Drew Smith
CDA 3100

CDA3100: Computer Organization

Assignment 03

The purpose of this assignment is to give you experience writing assembly using the MIPS instruction set. For each problem, you will need to either write a program from scratch or modify a program given to you. You will need to submit a separate file for each problem. QtSpim will be used to test your work, so be sure that you use this simulator when testing your software. There is a link to QtSpim in the Course Library. In addition to the assembly files, you will need to submit all of your answers as a single pdf, so there will be two copies of your assembly solutions – one in the pdf and one in the assembly file(s). For example, I would upload the following files: ch03c.hw3.pdf, ch03c.hw3.2.s, and ch03c.hw3.3.s. Do not put the files in an archive (no zip/tar).

1. A sample program called leaf_example.s is provided with this assignment. Download it and open it in QtSpim. Run to the first line in the example procedure, which should be:
   addi $sp, $sp, -4

   At what address is that instruction stored? What is the hexadecimal representation of the instruction? Is this an I-format or R-format instruction? What is the value (in binary) of the constant field? (2)

a. `[00400024] 23bdfffc = 0010 0011 1011 1101 1111 1111 1111 1100`

b. Itype

c. 1111 1111 1111 1111 1111 1111 1110 1001

1. Run the program using the values 19, 64, -23 and 72 (in that order). What is the value returned by leaf_example? 34

2. Even though the program has completed, the value of i (-23) should still be in register $a2. How is that value represented in hexadecimal? What is the binary equivalent of that value? (1)

   : FFFFFFFFFFFFFFE9

   In binary: 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1110 1001

3. Reinitialize and load the leaf_example.s code. Sometimes it is convenient to inspect the registers at a particular place in the execution of the code. Notice that the instruction immediately after the "jal leaf_example" instruction is "move $s0, $v0" (move is actually a pseudo-instruction; it is replaced by an "addu" instruction). This instruction is stored at address 0x004000c0. Also notice that the value returned by the leaf_example_ procedure should be in register $v0 when this instruction is about to execute. We can check this out

by setting a breakpoint in the code. You can set a breakpoint by right-clicking on the line of code and selecting "set breakpoint". Now execute the program, using the values 10, 20, 30 and 40 (in that order). The program will pause when it is about to execute the instruction at your breakpoint. Select the "Abort" option. What value is in $v0 at this point? What values are in $a0 through $a3?

$v0 = fffffd8

$a0 = a

$a1 = 14

$a2 = 1e

$a3 = 28

2. Another example program called, fact.s, is provided with this assignment.
    1. Start with the code as given and add a driver program that will allow you to test its functionality. Here is a sample execution of my program:

```
Welcome to the factorial tester!
Enter a value for n (or a negative value to exit): 0 0! is 1
Enter a value for n (or a negative value to exit): 6 6! is 720
Enter a value for n (or a negative value to exit): 9 9! is 362880
Enter a value for n (or a negative value to exit): 11 11! is 39916800
Enter a value for n (or a negative value to exit): 20 20! is -2102132736
Enter a value for n (or a negative value to exit): 100 100! is 0
Enter a value for n (or a negative value to exit): 3 3! is 6
Enter a value for n (or a negative value to exit): -1 Come back soon!
```

Your program's output should mirror the output above. Include a header at the top of your source code and comments where appropriate. Be sure to follow the file name format described above when submitting your solution. (18)

2. There is clearly a problem with the program, 20! != -2102132736. What caused this result and what is the largest input that can safely be used with this program? - An overload and the highest input is 12

3. Create a program that reads an unsigned integer from a user. The integer can be in the range 0-255. Take the user-entered value and reverse the order of the bits. Write the new result back to the console. Your output should mirror the one below. Include a header at the top of your source code and comments where appropriate. Be sure to follow the file name format described above when submitting your solution. (30)

```
Welcome to the amazing bit swapper!
Enter a number in the range 0-255:  7       Flippy-Do:  224
Enter a number in the range 0-255:  1       Flippy-Do:  128
Enter a number in the range 0-255:  255     Flippy-Do:  255
Enter a number in the range 0-255:  0       Flippy-Do:  0
```

Drew Smith
CDA 3100

```
Enter a number in the range 0-255:  3        Flippy-Do:  192
```