Computer Organization and Design

Homework 2

2.1 For the following C statement, what is the corresponding MIPS assembly code? Assume that the variables f, g, h, and i are given and could be considered 32-bit integers as declared in a C program. Use a minimal number of MIPS assembly instructions.

$$f = g + (h - 5)$$

Solution:

```
addi f, h, -5
add f, f, g
```

2.3 For the following C statement, what is the corresponding MIPS assembly code? Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

$$B[8] = A[i - j]$$

Solution:

```
sub $t0, $s3, $s4
sll $t0, $t0, 2
add $t0, $t0, $s6
lw $t1, 0($t0)
sw $t1, 32($s7)
```

2.6 The table below shows 32-bit values of an array stored in memory.

Address	Data
24	2
28	4
32	3
36	6
40	1

2.6.1 For the memory locations in the table above, write C code to sort the data from lowest to highest, placing the lowest value in the smallest memory location shown in the figure. Assume that the data shown represents the C variable called *Array*, which is an array of type *int*, and that the first number in the array shown is the first element in the array. Assume that this particular machine is a byte-addressable machine and a word consists of four bytes.

2.6.2 For the memory locations in the table above, write MIPS code to sort the data from lowest to highest, placing the lowest value in the smallest memory location. Use a minimum number of MIPS instructions. Assume the base address of *Array* is stored in register $s6.

Solution:

2.6.1

```
temp = Array [0];
temp2 = Array [1];
Array [0] = Array [4];
Array [1] = temp;
Array [4] = Array [3];
Array [3] = temp2;
```

2.6.2

```
lw $t0, 0($s6)
lw $t1, 4($s6)
lw $t2, 16($s6)
sw $t2, 0($s6)
sw $t0, 4($s6)
lw $t0, 12($s6)
sw $t0, 16($s6)
sw $t1, 12($s6)
```

2.12 Assume that registers $s0 and $s1 hold the values 0x80000000 and 0xD0000000, respectively.

2.12.1 What is the value of $t0 for the following assembly code?

```
add $t0, $s0, $s1
```

2.12.2 Is the result in $t0 the desired result, or has there been overflow?

2.12.3 For the contents of registers $s0 and $s1 as specified above, what is the value of $t0 for the following assembly code?

```
sub $t0, $s0, $s1
```

2.12.4 Is the result in $t0 the desired result, or has there been overflow?

2.12.5 For the contents of registers $s0 and $s1 as specified above, what is the value of $t0 for the following assembly code?

```
add $t0, $s0, $s1
add $t0, $t0, $s0
```

2.12.6 Is the result in $t0 the desired result, or has there been overflow?

Solution:

2.12.1 50000000

2.12.2 overflow

Overflow is usually when the sign of the result doesn't make sense compared to the operands. 0x80000000 and 0xD0000000 are both negative numbers. When you add two negative numbers you should get a negative number back. But instead you get 0x50000000, which is in fact a positive number. So yes, an overflow has occurred.

2.12.3 B0000000

2.12.4 no overflow

2.12.5 D0000000

2.12.6 overflow

2.16 Provide the type, assembly language instruction, and binary representation of instruction described by the following MIPS fields:

op=0, rs=3, rt=2, rd=3, shamt=0, funct=34

Solution:

Type: r-type

Assembly language instruction: sub $v1, $v1, $v0

Binary representation:

0000 0000 0110 0010 0001 1000 0010 0010

0x00621822