Mitchell Kolb
10/7/2020
CPTS 260 Homework 3
WSU

Part 1: Intro To Assembly

1.

f = $s0
g = $s1
h = $s2
i = $s3
j = $s4

Base address for arrays A = $s6, and B = $s7

$t0 = f * 4;
$t0 = &A[f];
$t1 = g * 4;
$t1 = &B[g];
f = A[f];
$t2 = &A[f + 1];
$t0 = A[f + 1];
$t0 = A[f + 1] + A[f];
B[g] = A[f + 1] + A[f];

2.

A.

$s0 = 0x80000000 in Hex
$s0 = 1000 0000 0000 0000 0000 0000 0000 0000 in Binary
$s0 = 2147483648 in Dec

$s1 = 0xD0000000 in Hex
$s1 = 1101 0000 0000 0000 0000 0000 0000 0000 in Binary
$s1 = 3489660928 in Dec

2147483648 + 3489660928 = 5637144576

1000 0000 0000 0000 0000 0000 0000 0000
1101 0000 0000 0000 0000 0000 0000 0000
+_____
1 0101 0000 0000 0000 0000 0000 0000 0000 = 5637144576

There is overflow because the carry in and carry out bits are not equal.

B.

$s0 = 0x80000000 in Hex
$s0 = 1000 0000 0000 0000 0000 0000 0000 0000 in Binary
$s0 = 2147483648 in Dec

$s1 = 0xD0000000 in Hex
$s1 = 1101 0000 0000 0000 0000 0000 0000 0000 in Binary
$s1 = 3489660928 in Dec

2147483648 - 3489660928 = -1342177280

1000 0000 0000 0000 0000 0000 0000 0000
0011 0000 0000 0000 0000 0000 0000 0000 2's compliment
+_____
1011 0000 0000 0000 0000 0000 0000 0000 = -1342177280 signed 2's compliment

There is no overflow because there were no extra bits that went into the carry in or carry out.

C.

First MiPs line
$s0 = 0x80000000 in Hex
$s0 = 1000 0000 0000 0000 0000 0000 0000 0000 in Binary
$s0 = 2147483648 in Dec

$s1 = 0xD0000000 in Hex
$s1 = 1101 0000 0000 0000 0000 0000 0000 0000 in Binary
$s1 = 3489660928 in Dec

2147483648 + 3489660928 = 5637144576

1000 0000 0000 0000 0000 0000 0000 0000
1101 0000 0000 0000 0000 0000 0000 0000
+_____
1 0101 0000 0000 0000 0000 0000 0000 0000 = 5637144576

Second MiPs line
$s0 = 0x80000000 in Hex
$s0 = 1000 0000 0000 0000 0000 0000 0000 0000 in Binary
$s0 = 2147483648 in Dec

$t0 = 0x150000000 in Hex
$t0 = 0001 0101 0000 0000 0000 0000 0000 0000 0000 in Binary
$t0 = 5637144576 in Dec

2147483648 + 5637144576 = 7784628224

0000 1000 0000 0000 0000 0000 0000 0000 0000
0001 0101 0000 0000 0000 0000 0000 0000 0000
+_____
0001 1101 0000 0000 0000 0000 0000 0000 0000 = 7784628224

There is no overflow because there were no extra bits that went into the carry in or carry out.

3.

```
while( 0 < i )
{
        temp = 1;
        if(temp = 0)
                break;
        i--;
        B = B + 2;
}

$s1 = A, $s2 = B, $t1 = i, $t2 = temp

LOOP:
        slt $t2, $0, $t1
        beq $t2, $0, DONE
        subi $t1, $t1, 1
        addi $s2, $s2, 2
        j LOOP
DONE:
```

Part 2: Intro to MIPS

1.
A.

0010 0001 0010 1001 0000 0000 0000 0001
= 001000 01001 01001 00000 00000 000001

op is the ADDi instruction
rs is 9 which is $t1
rt is 9 which is $t1
The rest of the 16 bits once translated to dec is 1 which is used as the constant for the addi command.

The instruction is ADDi $t1, $t1, 1

NAME	NUMBER
$zero	0
$at	1
$v0-$v1	2-3
$a0-$a3	4-7
$t0-$t7	8-15
$s0-$s7	16-23
$t8-$t9	24-25
$k0-$k1	26-27
$gp	28
$sp	29
$fp	30
$ra	31

B.

sw $t5, 32($t2)

sw according to the reference sheet has a hex of 2b.
$t5 = 21Dec
32 =
$t2 = 10Dec

- The offset is 6 bits so (31 through 26) is 101011 for the sw instruction
- The following next five bits is $t2 or 10 or 01010 in binary
- The following next five bits is $t5 or 21 or 01101 in binary
- The following 16 bits are used for the offset and it's 32 so it's 0000 0000 0010 0000 in binary

The whole string is 101011 01010 01101 0000 0000 0010 0000
When translated to hexadecimal it's AD5D0020

C.

op = 0, rs = 3, rt = 2, rd = 13, Shamt = 0, funct = 36

op is 6 bits so it's 000000
rs is 5 bits and 3 so it's 00011
rt is 5 bits and 2 so it's 00010
Shamt is 5 bits and 0 so it's 00000
funky is 6 bits and 36 so it's 100100

all together it's: 000000 00011 00010 01101 00000 100100
once translated to the MIPS command it's AND $t4, $v1, $v0

2.

A.

I think the I-format would be the best because there are arrays in the command and this instruction format has space set to keep the address of the array available.

B.

```
loop:
    slt $t1, $t2, 1
    beg #t1, $zero, Done
    addi $t2, $t2, -1
    j loop:
```

3.

x = $s1
y = $s2
A = $s3

```
add $t0, $s3, $t0
lw $t0, 5($t0)
```