

Homework #2 – MIPS Assembly Language

Due: at the start of lecture on Thursday 14 June 2007

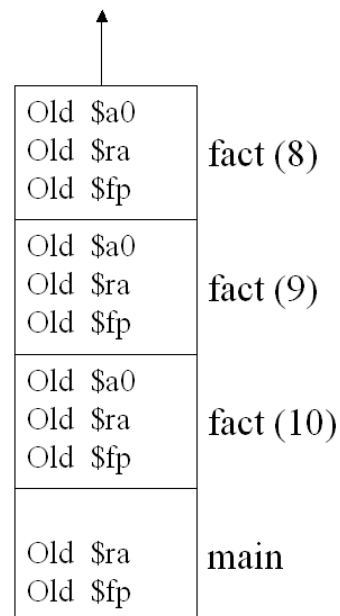
Your MIPS code should be properly commented. Points will be deducted for poor comments or no comments. The MIPS simulator (SPIM) accepts pseudo instructions, which are not part of the MIPS ISA. Do NOT use those pseudo instructions for the exercises that ask you to produce MIPS code. One of the goals of this assignment is to learn the real MIPS ISA. The MIPS simulator (SPIM) is described in the documentation included with the CD-ROM that accompanies your book.

Part I – Regular Problems

1. (10 pts) Comment the following code fragment (high-level code included for clarity):

```
main() {
    int f;
    f = fact (10);
    printf ("Fact(10) = %d\n", f);
}
```

```
int fact ( int n) {
    if (n < 1)
        return (1);
    else
        return (n * fact(n-1));
}
```



```

main:  subu   $sp, $sp, 32
      sw    $ra, 20($sp)
      sw    $fp, 16($sp)
      addiu $fp, $sp, 28
      li    $v0, 4
      la    $a0, str
      syscall
      li    $a0, 10
      jal   fact
      addu  $a0, $v0, $zero
      li    $v0, 1
      syscall
      lw    $ra, 20($sp)
      lw    $fp, 16($sp)
      addiu $sp, $sp, 32
      jr    $ra

fact:  subu   $sp, $sp, 32
      sw    $ra, 20($sp)
      sw    $fp, 16($sp)
      addiu $fp, $sp, 28
      sw    $a0, 0($fp)
      lw    $v0, 0($fp)
      bgtz  $v0, L2
      li    $v0, 1
      j     L1
L2:    lw    $v1, 0($fp)
      subu  $v0, $v1, 1
      move  $a0, $v0
      jal   fact
      lw    $v1, 0($fp)
      mul   $v0, $v0, $v1
L1:    lw    $ra, 20($sp)
      lw    $fp, 16($sp)
      addiu $sp, $sp, 32
      jr    $ra

```

2. (15 pts) Assume that the code from Problem 1, above, is run on a 4.27 GHz machine that requires the following number of cycles for each instruction (CPI):

| Instruction | CPI |
|--|-----|
| add, addi, addiu, sub, subu, subiu | 1 |
| j, jal, jr, syscall | 2 |
| lw, sw, li, la, bne, beq, mul | 3 |
| Any other instructions | 4 |

- (10pts) What is the execution time for that code if the pseudo-instructions *move* and *bgtz* each require 2 cycles? *Show your work to get full credit.*
- (5 pts) What is the execution time for that code if the preceding pseudo-instructions each require one cycle? *Show your work to get full credit.*

3. (5 pts) Write the single MIPS instruction or minimal sequence of instructions that implements the following C statement:

```
x[10] = x[20] - c;
```

and describe why this code works (include comments in the code). You may assume that the variable c corresponds to register $\$t0$ and the array x has a base address of $2,000,000_{ten}$.

4. (20 pts) The Single Instruction Computer (SIC) has one instruction, named **Subtract and Branch if Negative**, abbreviated **sbn**. This instruction is given with three memory addresses – for example, the instruction:

```
sbn a, b, c # Mem[a] = Mem[a] - Mem[b]; if (Mem[a] < 0) go to c;
```

subtracts the number in memory location **b** from the number in memory location **a**. It then stores the result in memory location **a**. If the result is greater than or equal to zero, then the computer takes its next instruction from the memory location just after the current instruction. If the result is less than zero, then the next instruction is taken from memory location **c**.

Here is a program to multiply the positive integer in **a** by the nonnegative integer in **b**, putting the result in **c**. Let us assume that memory location called **one** contains the number 1.

```
start: sbn temp, temp, .+1 # Sets temp to zero
loop:  sbn b, one, done    # Decrease b by one, branch if it was
zero                                     # zero
      sbn temp, a, loop   # Subtract a from temp and loop back
done:  sbn c, c, .+1      # Set c to zero
      sbn c, temp, .+1   # Set C to product
```

In this program, assume that the notation **.+1** means "the address of the instruction after this instruction," so the current instruction goes on to the next instruction in sequence, whether or not the result is negative.

During the middle part of this program, **temp** contains the negative of **a** times 0, then the negative of **a** times 1, and so on. Notice that this program destroys the value in **b**.

- (10 pts) Write a SIC program to compute the absolute value of **a**, putting the result in **b**.
- (5 pts) What will happen in the multiplication program above if the number in **a** is zero? What happens if the number in **a** is negative?
- (5 pts) Now that you know the answers to the preceding questions, rewrite the program so it works in all cases.

Part II – Extra-Credit Problem

5 (15 pts) Write a program using native MIPS code (no pseudoinstructions) to compute a Fibonacci number x , where x is an input argument. *Hint:* Using the code for

factorial in Problem 1, above, you might want to recall that the Fibonacci number $x_i = x_{i-1} + x_{i-2}$.

Thus, we have a recursion expression on which to build a recursive program structure similar to Problem 1. Alternatively, you can use iteration instead of recursion. **However, you must check your code on the SPIM simulator to make sure it works before including it in your homework problem answer.**
