CS2630: Computer Organization
Homework 2
MIPS assembly programming
Due February 2, 2017, 11:59pm

Instructions: You must complete both assembly files hw2_mult4.asm and hw2_inserts.asm and submit them to Assignments > Homework 2 on ICON. Make sure your programs assemble and run successfully, and make sure to test them on multiple inputs. We will test your code automatically, although we may also inspect your code.

Goals for this assignment
- Gain familiarity with the register-transfer style of programming
- Learn the MIPS instructions
- Get experience writing MIPS programs with load, store, and branch
- Apply the data representation concepts you’ve learned up to this point

Part 0

Do this part immediately!
This assignment requires the MARS simulator. Just as Eclipse may have been your IDE for Java programming, MARS will be your IDE for MIPS programming. Go to the resources page for directions on downloading and running MARS (http://homepage.cs.uiowa.edu/~bdmyers/cs2630_sp17/resources/).

Next, watch the MARS tutorial videos linked from http://homepage.cs.uiowa.edu/~bdmyers/cs2630_sp17/resources/. There are 3 right now, but lessons 01 and 02 are most relevant to HW2.

Part 1

1. Write a MIPS program that checks whether the sum of a pair of integers is a multiple of 4.

Assembly program hw1_mult4.asm contains skeleton code that reads two integers from stdin and stores them in $t0 and $t1. Your code must set $s0 to 1 if the sum of the two integers is a multiple 4 and set $s0 to -1 if not.
• TIP: If you are running your program in MARS, you have two choices for providing the two integers as input.
  
  o default: When you press run (කාටි) the cursor will go down to the console. Enter the first integer <ENTER> the second integer. Here is an example of entering 3 and 6.

  ![](image)

  o popup window: Check the option “Popup dialog for input sycalls”. This option will make it so that when the program runs, there will be two popup windows, one for each input.

  ![Settings](image)

  **Show Labels Window (symbol table)**
  **Program arguments provided to MIPS program**
  **Popup dialog for input sycalls (5,6,7,8,12)**

  TIP: For those students that are comfortable with the command line, alternatively use

  ```
  java -jar Mars4_5.jar '$s0' hw2_mult4.asm
  ```

to test your program quickly

(see [http://courses.missouristate.edu/KenVollmar/mars/Help/MarsHelpCommand.html](http://courses.missouristate.edu/KenVollmar/mars/Help/MarsHelpCommand.html) for more details). You can even give it a file as stdin

  ```
  cat testinput | java -jar Mars4_5.jar '$s0' hw2_mult4.asm
  ```

  2. In this problem, you’ll complete another assembly program, found in `hw2_inserts.asm`, where you will insert the integer 42 into the front of an array of 4-byte integers. This program takes command line arguments for an array.

  **An example**

  command line arguments: 1 2 3 4
  
  • beware of the restrictive formatting: (single-digit positive, decimal integers)

  Starting state for the example
• register $s0 = 3  (the number of elements in the array)
• register $s1 = 0x10040000  (address of the start of the array)
• input array (stored in memory)

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10040000</td>
<td>1</td>
</tr>
<tr>
<td>0x10040004</td>
<td>2</td>
</tr>
<tr>
<td>0x10040008</td>
<td>3</td>
</tr>
<tr>
<td>0x100400C</td>
<td>4</td>
</tr>
</tbody>
</table>

At the end of your program, the array should contain the number 42 at the front, with everything else moved down.

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10040000</td>
<td>42</td>
</tr>
<tr>
<td>0x10040004</td>
<td>1</td>
</tr>
<tr>
<td>0x10040008</td>
<td>2</td>
</tr>
<tr>
<td>0x100400C</td>
<td>3</td>
</tr>
<tr>
<td>0x1004010</td>
<td>4</td>
</tr>
</tbody>
</table>

Notes
• The program already runs (initializes the array but doesn’t do anything to it), so take some time to run it and poke around the memory and registers to understand what is going on before you start writing your solution.
• Your program must work on arrays of any integers of any length, so test it on enough inputs that you are sure of the correctness.
• The command line arguments must be formatted correctly, otherwise the program will end with ERROR (or worse it will silently corrupt memory).
• It is very important to check the Registers window and Data Segment window to view the values stored in different register and memory locations. The array address will start in the smallest addresses of the heap.
• You have two options for passing command line arguments to the program:
  A. In MARS: check Settings > “Program arguments provided to MIPS program”, and put the arguments in the Program Arguments text box at the top of the Text Segment window in Execute mode.
  B. On command line: here is an example invocation of the program
     java -jar Mars4_5.jar '0x10040000-0x10040020'
     hw2_inserts.asm pa 1 2 3 4
     You can provide a memory address range to print out the contents of memory at the end of the program. (more details at

http://courses.missouristate.edu/KenVollmar/mars/Help/MarsHelpCommand.html )