ECE 3056

Exam I - Solutions
February 27th, 2014
3:00 pm – 4:25pm
1. **(35 pts)** Consider the following block of SPIM code. The text segment starts at 0x00400000 and the data segment starts at 0x10010000.

```
.data
label: .word 8, 16, 0x32, 0x64
L1: .byte 64, 32
L2: .asciiz “End 1”

.text
.globl main
main:   la $a0, label    #pass array base address
li $a1, 16       #pass array count
jal func        #function call

li $v0, 10      #terminate program
syscall

func:   move $v0, $a0     #get array base address
        move $v1, $a1     # get array count
        add $v1, $v1, $v0  #array ending address
        move $t1, $0     #initialize loop sum

loop:   lw $s6, 0($v0)      #load first va
        add $t1, $t1, $s6    #update sum
        addi $v0, $v0, 4     #update pointer
        slt $t0, $v0, $v1     #check for termination
        bne $t0, $0, loop   #onto next element
        move $v0, $t1        #pass sum back to main
        jr $ra
```

a. **(10 pts)** Show the hexadecimal contents of the data segment. Assume **little endian** storage.

<table>
<thead>
<tr>
<th>Word Address</th>
<th>Word Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10010000</td>
<td>0x00000008</td>
</tr>
<tr>
<td>0x10010004</td>
<td>0x00000010</td>
</tr>
<tr>
<td>0x10010008</td>
<td>0x00000032</td>
</tr>
<tr>
<td>0x1001000c</td>
<td>0x00000064</td>
</tr>
<tr>
<td>0x10010010</td>
<td>0x6E452040</td>
</tr>
<tr>
<td>0x10010014</td>
<td>0x00312064</td>
</tr>
<tr>
<td>0x10010018</td>
<td></td>
</tr>
<tr>
<td>0x1001001c</td>
<td></td>
</tr>
</tbody>
</table>
b. (8 pts) Provide the hexadecimal encodings of the first two instructions in the body if the loop.

\[
\text{lw} \ $s6, \ 0($v0) \quad 0x8c560000
\]

\[
\text{add} \ $t1, \ $t1, \ $s6 \quad 0x01364820
\]

c. (4 pts) What value is returned in register $t0 for the following instruction.

i. \ \text{lw} \ $t0, \ \text{loop}($0) \quad 0x8c560000

The address that is computed will be that of the \text{lw} instruction.

d. (5 pts) Imagine that the procedure \text{func} is independently compiled and linked. It is placed in memory at a location 0x00402004. Provide the hexadecimal encoding of the instruction \text{jal func} after it is patched by the linker.

\[0x0c100801\]

e. (8 pts) Consider the \text{bne} $t0, \ $t1, \ \text{Loop} instruction which is located at address 0x00400024. Which of the following addresses are feasible branch targets? To receive credit you must have a valid reason.

i. \ 0x00444000 \quad \text{yes/no. Why?}

No. The target is out of range of the 16 bit 2’s complement offset value

ii. \ 0x00400480 \quad \text{yes/no. Why?}

Yes. The target is in range of the 16 bit 2’s complement offset value
2. **(10 pts)** Consider the following SPIM program. The procedure solo uses registers $21, $22, and $8.

```
.data
L1:   .word 0x32, 104
      .align 2
Blank: .word
L2:   .space 64

.text
main: li $t0, 4
      move $t2, $zero
      li $a0, 1

loop: jal solo
      addi $t0, $t0, -1
      addi $a0, $a0, 1
      add $t2, $t2, $v0
      slt $t1, $t0, $zero
      bne $t1, $zero, loop
      li $v0, 10
      syscall
```

a. **(6 pts)** What registers must be saved on the stack, if any, on each call to the procedure solo? Distinguish between caller and callee.

**Caller:** $t0, $a0, $t1, $t2.

Remember to follow the convention. The problem tells you what solo uses so you can address the problem from the perspective of the caller and callee. In practice, the caller cannot “know” what registers the callee is using. This is the reason for the convention. You could avoid saving $t1 since the caller can know that it does not need the value to be saved across the call.

**Callee:** $21 ($s5), $22 ($s6)

b. **(4 pts)** Is the above code relocatable? Justify your answer.

This code is relocatable as long as the location of solo does not change and the code is relocated to remain within the same 256Mbyte segment of the address space.
3. **(15 pts)** Consider the single cycle SPIM datapath shown overleaf. You wish to add support for the `lui` instruction, e.g., `lui $t0, 0x4004`. You should be able to determine which of the known instruction formats is utilized by this instruction. First define the behavior of `lui` that you are implementing. There are several possible solutions. Your solution will be graded accordingly.

Complete the following.

a. Describe (in bullet form) the basic actions that need to be added. Be brief.

There are many alternative solutions. This is one of them.

The instruction definition is one where the upper bits of \( r^t \) are set to the value of the immediate operand and the lower order bits are zero

- Have a unit similar to sign extension that produces a 32-bit quantity as 0xHHHH0000, where 0xHHHH is the immediate operand.
- Make this new operand a third input to the ALUSrc mux.
- Perform addition (the other input is the contents of register 0 which \( rs \)).
- Write \( rt \)

b. Show the required changes to the datapath overleaf. To receive credit, i) I must be able to read all notations you make and ii) all hardware modifications must be clear. There should be no ambiguity.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>RegDst</th>
<th>ALUSrc</th>
<th>MemToReg</th>
<th>RegWrite</th>
<th>MemRead</th>
<th>MemWrite</th>
<th>ALUOp</th>
<th>Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>lui</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00</td>
<td>0</td>
</tr>
</tbody>
</table>