Homework #1: MIPS Assembly Programming

Due: Oct 10, 11:59 PM
Responsible TA: Jihun Baek (qorwlgns444@unist.ac.kr)

1 Assignment Description

In this assignment, you will learn about the MIPS instruction set through hands-on coding in MIPS assembly language. Specifically, you will implement a program that generates the Fibonacci sequence using recursive function calls.

2 Submission

- Late submission will be assessed a penalty of 10% per day (We will only accept late submissions of up to 3 days).
- Submit one assembly file (`Your_ID-hw1.s`) to Blackbored. For example, if your ID is 20231234, then you should submit a file named `20231234-hw1.s`.
- The submitted file must include line-by-line comments, starting with `#`, explaining the meaning of each instruction.

3 MIPS Simulation Environment

In this assignment, you will use a simulator called QtSpim [3], which allows you to run MIPS assembly language in your computer environment without requiring actual MIPS CPU hardware. This tool enables you to execute and debug (step-by-step execution, setting breakpoints) your MIPS assembly (`.s`) files. You are required to install QtSpim version 9.1.24 in order to carry out this assignment. Here, we are sharing a video tutorial [2] on how to use QtSpim. Please refer to it for guidance.

4 Guidelines

You are required to write assembly code corresponding to the C code provided below. Specifically, you need to write a program that prints the Fibonacci number [4]. When the user inputs the index for the Fibonacci sequence (as shown in Line 19), your program must print the corresponding Fibonacci number (as shown in Line 20). §6 describes the possible range of the input index. Note that the Fibonacci number should be calculated using a recursive function (Line 5). As shown in Line 21, you must print the total number of recursive calls that invoked.

```
1  #include <stdio.h>
2
3  int cnt = 0;
4
5  int fibonacci(int n) {
6      cnt = cnt + 1;
7      if(n == 0)
8          return 0;
9      if (n == 1)
10         return 1;
```

```c
    else
        return fibonacci(n-1) + fibonacci(n-2);
}

int main() {
    int n;

    printf("The index for the Fibonacci sequence: ");
    scanf("%d", &n);
    printf("Fibonacci number: %d\n", fibonacci(n));
    printf("The number of fibonacci function calls: %d\n", cnt);
    return 0;
}
```

Through this assignment, you will learn how stack frames are created and utilized. You will need to implement a recursive function by using the `jal` instruction for function calls and `jr $ra` for returns, as well as managing the stack using the `$sp` register.

You may want to start the assignment from the assembly file provided below. Specifically, QtSpim offers a small set of operating-system-like services (e.g., `print`) through the MIPS system call (`syscall`) instruction. To request a service, a program loads the system call number [6] into register `$v0` and the arguments into registers `$a0`, `$a1`, `$a2`, and `$a3`. System calls that return values place their result in register `$v0`.

```
        .data                   # Data segment
prompt: .asciiz "The index for the Fibonacci sequence: " # Prompt for input
fib_result: .asciiz "Fibonacci number: "          # Format for Fibonacci result
call_count: .asciiz "The number of fibonacci function calls: " # Format for call count
newline: .asciiz "\n"                           # Newline string

        .text                   # Code segment
        .globl main             # Entry point

main:

        # Print "The index for the Fibonacci sequence: "
        li $v0, 4               # syscall for print_string
        la $a0, prompt          # Load address of prompt
        syscall

        # Read integer input (n)
        li $v0, 5               # syscall for read_int
        syscall
        move $a0, $v0           # Move input (n) to $a0 for fibonacci call

        # Call fibonacci function
        jal fibonacci           # Call fibonacci(n)


        # Exit program
        li $v0, 10              # syscall for exit
        syscall

fibonacci:
        . . .
```

5 Examples

Here are examples of the expected standard input and output. Note that `<EOF>` is specified to indicate the end of output rather than being part of the actual standard output. Therefore, at the very end of the program execution, the program

should print "`The number of Fibonacci function calls:  N`" followed by a newline, and then exit.

5.1 Example #1

```
The index for the Fibonacci sequence: 4
Fibonacci number: 3
The number of fibonacci function calls: 9
<EOF>
```

5.2 Example #2

```
The index for the Fibonacci sequence: 14
Fibonacci number: 377
The number of fibonacci function calls: 1219
<EOF>
```

5.3 Example #3

```
The index for the Fibonacci sequence: 21
Fibonacci number: 10946
The number of fibonacci function calls: 35421
<EOF>
```

6 Misc

- **Be careful about plagiarism!** We will conduct strict cross-plagiarism detection for evaluation, including a series of answer generated by ChatGPT, code found online, and your submissions. Therefore, we do not recommend consulting ChatGPT or online solutions. Last semester, we identified several cases of plagiarism using an automated tool. If you are found to be engaged in "deep collaboration" with other students, both the provider and the recipient will receive penalties, including, in the worst case, receiving an F grade.
- **MIPS instruction set.** Detailed information regarding the instructions can be found on the attached MIPS green sheet page [1]. You can also use MIPS pseudoinstructions [5].
- **Input index range.** The input index for the Fibonacci sequence is expected to be a positive integer, specifically in the range of 1 to 30. You do not need to account for exceptional cases, such as negative numbers or values outside this range, as we will only consider inputs within the specified range during grading.
- **Grading policy.** We will grade strictly based on the accuracy of the output relative to the input. No excuses will be accepted for discrepancies. We recommend starting your work from the assembly file provided in §4 to ensure the accuracy of the printed text.
- **Questions.** If you have any requests or questions (technical difficulties, late submission due to inevitable circumstances, etc.), please ask the TAs on Blackboard. We generally encourage the use of Blackboard for discussions. However, for urgent issues or secret issues, you can send an email to the responsible TA, Jihun Baek.

References

[1] 2009. MIPS Reference Data. https://websec-lab.github.io/courses/2024f-cse261/metarials/MIPS_Green_Sheet.pdf.

[2] 2015. How to Program in MIPS! (QTSpim) (Beginner). https://www.youtube.com/watch?v=wlssbwmPfnc.

[3] 2023. SPIM: A MIPS32 Simulator. https://spimsimulator.sourceforge.net/.

[4] 2024. Fibonacci sequence. https://en.wikipedia.org/wiki/Fibonacci_sequence.

[5] 2024. MIPS Pseudoinstructions. https://websec-lab.github.io/courses/2024f-cse261/metarials/MIPS_pseudoinstructions.pdf.

[6] 2024. MIPS system calls. https://peterfab.com/ref/mips/syscalls.html.