CIS 314, Fall 2001 QUIZ #1: MIPS and SPIM

1. (16 points) Given the following data segment declaration, what values are printed by each of the syscalls?

```plaintext
m:       .word    0xffffffec
n:       .word    0x0fac0033
p:       .word    5
list:    .word    m,n,p
authors: .asciiz  "Patterson and Hennessey"
```

(a) \text{lw} \hspace{1em} \$\text{a0}, \text{m} \hspace{1em} \text{li} \hspace{1em} \$\text{v0}, 1

(b) \text{lb} \hspace{1em} \$\text{t6}, \text{n} \hspace{1em} \text{srl} \hspace{1em} \$\text{a0}, \$\text{t6}, 2

\text{syscall}

\text{syscall}

\text{ANSWER: -20} \hspace{1em} \text{ANSWER: 3}

(c) \text{li} \hspace{1em} \$\text{t0}, 4

(d) \text{li} \hspace{1em} \$\text{t0}, 8

\text{lw} \hspace{1em} \$\text{t1}, \text{n}($\text{t0}) \hspace{1em} \text{lw} \hspace{1em} \$\text{t1}, \text{list}($\text{t0})

\text{la} \hspace{1em} \$\text{t2}, \text{authors} \hspace{1em} \text{lw} \hspace{1em} \$\text{a0}, \text{0}($\text{t1})

\text{add} \hspace{1em} \$\text{a0}, \$\text{t1}, \$\text{t2} \hspace{1em} \text{li} \hspace{1em} \$\text{v0}, 1

\text{li} \hspace{1em} \$\text{v0}, 4 \hspace{1em} \text{syscall}

\text{syscall}

\text{ANSWER: rson and Hennessey} \hspace{1em} \text{ANSWER: 5}

2. (54 points)
(a) Write a MIPS code segment that describes a complete, binary tree structure having 7 nodes with the integer constants 7, 8, 9, and 10 stored consecutively in its leaves from left-to-right.

\text{ANSWER:}

```
root: .word left, right
left: .word 7, 8
right: .word 9, 10
```

(b) What does it mean to say that an address is “aligned”? When is necessary?

\text{ANSWER:} An address is aligned if it falls on a word boundary, that is, if it is a multiple of 4. It is necessary if the data at that address is to be treated as a word (i.e., used with \text{lw}, \text{sw} operations) or as the address of an instruction.

c) Assume that \text{S} is defined as a string array of 16 characters. Give instructions for loading the character at position \text{S}[3] into register \$\text{t0}.

\text{ANSWER:}

```
\text{li} \hspace{1em} \$\text{t0}, 3
\text{lb} \hspace{1em} \$\text{t0}, \text{S}($\text{t0})
```
(d) Assume that $A$ is defined as an $n \times m$ integer matrix (meaning that it has $n$ rows and $m$ columns). Give an expression for the offset of element $(i, j)$ from the start of $A$, assuming that $A$ is stored in row major order and indexed starting at 0.
ANSWER: $4\times((i\times m) + j)$

(e) Assume that $\text{foo}$ is a procedure that uses only registers $s6$, $s7$, $t0$, and $t2$. It loads all four registers and then calls another procedure $g$; after the call to $g$, $\text{foo}$ uses the values previously loaded in registers $s6$ and $t0$ (but not $s7$ and $t2$). If there are no other procedure calls in either routine, show all of the code that $\text{foo}$ needs to conform to standard register usage; that is, show any saves/restores needed on entry/exit and around the call to $g$.
ANSWER:

<table>
<thead>
<tr>
<th>Code Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sub $sp,$sp,12</code></td>
<td>On entry to $\text{foo}$</td>
</tr>
<tr>
<td><code>sw $ra, 0($sp)</code></td>
<td></td>
</tr>
<tr>
<td><code>sw $s6,4($sp)</code></td>
<td></td>
</tr>
<tr>
<td><code>sw $s7,12($sp)</code></td>
<td></td>
</tr>
<tr>
<td><code>lw $s6,0($sp)</code></td>
<td>On exit from $\text{foo}$</td>
</tr>
<tr>
<td><code>lw $s7,12($sp)</code></td>
<td></td>
</tr>
<tr>
<td><code>lw $ra, 0($sp)</code></td>
<td></td>
</tr>
<tr>
<td><code>add $sp,$sp,12</code></td>
<td></td>
</tr>
</tbody>
</table>

Before call to $g$

<table>
<thead>
<tr>
<th>Code Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sub $sp,$sp,4</code></td>
<td></td>
</tr>
<tr>
<td><code>sw $t0,0($sp)</code></td>
<td></td>
</tr>
</tbody>
</table>

After call to $g$

<table>
<thead>
<tr>
<th>Code Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lw $t0,0($sp)</code></td>
<td></td>
</tr>
<tr>
<td><code>add $sp,$sp,4</code></td>
<td></td>
</tr>
</tbody>
</table>

(f) Write a segment of code that prints the middle sixteen bits of the word in register $t3$ as an unsigned integer (that is, it should print bits 8-24).

ANSWER

<table>
<thead>
<tr>
<th>Code Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sll $t3,$t3,8</code></td>
<td># remove top 8 bits</td>
</tr>
<tr>
<td><code>srl $a0,$t3,16</code></td>
<td># remove lower 8 bits</td>
</tr>
<tr>
<td><code>li $v0,1</code></td>
<td></td>
</tr>
<tr>
<td><code>syscall</code></td>
<td></td>
</tr>
</tbody>
</table>

(g) What is a pseudo instruction? Give an example.
ANSWER: A pseudo instruction is an instruction accepted by the assembler that does not correspond directly to a machine language instruction. There are many examples: `sub $t0,$t1,4`; `div $t1,$t2,$t3`; `li $t0,34`

(h) Both $s$ and $t$ registers are general registers that are available to the programmer. What is the distinction? Why is it useful to have?
ANSWER: By standard programming conventions, the $s$ registers are preserved across a procedure call while the $t$ are not. This means that the caller does not have to worry about saving the $s$ registers and the callee does not have to worry about saving the $t$ registers. This convention says us from having to save every register on every procedure call.

(i) What is a stack frame?
ANSWER: A stack frame is a contiguous section of space allocated on top of the runtime stack, usually upon procedure (or block) entry. It may have space for arguments, return values, local variables, registers, etc.
3. (30 points) Write a program segment that modifies an integer array, \( A \), of 25 elements so that each element of \( A \) (except for the first one) contains the sum of itself and the element below it. Thus you should implement

\[
\text{for } (i=1; i<=24; ++i) \ A[i] = A[i-1]+A[i]
\]

Assume that the values of the array \( A \) have already been set by some other code. Show the `.data` segment for any variables that you use.

COMMENT your program well; if I can’t understand it, I may assume its wrong!

ANSWER:

```assembly
# program to add adjacent elements of an array of integers
.data
A: .space 100
.text

__start: .globl __start
    li $t0,0       # initialize offset of A[i-1]
    li $t1,4       # initialize offset of A[i]
    li $t2,96      # offset of last element in array

    lw $t6,A($t0)  # load A[i-1]
    lw $t7,A($t1)  # load A[i]
    add $t7,$t7,$t6 # add 'em
    sw $t7,A($t1)  # store result in A[i]

    add $t0,$t0,4    # increment offset for A[i-1]
    add $t1,$t1,4    # increment offset for A[i]

out:
    j loop          # repeat
```