Week 1 - Assembly Assignment

Convert a 32-bit value from Little Endian to Big Endian format using RISC-V assembly

```
.data
a:.word 0x12345678
.text
la x10,a
lw x11,0(x10)
andi x12,x11,0xFF
slli x12,x12,24

srli x13,x11,8
andi x13,x13,0xFF
slli x13,x13,16
add x12,x12,x13

srli x13,x11,16
andi x13,x13,0xFF
slli x13,x13,8
add x12,x12,x13

srli x13,x11,24
add x12,x12,x13
sw x12,0(x10)
```

Write an Assembly Program for addition of 2 64-bit numbers on RV32I

```
    .data
a_low:  .word 0x00000001        # Lower 32 bits of first 64-bit number (1)
a_high: .word 0x00000000        # Upper 32 bits of first 64-bit number (0)
b_low:  .word 0x00000002        # Lower 32 bits of second 64-bit number (2)
b_high: .word 0x00000000        # Upper 32 bits of second 64-bit number (0)
result_low: .word 0             # To store the lower 32 bits of the result
result_high: .word 0            # To store the upper 32 bits of the result

    .text
    li x5, 0                    # Start of main code

    # Load the lower and higher 32 bits of the first number (a)
    la x6, a_low                # Load address of a_low into x6
    lw x7, 0(x6)                # Load a_low into x7 (lower 32 bits of a)
    la x6, a_high               # Load address of a_high into x6
    lw x8, 0(x6)                # Load a_high into x8 (upper 32 bits of a)

    # Load the lower and higher 32 bits of the second number (b)
    la x6, b_low                # Load address of b_low into x6
    lw x9, 0(x6)                # Load b_low into x9 (lower 32 bits of b)
```

```
la x6, b_high               # Load address of b_high into x6
lw x10, 0(x6)               # Load b_high into x10 (upper 32 bits of b)

# Add the lower 32 bits
add x11, x7, x9             # Add lower 32 bits of a and b
mfhi x12                    # Move carry (high part) from the addition

# Add the upper 32 bits with carry
add x13, x8, x10            # Add the upper 32 bits of a and b
add x13, x13, x12           # Add the carry from the lower 32-bit addition

# Store the results in result_low and result_high
la x6, result_low           # Load address of result_low into x6
sw x11, 0(x6)               # Store the lower 32 bits of the result in result_low
la x6, result_high          # Load address of result_high into x6
sw x13, 0(x6)               # Store the upper 32 bits of the result in result_high

# Exit (typically with an exit system call, but here it is just an infinite loop)
j x5                        # Jump to x5 to create an infinite loop, ending the program
```