CS/COE 0447 Fall 2009
Lab 2: Immediate values, memory, system calls and endianness
Solution

Part 1: Immediate Values

What is the machine code (in hexadecimal) of these instructions? Is the immediate field (the last 16 bits) the same in both instructions?

0x2009FFFF and 0x340AFFFF. The immediate field is the same (0xFFFF) for both instructions.

What are the values of the registers? Why are they different?

$t1 = 0xFFFFFFFF and $t2 = 0x0000FFFF. The values are different because the first operation interprets the immediate value as a signed number and therefore extends the value -1 to 32 bits. 0xFFFFFFFF is the result of adding -1 to 0. Instead, the second operation interprets the immediate value as an unsigned number with value 65535 and extends it to 32 bits, which yields 0x0000FFFF. This value is then “ored” with zero, which gives 0x0000FFFF.

Part 2: Memory

```
.text
     la $t0, x
     lw $t1, 0($t0)
     la $t0, y
     lw $t2, 0($t0)
     add $t3, $t1, $t2
     la $t0, z
     sw $t3, 0($t0)
```

```
.text
     la $t0, x
     lh $t1, 0($t0)
     la $t0, y
     lh $t2, 0($t0)
     add $t3, $t1, $t2
     la $t0, z
     sh $t3, 0($t0)
```

```
.text
     la $t0, x
     lb $t1, 0($t0)
     la $t0, y
     lb $t2, 0($t0)
     add $t3, $t1, $t2
     la $t0, z
     sb $t3, 0($t0)
```

Part 3: System Calls

```
     .data
str0:    .asciiz  "The sum of "
str1:    .asciiz  " and "
str2:    .asciiz  " is "
     .text
     li $v0, 5
     syscall
     move $t0, $v0
     li $v0, 5
     syscall
     move $t1, $v0
     add $t2, $t0, $t1
     li $v0, 4
     la $a0, str0
     syscall
     li $v0, 1
     move $a0, $t0
     syscall
     li $v0, 4
     la $a0, str1
     syscall
     li $v0, 1
     move $a0, $t1
     syscall
     li $v0, 4
     la $a0, str2
     syscall
     li $v0, 1
     move $a0, $t2
     syscall
```

Part 4: Endianness

What is the address of the byte with value 0x04?

0x10010003

What is now the address of the byte with value 0x04?

0x10010000

Is the simulator little endian or big endian?

Little endian, because the last byte of the word is stored in the first position in memory.