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

Comp 411 Computer Organization

Spring 2010

Problem Set #4

Issued Wednesday, 17 February 2010; Due Wednesday, 3 March 2010

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.

You may work together on this assignment in small teams if you wish but the work you turn in must be your own. Include in the comments at the top of your program the names of the other students on your team. It is not acceptable to divide the programming among team members. To get maximum benefit from the exercise you need to struggle with these ideas yourself.

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 would 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 B-43 to B-45, 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 B-43 to B-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. Putting it all together!

The final objective is to put all of the above pieces together: given strings representing two decimal numbers, convert them into an integers, compute their product, and convert the result into string form, and print the result on the console.

Write a main program which uses atoi to convert each input string into a number, multiply them using the MIPS multiplier, and then convert the result back to a string using num2str. Display the result using syscall 4.

Your program should work like this:

```
char num1[] = "1234";
char num2[] = "4321";
char numres[11];

main() {
  int n1 = atoi(num1);
  int n2 = atoi(num2);
  int nr = n1 * n2;
  num2str(nr, numres);
  syscall(4, numres);
}
```

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: Turn in your 3 “asm” files on Blackboard. Include your name and anyone you collaborated with in the comments at the top of each file.

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 3, both using our data sets, and by examining your code. If your solution is correct, we may not need to evaluate Problems 1-2 to give you full credit.
- If your solution to Problem 3 is not correct, we will evaluate your solutions to Problems 1-2 to determine your partial credit.