CSE 410 Assignment 2 (Program)

Spring 2008

Due: Midnight, Thursday 4/17/2008

Download the file ext_euclid.s, which implements the extended Euclidian algorithm for computing the greatest common divisor of a given two numbers.
Here is the pseudocode for the algorithm:
http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm#Iterative_method

Open the program up in SPIM and use SPIM to analyze it.

Answer the following questions.

1. What instructions replace the pseudo-instructions **`move`**, **`li`** when the code is stored in memory? What happens to the load address instruction **`la $a0,newline`**, and why?
2. What happens to the instructions **`divu $t6, $s0, $s1`** and **`remu $s1, $s0, $s1`** when they are loaded to SPIM? What is the difference between the new code inserted by SPIM and the original instructions?
3. Before the program reaches **`main`**, it executes several instructions that initialize the stack pointers and argument pointers. These are inserted automatically by SPIM. The program then jumps to the beginning of `main` with the **`jal`** instruction at location `0x00400014`. What happens to **`R31 (ra)`** after this instruction is executed? What is the significance of this value?
4. When the program terminates, what values are stored in the following registers (use hexadecimal)?
   a. **`R8 (t0)`**
   b. **`R9 (t1)`**
   c. **`R16 (s0)`**
   d. **`R17 (s1)`**
5. The current code contains four branch instructions to implement the loop. Modify the program so that each iteration of the loop only executes **at most one** branch instruction instead of four. Put a comment that begins with the note **`#CHANGED`** to the right of each line of code that you change to do this. Run your new code in SPIM to ensure that it produces the same results as the original version.
6. The current program throws an exception if the user enters a negative number for any of the inputs. Modify the program so that if the user enters a negative number for any of the inputs, the program displays the message "Unexpected input, exiting." and terminates (simply, goes to the label "**`end`**").
7. [**Extra credit**] Rewrite the code so that the code between the labels **`initialize`** and **`store_result`** is contained in a separate procedure with label **`gcd,`** which takes 2 integer arguments and returns the greatest common divisor of the given numbers. Rewrite the code starting from the label **`main`** so that it calls **`gcd`** with the numbers given by the user and prints the returned value of **`gcd`** at the end. Use the calling convention taught in class, including argument registers, stacks and stored registers.

Use electronic submission via the Catalyst tool (your modified programs from questions 5, 6 and 7 along with written answers in a text document):
https://catalysttools.washington.edu/collectit/dropbox/telmas/2218