PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Feet Ring Road, BSK III Stage, Bengaluru-560 085

Department of Electronics
and Communication Engineering

Course Title: RISC V Architecture
Course Code: UE21EC352A

Teacher: Dr. Santhameena S

Project Title:

CIRCULAR CONVOLUTION, HCF and LCM, and HARMONIC MEAN

Made By:
Sharvi Ranjan- PES1UG22EC918
Varada Dhabe- PES1UG22EC326

Problem Statement:

Development of a RISC-V Assembly Program for Hybrid Numerical and Signal Processing Tasks

- **Circular convolution**, also known as **cyclic convolution**, is a special case of **periodic convolution**, which is the convolution of two periodic functions that have the same period. Periodic convolution arises, for example, in the context of the discrete-time Fourier transform (DTFT).Circular convolution plays an important role in maximising the efficiency of a certain kind of common filtering operation.
- **Compute the Greatest Common Divisor (GCD) and Least Common Multiple (LCM)** of two given integers using an iterative algorithm.
- **Calculate the Harmonic Mean** of the computed GCD and LCM, demonstrating integer arithmetic and memory handling.

Program:

LOGIC:

```
# Input:
#   N: Size of the arrays
#   A: Array of size N
#   B: Array of size N
# Output:
#   C: Array of size N (result of circular convolution)

# Initialize data
C = [0] * N   # Output array initialised to zeros

# Outer loop: Iterate over i (output index)
for i in range(0, N):
    sum = 0   # Initialize sum for C[i]

    # Inner loop: Iterate over j (input index)
    for j in range(0, N):
        k = (i - j + N) % N   # Circular index calculation
        sum += A[k] * B[j]    # Accumulate product of A[k] and B[j]

    C[i] = sum   # Store the result in output array

# End of program
return C
```

CODE:

```
.data
```

```
# Data for GCD, LCM, and Harmonic Mean calculations
num1: .word 56 # Example number 1
num2: .word 98 # Example number 2
gcd_result: .word 0 # To store GCD result
lcm_result: .word 0 # To store LCM result
harmonic_mean: .word 0 # To store Harmonic Mean result

# Data for Circular Convolution
N: .word 10 # Length of sequences
x: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 # Input sequence x
h: .word 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 # Input sequence h
y: .word 40 # Output array for convolution result, size = N * 4 bytes

.text
.globl _start

# Start of program
_start:

# Part 1: Calculate GCD, LCM, and Harmonic Mean
la x5, num1 # Load address of num1
lw x6, 0(x5) # Load num1 into x6 (a)
la x7, num2 # Load address of num2
lw x8, 0(x7) # Load num2 into x8 (b)
mv x10, x6 # Save num1 in x10
mv x11, x8 # Save num2 in x11

gcd_loop:
beq x8, x0, gcd_done # If x8 == 0, exit loop
rem x9, x6, x8 # x9 = x6 % x8
mv x6, x8 # x6 = x8
mv x8, x9 # x8 = x9
j gcd_loop

gcd_done:
la x5, gcd_result
sw x6, 0(x5) # Store GCD in gcd_result
mul x12, x10, x11 # x12 = num1 * num2

div x13, x12, x6 # x13 = LCM = (a * b) / gcd
la x5, lcm_result
sw x13, 0(x5) # Store LCM in lcm_result

# Calculate Harmonic Mean
la x5, gcd_result
lw x14, 0(x5) # Load GCD
la x5, lcm_result
lw x15, 0(x5) # Load LCM
mul x16, x14, x15 # x16 = gcd * lcm
slli x16, x16, 1 # x16 = 2 * (gcd * lcm)
add x17, x14, x15 # x17 = gcd + lcm
div x18, x16, x17 # x18 = Harmonic Mean
la x5, harmonic_mean
sw x18, 0(x5) # Store Harmonic Mean

# Part 2: Circular Convolution
la x1, N
lw x2, 0(x1) # x2 = N
```

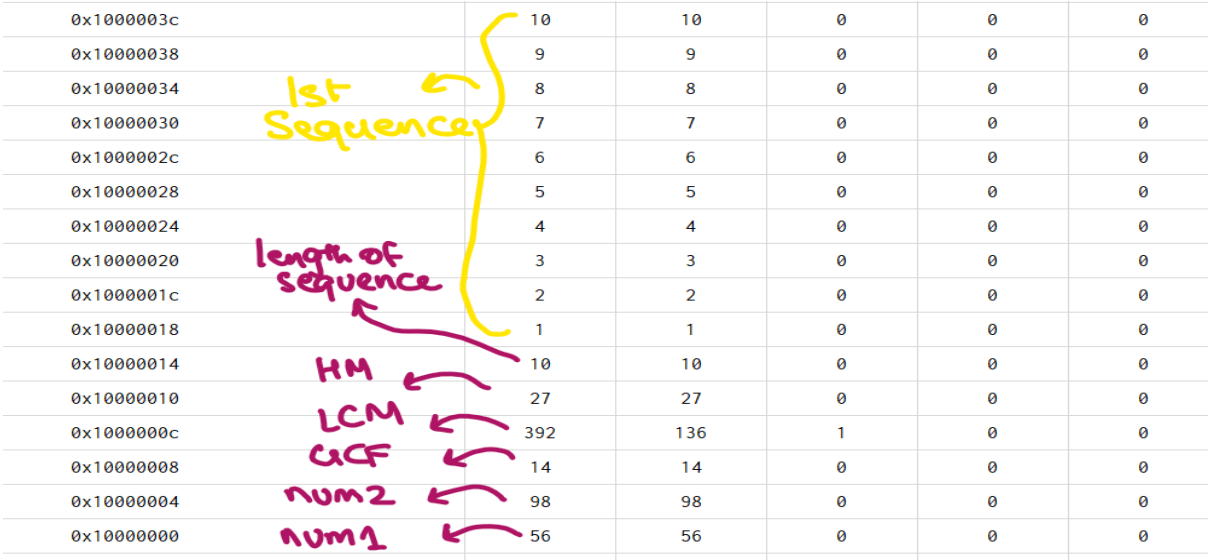
```
la x3, x # x3 = address of x
la x4, h # x4 = address of h
la x5, y # x5 = address of y
li x6, 0 # Outer loop counter

outer_loop:
li x7, 0 # Reset accumulator
la x8, x # Reset x pointer
la x9, h # Reset h pointer
li x10, 0 # Inner loop counter

inner_loop:
lw x11, 0(x8) # Load x[x10]
lw x12, 0(x9) # Load h[(x6 - x10) % N]
mul x13, x11, x12 # Multiply x[x10] * h[(x6 - x10) % N]
add x7, x7, x13 # Accumulate result
addi x8, x8, 4 # Move to next x
addi x10, x10, 1 # Increment inner loop counter
sub x14, x6, x10 # Calculate (x6 - x10)
rem x14, x14, x2 # Modulo N
bgez x14, pos_index # If positive, skip adjustment
add x14, x14, x2 # Adjust for negative index

pos_index:
slli x14, x14, 2 # Convert index to byte offset
la x9, h # Reset h pointer
add x9, x9, x14 # h[(x6 - x10) % N]
bne x10, x2, inner_loop # Repeat inner loop
sw x7, 0(x5) # Store result in y[x6]
addi x5, x5, 4 # Move to next y
addi x6, x6, 1 # Increment outer loop counter
bne x6, x2, outer_loop # Repeat outer loop
# Halt program for inspection
halt:
j halt
```

Output:

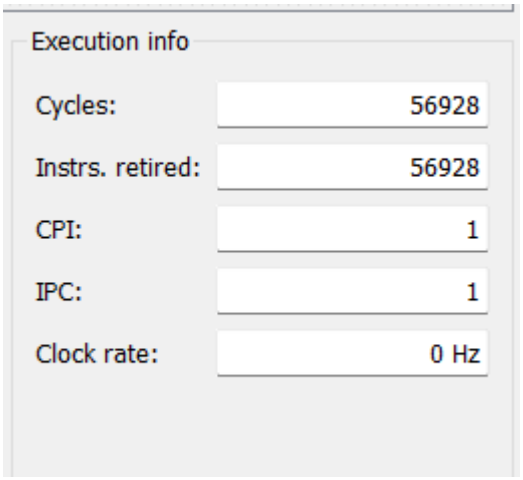


0x1000003c	10	10	0	0	0
0x10000038	9	9	0	0	0
0x10000034	8	8	0	0	0
0x10000030	7	7	0	0	0
0x1000002c	6	6	0	0	0
0x10000028	5	5	0	0	0
0x10000024	4	4	0	0	0
0x10000020	3	3	0	0	0
0x1000001c	2	2	0	0	0
0x10000018	1	1	0	0	0
0x10000014	10	10	0	0	0
0x10000010	27	27	0	0	0
0x1000000c	392	136	1	0	0
0x10000008	14	14	0	0	0
0x10000004	98	98	0	0	0
0x10000000	56	56	0	0	0

Address	Word	Byte 0	Byte 1	Byte 2	Byte 3
0x10000090	0	0	0	0	0
0x1000008c	55	55	0	0	0
0x10000088	55	55	0	0	0
0x10000084	55	55	0	0	0
0x10000080	55	55	0	0	0
0x1000007c	55	55	0	0	0
0x10000078	55	55	0	0	0
0x10000074	55	55	0	0	0
0x10000070	55	55	0	0	0
0x1000006c	55	55	0	0	0
0x10000068	55	55	0	0	0
0x10000064	1	1	0	0	0
0x10000060	1	1	0	0	0
0x1000005c	1	1	0	0	0
0x10000058	1	1	0	0	0
0x10000054	1	1	0	0	0
0x10000050	1	1	0	0	0
0x1000004c	1	1	0	0	0
0x10000048	1	1	0	0	0
0x10000044	1	1	0	0	0
0x10000040	1	1	0	0	0

Circular Convolution result

2nd Sequence



Conclusion:

In conclusion, this RISC-V assembly program effectively combines numerical computation and signal processing tasks, demonstrating the versatility and computational capability of low-level programming. By calculating GCD, LCM, and Harmonic Mean alongside performing circular convolution, the program integrates mathematical precision with efficient sequence processing. It serves as a foundational example of handling complex operations within memory-constrained and performance-critical systems. This project not only highlights the practical application of RISC-V assembly language in solving hybrid computational problems but also provides a strong base for further exploration in embedded systems and digital signal processing domains.