CIS 314 WINTER 2004           QUIZ #1: MIPS and SPIM   SOLUTION

1. (20 points) Given the following data segment declaration, what values are printed by each of
the following syscalls? Assume big endian.

Root: .word Left,Right,45
Left: .word -1,-1,16
Right: .word -1,-1,3
Authors: .asciiz “Patterson and Hennessey”

(a)
la $t0,Root
lw $t0,4($t0)
lw $a0,8($t0)
l $v0,1
syscall

ANS: 3

(b)
lb $a0,Right
li $v0,1
syscall

ANS: -1

(c)
lw $t2,Right
add $t2,$t2,5
la $a0,Authors
add $a0,$a0,$t2
li $v0,4
syscall

ANS: erson and Hennessey

(d)
lw $t0,Right
add $t0,$t2,5
la $t1,Left
la $t0,Authors
sub $a0,$t0,$t1
li $v0,1
syscall

ANS: 24

(e)
lbu $a0,Left
li $v0,1
syscall

ANS: 255

2. (40 points) Short answer.

(a) Almost all MIPS instructions are of the I or R-type. Why did the designers decide to make a
separate type for the jump instruction?
ANS: The jump instruction needs only 6 bits to specify the op code and the rest of the
instruction can be address; defining a format especially for this instruction allows it to
have the maximum range for a jump.

(b) The register $1 is reserved for the assembler. Why does the assembler need to use one of the
program registers? Give a plausible example of such a situation.
ANS: The assembler must generate code for pseudo instructions which often do not
correspond exactly to a machine language instruction and require some runtime
calculations. The assembler does these calculations in register $1. An example would be
in calculating the address for an instruction like “la $9,x” where the address for x is
computed in the two steps:
  lui $1,4097
  add $9,$t1,4

(c) If you lay out an nxm array of integers in row major order, what is the address of element [i,j]?
(Assume indexing starts at 0.) Give your answer in expression form.
ANS: (i*4*m) + j*4
(d) Write the declaration for a doubly linked, three node circular list in which each element stores one integer (pick the values 3, 6, and 9 for those integers).

\[
\text{ANS.} \quad \text{.data} \\
N1: \quad \text{.word} \quad \text{N2, N3, 3} \\
N2: \quad \text{.word} \quad \text{N1, N3, 6} \\
N3: \quad \text{.word} \quad \text{N2, N1, 9}
\]

(e) What is the difference between strings declared as `.ascii` and strings declared as `.asciiz`? When is it useful to declare something as `.asciiz` and why.

\[
\text{ANS:} \quad \text{Strings declared as `.ascii` are not terminated, strings declared as `.asciiz` are terminated with a byte of all 0s. Terminated strings are usual in output as the syscall is given the address of the start of the string and it continues printing until the end of the string is reached.}
\]

(f) Procedures get their arguments from the $a registers but there are only four of them. How does a procedure get its arguments if it has more than 4?

\[
\text{ANS:} \quad \text{The first four parameters are passed in the registers and the remaining are pushed on top of the stack by the calling program and removed from the stack by the callee.}
\]

(g) The conditional branch instructions all require an address which is specified in 16 bits. How is the full 32 bit address constructed?

\[
\text{ANS:} \quad \text{The 16 bit number specified is added to the current program counter value meaning that the address is relative to the current location.}
\]

(h) Write a segment of code to print the third character of a string named NAME, assuming it is declared to be an asciiz string of 20 characters.

\[
\text{ANS:} \quad \text{add a character to .data section to get termination character} \\
\text{Char:} \quad \text{.asciiz "\"} \\
\text{then code is} \\
\text{la} \quad \text{$t0, NAME} \quad \# \text{get addr of string} \\
\text{lb} \quad \text{$t0, 2($t0)} \quad \# \text{get third character} \\
\text{sb} \quad \text{$t0, Char} \quad \# \text{store it in Char to get termination} \\
\text{la} \quad \text{$a0, Char} \quad \# \text{print it} \\
\text{li} \quad \text{$v0, 4} \\
\text{syscall}
\]

3. (25 points) Write a program segment that reads through an array of 100 elements (assume it has been initialized by some other code) and sums all of the elements that are multiples of 3. You don’t have to write the entire program, just this one segment and any definitions it needs in the .data segment. COMMENT YOUR CODE APPROPRIATELY.

\[
\text{.data} \\
\text{List:} \quad \text{.space 400} \\
\text{Sum:} \quad \text{li} \quad \text{$t0, 0} \quad \# \text{initialize first array offset} \\
\text{li} \quad \text{$t1, 1396} \quad \# \text{final offset} \\
\text{li} \quad \text{$t3, 0} \quad \# \text{initialize sum} \\
\text{Loop:} \quad \text{bge} \quad \text{$t0, $t1, Endlt} \quad \# \text{test for array completed} \\
\text{lw} \quad \text{$t4, List($t0)} \quad \# \text{load array element} \\
\text{div} \quad \text{$t5, $t5, 3} \quad \# \text{divide by 3} \\
\text{mfhi} \quad \text{$t5} \quad \# \text{get remainder} \\
\text{bnez} \quad \text{inc} \quad \# \text{skip summing if divisible by 3}
\]
add $t3,$t3,$t4      # add to running sum
Inc:    add $t0,$t0,4       # increment offset for next element
j    Loop
EndIt:
move $v0,$t3  # return result
jr   $ra
4. (15 points) Below are the skeletons of a program and two procedures: P1 and P2. The main program calls P1 and P1 calls P2; P2 does not call any procedures.

You do not need to know the details of these procedures. You can assume that all relevant register usage is shown. The procedures are missing code to save and restore registers. Add that code. FOLLOW STANDARD CONVENTIONS: save only those registers that need it!
(NOTE: do not just add the words "save" and "restore," add the MIPS code to perform those actions.)

<table>
<thead>
<tr>
<th>Skeleton for Main Pgm</th>
<th>Skeleton for P1</th>
<th>Skeleton for P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>P1:</td>
<td>P2:</td>
</tr>
<tr>
<td>lw $s0, ...</td>
<td>sub $sp,$sp,4</td>
<td>sub $sp,$sp,8</td>
</tr>
<tr>
<td>lw $t0, ...</td>
<td>sw $ra,0($sp)</td>
<td>sw $s0,0($sp)</td>
</tr>
<tr>
<td>lw $t1, ...</td>
<td>...</td>
<td>sw $s1,4($sp)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>jal P1</td>
<td>lw $t1, ...</td>
</tr>
<tr>
<td>lw $t1,0($sp)</td>
<td>lw $t0, ...</td>
<td>lw $s0, ...</td>
</tr>
<tr>
<td>add $sp,$sp,4</td>
<td>lw $t1, ...</td>
<td>lw $s1, ...</td>
</tr>
<tr>
<td>add $t0,$t1,$s0</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>jal P2</td>
<td>lw $s0,0($sp)</td>
</tr>
<tr>
<td>jal P2</td>
<td>lw $ra,0($sp)</td>
<td>lw $s1,4($sp)</td>
</tr>
<tr>
<td>...</td>
<td>add $sp,$sp,4</td>
<td>add $sp,$sp,8</td>
</tr>
<tr>
<td>...</td>
<td>jr $ra</td>
<td>jr $ra</td>
</tr>
</tbody>
</table>