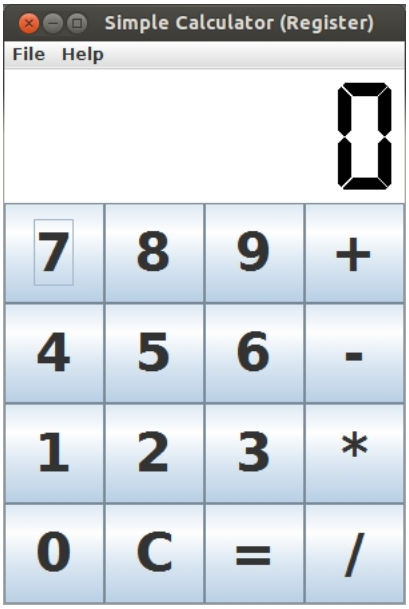
Project 1 - Calculator

CS 0447 — Computer Organization & Assembly Language

Check the Due Date on the CourseWeb

The purpose of this project is for you to practice writing assembly language to interact with output/input hardware. The hardware for this project is a dollar store calculator as shown below (on left):



The Calculator (Mars Tool) that we are going to use for this project is shown above (on right). This tool can be found in `SimpleCalculatorRegister.zip` located in the CourseWeb under this project. Extract all files to your `[..]/mars4_5/mars/tools` directory. If you extract all files to the right directory, when you run the MARS program, you should see "Simple Calculator (Register) V0.1" under the "Tools" menu.

Introduction to the Simple Calculator (Mars Tool)

The Simple Calculator (Mars Tool) consists of two parts, LCD display and a key pad. The LCD display is connected directly to the register `$t8`. It simply display the value of the register `$t8` on the LCD screen. Thus, whatever number your program wants to show on the calculator's LCD screen, simply put it in the register `$t8`. This display supports both positive and negative number. If the value stored in the register `$t8` is a negative value, a minus symbol will be displayed before the value (e.g., -123). **Note** that there is no dot symbol (.) on the key pad. This Simple Calculator can only display integer numbers. For this project, we will focus only on integer operations. Thus, the division must be the integer division (e.g., 5 / 2 = 2).

There are 16 buttons on the key pad, 0 to 9, +, −, ∗, /, =, and C (clear). This key pad is connected directly to the register `$t9`. If the value of the register `$t9` is currently equal to 0 and a button is pressed, the Most Significant Bit (MSB) of the register `$t9` (bit 31) will be 1 and the last four bit of the register `$t9` (bit 0 to bit 3) will be changed according to the pressed button as shown in Table 1. **Note** that these buttons are disabled if the content of the register `$t9` is

Button	$b_3b_2b_1b_0$
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
+	1010
−	1011
∗	1100
/	1101
=	1110
C	1111

Table 1: Associated Values of Buttons

not equal to 0. Thus, it is your responsibility to change the content of the register `$t9` to 0 when your program is ready to get a new key pad input. **Note** that the Simple Calculator does not use `syscall`. Thus, there is no special instruction for your program to wait for an input from the key pad. Your program has to loop checking the value of the register `$t9` until it is not zero. The non zero value will be the input. Again, **do not** forget that when an input exists, the MSB of the register `$t9` will be 1 and the last four digits are used to represent a pressed button.

What to Do?

For this project, write a MIPS assembly program such that when the program is running, the Simple Calculator will behave just like a dollar store calculator. **Note** that there are various kind of dollar store calculators. They may behave differently. So, let's follow the behavior as shown in Table 2. According to the Table 2, we assume that to calculate a `result`, we need three variables, two for operands and one for an operator. Let's call these `operand1`, `operand2`, and `operator`, respectively. According to the Table 2, if a user press operator key more than one time consecutively, this calculator will use the last pressed operator key as the operator. For example, if user press the following key 5, *, -, 9, and =, (in that order) your calculator should show the result -4 because 5 - 9 = -4. As you may notice, because of this type of behavior, we cannot have a negative number as the second operand. For example, we a user wants to calculate 49 * (-52), the user has to calculate (-52) * 49 instead which can be achieve by the following series of key presses, C, -, 5, 2, *, 4, 9, and =. Note that when user press C, -, 5, and 2, the screen should show 52 because user is actually

calculating 0 - 52.

Requirements

The following are requirements that you **must** follow:

- Your program should be named `calculator.asm`
- Since the Simple Calculator (Mars Tool) uses registers `$t8` and `$t9`, **do not use** these registers for any other purpose.
- You can only use registers `$s0` to `$s7` and `$t0` to `$t9` in your program.
- Do not use any system calls (`syscall`). Do not worry that your calculator will run indefinitely.
- No functions in your program. In other words, do not use instructions `jal` and `jr` or stack pointer (`$sp`). You may already learn how to create a function in assembly language and feel tempted to do so. **DON'T**.
- No memory reference instructions are allowed (no load and store instructions).
- Your program should have only the text segment (`.text`) and no data segment (`.data`).
- You are not allowed to use multiplication and division instructions. Simply use loop to achieve those type of calculations.
- Try not to use pseudo instructions. This project can be done by using only `add`, `addi`, `sll`, `srl`, `slt`, `beq`, `bne`, and `j` instructions and a lot of labels. Note that this requirement is not a definite requirement. If you need to use other instructions, you are allowed to do so. The main reason why we should not use pseudo instruction is because we **may** use this program for your last project. If you use a lot of pseudo instructions or a complex instructions, you may have to rewrite your program for your last project.

Hints

- Focus on a small task at a time. For example:
  1. Write the program to receive a numeric keypad and show the correct number of the LCD screen
  2. Extend from 1 to support a series of numeric keypad (e.g., 5, 2, and 9 should result in 529 on the LCD screen). Note that multiply by 10 can be easily achieved by shift left three follows by adding with the original number twice.
  3. Extend from 2 to support C (clear) button.
  4. Practice writing multiplication and division.

Submission

The due date of this project is stated on the CourseWeb. Late submissions will not be accepted. You should submit the file `calculator.asm` via CourseWeb.

State	Input	Actions
0 (Start)	N/A	Set `operand1` and `operand2` to 0 Set `operator` to nothing Display `operand1` Go to state 1
1	0 - 9	`operand1` = (`operand1 * 10`) + Input Display `operand1` Go back to State 1
	+,−,∗,/	`operator` = Input Display `operand1` Go to State 2
	=	`result` = `operand1` Display `result` Go to State 4
	C	Go to State 0
2	0 - 9	`operand2` = (`operand2 * 10`) + Input Display `operand2` Go to State 3
	+,−,∗,/	`operator` = Input Display `operand1` Go back to State 2
	=	`result` = `operand1` Display `result` Go to State 4
	C	Go to State 0
3	0 - 9	`operand2` = (`operand2 * 10`) + Input Display `operand2` Go back to State 3

	+,−,∗,/	`result` = `operand1 operator operand2`
		Display `result`
		`operand1` = `result`
		`operand2` = 0
		`operator` = Input
		Go to State 2

	=	`result` = `operand1 operator operand2` Display `result` `operand2` = 0 Go to State 4
	C	Go to State 0
4	0 - 9	`operand1` = Input Display `operand1` Go to State 1
	+,−,∗,/	`operand1` = `result` `operator` = Input Go to State 2
	=	Display `result` Go back to State 4
	C	Go to State 0

Table 2: Behavior of a Dollar Store Calculator