CS1216 - Monsoon 2022 - Homework 3

Jivansh Sharma
UG 24 1020211193

16/10/2022

Collaborators: None

1. Read the assembly code below; add comments to explain what each line of code is doing; in one sentence, explain what this procedure is trying to accomplish.

```
myProcedure:
    1. sll $a0, $a0, 24
# This instruction shifts the address at a0 by 24 bits to the left,
losing on the first 24 bits of the number -> XXX(8 bits occupied)000(24 times)
# In case the first 24 bits of the number weren't occupied it will also
multiply the number stored by 2*24

    2. srl $a0, $a0, 24
# This instruction shifts the address at a0 by 24 bits to the right,
losing on the last 24 bits of the number -> 000(24 times)XXX(8 bits occupied)
# In case the first 24 bits of the number weren't occupied it will also
divide the number stored by 2*24

    3. add $v0, $a0, $zero # setting v0 = a0 + 0
    4. jr $ra              # return statement indicating procedure is over.
```

2. For the (pseudo) assembly code below, replace L, M, U, and X with the smallest set of instructions to save/restore values on the stack and update the stack pointer. Assume that procA and procB were written independently by two different programmers who are following the MIPS guidelines for caller-saved and callee-saved registers. In other words, the two programmers agree on the input arguments and return value of procB, but they can't see the code written by the other person. Be sure to read the textbook and lecture slides so you understand the MIPS guidelines for caller-saved and callee-saved registers.

L

```
addi $sp, $sp, -16
sw, $a0, 12($sp)
sw, $ra, 8($sp)
sw, $t0, 4($sp)
sw, $t1, 0($sp)
```

M

```
lw, $t1, 0($sp)
lw, $t0, 4($sp)
lw, $ra, 8($sp)
lw, $a0, 12($sp)
addi $sp, $sp, 16
```

U

```
addi $sp, $sp, -12
sw, $s0, 8($sp)
sw, $s1, 4($sp)
sw, $s2, 0($sp)
```

X

```
lw, $s2, 0($sp)
lw, $s1, 4($sp)
lw, $s0, 8($sp)
addi $sp, $sp, 12
```

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

(a) Assuming that $t0 is represented as a 32-bit signed integer (2's complement representation), state the value that will be stored in $t0 after the following instruction. Is this the value that was desired by the programmer? State your reasons.

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

No, this will not give the desired value. Rather it will result in integer overflow. This happens because `-2147483648` + `-805306368` overflows past the limits of a 32-bit number. The desired value was `1342177280` which is not possible to store in a 32-bit number.

Incase, we wanted to ignore the overflow, we could have used the equivalent unsigned instruction: `addu`.

(b) What about the following instruction? State whether $t0 will store the desired value. Why?

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

Yes, this will give the desired value. Rather it will `NOT` result in integer overflow. The desired and returned value is `-1342177280`

This happens because integer overflow is not possible in subtraction. Even if you add the largest negative number to the largest positive number, the result will be a 32 bit positive number.