The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Comp 411 Computer Organization
Spring 2009

Problem Set #4
Issued Tuesday, 3 March 2009; Due Tuesday, 24 March 2009

Homework Information: Some of the problems are probably too time-consuming to be done the night before the due date, so plan accordingly. Late homework will not be accepted.

Problem 0. Review Honor Code

Please review the Honor Code policies on the class website. Our department and the university have clearly described what types of collaboration are allowed or prohibited. Our department website explains how the Honor Code is to be observed with regard to programming assignments. For the programming part of Problem Set #3, you were allowed unlimited help from others. For this problem set, however, you are not allowed to collaborate, although you are free to seek help with concepts, programming style, and reasonable help with debugging your code, as long as you fully understand what you do and what you submit. Please take a moment to review the policies on the class website.

Problem 1. ASCII decimal numbers to binary numbers

Write a function `atoi` that converts a string of up to 8 ASCII digits ('0' through '9') into a 32-bit integer. In C, the declaration of this function might look like:

```
int atoi(char str[]);
```

The string ends with a null character (i.e., its ASCII code is 0), which will be the N+1th character, i.e., if the string is 4 digits, then the null will be the 5th character; if the string is 8 digits, the null will be the 9th character, and so on. You don't need to check for errors in the string, i.e., you may assume the string contains only characters '0' through '9'.

Then write the main program to test the above function. In particular, `main` calls the `atoi` function by placing the starting address of the string in $a0 and displays the returned integer value in the console window. Use syscall 1 to display the integer value, as described in the textbook pages A-43 to A-45 (on the CD), and verify that your `atoi` function is correct. The `main` program must use a variable `string1` as follows, and call `atoi` with it:

```
.data
string1: .asciiz "1237583"
```

Your program must run in the MARS simulator without error. We will test it by changing the value of `string1`.

Problem 2. Binary numbers to ASCII decimal numbers

Write a function `num2str` that converts a 32 bit integer into a null-terminated ascii string of characters (i.e., does the opposite of `atoi)`.

```
      void num2str(int num, char str[]);
```

Inside your function you may assume that there is sufficient space allocated in the output argument string to store all the characters of `num`.

Then write the `main` program to test this new function. In particular, `main` calls the num2str function and displays the generated string in the console window. Use syscall 4 to display the string, as described in the textbook pages A-43 to A-45. You'll want to make sure that the address you pass to your function points to a sufficiently large area to contain the resulting string (provide 11 bytes, so you can return a string with up to 10 digits plus a null).

The `main` program must declare a variable `num1` as follows, and call `num2str` with it:

```
      .data
num1: .word 254367
```

Your program must run in the MARS simulator without error. We will test it by changing the value of `num1`.

Problem 3. Computing Fibonacci numbers

The goal of this problem is to compute the Nth Fibonacci number, Fib(N), where Fib(0)=0, Fib(1)=1, and Fib(n)=Fib(n-1)+Fib(n-2).

```
      int fib(int n);
```

First, write a ***recursive procedure*** to compute Fibonacci numbers, that is, one that calls itself. You will need to manage the stack appropriately. Follow all conventions discussed in class regarding saved and temporary registers, passing arguments and values, etc.

You can assume that the Fibonacci number computed will fit within 32 bits (unsigned).

Once again, write the `main` program to test the above function. The `main` program must use a variable `num1` as follows, and call your Fibonacci function with it:

```
      .data
num1: .word 11
```

The result returned by the Fibonacci function must be printed by `main` using the syscall 1 method. Your program must run in the MARS simulator without error. We will test it by changing the value of `num1`.

Problem 4. Putting it all together!

The final objective is to put all of the above pieces together: given a string representing a decimal number, convert it into an integer, compute its Fibonacci number, convert the result into string form, and print the result on the console.

In particular, use the three functions you wrote above---`atoi`, `num2str` and `fib`---but leave the `main` functions you wrote to test them. Write a *new* `main` function to test everything together. Declare the following two string variables, one for the input (n), and the other to hold the result:

```
      .data
      string1: .asciiz "12"
```

```
    string2: .space 11
```

The `main` program will pass string1 to atoi to convert it into an integer, which is then passed to fib() to compute the Fibonacci number, whose result is then pass to num2str to convert to a string again and store in string2, which is finally printed to the console by main using syscall 4.

For full credit, your program should accept input numbers that have leading zeros to the left (e.g., 008, 00001234), but should print the answer with no leading zeros (i.e., 34, not 00000034).

Submission: Test your code using MARS and turn in a copy of the assembly language file (.asm) by email to our TA. Your email should have four attachments, one for each of the problems. Put "hw4" in the subject line, and be sure to email it by start of class on the due date. Do not turn in hard copy.

Grading: Your program must run in the MARS simulator without error. We will test it with our own data sets to determine your grade.

- We will first evaluate your solution to Problem 4, both using our data sets, and by examining your code. If your solution is correct, we may not need to evaluate Problems 1-3 to give you full credit.
- If your solution to Problem 4 is not correct, we will evaluate your solutions to Problems 1-3 to determine your partial credit.