Lab 2
Load Store Unit

This lab session will demonstrate the usage of the Load/Store Unit. The Load/Store Unit (LSU) executes all load and store instructions and provides the data transfer interface between the GPRs, FPRs, and the cache/memory subsystem. The LSU calculates effective addresses, performs data alignment, and provides sequencing for load/store string and multiple instructions.

In the following exercises all inputs are assumed to be valid. i.e. no need to check that the input values are out of range.

Arrays

1. Write a program that emulates a two dimensional array. The array size is 5x20 cells, each cell 32 bit wide. Register R5 will be used to define the action performed using the array:

   if R5=0 then the action is READ:
      R6 = Array[R3][R4]
   Else the action is WRITE:
      Array[R3][R4] = R6

Use the following registers as parameters:

<table>
<thead>
<tr>
<th>Register</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3</td>
<td>Row Number</td>
</tr>
<tr>
<td>R4</td>
<td>Column Number</td>
</tr>
<tr>
<td>R5</td>
<td>Action</td>
</tr>
<tr>
<td>R6</td>
<td>Data</td>
</tr>
</tbody>
</table>
Multiple word access

2. Write a program that copies memory cells from source to destination address, use the following registers as parameters:

<table>
<thead>
<tr>
<th>Register</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3</td>
<td>Source address</td>
</tr>
<tr>
<td>R4</td>
<td>Destination address</td>
</tr>
<tr>
<td>R5</td>
<td>Number of bytes to copy</td>
</tr>
</tbody>
</table>

R5 is an unsigned number with a max value of 40.
Source & Destination memory addresses are aligned.

Try to make the code as effective (fast) as possible.
Read Modify Write

3. Write a program that will receive the following parameters

<table>
<thead>
<tr>
<th>Register</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3</td>
<td>Destination address</td>
</tr>
<tr>
<td>R4</td>
<td>Mask</td>
</tr>
<tr>
<td>R5</td>
<td>Value</td>
</tr>
</tbody>
</table>

The program will overwrite the data (32 bit data) in the memory destination address (R3) according to the following rule: If the corresponding bit in the Mask register (R4) is '1' then overwrite the corresponding bit from the Value register (R5) otherwise keep the original bit's value.

For example, suppose that the 32 bit value 0x0000_00F0 is stored at memory address 0x0000_1000 pointed by R3. The following table shows the new value that is to be copied to the memory address 0x0000_1000 pointed by R3 at the end of the run.

<table>
<thead>
<tr>
<th>Original Value</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R4 (Mask)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R5 (Value)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>New Value</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Note!
1. The values shown are in binary base
2. For clarity only 8 bits are shown (your program should deal with 32 bit data)
3. Red marked cells indicate that the bit was copied from the Value register (R5).

Good Luck