Homework 3

ECE2500 CRN:82943

Jacob Abel

September 25, 2018

Problem 1: Arrange the following branch/jump instructions in descending order of their maximum possible jump target. That is, the instruction that allows the farthest possible jump must go first, followed by the next farthest, and so on. If two instructions allow the same amount of jump, list them in any order. Also state encoding format (R/I/J) and the number of bits used to represent the target address for each instruction. Finally, list for each instruction the maximum distance that can be jumped, both forwards and backwards in memory.

j , beq , jr , jal , bne

jr **Encoding Format:** Register

Address Bits: 32-bit

Max Direct Jump Address: `0xFFFFFFFF`

Range Any location within 32-bit addressable memory.

jal **Encoding Format:** Jump

Address Bits:

Max Direct Jump Address: `0xFFFFFFF`

Range Any location with the same upper 4 bits.

j **Encoding Format:** Jump

Address Bits: 26-bit

Max Direct Jump Address: `0xFFFFFFF`

Range Any location with the same upper 4 bits.

beq **Encoding Format:** Immediate

Address Bits: 16-bit

Max Forward Distance: `0x7FFF`

Max Backward Distance: `0x8000`

bne **Encoding Format:** Immediate

Address Bits: 16-bit

Max Forward Distance: `0x7FFF`

Max Backward Distance: `0x8000`

Problem 2: Answer true or false to the following questions and give reasons:

1. The immediate constant operand in the sltiu instruction is treated as an unsigned 16-bit integer.

   Yes as it is an unsigned instruction.

2. The immediate constant operand in the sltiu instruction is sign extended.

   Yes as all arithmetic immediate instructions are sign extended whether or not they are signed.

Problem 3: Suppose the program counter (PC) is set to `0x30002000`. Is it possible to use the jump (j) MIPS assembly instruction to jump directly to the instruction at `0x50003000`? If yes, write the corresponding assembly instruction(s). If not, explain why and give other MIPS instruction(s) that can jump to this target address. (For this problem, assume that the instruction memory starts from `0x00000000` and goes to `0xFFFFFFFF`.)

No as jump can only access the same 256 MB_2 of memory block as the program counter and the intended location is in a different block. Jump register has full 32-bit memory access and can reach said address. The following would get to the address.

```
lui   $t0  0x5000
ori   $t0  0x3000
jr    $t0
```

Problem 4: The mul3.asm file gives the assembly code that implements the main function which calls the mul3 function by passing three arguments in $\$a0$, $\$a1$, $\$a2$. mul3 returns the solution (product of the three input arguments) in $\$v0$ (which is then saved in $\$s0$). Modify this function so that instead of taking in three arguments, the function takes in six input arguments. The output should be the product of all six input arguments. You must use the stack (i.e., use the memory with the $\$fp$ register and/or $\$sp$ register) to pass the input arguments. Your submission must consist of a file named mul6.asm with the modified code. Test your code in QtSPIM. In both the screenshots, mark the data that your code pushed onto the stack. In addition, in the pdf report state the following values from QtSPIM: $\$ra$ register before and after executing the jal instruction and $\$fp$ and $\$sp$ registers before pushing any argument on the stack, after pushing all six arguments on the stack (but before jal), and after popping all six arguments from the stack in mul6.

$\$fp$ Unused for this problem.

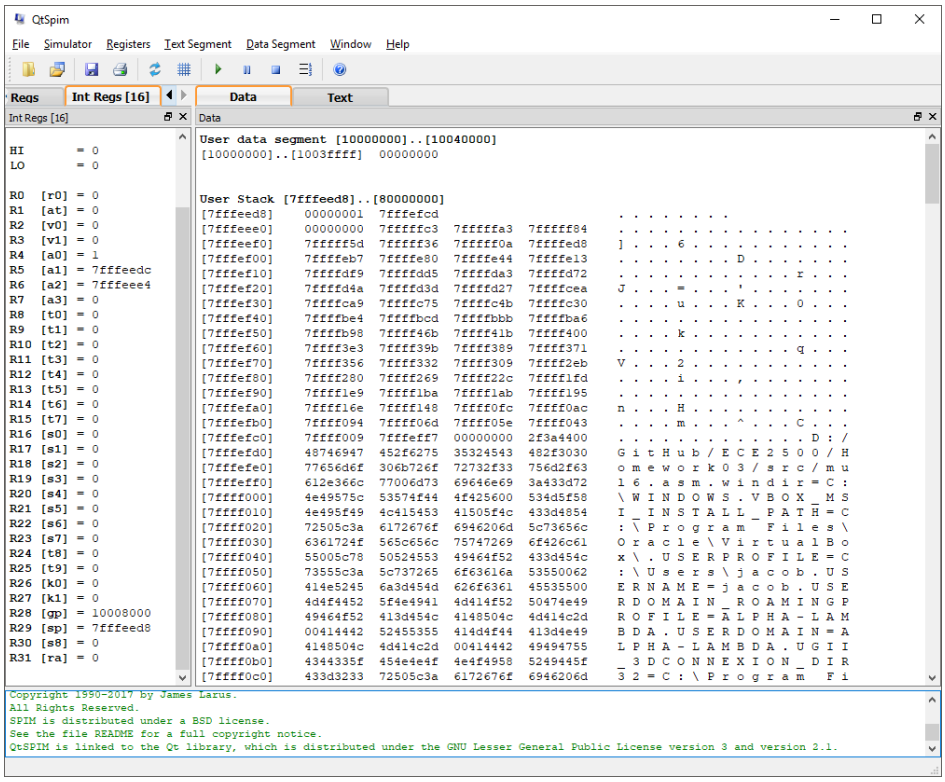
$\$sp$

Before: `0x7FFFEED0`
After: `0x7FFFEEC0`

$\$ra$

Before: `0x400018`
After: `0x40005c`

1. User stack at the start of the program (before pushing anything on the stack).



2. User stack after pushing all six arguments on the stack.