t0, $t0, 1
```

ICS 233: Computer Architecture & Assembly Language
Fall Semester 2021 (211) – Section 02
Quiz 2

Name: *ID#*

Q1. (4 points) Below is a MIPS assembly code. For each instruction, determine the register that is modified, and its new value (in hexadecimal). Indicate if any instruction causes an exception. Assume the initial values stored in **`$t0 = 0xFFFF_FFF0, $t1 = 0x8765_4321, $t2 = 0x0000_0006, $t3 = $t4 = 0x0000 0000, and $t5 = $t6 = 0xFFFF_FFFF`** initially.

Code	Reg.	New Value
`addi $t3, $t0, 20`	`$t3`	`0x0000 0004 (4)`
`subu $t4, $t0, $t2`	`$t4`	`0xFFFF FFEA (-22)`
`nor $t5, $t0, $t1`	`$t5`	`0x0000 000E`
`sra $t6, $t1, 1`	`$t6`	`0xC3B2 A190`

Q2. (4 points) Translate the following high-level language expression into the **shortest** sequence of assembly language instructions:

`a. $t0 = 56 * $t1` (you cannot use multiplication instructions)

```
# 56*$t1 = (32 + 16 + 8) *$t1
sll $t2, $t1, 5 # 32*$t1
sll $t3, $t1, 4 # 16*$t1
sll $t4, $t1, 3 # 8*$t1
addu $t0, $t2, $t3 # 48 * $t1
addu $t0, $t0, $t4 # 56 * $t1
```

```
# 56*$t1 = (64 - 8) *$t1
sll $t2, $t1, 6 # 64*$t1
sll $t3, $t1, 3 # 8*$t1
subu $t0, $t3, $t2 # 56 *
$t1
```

`b. $t2 = NOT $t3`

```
nor $t2, $t3, $t3
or
nor $t2, $t3, $zero
```

(2 points) Rewrite a pseudo-instruction (**`odd $t0, $t1`**) that sets $t0 to 1 if $t1 is odd and resets it to 0 otherwise. You may use ONLY the $at register as a temporary register for intermediate results.

```
andi $t0, $t1, 1
```