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) * (C - D);
Y = X / (A * D);

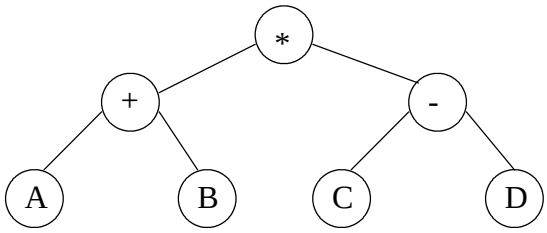
for each of the five machines. The programs for each of the five machines are as follows:

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

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

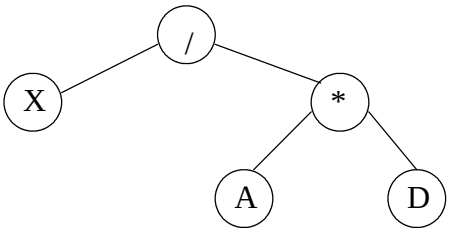
For the 0-address code, I first drew expression trees for each calculation and then performed a post-order traversal.

X = (A+B)*(C-D)



Post-order Traversal: A B + C D - *

Y = X / (A*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.

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