CS61C Homework 3 - C to MIPS Practice Problems

TA: Sagar Karandikar

Spring 2015

This homework is an ungraded assignment designed to familiarize you with translating C code to MIPS. We will release solutions on Sunday, Feb 22nd, so that you may use them to study for the exam.

Problem 1 - Useful Snippets

In this section, we'll take the same problem (that of printing a string) and approach it using different C constructs. This should allow you to see how various C constructs are translated into MIPS.

Suppose that we have a `print` function, but that this function only takes one character and prints it to the screen. It expects the character to be in the lower 8 bits of $a0.

A) Translate into MIPS, while preserving the while loop structure:

```
void string_print(char *print_me) {
    while (*print_me != '\0') {
        print(*print_me);
        print_me++;
    }
}
```

B) Translate into MIPS, while preserving the for loop structure (your function is given the string length):

```
void string_print(char *print_me, int slen) {
    for (int i = 0; i < slen; i++) {
        print(*(print_me+i));
    }
}
```

C) Translate into MIPS, while preserving the do while loop structure:

```
void string_print(char *print_me) {
    if (!(*print_me)) {
        return;
```

```
    }
    do {
        print(*print_me);
        print_me++;
    } while (*print_me);
}
```

Problem 2 - Recursive Fibonacci

Convert the following recursive implementation of Fibonacci to MIPS. Do not convert it to an iterative solution.

```
int fib(int n) {
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }
    return fib(n-1) + fib(n-2);
}
```

Problem 3 - Memoized Fibonacci

Now, modify your recursive Fibonacci implementation to memoize results. For the sake of simplicity, you can assume that the array given to you (memolist) is at least n elements long for any n. Additionally, the array is initialized to all zeros.

```
int fib(int n, int* memolist) {
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }
    if (memolist[n]) {
        return memolist[n];
    }
    memolist[n] = fib(n-1, memolist) + fib(n-2, memolist);
    return memolist[n];
}
```

Problem 4 - Self-Modifying MIPS

Write a MIPS function that performs identically to this code when called many times in a row, but does not store the static variable in the static segment (or even the heap or stack):

```
short nextshort() {
    static short a = 0;
    return a++;
}
```

Tips/Hints:

- You can assume that the `short` type is 16 bits wide. `short`s represent signed numbers.
- You can assume that your MIPS code is allowed to modify any part of memory.
- See the title of this question.