# Mark Fienup
# HW 1 Question 1 solution
# High-level algorithm to count the number of positive, negative, & zero elements.

# posCount = 0
# negCount = 0
# zeroCount = 0
# for i = 0 to length-1 do
#     if numbers[i] < 0 then
#         negCount = negCount + 1
#     else if numbers[i] > 0 then
#         posCount = posCount + 1
#     else
#         zeroCount = zeroCount + 1
#     end if
# end for

# MIPS Assembly Language code for the above algorithm:
.data
defaultnumbers: .word 2, 0, 3, -1, 10, 0, 0, 12, -5, -6, 3, 5, 1, -5
defaultnegCount: .word 0  # negCount stored in register $8
defaultposCount: .word 0  # posCount stored in register $9
defaultzeroCount: .word 0  # zeroCount stored in register $10
defaultlength: .word 14  # value of (length - 1) stored in register $11
.text
.globl main
main:
    li $9, 0
    li $8, 0
    li $10, 0
for:
    lw $11, length
    la $12, numbers  # base address of numbers stored in $12
    sub $11, $11, 1
    li $15, 0  # i stored in register $15
for_loop: bgt $15, $11, end_for
if:
    mul $13, $15, 4 # address of numbers[i] calculated in $13
    add $13, $12, $13
    lw $14, 0($13) # value of numbers[i] stored in $14
    bge $14, 0, else_if
    addi $8, $8, 1
    j end_if
else_if:
    ble $14, 0, else
    addi $9, $9, 1
    j end_if
else:
    addi $10, $10, 1
end_if:
    addi $15, $15, 1
    j for_loop
end_for:
    sw $8, negCount
    sw $9, posCount
    sw $10, zeroCount
    li $v0, 10           # system call for exiting the program
    syscall
2. Compare zero-, one-, two-, three-address, and the load & store machines by writing a single program to compute both lines combined

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

for each of the five machines. The instructions available for use are as follows:

<table>
<thead>
<tr>
<th>3 Address Program</th>
<th>2 Address Program</th>
<th>1 Address Program (Accumulator machine)</th>
<th>0 Address Program (Stack machine)</th>
<th>Load/Store Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD X, A, B</td>
<td>MOVE X, A</td>
<td>LOAD C</td>
<td>PUSH A</td>
<td>LOAD R1, A</td>
</tr>
<tr>
<td>ADD X, X, C</td>
<td>ADD X, B</td>
<td>SUB D</td>
<td>PUSH B</td>
<td>LOAD R2, B</td>
</tr>
<tr>
<td>SUB Y, C, D</td>
<td>ADD X, C</td>
<td>STORE X</td>
<td>ADD</td>
<td>ADD R3, R1, R2</td>
</tr>
<tr>
<td>MUL X, X, Y</td>
<td>MOVE Y, C</td>
<td>LOAD A</td>
<td>PUSH C</td>
<td>LOAD R2, C</td>
</tr>
<tr>
<td>SUB Y, A, D</td>
<td>SUB Y, D</td>
<td>ADD B</td>
<td>ADD</td>
<td>ADD R3, R3, R2</td>
</tr>
<tr>
<td>DIV Y, C, Y</td>
<td>MUL X, Y</td>
<td>ADD C</td>
<td>PUSH C</td>
<td>LOAD R4, D</td>
</tr>
<tr>
<td></td>
<td>MOVE T, A</td>
<td>MUL X</td>
<td>PUSH D</td>
<td>SUB R5, R2, R4</td>
</tr>
<tr>
<td></td>
<td>SUB T, D</td>
<td>STORE X</td>
<td>SUB</td>
<td>MUL R5, R3, R5</td>
</tr>
<tr>
<td></td>
<td>MOVE Y, C</td>
<td>LOAD A</td>
<td>POP X</td>
<td>STORE R5, X</td>
</tr>
<tr>
<td></td>
<td>DIV Y, T</td>
<td>SUB D</td>
<td>PUSH C</td>
<td>SUB R1, R1, R4</td>
</tr>
<tr>
<td></td>
<td>STORE Y</td>
<td>STORE Y</td>
<td>PUSH D</td>
<td>DIV R1, R2, R1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUB</td>
<td>STORE R1, Y</td>
</tr>
</tbody>
</table>

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 needed to store the program</td>
<td>6 (8 + 3 * 32) = 624</td>
<td>10 (8 + 2 * 32) = 720</td>
<td>14 (8 + 32) = 560</td>
<td>10 (8 + 32) + 6 * 8 = 448</td>
<td>6 (8 + 5 + 32) + 6 (8 + 3 * 5) = 408</td>
</tr>
<tr>
<td>Number of bits of data transferred to and from memory</td>
<td>6 (3 * 32) = 576</td>
<td>4 (2 * 32) + 6 (3 * 32) = 832</td>
<td>14 * 32 = 448</td>
<td>10 * 32 = 320</td>
<td>6 * 32 = 192</td>
</tr>
<tr>
<td>Total number of bits read and written while the program executes (sum of above rows)</td>
<td>1,200</td>
<td>1,552</td>
<td>1,008</td>
<td>768</td>
<td>600</td>
</tr>
</tbody>
</table>