

**ECE2030A Introduction to Computer Engineering
Fall 2008**

Homework Assignment #4

Assigned 11/24/08

Due 12/08/08 11:59am (See instructions)

No late turn-in accepted

Objective: In this assignment, you will learn to use an architecture simulator called “SPIM” and implement MIPS assembly programs running on top of SPIM to solve the following two problems. The information of the toolkit including installation and download instructions can be found at <http://pages.cs.wisc.edu/~larus/spim.html>. The tool can be installed on several platforms. You may want to start this assignment early since it may take a longer ramp for some in learning assembly programming.

Material: For the two problems below, you will use MIPS R2000/3000 assembly language. You are allowed to use the “pseudo instruction” provided by SPIM. Note that pseudo instructions are not actual instructions defined and assigned encoding space in the MIPS ISA but will be interpreted into legitimate MIPS instructions by the SPIM assembly when you execute them. Please read the document “Assemblers, Linkers, and the SPIM Simulator” post on our course website for more information.

Turn-in and Checkoff: There are **two steps** for turning in this assignment. First, you need to email your assembly programs to the TA Thanh Tran (thanh.tran@gatech.edu) prior to the due time. All late emails (based on the timestamp) will not receive any credit. Please type your name and email address in the header comment area in your programs. Second, an in-person check-off in the lab will be scheduled. Our TA will go through your code one-on-one, ask you some questions, and give you credit based on the codes you email him. TA will send out further instructions regarding the checkoff procedure. It will take place in the final week.

1. (50%) **Finding “perfect numbers”.** You can download the program “perfectnum.s” from the link in course website as the template to develop your code, or you can choose to start from scratch. Please write a MIPS assembly program to find all the perfect numbers in-between a given range $[x, y]$. A “*perfect number*” is a positive integer which is equivalent to the summation of all its factors (excluded itself.) For example, the smallest “*perfect number*” is 6 since the factors of 6 are 1, 2, 3 and $6 = 1 + 2 + 3$. (Hint: To be able to find all the perfect numbers, your program should factorize each given integer number in the range.) Your MIPS assembly program needs to do the following

- (1) Take two integer numbers x and y as input that forms the range
- (2) Print out all the perfect numbers in the range (there are not too many of them.)
- (3) Print a comma “,” (as provided in template as **comma**) every 5000 numbers, so we know your program is still alive and running if no perfect number is found amid execution.
- (4) At the end of your program execution, print out “*Program Ends*” (**pend** as defined in template program) on the SPIM console so we know when your program stops”.

2. (50%) **Cracking message.** Download the program “decipher.s” from the course website. This program contains an encrypted message in the .data segment labeled “crypt”. If you execute this program directly with SPIM, it will print out unreadable characters in the SPIM console window. Your job is to add a sequence of MIPS instructions in the decipher.s to crack the encrypted message.

My encryption was done by the following steps for each “32-bit word” message and stored in the **crypt** part of the **.data** segment in hex.

1. Rotate each 32-bit word to the left by 3 bits.
2. Bitwise eXclusive-OR (XOR) this 32-bit word with one word from the “One-Time-Pad” key (*OneTimePad* is also declared in the .data segment).
3. Move to the next 32-bit word of the message and the One-Time Pad. Repeat 1 and 2 till the end of the message. (you can consider 0x00000000 ends the encrypted message.)

Now you need to reverse the encryption process above to decrypt the message. You have to (1) **decrypt** each word-by-word, and (2) **overwrite** the memory location where the ciphertext named “**crypt**” was stored with your decrypted plaintext. After you are done with all the words (by detecting a zero word 0x00000000), you call “`syscall`” to print out your decrypted new message on SPIM console (Actually, this part of the code is provided at the end of the file decipher.s). The plaintext message is in English once you decrypt it. We will validate your program based on the printout in SPIM console.