Computer Architecture Homework #1 Solution

```
# 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
numbers:     .word 2, 0, 3, -1, 10, 0, 0, 12, -5, -6, 3, 5, 1, -5
negCount:    .word 0       # negCount stored in register $8
posCount:    .word 0       # posCount stored in register $9
zeroCount:   .word 0       # zeroCount stored in register $10
length:      .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) * (C - D);
Y = C / (A - D);

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

3 Address Program	2 Address Program	1 Address Program (Accumulator machine)	0 Address Program (Stack machine)	Load/Store Program

ADD X, A, B
ADD X, X, C
SUB Y, C, D
MUL X, X, Y
SUB Y, A, D
DIV Y, C, Y
	MOVE X, A
	ADD X, B
	ADD X, C
	MOVE Y, C
	SUB Y, D
	MUL X, Y
	MOVE T, A
	SUB T, D
	MOVE Y, C
	DIV Y, T
		LOAD C
		SUB D
		STORE X
		LOAD A
		ADD B
		ADD C
		MUL X
		STORE X
		LOAD A
		SUB D
		STORE Y
		LOAD C
		DIV Y
		STORE Y
			PUSH A
			PUSH B
			ADD
			PUSH C
			ADD
			PUSH C
			PUSH D
			SUB
			MUL
			POP X
			PUSH C
			PUSH A
			PUSH D
			SUB
			DIV
			POP Y
				LOAD R1, A
				LOAD R2, B
				ADD R3, R1, R2
				LOAD R2, C
				ADD R3, R3, R2
				LOAD R4, D
				SUB R5, R2, R4
				MUL R5, R3, R5
				STORE R5, X
				SUB R1, R1, R4
				DIV R1, R2, R1
				STORE R1, Y

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.

	3 Address	2 Address	1 Address	0 Address	Load & Store
Number of bits needed to store the program	6 (8 + 3 * 32) = 624	10 (8 + 2 *32) = 720	14 (8 + 32) = 560	10 (8 + 32) + 6 * 8 = 448	6 (8 + 5 + 32) + 6 (8 + 3 * 5) = 408
Number of bits of data transferred to and from memory	6 (3 * 32) = 576	4(2 * 32) + 6(3 * 32) = 832	14 * 32 = 448	10 * 32 = 320	6 * 32 = 192
Total number of bits read and written while the program executes (sum of above rows)	1,200	1,552	1,008	768	600