Computer Architecture Homework #1 Solution

sumPos = 0;
sumNeg = 0;
for i = 0 to length-1 do
    if numbers[i] < 0 then
        sumNeg = sumNeg + numbers[i]
    else
        sumPos = sumPos + numbers[i]
    end if
end for

1. Write MIPS Assembly Language code for the above algorithm that sums the array's elements.

.data
numbers:    .word 2, 3, -1, 10, -5, -6, 3, 5, 1, -5
length:     .word 10
sumPos:     .word 0
sumNeg:     .word 0

.text
.globl main
main:
    li $8, 0 # Register $8 holds sumPos
    li $9, 0 # Register $9 holds sumNeg
for:
    lw $10, length
    li $11, 0 # Register $11 holds i
    la $12, numbers # Register $12 holds address of numbers[i] element
for_loop:
    bge $11, $10, end_for
if:
    lw $24, 0($12) # Read numbers[i] element into register $24
    bge $24, 0, else
then:
    add $9, $9, $24
    j end_if
else:
    add $8, $8, $24
end_if:
    addi $11, $11, 1 # increment loop control variable i
    addi $12, $12, 4 # move pointer to numbers[i] by the length of a word (4 bytes)
    j for_loop
end_for:
    sw $8, sumPos
    sw $9, sumNeg
    li $v0, 10           # system call for exiting the program
    syscall
2. Compare zero-, one-, two-, three-address, and the load & store machines by writing programs to compute

\[ X = (A + B) \times (C - D); \]
\[ Y = X / (A \times D); \]

for each of the five machines. The programs for each of the five machines are as follows:

<table>
<thead>
<tr>
<th>3 Address</th>
<th>2 Address</th>
<th>1 Address (Accumulator machine)</th>
<th>0 Address (Stack machine)</th>
<th>Load &amp; Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD X, A, B</td>
<td>MOVE X, A</td>
<td>LOAD A</td>
<td>PUSH A</td>
<td>LOAD R1, A</td>
</tr>
<tr>
<td>SUB T, C, D</td>
<td>ADD X, B</td>
<td>MUL D</td>
<td>PUSH B</td>
<td>LOAD R2, B</td>
</tr>
<tr>
<td>MUL X, X, T</td>
<td>MOVE T, C</td>
<td>STORE Y</td>
<td>ADD</td>
<td>ADD R2, R1, R2</td>
</tr>
<tr>
<td>MUL Y, A, D</td>
<td>SUB T, D</td>
<td>LOAD C</td>
<td>PUSH C</td>
<td>LOAD R3, C</td>
</tr>
<tr>
<td>DIV Y, X, Y</td>
<td>MUL X</td>
<td>SUB D</td>
<td>PUSH D</td>
<td>LOAD R4, D</td>
</tr>
<tr>
<td>MOVE T, A</td>
<td>STORE X</td>
<td>LOAD A</td>
<td>LOAD R3, R4</td>
<td>MUL R2, R2, R3</td>
</tr>
<tr>
<td>MUL T, D</td>
<td>SUB B</td>
<td>MUL</td>
<td>LOAD R2, X</td>
<td>STORE R2, X</td>
</tr>
<tr>
<td>MOVE Y, X</td>
<td>MUL X</td>
<td>POP X</td>
<td>MUL R1, R1, R4</td>
<td>MUL R1, R2, R1</td>
</tr>
<tr>
<td>DIV Y, T</td>
<td>STORE X</td>
<td>PUSH X</td>
<td>DIV R1, R2, R1</td>
<td>DIV R1, R2, R1</td>
</tr>
</tbody>
</table>

For the 0-address code, I first drew expression trees for each calculation and then performed a post-order traversal.

\[ X = (A+B)*(C-D) \]

\[ Y = X / (A*D) \]

Post-order Traversal: \( A \ B \ C \ D \ - \ * \)

Post-order Traversal: \( X \ A \ D \ * \ / \)

3. Assume 8-bit opcodes, 32-bit absolute addressing, 5-bit register numbers, and 32-bit operands. Compute the number of bits needed in programs from question 1 by completing the following table.

<table>
<thead>
<tr>
<th></th>
<th>3 Address</th>
<th>2 Address</th>
<th>1 Address</th>
<th>0 Address</th>
<th>Load &amp; Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of bits</td>
<td>(8+3*32)*5 = 520</td>
<td>(8+2*32)*9 = 648</td>
<td>(8+32)*12 = 480</td>
<td>5*8+9*40 = 400</td>
<td>5*(8+3*5) + 6*(8+5+32) = 385</td>
</tr>
<tr>
<td>needed to store the</td>
<td>(8+3*32)*5 = 520</td>
<td>(8+2*32)*9 = 648</td>
<td>(8+32)*12 = 480</td>
<td>5*8+9*40 = 400</td>
<td>5*(8+3*5) + 6*(8+5+32) = 385</td>
</tr>
<tr>
<td>program</td>
<td>(8+3*32)*5 = 520</td>
<td>(8+2*32)*9 = 648</td>
<td>(8+32)*12 = 480</td>
<td>5*8+9*40 = 400</td>
<td>5*(8+3*5) + 6*(8+5+32) = 385</td>
</tr>
<tr>
<td>Number of bits of</td>
<td>(3*32)*5 = 480</td>
<td>4*64+5*96 = 736</td>
<td>12*32 = 384</td>
<td>9*32 = 288</td>
<td>6*32 = 192</td>
</tr>
<tr>
<td>data transferred to</td>
<td>(3*32)*5 = 480</td>
<td>4*64+5*96 = 736</td>
<td>12*32 = 384</td>
<td>9*32 = 288</td>
<td>6*32 = 192</td>
</tr>
<tr>
<td>and from memory</td>
<td>(3*32)*5 = 480</td>
<td>4*64+5*96 = 736</td>
<td>12*32 = 384</td>
<td>9*32 = 288</td>
<td>6*32 = 192</td>
</tr>
<tr>
<td>Total number of bits</td>
<td>1000</td>
<td>1,384</td>
<td>864</td>
<td>688</td>
<td>577</td>
</tr>
<tr>
<td>read and written</td>
<td>1000</td>
<td>1,384</td>
<td>864</td>
<td>688</td>
<td>577</td>
</tr>
<tr>
<td>while the program</td>
<td>1000</td>
<td>1,384</td>
<td>864</td>
<td>688</td>
<td>577</td>
</tr>
<tr>
<td>executes</td>
<td>1000</td>
<td>1,384</td>
<td>864</td>
<td>688</td>
<td>577</td>
</tr>
</tbody>
</table>