Homework 3

Topics covered: Arithmetic & bitwise logic, register usage, function calls

1) Perform the twos complement binary additions assuming 8-bit registers. Indicate overflow, if any.

\[
\begin{align*}
1000 & 1100 + 1000 & 1101 & = 0011 & 0010 \\
1000 & 1001 & & 1001 & 1001 & & 1011 & 0000 \\
\end{align*}
\]

Overflow: two negatives gave positive answer
No overflow
Overflow: two positives gave negative answer

2) Perform the unsigned binary additions assuming 8-bit registers. Indicate overflow, if any.

\[
\begin{align*}
0010 & 0100 + 1111 & 0100 & = 1111 & 0010 \\
1001 & 0000 & & 1001 & 0110 & & 1010 & 1110 \\
\end{align*}
\]

Overflow: carry-out at MSB
Overflow: carry-out at MSB

3) Our MIPS machine has the following values in these three registers.

$\text{\$s0} \quad \text{0xf809fab}\text{c}$
$\text{\$s1} \quad \text{0x70d74931}$
$\text{\$s2} \quad \text{0x5fffffff}$

What are the contents of $\text{\$s3}$, $\text{\$s4}$, and $\text{\$s5}$ after executing the following code?

add $\text{\$s3, \$s0, \$s1}$ $\text{\$s3} = \text{0x68e143ed}$
add $\text{\$s4, \$s1, \$s2}$ $\text{\$s4} = \text{0xd0d7491a}$
add $\text{\$s5, \$s0, \$s2}$ $\text{\$s5} = \text{0x5809faa5}$
4) Read over the following code. What are the contents of the global ($s$) registers after the code is done executing? What are the contents of array?

<table>
<thead>
<tr>
<th>$s0$</th>
<th>3 (last calculation)</th>
<th>$t0$</th>
<th>Temporary registers don’t matter!!</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s1$</td>
<td></td>
<td>$t1$</td>
<td></td>
</tr>
<tr>
<td>$s2$</td>
<td></td>
<td>$t2$</td>
<td></td>
</tr>
<tr>
<td>$s3$</td>
<td>Address of array</td>
<td>$t3$</td>
<td></td>
</tr>
<tr>
<td>$s4$</td>
<td>Address of endar (end of array)</td>
<td>$t4$</td>
<td></td>
</tr>
<tr>
<td>$s5$</td>
<td></td>
<td>$t5$</td>
<td></td>
</tr>
<tr>
<td>$s6$</td>
<td>3 (counter)</td>
<td>$t6$</td>
<td></td>
</tr>
<tr>
<td>$s7$</td>
<td></td>
<td>$t7$</td>
<td></td>
</tr>
<tr>
<td>$s8$</td>
<td></td>
<td>$t8$</td>
<td></td>
</tr>
</tbody>
</table>

Array: [13, 7, 9]

// The algorithm:
n=6;
if (n % 2 == 0) // n is even
    return (2n+1);
else // n is odd
{}
    store (n, array); // store n into array
    return ((n+1) / 2);

So each iteration through produces the following ‘n’: 6, 13, 7, 4, 9, 5.
.data
array: .word 0:3
endar: .byte

.text
__start:
li $s0, 6
li $s6, 0
la $s3, array
la $s4, endar

checknum:
rem $t3, $s0, 2
beqz $t3, even
b odd
even:
jal makebigger
b checknum

odd:
jal makesmaller
move $a1, $s3
jal store
move $s3, $a1
b checknum

makebigger:
add $t2, $s0, $0
sll $t1, $t2, 1
add $s0, $t1, 1
jr $ra

makesmaller:
add $t2, $s0, $0
add $t1, $t2, 1
div $s0, $t1, 2
jr $ra

store:
move $t2, $s0
move $t0, $a1
beq $t0, $s4, full
add $t0, $t0, 4
sw $t2, 4($s3)
add $s6, $s6, 1
move $a1, $t0
jr $ra

full:
done