Homework #7: MAL/TAL and interrupts

1. Hand assemble the following MAL code, that means convert it to TAL, then convert it to machine code (in HEX). Start instructions at the address 0x0004 4800 and the procedure “bob” starts at address 0x0008 8000.

   Add $s1, $s2, 5
   bgt $s1, $s3, no
   jal bob
   no: mul $s4, $s4, $s1 # need only 32-bit result

2. Hand assemble the following MAL .text section. That is, convert to TAL, then to binary, then to HEX. Start instructions at the address 0x0008 88000 and the date starts at 0x0004 4400. Show what the .data section looks like.

   .data
   c .byte 'a'
   match .byte 5

   .text
   li $v0, 12 #getc code
   syscall
   move $t3, $v0
   sb $0, match #default failed
   lb $t0, c
   bne $t1, $t3, nope
   li $t2, 1
   sb $t1, match
   nope:

3. Give a correct TAL translation of the following MAL instruction, if “variable” has been assigned the address of 0x0104 8800

   la $t0, variable
4. How can the MAL “move” instruction be implemented in TAL?

5. Give two different ways that the following MAL instruction can be translated into TAL instructions, leave “fred” unresolved.
   \[
   \text{bgt } \$s1, \$s2, \text{fred}
   \]

6. Disassemble the following MIPS Machine code into TAL instructions. Make up label names when needed. The code starts at 0x0040 0000

<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0040 0000</td>
<td>0x1232 0002</td>
</tr>
<tr>
<td>0x0040 0004</td>
<td>0x0232 8824</td>
</tr>
<tr>
<td>0x0040 0008</td>
<td>0x0011 8882</td>
</tr>
<tr>
<td>0x0040 000C</td>
<td>0x2231 FFFD</td>
</tr>
</tbody>
</table>

7. What is polling (sometimes called spin waiting)?

8. What are the advantages and disadvantages of implementing polling and interrupts? What are the overheads of each?