CS3350B Computer Architecture **Quiz 2** **February 26, 2015**

Student ID number:
Student Last Name:

MIPS Cheat Sheet

Instruction	Example	Meaning
load word	`lw $s0, 4($s1)`	$s0 = Memory($s1+4)
store word	`sw $s0, 4($s1)`	Memory($s1+4) = $s0
add	`add $s0, $s1, $s2`	$s0 = $s1 + $s2
subtract	`sub $s0, $s1, $s2`	$s0 = $s1 - $s2
add immediate	`addi $s0, $s1, 2`	$s0 = $s1 + 2
subtract immediate	`subi $s0, $s1, 2`	$s0 = $s1 - 2
shift left	`sll $s0, $s1, 2`	$s0 = $s1 $\ll$ 2 bits
branch on equal	`beq $s0, $s1, L`	if ($s0 == $s1) go to L
branch on not equal	`bne $s0, $s1, L`	if ($s0 != $s1) go to L
set on less than	`slt $s0, $s1, $s2`	if ($s1 $<$ $s2) $s0 = 1 else $s0 = 0
jump	`j L`	go to L
jump register	`jr $s0`	go to $s0
jump and link	`ja 250`	go to 1000; $ra=PC+4

Exercise 1. Assume that `$t0` holds the value `5` (`101` in binary). What is the value of `$t2` after each of the following instructions?

1.1
```
sll $t2, $t0, 2
andi $t2, $t2, 55    # 55 in binary is 110111
```

`$t2` = 10100 in binary
010100 & 110111 = 010100 in binary
`$t2` = $2^4 + 2^2$ = 20

1.2
```
slt $t2, $zero, $t0
bne $t2, $zero, ELSE
j DONE
ELSE: addi $t2, $t2, 2
DONE:
```

Since `$t0 > 0`, `$t2` = 1
Then `$t2` != 0, go to `ELSE`
`$t2` += 2
`$t2` = 3

Exercise 2. For the following `C` statements, what is the corresponding `MIPS` assembly code? Try to use a minimal number of `MIPS` assembly instructions.

2.1 `f = (g << 1) + (h - 5);`
Assume that the variables `g` and `h` are given and stored in `$t0` and `$t1` respectively. Use `$s0` to represent the variable `f`.

```
sll $t0, $t0, 1
subi $t1, $t1, 5
add $s0, $t0, $t1
```

2.2 `B[4] = A[i-j];`

`A` and `B` are arrays of 32-bit integers. Assume that the variables `i` and `j` are assigned to registers `$t0` and `$t1` respectively, and that the base address of `A` and `B` are in registers `$s4` and `$s5` respectively.

```
sub $t0, $t0, $t1   # i - j
sll $t0, $t0, 2   # (i - j) * 4
add $s4, $s4, $t0
lw $s0, 0($s4)
sw $s0, 16($s5)
```

Exercise 3. Consider the following `C` function computing the sum of the sequence: 1, 2, 3, . . . , n.

```
int sum (int n) {
    int s = 0;
    for (int i = 0; i < n; ++i)
        s += i + 1;
    return s;
}
```

Write a minimal sequence of `MIPS` assembly instructions that does the identical operation. Assume that the registers `$t0` and `$s0` hold the values of the variables `i` and `s`, respectively. As usual, we assume that `$s0` has its own value before the function `sum` is called, thus this value must be saved on stack (given by the register `$sp`) during the execution of the function `sum`.

Note: the argument, the return value and the return address are in `$a0`, `$v0` and `$ra`, respectively.

```
sum:
addi $sp, $sp, -4
sw $s0, 0($sp)    # Save $s0
add $s0, $0, $0    # $s0 = 0
add $t0, $s0, $0    # $t0 = 0
loop:
slt $t2, $t0, $a0    # If $t0 ≥ n
beq $t2, $0, done    # Go to done
addi $t0, $t0, 1    # $t0 += 1
add $s0, $s0, $t0    # $s0 += $t0
j loop
done:
add $v0, $s0, $0    # Save return
lw $s0, 0($sp)    # Pop $s0
addi $sp $sp 4
jr $ra    # Return
```