

Homework 5

Due: 4 Nov 2009 1:55pm

All homeworks are due at 1:55pm in the **CS 31 bin** on the second floor. No late homeworks are accepted.

Please include your *login name* on each piece of paper you hand in, and please staple your pages together before handing in.

Some of the problems on this homework must be handed in electronically. Please do each problem in a separate file. Run `cs031_handin hw05` in the directory where your files are stored to hand in. You will be prompted for the name of each file.

Your assembly code should not look like it was compiled from higher-level code, please do not mess with the stack in strange ways and make sure to return properly from functions.

Problem 5.1

In CS32 you'll become quite familiar with two types of errors: bus errors and segmentation faults. A segmentation fault is caused when you try to access memory at an address that is valid, but that your program is not allowed to access. A bus error is caused when you try to access an invalid address. Write two small segments of MIPS code (a few lines each) that will cause each of these errors.

Problem 5.2

A Horcrux allows a powerful wizard to save a piece of his soul in an object. This was actually the inspiration for the muggle device of saved registers. Luckily, you do not need to commit murder to use registers.

Write two segments of code: one for the beginning and one for the end of a subroutine that uses the saved registers `$s0`, `$s1`, and `$s2`. Your first segment of code is the beginning of the subroutine, and your second segment finishes off the subroutine. Assume that any number of next-level subroutine calls are made from the subroutine, so you should save and restore all necessary aspects of the state.

You do not need to hand in an electronic file for this question. A paper printout of your assembly code will do.

Problem 5.3

For this problem you will be writing a recursive subroutine in assembly and some code that will come in very handy for Life.

You will perform the following tasks:

- Write a recursive subroutine to multiply two numbers using repeated addition. You can assume that the numbers are non-negative and relatively small (ie., there will be no need to check for overflows). No credit will be given for a non-recursive solution. This means **the procedure must use jal to call itself**. (This also means you also shouldn't perform tail recursion optimization, since it would make your solution identical to an iterative one, (and you will lose points). If you don't know what tail recursion optimization is, don't worry about it.)
- Write a subroutine to print out a two-dimensional array of numbers where the address of the array and its dimensions are declared in the data section of your program.
- Write a program using these two subroutines that will create and print a simple multiplication table. The dimensions of the table should be declared in the data section of your program and easily changed. The table may or may not be square. There's no need to be clever here, just iterating over the entire table and performing the appropriate multiplication will be fine.

As an example, the output of your program for a 4x5 multiplication table should look something like this:

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20

You can use a tab character to make your numbers align.

Hand this in electronically, as described above.