RISC V Assembly Programs

Dr. Girish H
Professor
Department of ECE
Cambridge Institute of Technology
&
Kavinesh
Research Staff
CCCIR
Cambridge Institute of Technology

Hello World

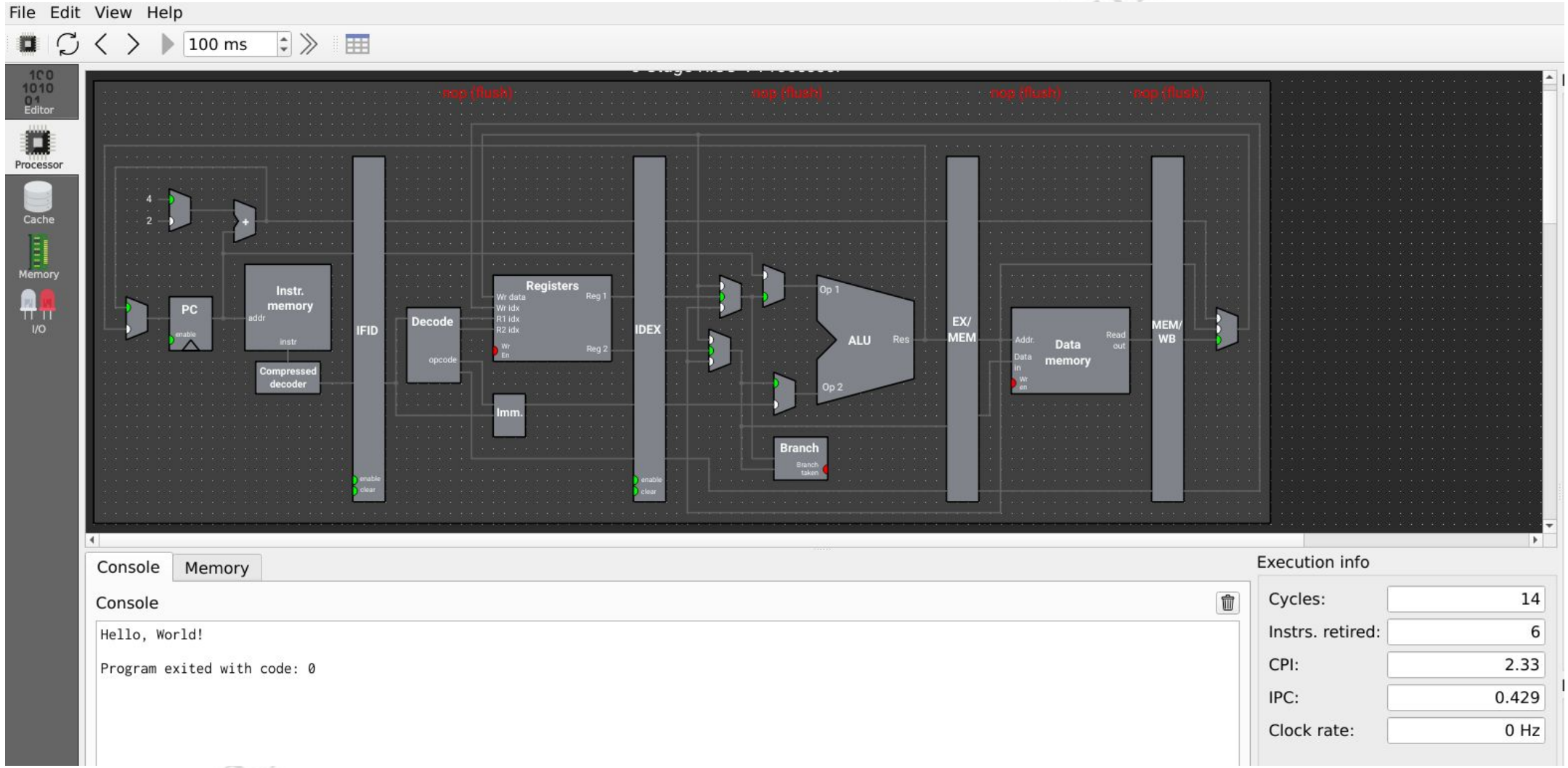
```
// Header file for input output functions
#include <stdio.h>

// Main function: entry point for execution
int main() {

    // Writing print statement to print hello world
    printf("Hello World");

    return 0;
}
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Hello World

```
.data
msg: .string "Hello, World!\n"

.text
main:
        la a0, msg  # Load address of message
        li a7, 4    # System call for print string
        ecall

        li a7, 10   # System call for exit
        ecall
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT


File Edit View Help
100 ms
Editor
Processor
Cache
Memory
I/O
nop (flush)
nop (flush)
nop (flush)
nop (flush)
PC
Instr. memory
Compressed decoder
IFID
Decode
Registers
Imm.
IDEX
ALU
Branch
EX/ MEM
Data memory
MEM/ WB
Console
Memory
Hello, World!
Program exited with code: 0
Execution info
Cycles: 14
Instrs. retired: 6
CPI: 2.33
IPC: 0.429
Clock rate: 0 Hz

Addition of two numbers

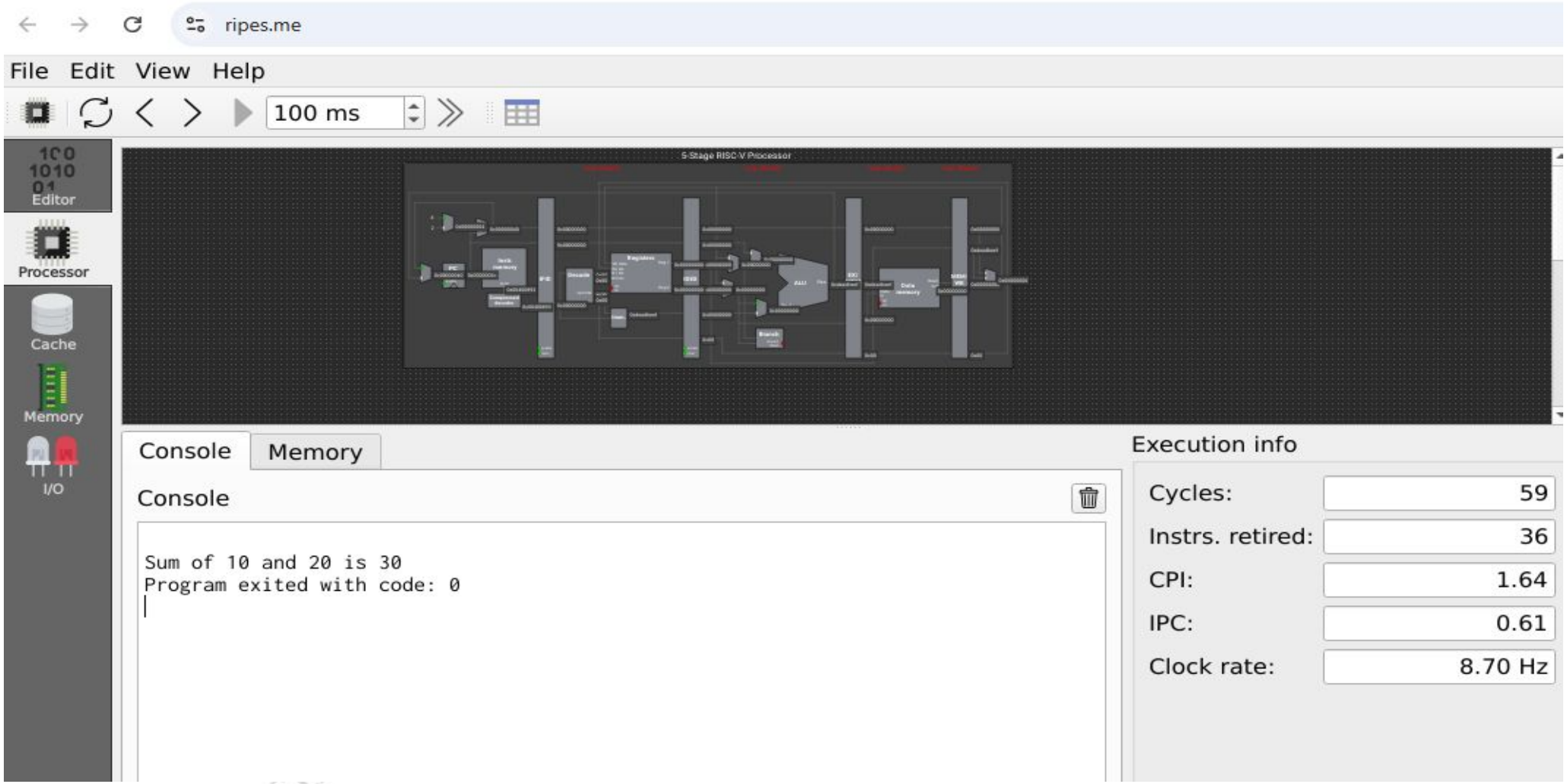
```
#include <stdio.h>

int main() {
    int a=10 , b = 20, sum = 0;

    // Calculate the addition of a and b
    // using '+' operator
    sum = a + b;

    printf("Sum: %d", sum);

    return 0;
}
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Addition of two numbers

```
.data
num1:  .word  10
num2:  .word  20
str1:  .string "Sum of "
str2:  .string " and "
str3:  .string " is "

.text
main:
        # Load numbers from static data
        lw  a0, num1
        lw  a1, num2
        # Perform addition
        add a2, a0, a1  # a2 = a0 + a1
        # Print the result to console
        mv  a1, a2
        lw  a0, num1
        jal ra, printResult
        # Exit program
        li a7, 10
        ecall

printResult:
        mv t0, a0
        mv t1, a1
        la a0, str1
        li a7, 4
        ecall
        mv a0, t0
        li a7, 1
        ecall
        la a0, str2
        li a7, 4
        ecall
        lw a0, num2
        li a7, 1
        ecall
        la a0, str3
        li a7, 4
        ecall
        mv a0, t1
        li a7, 1
        ecall
        ret
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT


ripes.me
File Edit View Help
100 ms
Editor
Processor
Cache
Memory
I/O
5-Stage RISC-V Processor
Console
Memory
Console
Sum of 10 and 20 is 30
Program exited with code: 0
Execution info
Cycles: 59
Instrs. retired: 36
CPI: 1.64
IPC: 0.61
Clock rate: 8.70 Hz

Subtraction of two numbers

```
#include <stdio.h>

int main() {
    int a=10 , b = 20, sum = 0;

    // Calculate the subtraction of a and b
    // using '-' operator
    difference = a - b;

    printf("Difference: %d", difference);

    return 0;
}
```

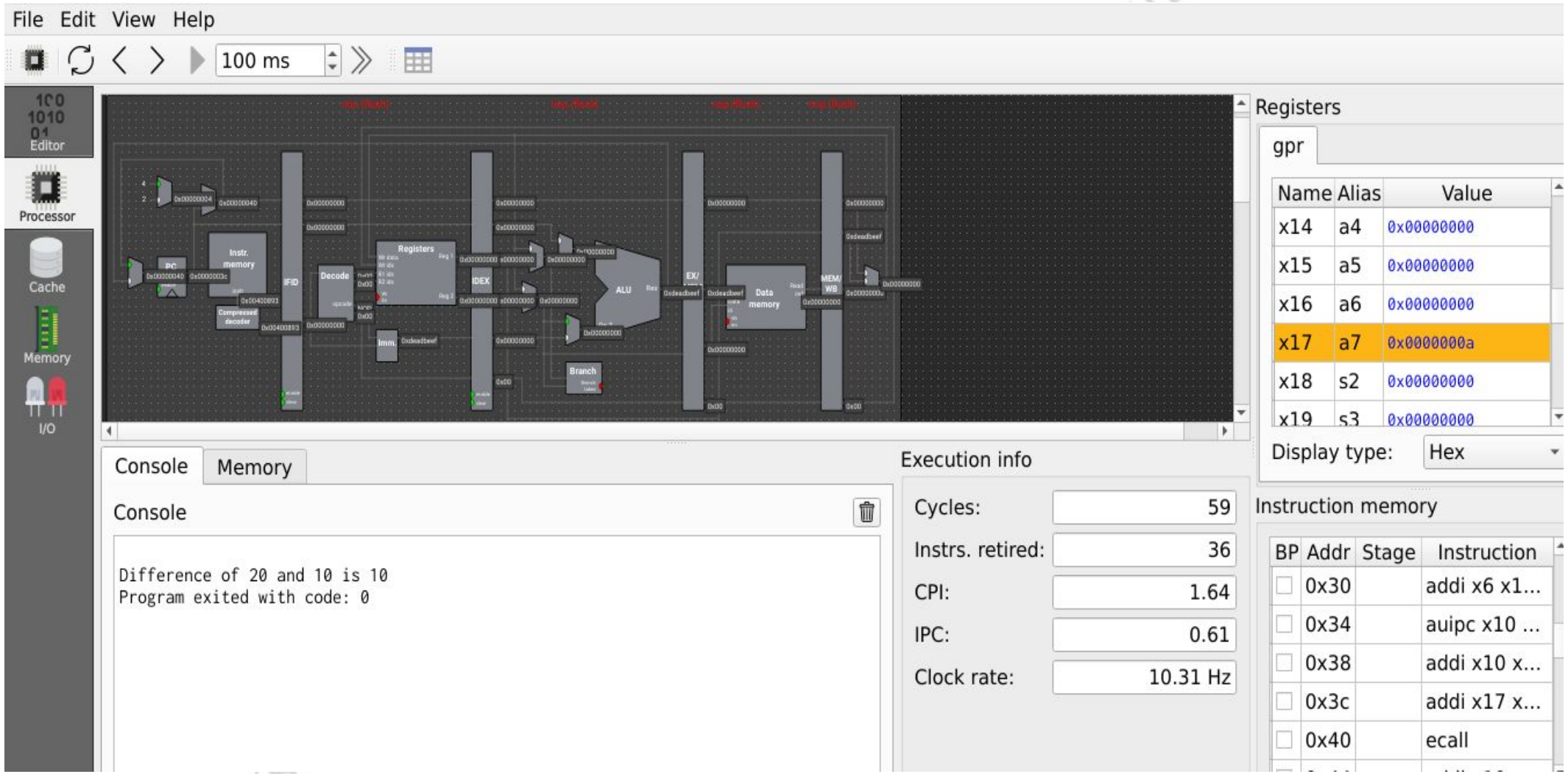
Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Difference of two numbers

```
.data
num1:  .word  20
num2:  .word  10
str1:  .string "Difference of "
str2:  .string " and "
str3:  .string " is "
.text
main:
        # Load numbers from static data
        lw  a0, num1
        lw  a1, num2
        # Perform subtraction
        sub a2, a0, a1  # a2 = a0 - a1
        # Print the result to console
        mv  a1, a2
        lw  a0, num1
        jal ra, printResult

        # Exit program
        li a7, 10
        ecall
```

```
printResult:
        mv t0, a0
        mv t1, a1
        la a0, str1
        li a7, 4
        ecall
        mv a0, t0
        li a7, 1
        ecall
        la a0, str2
        li a7, 4
        ecall
        lw a0, num2
        li a7, 1
        ecall
        la a0, str3
        li a7, 4
        ecall
        mv a0, t1
        li a7, 1
        ecall
        ret
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT


File Edit View Help
100 ms
Editor
Processor
Cache
Memory
I/O
Registers
gpr
Name Alias Value
x14 a4 0x00000000
x15 a5 0x00000000
x16 a6 0x00000000
x17 a7 0x0000000a
x18 s2 0x00000000
Display type: Hex
Console Memory
Console
Difference of 20 and 10 is 10
Program exited with code: 0
Execution info
Cycles: 59
Instrs. retired: 36
CPI: 1.64
IPC: 0.61
Clock rate: 10.31 Hz
Instruction memory
BP Addr Stage Instruction
0x30 addi x6 x1...
0x34 auipc x10 ...
0x38 addi x10 x...
0x3c addi x17 x...
0x40 ecall

Multiplication of two numbers

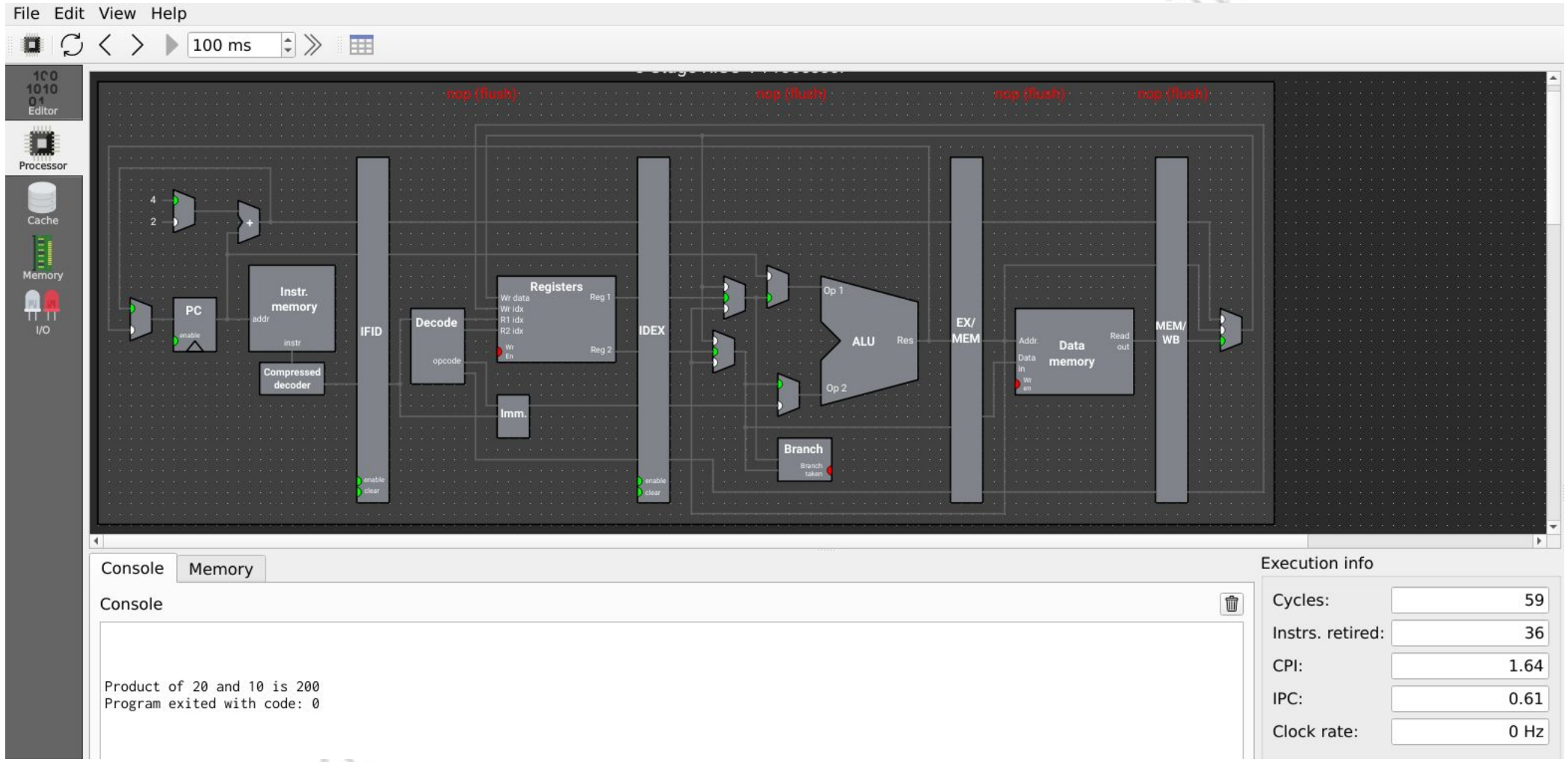
```
#include <stdio.h>

int main() {
    int a=10 , b = 20, sum = 0;

    // Calculate the multiplication of a and b
    // using '*' operator
    product = a * b;

    printf("product: %d", product);

    return 0;
}
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Multiplication of two numbers

```
.data
num1:  .word  20
num2:  .word  10
str1:  .string "Product of "
str2:  .string " and "
str3:  .string " is "
.text
main:
        # Load numbers from static data
        lw  a0, num1
        lw  a1, num2

        # Perform multiplication
        mul a2, a0, a1  # a2 = a0 * a1
        # Print the result to console
        mv  a1, a2
        lw  a0, num1
        jal ra, printResult
        # Exit program
        li a7, 10
        ecall
printResult:
        mv t0, a0
        mv t1, a1
        la a0, str1
        li a7, 4
        ecall
        mv a0, t0
        li a7, 1
        ecall
        la a0, str2
        li a7, 4
        ecall
        lw a0, num2
        li a7, 1
        ecall
        la a0, str3
        li a7, 4
        ecall
        mv a0, t1
        li a7, 1
        ecall
        ret
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT


File Edit View Help
100 ms
Editor
Processor
Cache
Memory
I/O
nop (flush)
nop (flush)
nop (flush)
nop (flush)
PC
Instr. memory
Compressed decoder
IFID
Decode
Registers
Imm.
IDEX
ALU
Branch
EX/ MEM
Data memory
MEM/ WB
Console
Memory
Console
Product of 20 and 10 is 200
Program exited with code: 0
Execution info
Cycles: 59
Instrs. retired: 36
CPI: 1.64
IPC: 0.61
Clock rate: 0 Hz

Division of two numbers

```
#include <stdio.h>

int main() {
    int a=10 , b = 20, sum = 0;

    // Calculate the division of a and b
    // using '/' operator
    quotient = a / b;

    printf("quotient: %d", quotient);

    return 0;
}
```

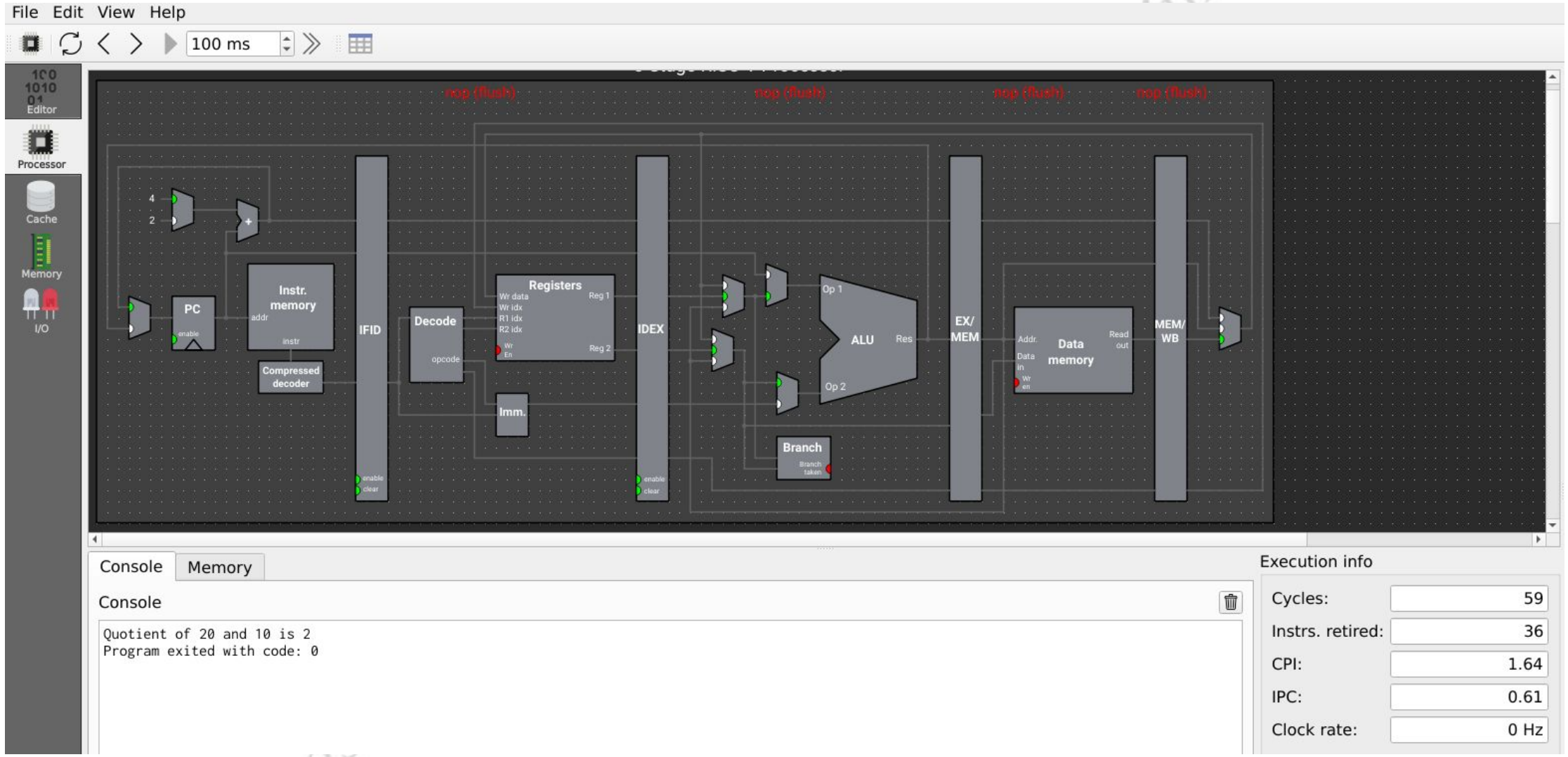
Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Division of two numbers

```
.data
num1:  .word  20
num2:  .word  10
str1:  .string "Quotient of "
str2:  .string " and "
str3:  .string " is "
.text
main:
        # Load numbers from static data
        lw  a0, num1
        lw  a1, num2
        # Perform division
        div a2, a0, a1  # a2 = a0 / a1
        # Print the result to console
        mv  a1, a2
        lw  a0, num1
        jal ra, printResult

        # Exit program
        li a7, 10
        ecall
```

```
printResult:
        mv t0, a0
        mv t1, a1
        la a0, str1
        li a7, 4
        ecall
        mv a0, t0
        li a7, 1
        ecall
        la a0, str2
        li a7, 4
        ecall
        lw a0, num2
        li a7, 1
        ecall
        la a0, str3
        li a7, 4
        ecall
        mv a0, t1
        li a7, 1
        ecall
        ret
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

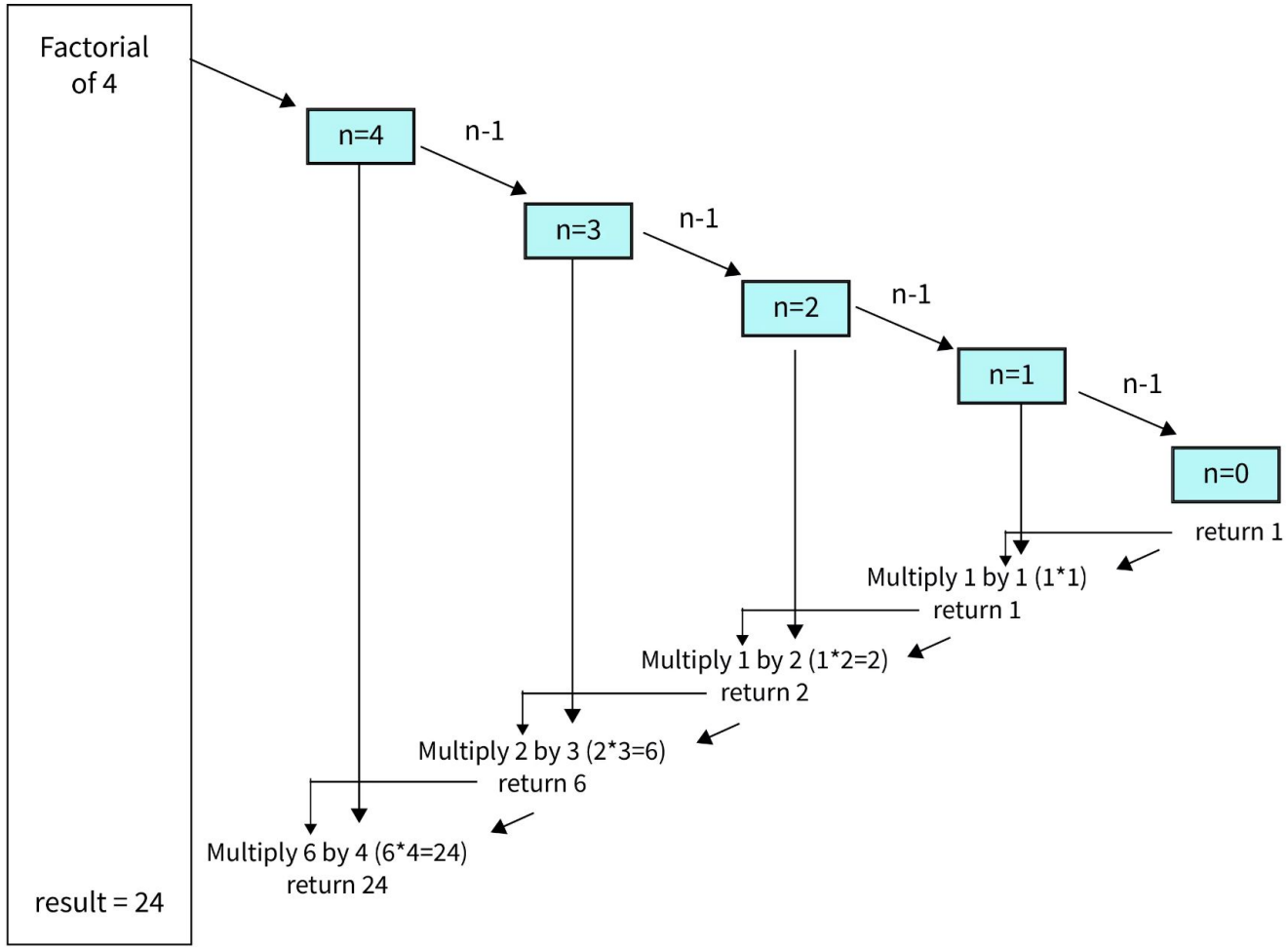


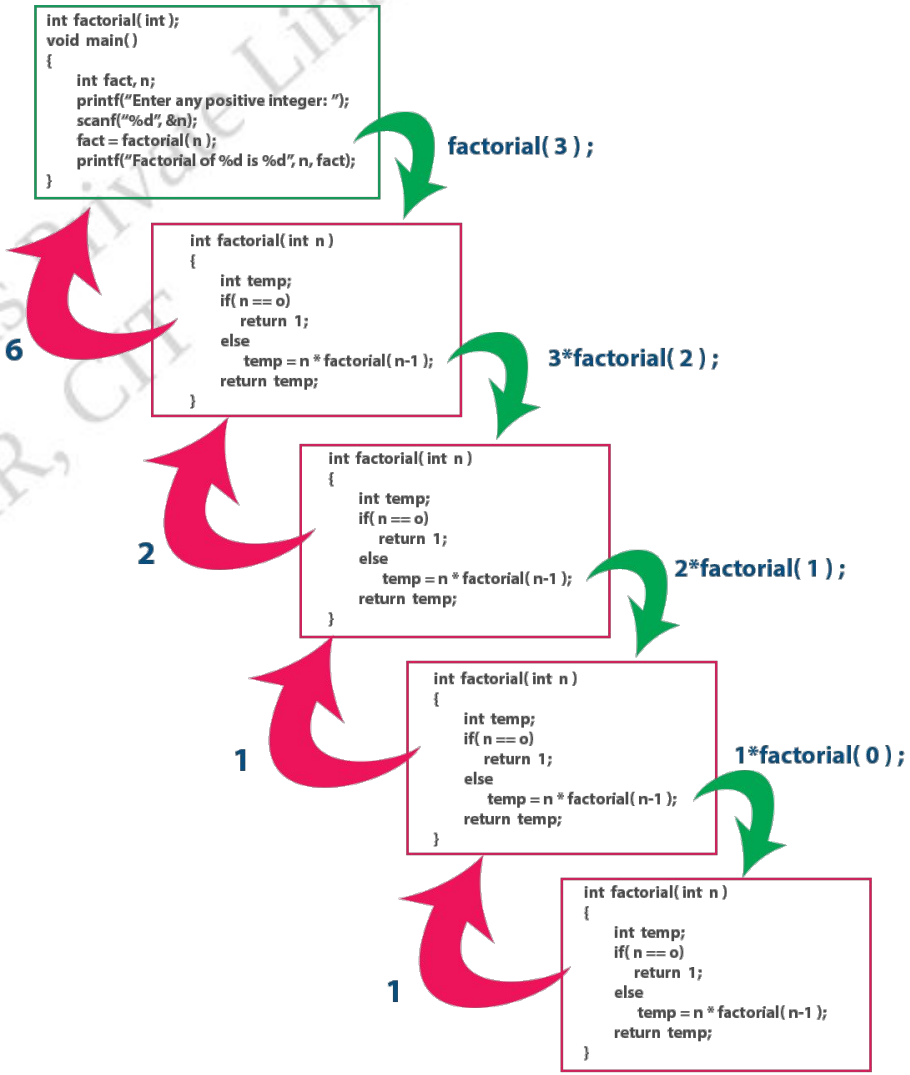
ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Program to Find the Factorial Using Recursion

```
#include <stdio.h>
unsigned int factorial(unsigned int n) {
    // Base Case:
    if (n == 1) {
    return 1;
    }
    // Multiplying the current N with the previous product
    // of Ns
    return n * factorial(n - 1);
}
int main() {
    int num = 5;
    printf("Factorial of %d is %d", num, factorial(num));
    return 0;
}
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR CIT



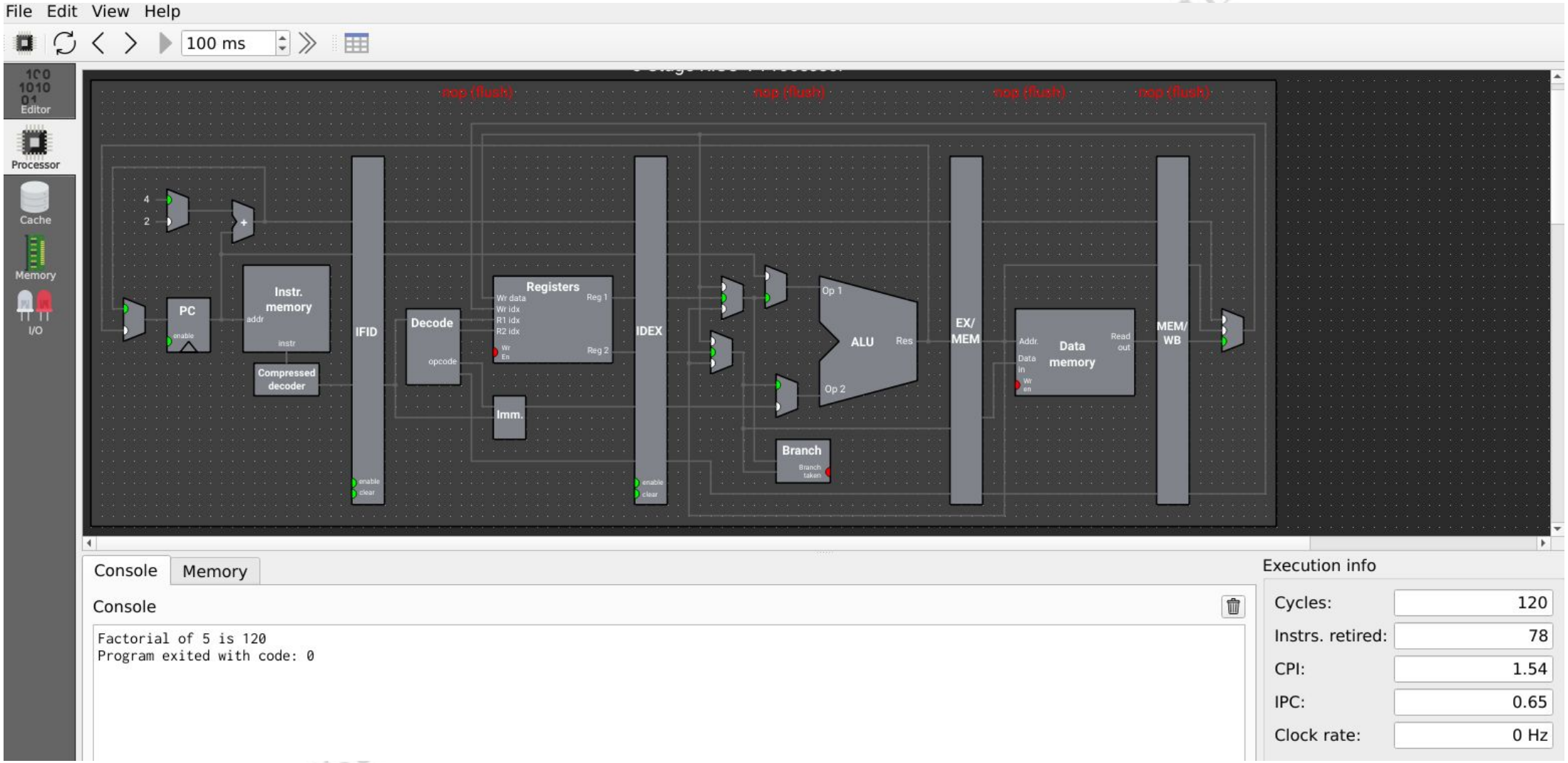


ANGSTROMERS Navigating Silicon Waves
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

Factorial of a number with recursion

```
.data
num:   .word  5
str1:  .string "Factorial of "
str2:  .string " is "
.text
main:
        lw a0, num
        jal ra, factorial
        mv a1, a0
        j printResult
factorial:
        addi sp, sp, -8
        sw ra, 4(sp)
        sw a0, 0(sp)
        li t0, 1
        beq a0, t0, base_case  # If a0 == 1, return 1
        addi a0, a0, -1
        jal ra, factorial  # Recursive call
        lw a1, 0(sp)
        mul a0, a0, a1
base_case:
        lw ra, 4(sp)
        addi sp, sp, 8
        ret
printResult:
        la a0, str1
        li a7, 4
        ecall
        lw a0, num
        li a7, 1
        ecall
        la a0, str2
        li a7, 4
        ecall
        mv a0, a1
        li a7, 1
        ecall
        li a7, 10
        ecall
```

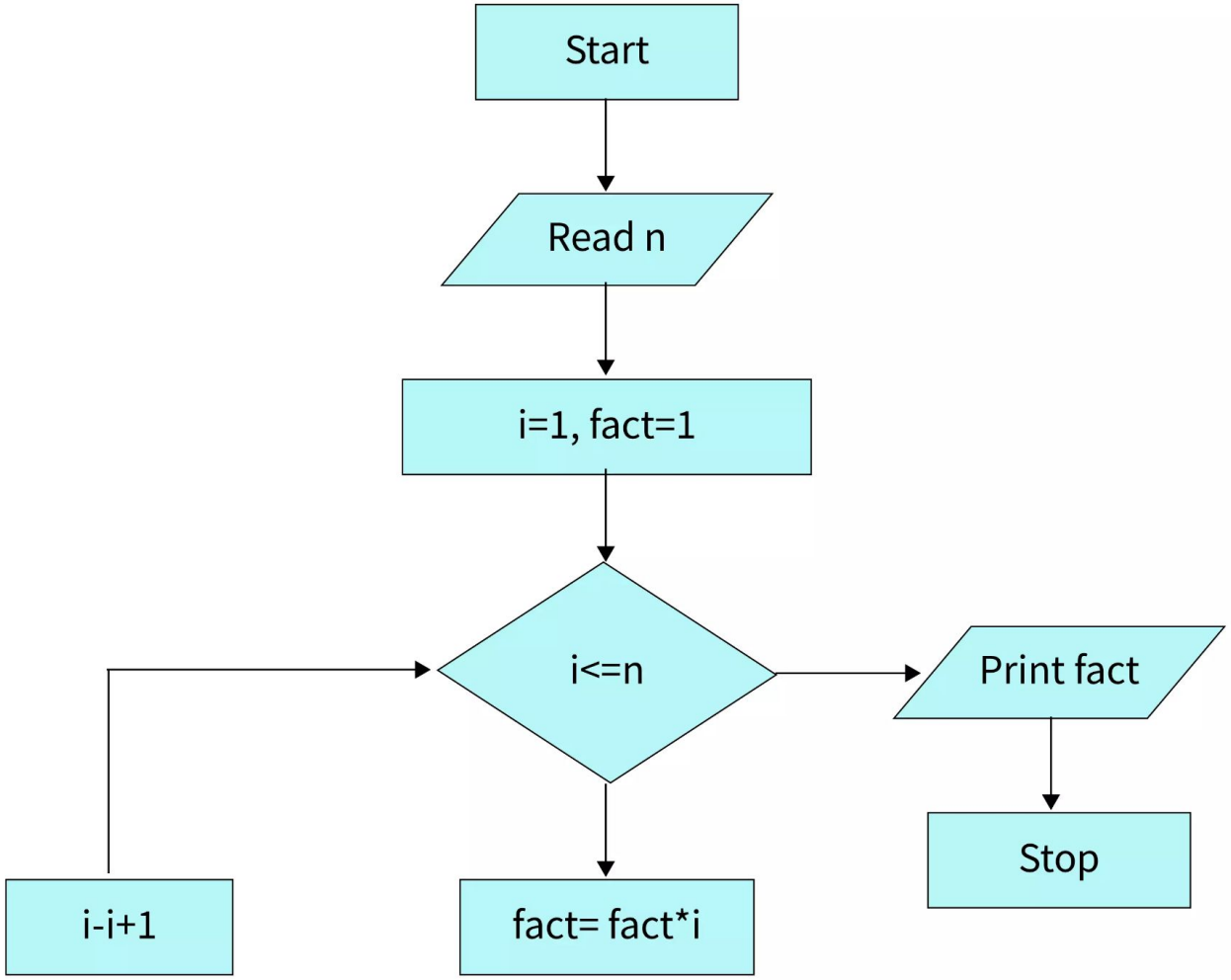
Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT


File Edit View Help
100 ms
Editor
Processor
Cache
Memory
I/O
nop (flush)
nop (flush)
nop (flush)
nop (flush)
PC
Instr. memory
Compressed decoder
IFID
Decode
Registers
Imm.
IDEX
ALU
Branch
EX/ MEM
Data memory
MEM/ WB
Console
Memory
Console
Factorial of 5 is 120
Program exited with code: 0
Execution info
Cycles: 120
Instrs. retired: 78
CPI: 1.54
IPC: 0.65
Clock rate: 0 Hz

Program to Find the Factorial without Recursion

```
#include <stdio.h>
unsigned int factorial(unsigned int N) {
    int fact = 1, i;
    // Loop from 1 to N to get the factorial
    for (i = 1; i <= N; i++) {
    fact *= i;
    }
    return fact;
}
int main() {
    int N = 5;
    int fact = factorial(N);
    printf("Factorial of %d is %d", N, fact);
    return 0;
}
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

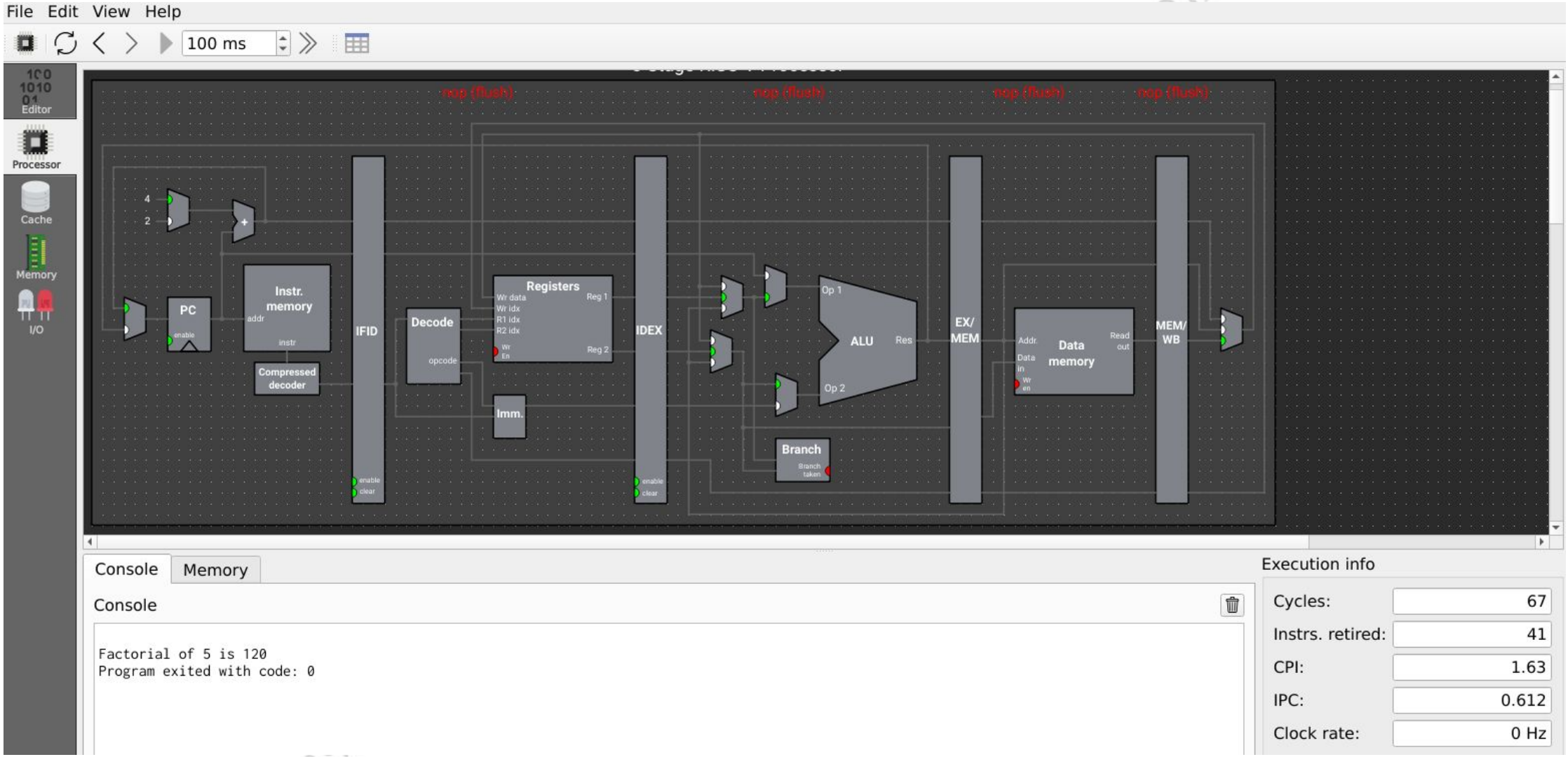

Start
Read n
i=1, fact=1
i<=n
Print fact
Stop
fact= fact*i
i-i+1

Factorial of a number without recursion

```
.data
num:   .word  5
str1:  .string "Factorial of "
str2:  .string " is "

.text
main:
        # Load number from static data
        lw  a0, num
        li  a1, 1  # Initialize factorial result to 1

        # Compute factorial iteratively
loop:
        beqz a0, printResult  # If a0 == 0, go
to printResult

        mul a1, a1, a0  # a1 = a1 * a0
        addi a0, a0, -1  # a0 = a0 - 1
        j loop
```

```
# --- printResult ---
# a1: Computed factorial
printResult:
        la a0, str1
        li a7, 4
        ecall
        lw a0, num
        li a7, 1
        ecall
        la a0, str2
        li a7, 4
        ecall
        mv a0, a1
        li a7, 1
        ecall
        li a7, 10
        ecall
```

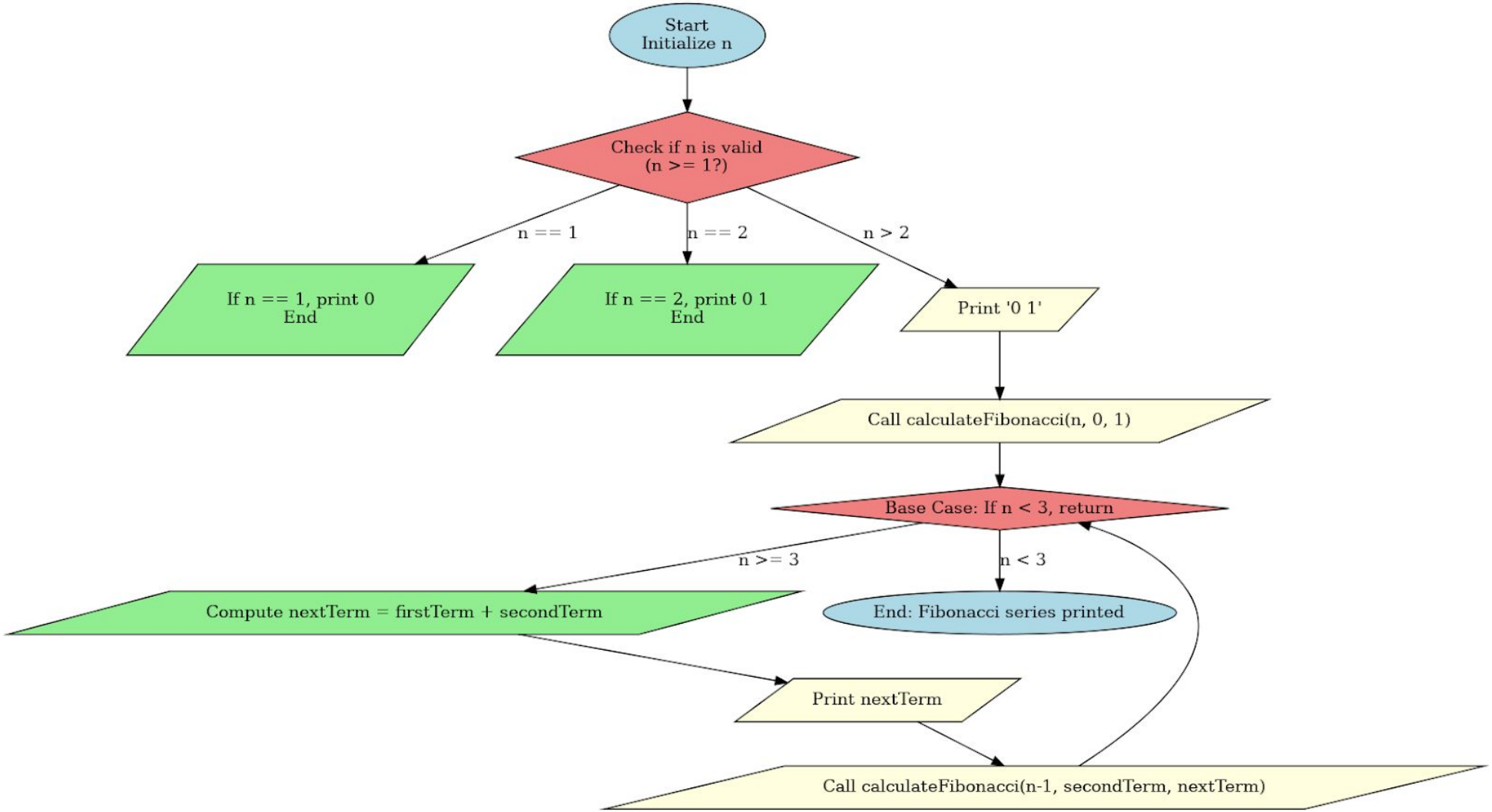
Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT


File Edit View Help
100 ms
Editor
Processor
Cache
Memory
I/O
nop (flush)
nop (flush)
nop (flush)
nop (flush)
PC
Instr. memory
Compressed decoder
IFID
Decode
Registers
Imm.
IDEX
ALU
Branch
EX/ MEM
Data memory
MEM/ WB
Console
Memory
Console
Factorial of 5 is 120
Program exited with code: 0
Execution info
Cycles: 67
Instrs. retired: 41
CPI: 1.63
IPC: 0.612
Clock rate: 0 Hz

Program to Find the Fibonacci with Recursion

```
#include <stdio.h>
void calculateFibonacci(int n, int firstTerm, int
secondTerm) {
if (n < 3) {
return;
}
 // Calculate the next Fibonacci term
int nextTerm = firstTerm + secondTerm;
printf("%d ", nextTerm);  // Print the current term
// Recursive call with updated terms for the next iteration
calculateFibonacci(n - 1, secondTerm, nextTerm);
}
// Function to handle the first two terms and call the
recursive function
void printFibonacci(int n) {
// Handle edge cases for invalid input
if (n < 1) {
printf("Invalid input: Number of terms should be greater
than or equal to 1\n");
return;
}
// Handle the case when only one term is requested
if (n == 1) {
printf("0 ");
return;
}// Handle the case when two terms are requested
if (n == 2) {
printf("0 1 ");
return;
}
// Print the first two terms and then call the recursive
function for the rest
printf("0 1 ");
calculateFibonacci(n, 0, 1);  // Start the recursive calculation
}
int main() {
int n = 9;  // Set the number of terms in the Fibonacci series
printFibonacci(n);  // Print the Fibonacci series up to the nth
term
return 0;
}
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT


Start
Initialize n
Check if n is valid
(n >= 1?)
n == 1
n == 2
n > 2
If n == 1, print 0
End
If n == 2, print 0 1
End
Print '0 1'
Call calculateFibonacci(n, 0, 1)
Base Case: If n < 3, return
n >= 3
n < 3
Compute nextTerm = firstTerm + secondTerm
End: Fibonacci series printed
Print nextTerm
Call calculateFibonacci(n-1, secondTerm, nextTerm)

Fibonacci series with recursion

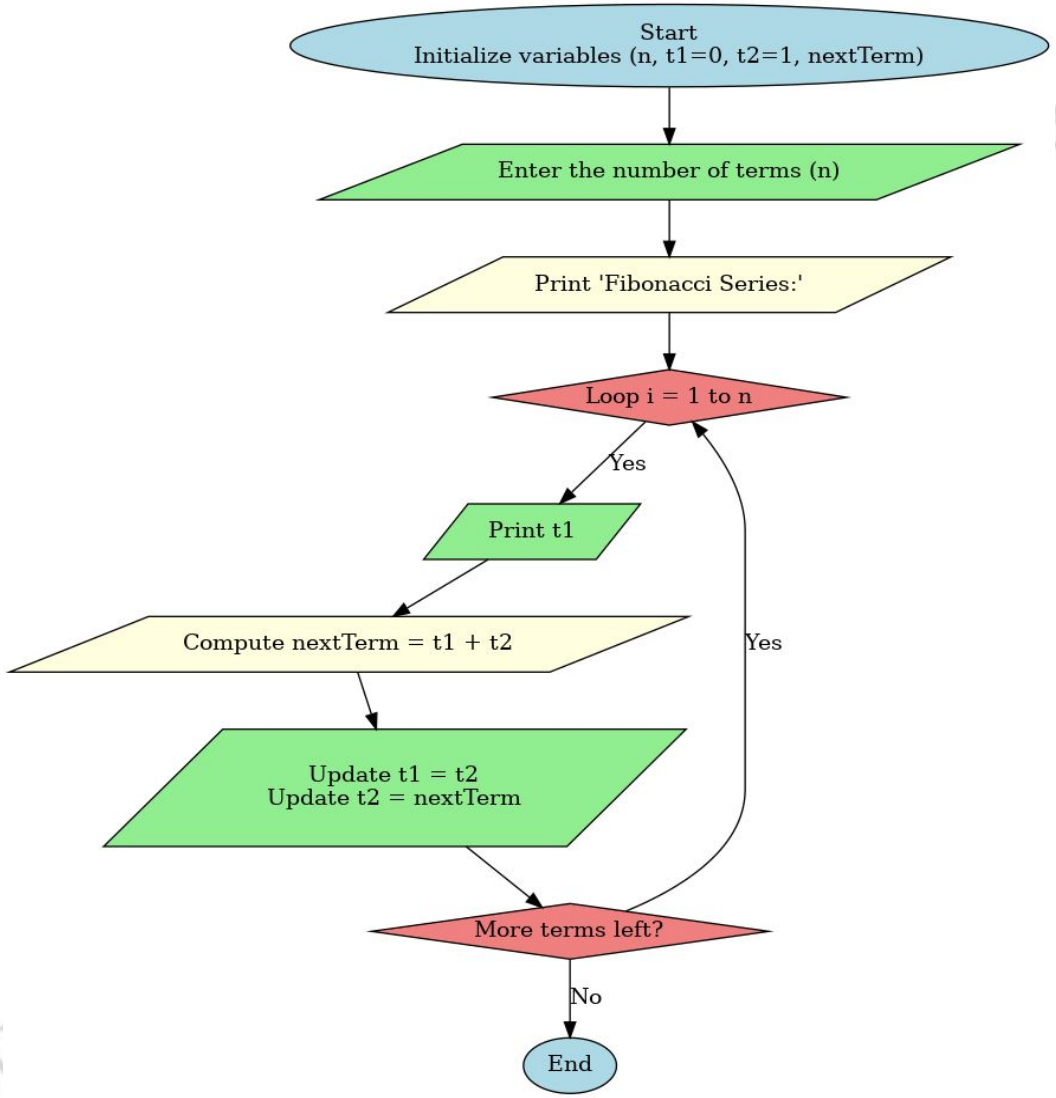
```
.data
num:   .word  5
str1:  .string "Fibonacci series up to "
str2:  .string " terms: "
space: .string " "
.text
main:
        lw a0, num
        li a1, 0   # First term
        li a2, 1   # Second term
        li a3, 0   # Counter
        jal ra, fibonacci_series
        li a7, 10  # Exit
        ecall
# --- Recursive Fibonacci Series ---
fibonacci_series:
        addi sp, sp, -12
        sw ra, 8(sp)
        sw a0, 4(sp)
        sw a3, 0(sp)
        lw t0, num
        bge a3, t0, end_recursion  # If counter >= num, return
        mv a0, a1
        li a7, 1  # Print Fibonacci number
        ecall
        la a0, space
        li a7, 4  # Print space
        ecall
        mv t1, a1
        mv a1, a2
        add a2, a2, t1
        addi a3, a3, 1
        jal ra, fibonacci_series
end_recursion:
        lw ra, 8(sp)
        lw a0, 4(sp)
        lw a3, 0(sp)
        addi sp, sp, 12
        ret
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Program to Find the Fibonacci without Recursion

```
#include <stdio.h>
int main() {
int n, t1 = 0, t2 = 1, nextTerm;
                                        // Get the number of terms from the user
printf("Enter the number of terms: ");
scanf("%d", &n);
printf("Fibonacci Series: ");
for (int i = 1; i <= n; i++) {
printf("%d ", t1);
nextTerm = t1 + t2;                     // Compute the next term
t1 = t2; // Update previous term
t2 = nextTerm;                          // Move to the next term
}
return 0;
}
```

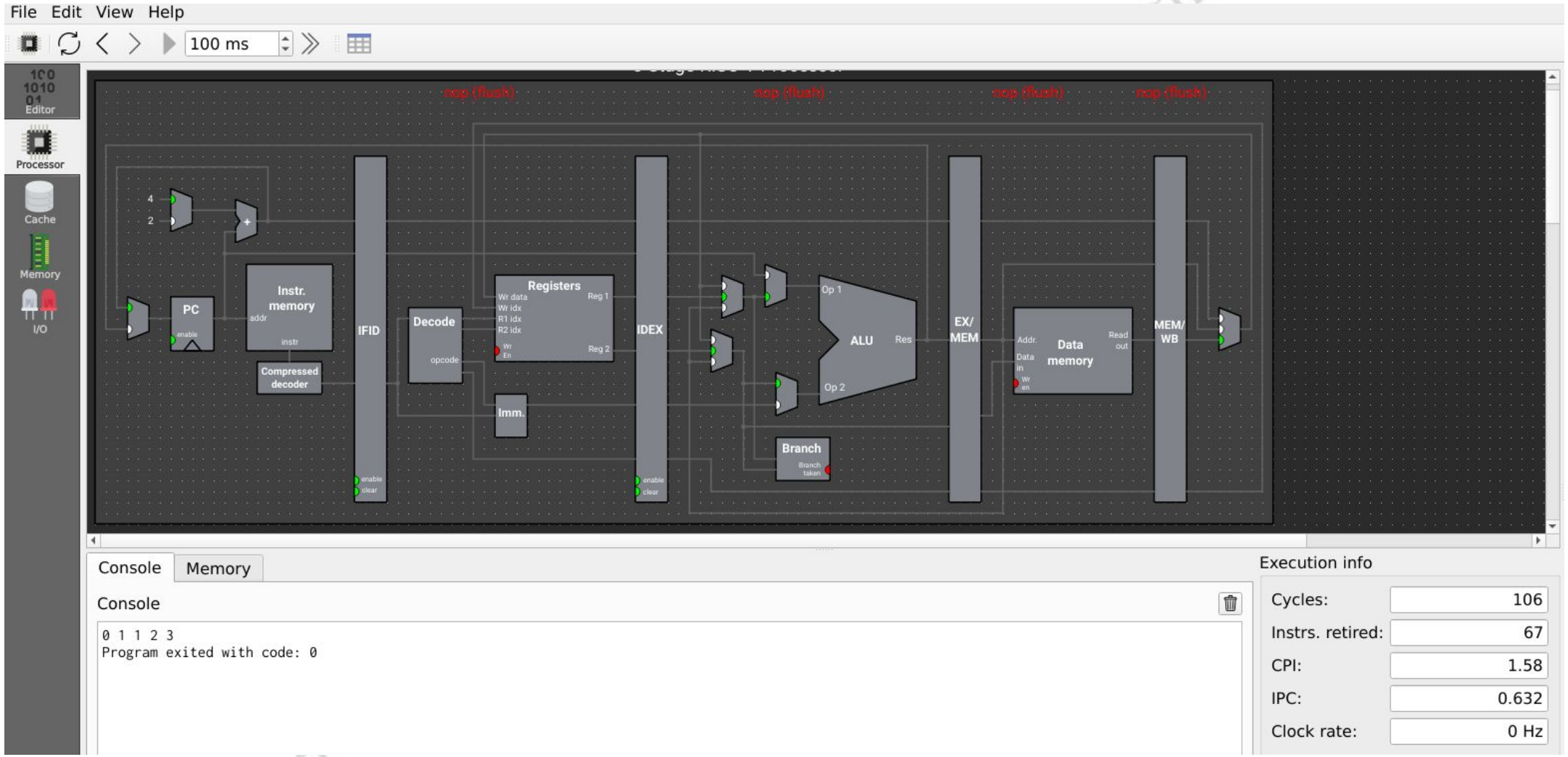
Angstromers Engineering Solutions Private Limited
Incubated of CCCIR, CIT


Start
Initialize variables (n, t1=0, t2=1, nextTerm)
Enter the number of terms (n)
Print 'Fibonacci Series:'
Loop i = 1 to n
Yes
Print t1
Compute nextTerm = t1 + t2
Update t1 = t2
Update t2 = nextTerm
Yes
More terms left?
No
End

Fibonacci series without recursion

```
.data
num:   .word  5
str1:  .string "Fibonacci series up to "
str2:  .string " terms: "
space: .string " "
.text
main:
        lw a0, num
        li a1, 0   # First term
        li a2, 1   # Second term
        li a3, 2   # Counter(first two terms are known)
        j print_first_two
loop:
        add a4, a1, a2   # Compute next Fibonacci number
        mv a1, a2        # Update previous terms
        mv a2, a4
        mv a0, a4
        li a7, 1         # Print integer
        ecall
        la a0, space
        li a7, 4         # Print space
        ecall
        addi a3, a3, 1
        lw t0, num
        bge a3, t0, exit # If counter >= num,
exit loop
        j loop
print_first_two:
        mv a0, a1
        li a7, 1         # Print first term
        ecall
        la a0, space
        li a7, 4         # Print space
        ecall
        mv a0, a2
        li a7, 1         # Print second term
        ecall
        la a0, space
        li a7, 4         # Print space
        ecall
        j loop
exit:
        li a7, 10
        ecall
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT


File Edit View Help
100 ms
Editor
Processor
Cache
Memory
I/O
nop (flush)
nop (flush)
nop (flush)
nop (flush)
4
2
PC
enable
Instr. memory
addr
instr
Compressed decoder
IFID
enable
clear
Decode
opcode
Registers
Wr data
Wr idx
R1 idx
R2 idx
Wr En
Reg 1
Reg 2
Imm.
IDEX
enable
clear
Op 1
ALU
Res
Op 2
Branch
Branch taken
EX/ MEM
Addr.
Data in
Wr en
Data memory
Read out
MEM/ WB
Console
Memory
Console
0 1 1 2 3
Program exited with code: 0
Execution info
Cycles: 106
Instrs. retired: 67
CPI: 1.58
IPC: 0.632
Clock rate: 0 Hz

Thank you

Angstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT