CSC 205: Homework 2
Connor Baker, February 2017

Instructions *Answer all problems as directed. Problems where SPIM source code is requested will NOT receive credit without the source code. All SPIM programs should not use any instructions not covered in class and should not use any pseudo instructions. Points will be deducted for any use of pseudo instructions.*

Problem 1 Suppose the MIPS processor contained half as many registers as stated in your text (16 versus 32). What impact would this have on the instruction formats? What instruction classes are affected?

Work The only instruction that does not use registers is the J type instruction. As such, both the R and I type instruction formats would see changes in terms of the registers available to them, as well as the amount of bits in the instruction allocated to the number of the register. After all, if there are only 16 available registers, there is no need for four bits in the instruction. As such, the R type instruction would have three more bits available (as one of each of the bits allocated to a register address would be freed up), and the I type instruction would have two more bits available, most likely given to the constant.

More than a few registers are reserved for the operating system, so most likely if the number of registers were to be halved there would be a massive reduction in the number of registers that store temporary information. There would also be way more register overspill – the transfer of information stored in the registers to main memory.

Problem 2 Add comments to the following MIPS code and describe in one sentence what it computes. Assume that \$a0 is used for the input and initially contains n, a positive integer. Assume that \$v0 is used for the output.

```
begin:      addi    $t5,$zero,0
            addi    $t6,$zero,1
            addi    $t1,$zero,1

loop:       slt     $t2,$a0,$t1
            bne     $t2,$zero,finish
            add     $t7,$t5,$t6
            add     $t5,$zero,$t6
            add     $t6,$zero,$t7
            addi    $t1,$t1,1
            j loop

finish:     add     $v0,$t7,$zero
```

In addition to what you are asked for above, what value does \$v0 contain when the program finishes?

Work The program computes the $(n-1)$th element (where \$a0 is n) of the Fibonacci sequence (generating function $F_n = F_{n-1} + F_{n-2}$), where $F_0 = 1, F_1 = 2$ (so it generates the zero-indexed set $\{1, 2, 3, 5, 8, \dots\}$, instead of the more widely used set for the Fibonacci sequence, $\{0, 1, 1, 2, 3, \dots\}$, which requres that $F_0 = 0$ and $F_1 = 1$). That $(n-1)$th element is what \$v0 contains.

The commented code is in Problem2.asm, which is attached.

Problem 3 Using SPIM, write the MIPS instruction(s) for this C statement:

```
A = (B - 210) * C;
```

Turn in your source code and SPIM output showing your results.

Work See attached file, Problem3.asm.

SPIM Output (356 - 210) * 2 = 292

Problem 4 Using SPIM, write the MIPS instruction(s) for this C statement:

```
if (q == r)
    r = r + q;
else
    q = r - q;
```

Turn in your source code and SPIM output showing your results.

Work See attached file, Problem4.asm.

SPIM Output q is -2

r is 3

Problem 5 The instructions below are from the MIPS program in problem 2. Determine the instruction type (R, I, or J) and field values for these instructions. Use decimal values for each field.

Instruction	Type	Opcode			Byte-Address		
j loop*	J	2			20		
Instruction	Type	Opcode	RS	RT	Address/Immediate		
addi \$t5,\$zero,0	I	8	0	13	0		
bne \$t2,\$zero,finish	I	5	0	10	5		
Instruction	Type	Opcode	RS	RT	RD	SHAMT	FUNCT
slt \$t2,\$a0,\$t1	R	0	4	9	10	0	42
add \$t7,\$t5,\$t6	R	0	13	14	15	0	32

*For this instruction, you may assume that the byte address of the instruction at the label loop is 20 (decimal).

Work Filled out table above.

Problem 6 Using SPIM, write MIPS assembly code for the following C segment:

```
for (int i = 0; i < N; i++)
    if (list[i] == key)
        Break; // Immediate for loop exit
key = i;
```

You may code as many branch or jump instruc6ions in the loop as you need; however, each loop iteration can, at most, execute only one branch and one jump (total of two). Note that a branch that "falls through" (e.g. does not branch) on each pass does not count toward this requirement as an executed branch.

Value **N** is positive number that can be any number greater than or equal to 10. Be aware that large values of N will require a very large array.

You may assume that the key is always found in the array (e.g. you do not have to account for the key not being found).

Turn in your source code and SPIM output showing your results.

Below is an example of how to access an array in MIPS.

```
          .data
myList    .word     1,2,3,4,5,6,7,8,9,10 #10 element array
myListA:  .word     myList               # get address
of
myList
          .text
lw        $s3,myListA                    # set $s3 to the
                                         beginning address
                                         of the array
lw        $t1,0($s3)                     # get list item
addi      $s3,$s3,4                      # increment list
                                         index
```

The LW instruction obtains an item from the list while the ADDI instruction advances the pointer register $s3 to the next element of the array. Put these instructions in a loop and you can "walk" through an entire array!

Work See Problem6.asm, attached.

SPIM Output

Problem 7 Show the single MIPS instruction or minimal sequence of instructions for this C statement:

```
B = 25 | A;
```

Note: the operator — is the C ”bitwise” OR operator.

Turn in your source code and SPIM output showing your results.

Work See attached Problem7.asm.

SPIM Output The result of B = 25 — A is: 29