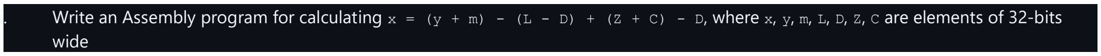
Week 2 - Assembly Assignment

i. Write an Assembly Program for:
   a. addition of N words
   b. addition of N half words
   c. addition of N bytes

```
    .data
words:      .word 5, 10, 15, 20, 25    # Array of 5 words (32-bit)
halfwords:  .half 100, 200, 300, 400, 500  # Array of 5 half-words (16-bit)
bytes:      .byte 1, 2, 3, 4, 5       # Array of 5 bytes (8-bit)

n:          .word 5                   # Number of elements (N)

    .text
    li x5, 0                    # Initialize sum to 0 (for words, half-words, and bytes)
    la x6, n                    # Load address of N
    lw x7, 0(x6)                # Load the value of N into x7 (N = 5)

    # Addition of N Words (32-bit)
    la x8, words                # Load address of words array into x8
    li x9, 0                    # Initialize sum of words (32-bit) to 0

word_addition_loop:
    lw x10, 0(x8)               # Load word from address x8
    add x9, x9, x10             # Add the word to the sum
    addi x8, x8, 4              # Move to the next word (4 bytes)
    addi x7, x7, -1             # Decrement N
    bnez x7, word_addition_loop # Repeat until N = 0

    # Store the result of word addition
    la x6, result_word          # Store the result in result_word
    sw x9, 0(x6)

    # Reset N and x7
    la x6, n                    # Load address of N
    lw x7, 0(x6)                # Reload N

    # Addition of N Half-Words (16-bit)
    la x8, halfwords            # Load address of half-words array into x8
    li x9, 0                    # Initialize sum of half-words (16-bit) to 0

halfword_addition_loop:
    lh x10, 0(x8)               # Load half-word from address x8
    add x9, x9, x10             # Add the half-word to the sum
    addi x8, x8, 2              # Move to the next half-word (2 bytes)
    addi x7, x7, -1             # Decrement N
    bnez x7, halfword_addition_loop # Repeat until N = 0

    # Store the result of half-word addition
    la x6, result_halfword      # Store the result in result_halfword
    sh x9, 0(x6)

    # Reset N and x7
    la x6, n                    # Load address of N
    lw x7, 0(x6)                # Reload N

    # Addition of N Bytes (8-bit)
    la x8, bytes                # Load address of bytes array into x8
    li x9, 0                    # Initialize sum of bytes (8-bit) to 0

byte_addition_loop:
    lb x10, 0(x8)               # Load byte from address x8
    add x9, x9, x10             # Add the byte to the sum
    addi x8, x8, 1              # Move to the next byte (1 byte)
    addi x7, x7, -1             # Decrement N
    bnez x7, byte_addition_loop # Repeat until N = 0

    # Store the result of byte addition
    la x6, result_byte          # Store the result in result_byte
    sb x9, 0(x6)

    # End of program, infinite loop
exit:
    j exit

    .data
result_word:  .word 0           # To store the result of word addition
result_halfword: .half 0        # To store the result of half-word addition
result_byte: .byte 0            # To store the result of byte addition
```



Write an Assembly program for calculating `x = (y + m) - (L - D) + (Z + C) - D`, where `x`, `y`, `m`, `L`, `D`, `Z`, `C` are elements of 32-bits wide

```
    .data
y:  .word 10        # Value of y
m:  .word 20        # Value of m
L:  .word 30        # Value of L
D:  .word 5         # Value of D
Z:  .word 50        # Value of Z
C:  .word 40        # Value of C
x:  .word 0         # To store the result (x)

    .text

# Load values into registers
    la x1, y         # Load address of y
    lw x2, 0(x1)     # Load value of y into x2

    la x3, m         # Load address of m
    lw x4, 0(x3)     # Load value of m into x4
```

```
# Calculate y + m
add x5, x2, x4      # x5 = y + m

la x6, L            # Load address of L
lw x7, 0(x6)        # Load value of L into x7

la x8, D            # Load address of D
lw x9, 0(x8)        # Load value of D into x9

# Calculate L - D
sub x10, x7, x9     # x10 = L - D

# Calculate (y + m) - (L - D)
sub x11, x5, x10    # x11 = (y + m) - (L - D)

la x12, Z           # Load address of Z
lw x13, 0(x12)      # Load value of Z into x13

la x14, C           # Load address of C
lw x15, 0(x14)      # Load value of C into x15

# Calculate Z + C
add x16, x13, x15   # x16 = Z + C

# Calculate ((y + m) - (L - D)) + (Z + C)
add x17, x11, x16   # x17 = ((y + m) - (L - D)) + (Z + C)

# Calculate ((y + m) - (L - D)) + (Z + C) - D
sub x18, x17, x9    # x18 = ((y + m) - (L - D)) + (Z + C) - D

# Store result in x
la x19, x           # Load address of x
sw x18, 0(x19)      # Store the result of the calculation in x

# Exit the program (system call for exit)
li a7, 10           # Exit system call
ecall
```