CS 341 - Lab 04 - Using a gcc Cross-Compiler to generate MIPS Assembly Code and Understand Optimizations

General Instructions:

1. Create a directory <roll_number>_<lab_no>. Store all the relevant files in this directory.
2. There are various questions in this lab which you need to answer in a text file <roll_number>_<lab_no>.txt
3. While submitting (on moodle), you have to create a tar.gz or zip of the entire <rollno>_<labno> directory in which all your relevant files reside.
4. The code you write should be well commented and easily readable. Violation of this could result in a penalty of half the points.

[5+5 marks] Question 1. Convert(Blindly translate) the following segments of C++ code to MIPS assembly code. Write the two codes in files gcd_basic.s and fibonacci_basic.s
You can store the result of the function calls in any 't' register.
GCD of two numbers(gcd.c):

```
int gcd(int a, int b){
        for(;;;){
                if(x==y) return x;
                else if(x>y) x = x-y;
                else y = y-x;
        }
}
int main(){
        int a = 10, b = 75;
        int  c = gcd(a,b);
        syscall_print_int(c);
        syscall_print_newline();
}
```

Fibonacci sequence(fibonacci.c):

```
int fibonacci(int first, int second, int  n){
        if(n<1) return -1;
        if(n==1) return first;
        if(n==2) return second;
        return fibonacci(first,second,n-1) + fibonacci(first,second,n-2);
}
```

```
int main(){
        int a = fibonacci(1,2,10);
        syscall_print_int(a);
        syscall_print_newline();
}
```
Note: If you do not follow the standard MIPS conventions of caller and callee saved registers, and argument/return registers, you won't get credit for this question.

Introduction To Cross-Compiler
[2.5 marks] Question 2[A]: What is a cross-compiler? What is the host platform? What is the target platform? Refer to the internet for more information.

You can answer this question after you complete your lab:
[2.5 marks] Question 2[B]: What is the host platform you have used? What is the target platform you have used?

Compiling The Code Using Cross-Compiler
Compile both the programs gcd.c and fibonacci.c using the cross-compiler mips-linux-gcc, using the "-S" option. Use the command "./opt/mips-cross/bin/mips-linux-gcc -S <filename>". Here <filename> will be substituted by gcd.c and fibonacci.c. This should give you gcd.s and fibonacci.s. Note that you cannot take input from console but you have to statically insert it, like in the given **gcd.c** file. Also, for printing output, use syscall_print_int() and syscall_print_newline() as used in the above files and **DON'T** use library calls like cout and printf.

The above assembly codes need a few modifications before they can be run on qtspim/spim:

1. There are a few assembler directives which spim/xspim do not understand. Use the program "remove_asm_directives.pl", available on piazza, to remove these. Call the new file as "gcd_noopt.s" and "fibonacci_noopt.s". Command line statement is "./remove_asm_directives.pl gcd.s gcd_noopt.s". Similarly for fibonacci.s.
2. Even now, you cannot load the program, since one of the label formats used by gcc is not understood by spim/xspim. This is the "1f" and "1" labels. Edit the "gcd_noopt.s" & "fibonacci_noopt" files and make these both "L1F"
3. You should also define "result" in the data segment. And also define the "syscall_print" procedure. You can use the "asmutils.s" file from piazza for these. You can append the contents of "asmutils.s" to both the files.

You should be able to load both the programs and run them.

[5+5 marks] Question 3: Comment both the files. Specifically comment on the various parts of the stack frame for the functions 'gcd' and 'fibonacci'. Also mention which parts you didn't understand.

Getting the Instruction Execution Count
For getting the instruction execution count, it is preferable to use the command-line spim tool. If it is not present in your path variable, you can access its binary on the path "/usr/bin/spim" on the Lab machines. Add this binary to your path variable for easier use.

Learn to use the command-line "spim" (not qtspim/xspim) and use appropriate arguments to make it run gcd_noopt.s and fibonacci_noopt.s

GCC Compiler Optimization
Now compile both the C files using the "-O1" option, i.e. optimization level 1, "gcd-opt.s" and "fibonacci-opt.s". As earlier use the "remove_asm_utils.pl" and "asmutils.s" files to run the files on spim.

[15 marks] Question 4: Execute the assembly codes on the command-line spim and use appropriate arguments to get the exact instruction execution count for(equal points for both fibonacci and gcd):

1. Your handwritten MIPS code**[5 marks]**
2. Generated assembly code without optimization**[5 marks]**
3. Generated assembly code with "-O1" optimization**[5 marks]**

Prepare a table of instruction counts for your handwritten programs, the generated assembly with and without optimization turned on.

[10 marks] Question 5: Explain the differences(e.g. Use of stack, use of registers, etc), if any that you observed in the three versions of the files.
[Bonus][5 mark] Question 6: Try to reason out the optimizations made using the flag "-O1".