RISC V Assembly Programs

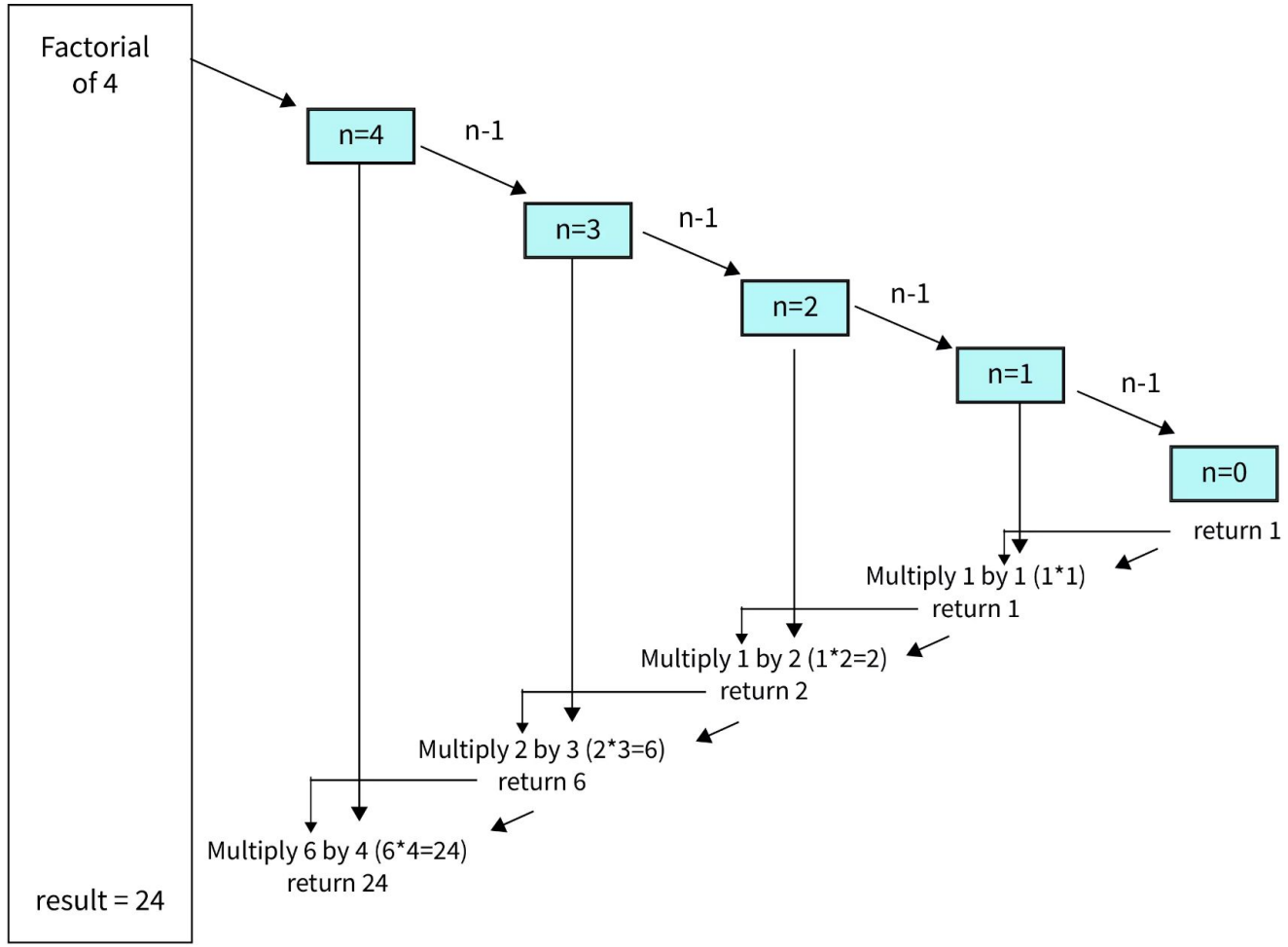
Dr. Girish H
Professor
Department of ECE
Cambridge Institute of Technology
&
Kavinesh
Research Staff
CCCIR
Cambridge Institute of Technology

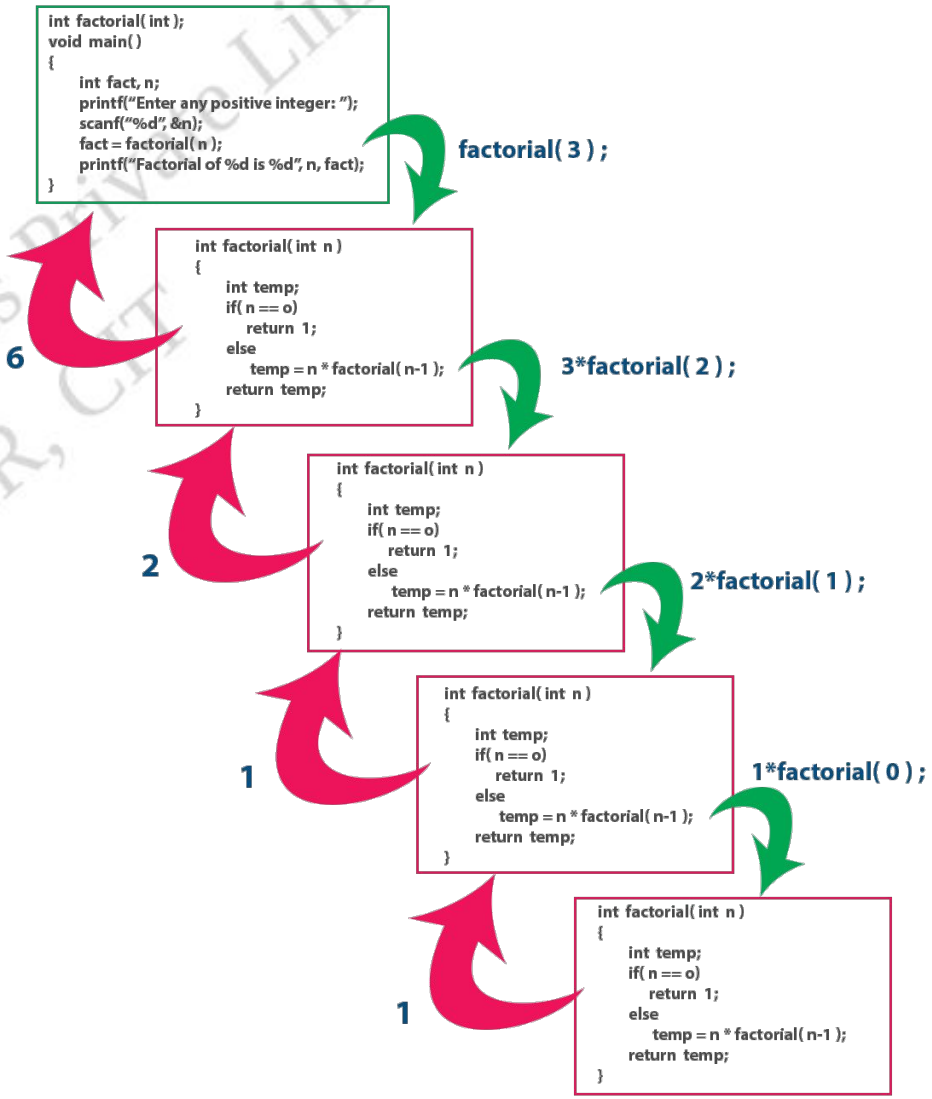
ANGSTROMERS
Navigating Silicon Waves
CCCIR CAMBRIAN
CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY
An Autonomous Institution
Chips to Startup Programme

Program to Find the Factorial

```
#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;
}
```

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

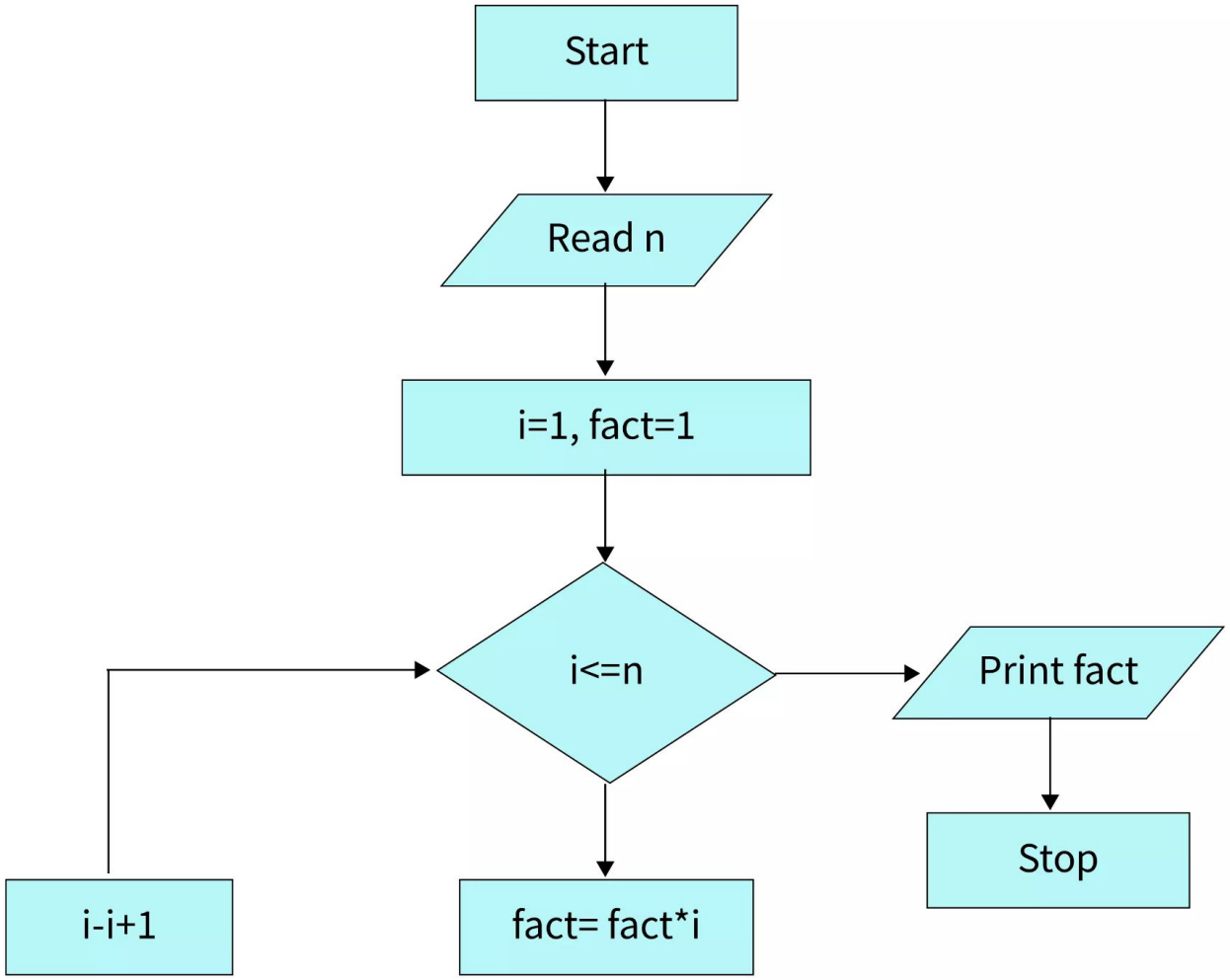

Factorial of 4
n=4
n-1
n=3
n-1
n=2
n-1
n=1
n-1
n=0
return 1
Multiply 1 by 1 (1*1)
return 1
Multiply 1 by 2 (1*2=2)
return 2
Multiply 2 by 3 (2*3=6)
return 6
Multiply 6 by 4 (6*4=24)
return 24
result = 24


int factorial(int);
void main( )
{
int fact, n;
printf("Enter any positive integer: ");
scanf("%d", &n);
fact = factorial(n);
printf("Factorial of %d is %d", n, fact);
}
factorial(3) ;
int factorial(int n)
{
int temp;
if(n == o)
return 1;
else
temp = n * factorial(n-1);
return temp;
}
6
3*factorial(2) ;
2
2*factorial(1) ;
1
1*factorial(0) ;
1

Program to Find the Factorial using FSM

```
#include <stdio.h>
typedef enum {
        INIT,
        CALC,
        DONE,
        ERROR
} State;
int main() {
        int n = 5;  /
        int i;
        unsigned long long result;
        State state = INIT;
        while (1) {
        switch (state) {
                case INIT:
                if (n < 0) {
 state = ERROR;
}
  } else {
                result = 1;
                i = 1;
                state = CALC;
        }
        break;
        case CALC:
        if (i > n) {
                state = DONE;
        } else {
                result *= i;
                I++;        }
        break;
        case DONE:
printf("Factorial of %d is %llu\n", n, result);
        return 0
        case ERROR:
        printf("Invalid hardcoded input!\n");
        return 1;   } }
  return 0;   }
```

ngstromers 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

Result

Factorial simulation using spike

```
copi-001@copi001-OptiPlex-7050:~/spike/fact$ spike pk -s a.out
bbl loader
Factorial of 5 is 1201400 ticks
57531 cycles
57531 instructions
1.00 CPI
```

FSM factorial

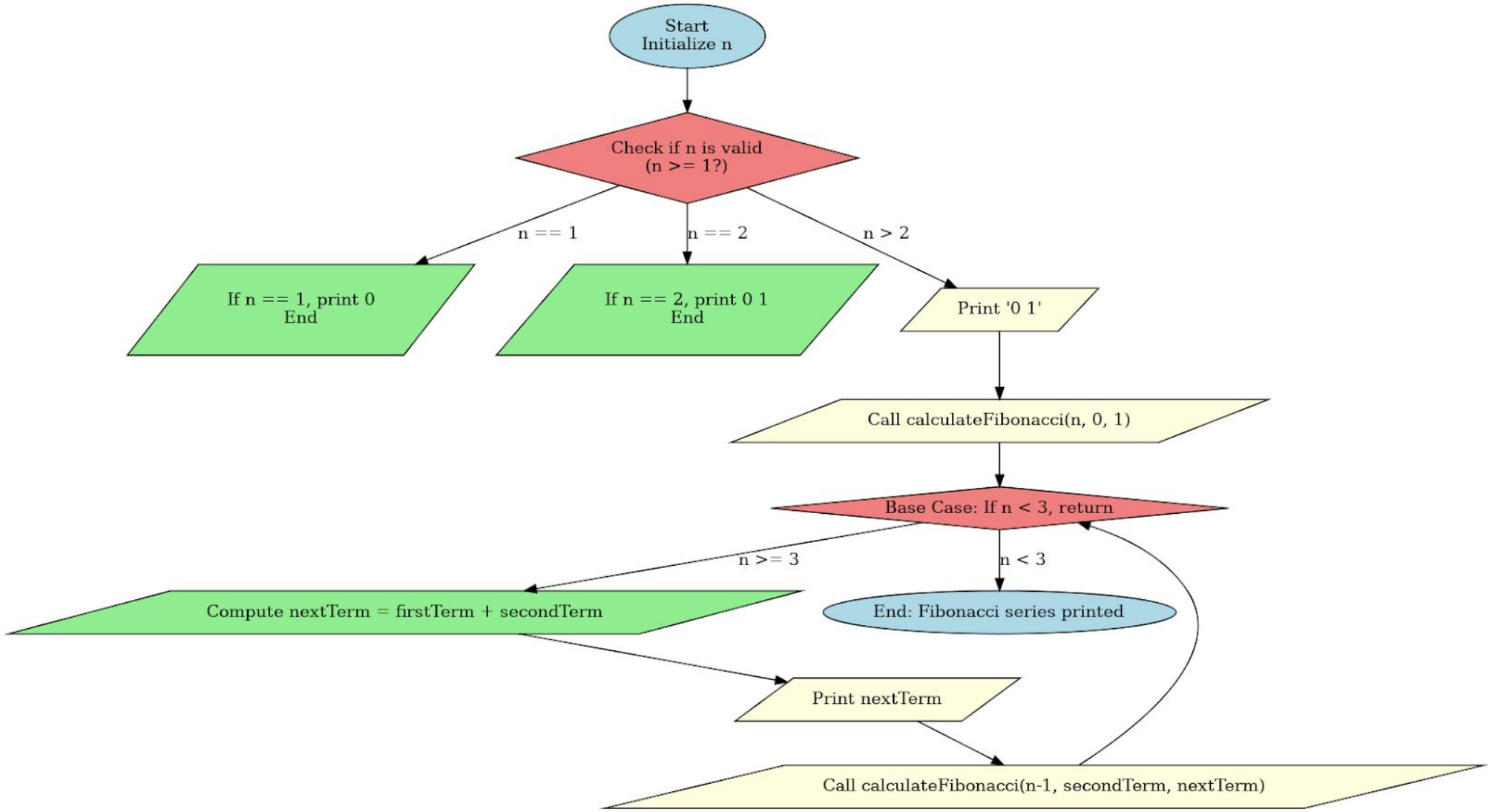
```
copi-001@copi001-OptiPlex-7050:~/spike/fsmfact$ spike pk -s a.out
bbl loader
Factorial of 5 is 120
1400 ticks
57900 cycles
57900 instructions
1.00 CPI
```

Private Limited
of CCCIR, CIT

Program to Find the Fibonacci

```
#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;
}
```

ngstromers Engineering Solutions Private Limited
Incubatee of CCIR, CIT



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

FSM Program Fibonacci

```
#include <stdio.h>
typedef enum {
        INIT,
        PRINT_TERM,
        CALCULATE_NEXT,
        DONE,
        ERROR
} State;
int main() {
        int n = 10;
        int t1 = 0, t2 = 1, nextTerm;
        int count = 1;
        State state = INIT;
        while (1) {
        switch (state) {
        case INIT:
                if (n <= 0) {
                state = ERROR;
                } else {
                printf("Fibonacci Series: ");
                state = PRINT_TERM;
                }
                break;
        case PRINT_TERM:
printf("%d ", t1);
                state = CALCULATE_NEXT;
                break;
        case CALCULATE_NEXT:
                nextTerm = t1 + t2;
                t1 = t2;
                t2 = nextTerm;
                count++;
                if (count > n) {
                state = DONE;
                } else {
                state = PRINT_TERM;
                }
                break;
        case DONE:
                printf("\n");
                return 0;
        case ERROR:
                printf("Invalid number of terms!\n");
                return 1;
        }}
        return 0;
}
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCIR, CIT

Result

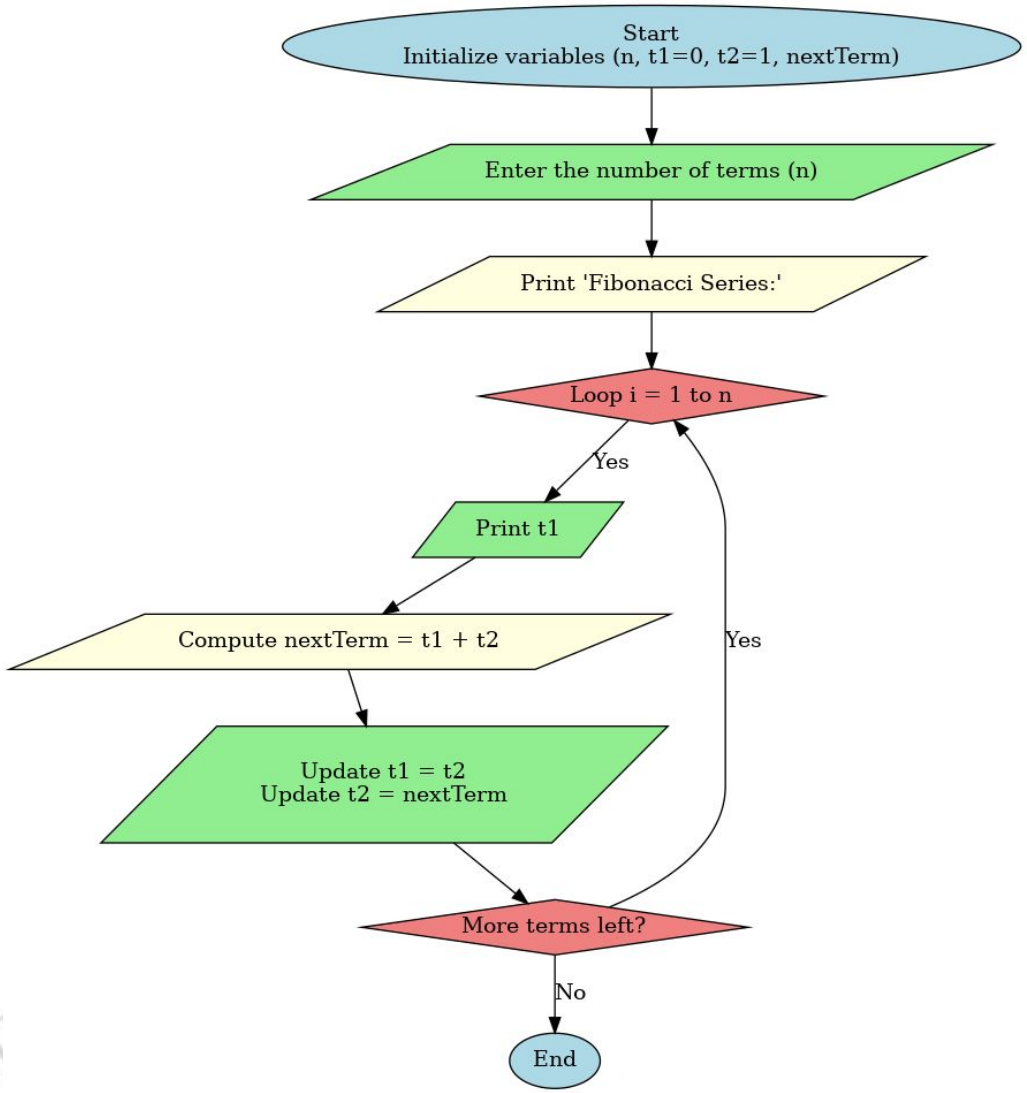
Fibonacci simulation using spike

```
copi-001@copi001-OptiPlex-7050:~/spike/fib$ spike pk -s a.out
bbl loader
0 1 1 2 3 5 8 13 21 1400 ticks
60860 cycles
60860 instructions
1.00 CPI
```

FSM Fibonacci

```
copi-001@copi001-OptiPlex-7050:~/spike/fsmfib$ spike pk -s a.out
bbl loader
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
1500 ticks
67318 cycles
67318 instructions
1.00 CPI
```

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


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
More terms left?
Yes
No
End

Matrix Multiplication

```
#include <stdio.h>
#define N 3  // Matrix size (N x N)

void matrix_multiply(int A[N][N], int
B[N][N], int C[N][N]) {
    // Standard triple-nested loop for matrix
multiplication
    for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        C[i][j] = 0;  // Initialize the result
element
        for (int k = 0; k < N; k++) {
        C[i][j] += A[i][k] * B[k][j];
        } } } }
int main() {
    int A[N][N] = {{1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}};
    int B[N][N] = {{9, 8, 7},
        {6, 5, 4},
        {3, 2, 1}};
    int C[N][N] = {0};  // Result matrix

    matrix_multiply(A, B, C);

    // Print the resulting matrix
    printf("Resultant Matrix C:\n");
    for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        printf("%d ", C[i][j]);
    }
    printf("\n");
    }
    return 0;
}
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCIR, CIT

Matrix multiplication using FSM

```
#include <stdio.h>
#define N 3
typedef enum {
        INIT,
        MULTIPLY,
        ACCUMULATE,
        DONE
} State;
void matrix_multiply_fsm(int A[N][N],
                int B[N][N], int C[N][N]) {
        int i = 0, j = 0, k = 0;
        int temp_sum = 0;
        State state = INIT;  // Initial state
        while (state != DONE) {
        switch (state) {
        case INIT:
                temp_sum = 0;
                if (i < N) {
                if (j < N) {
                if (k < N) {
                        state = MULTIPLY;
                } else {
                        state = ACCUMULATE;
}
        } else {
                j = 0;
                i++;
                state = INIT;
                }
                } else {
                state = DONE;
                }
                break;
case MULTIPLY:
temp_sum += A[i][k] * B[k][j];
        k++;
if (k < N) {
 state = MULTIPLY;
 } else {
  state = ACCUMULATE;
 } break;
case ACCUMULATE:
        C[i][j] = temp_sum;
        j++;
        k = 0;
        state = INIT;
break;
case DONE:
printf("Matrix multiplication complete.\n");
         break;
        default:
         state = DONE;
         break;
        }}}
int main() {
int A[N][N] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
int B[N][N] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
int C[N][N] = {0};  // Result matrix
        matrix_multiply_fsm(A, B, C);

        printf("Resultant Matrix C:\n");
        for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
        printf("%d ", C[i][j]);
        }
        printf("\n");
        }
        return 0;
}
```

Angstromers Engineering Solutions Private Limited
Incubatee of CCEIR, CIT

Result

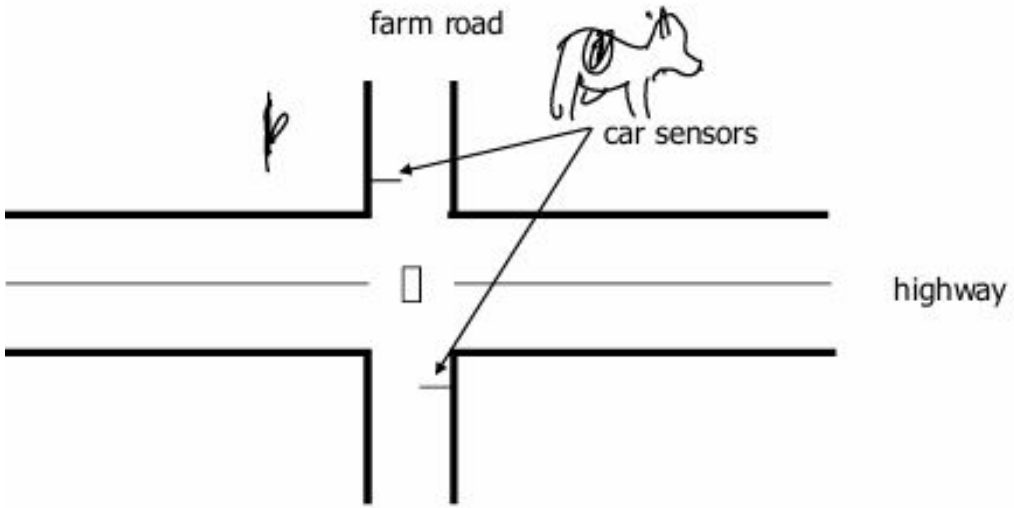
Matrix simulation using spike

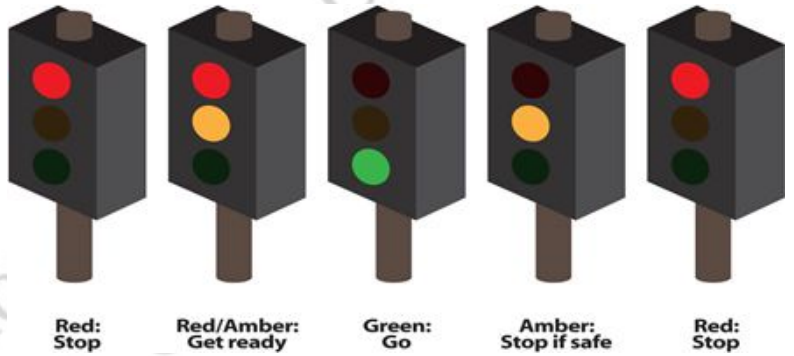
```
copi-001@copi001-OptiPlex-7050:~/spike/mat$ spike pk -s a.out
bbl loader
Resultant Matrix C:
30 24 18
84 69 54
138 114 90
1750 ticks
84242 cycles
84242 instructions
1.00 CPI
```

FSM Matrix multiplication

```
copi-001@copi001-OptiPlex-7050:~/spike/fsmmat$ spike pk -s a.out
bbl loader
Resultant Matrix C:
30 24 18
84 69 54
138 114 90
1750 ticks
84918 cycles
84918 instructions
1.00 CPI
```

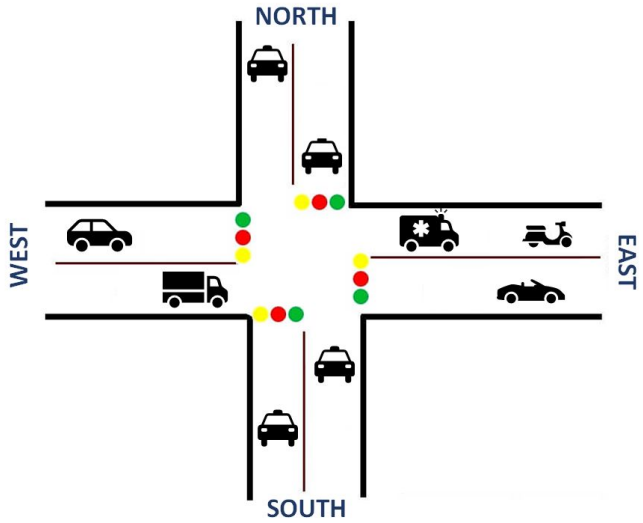
Finite State Machine model for Traffic Light Controller



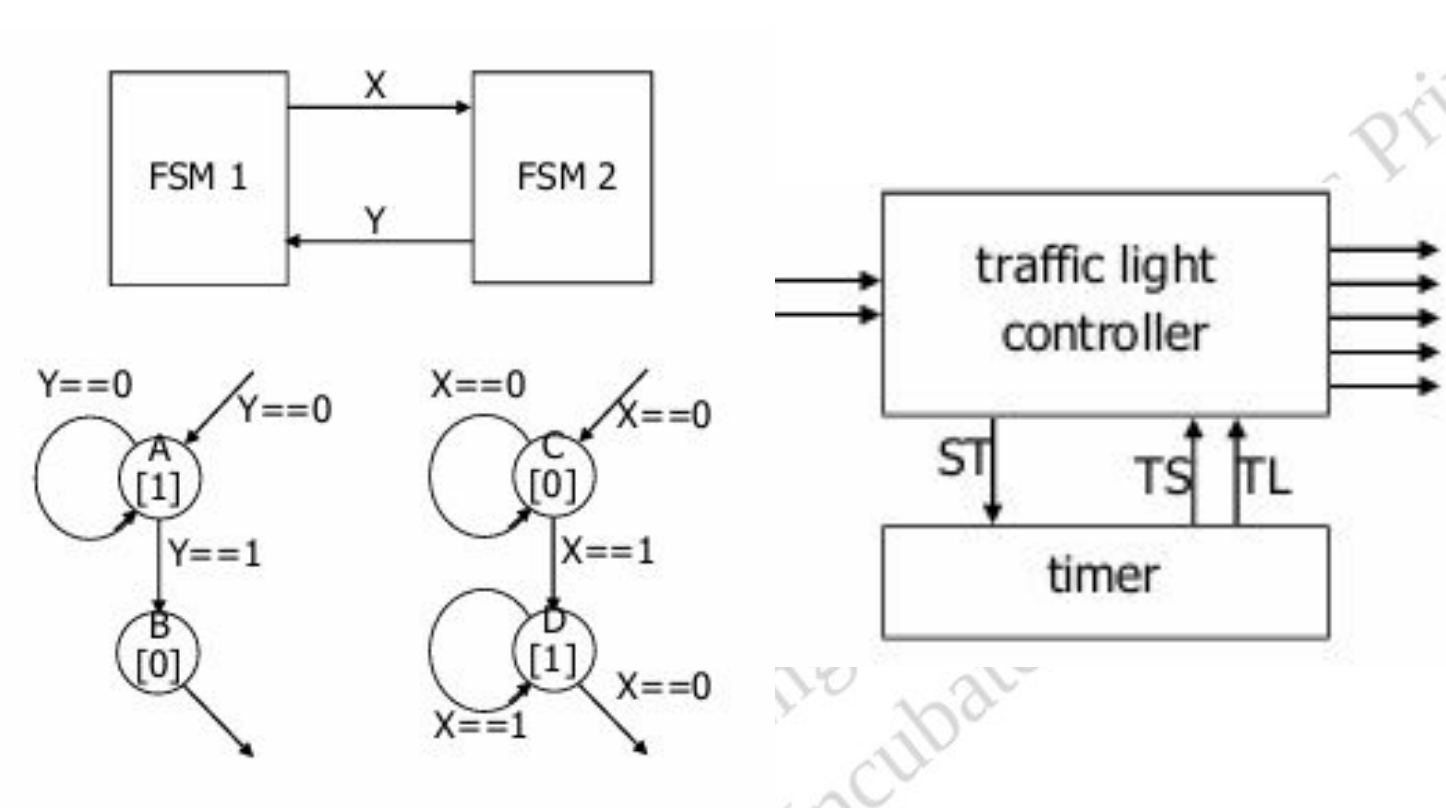


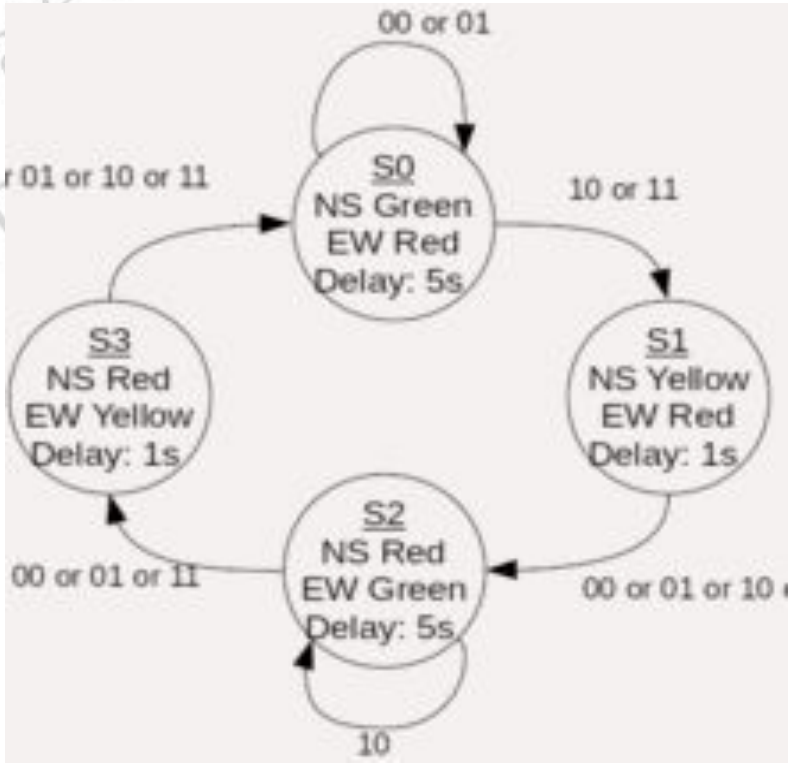
Different state of input

State	State (Binary)	State Name	East	North	West	South
0	000	East_green	Green	Red	Red	Red
1	001	East_yellow	Yellow	Red-Yellow	Red	Red
2	010	North_green	Red	Green	Red	Red
3	011	North_yellow	Red	Yellow	Red-Yellow	Red
4	100	West_green	Red	Red	Green	Red
5	101	West_yellow	Red	Red	Yellow	Red-Yellow
6	110	South_green	Red	Red	Red	Green
7	111	South_yellow	Red-Yellow	Red	Red	Yellow



Finite state machine for Traffic Light controller





State Diagram

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

FSM code for Traffic Light controller

```
#include <stdio.h>

typedef enum { RED, YELLOW, GREEN } State;

void trafficLight(State *state) {
    switch (*state) {
        case RED:
        printf("Stop! The light is RED.\n");
        *state = GREEN; // Transition to GREEN
         break;
        case GREEN:
        printf("Go! The light is GREEN.\n");
       *state = YELLOW; // Transition to YELLOW
            break;
        case YELLOW:
       printf("Caution! The light is YELLOW.\n");
        *state = RED; // Transition to RED
            break;
    }
}

int main() {
    State currentState = RED; // Initial state
    for (int i = 0; i < 6; i++) {
        trafficLight(&currentState);
    }
    return 0;
}
```

ngstromers Engineering Solutions Private Limited
Incubatee of C-CIR, CIT

Simulation using spike

Compilation command: **riscv64-unknown-elf-gcc main.c**

Simulation command : **spike pk a.out**

```
copi-003@copi003-OptiPlex-7040:~/Music/spike/traffic$ spike pk -s a.out
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
1700 ticks
80944 cycles
80944 instructions
1.00 CPI
```

Compilation command for Assembly : **riscv64-unknown-elf-gcc main.s**

Simulation command : **spike pk a.out**

Simulation using spike

Assembly Code generation command : **riscv64-unknown-elf-gcc –S main.c**

Compilation command for Assembly : **riscv64-unknown-elf-gcc main.s**

Simulation command : **spike pk a.out**

```
copi-003@copi003-OptiPlex-7040:~/Music/spike/traffic$ spike pk -s a.out
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
1700 ticks
80944 cycles
80944 instructions
1.00 CPI
```

Command to view generated asm file: **cat main.s**

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

Enable profiling metrics in FSM code for Traffic Light controller

```
#include <stdio.h>
unsigned long read_cycles(void){
unsigned long cycles;
asm volatile ("rdcycle %0" : "=r" (cycles));
 return cycles;
}
typedef enum { RED, YELLOW, GREEN } State;
void trafficLight(State *state) {
    switch (*state) {
        case RED:
        printf("Stop! The light is RED.\n");
        *state = GREEN; // Transition to GREEN
         break;
        case GREEN:
        printf("Go! The light is GREEN.\n");
       *state = YELLOW; // Transition to YELLOW
            break;
        case YELLOW:
       printf("Caution! The light is YELLOW.\n");
        *state = RED; // Transition to RED
            break;
    }
}
int main() {
    State currentState = RED; // Initial state
   unsigned long start, stop;
    start = read_cycles();
    for (int i = 0; i < 6; i++) {
        trafficLight(&currentState);
    }
    stop = read_cycles();
   printf(" cycle :%ld\n", stop - start);
    return 0;
}
```

ngstromers Engineering Solutions Private Limited
Incubatee of Cambridge CIT

Result

Simulation using spike

```
copi-003@copi003-OptiPlex-7040:~/Music/spike$ spike pk a.out
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
 cycle :68480
copi-003@copi003-OptiPlex-7040:~/Music/spike$
```

Assembly file generation : **riscv64-unknown-elf-gcc –S main.c**

Command to view generated asm file: **cat main.s**

ngstromers Engineering Limited
Incubated of

Result

Simulation using spike

```
copi-003@copi003-OptiPlex-7040:~/Music/spike$ spike pk a.out
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
 cycle :68384
```

Optimisation command : **riscv64-unknown-elf-gcc –O1 main.c**

Simulation command : **spike pk a.out**

Assembly file generation : **riscv64-unknown-elf-gcc –S main.c**

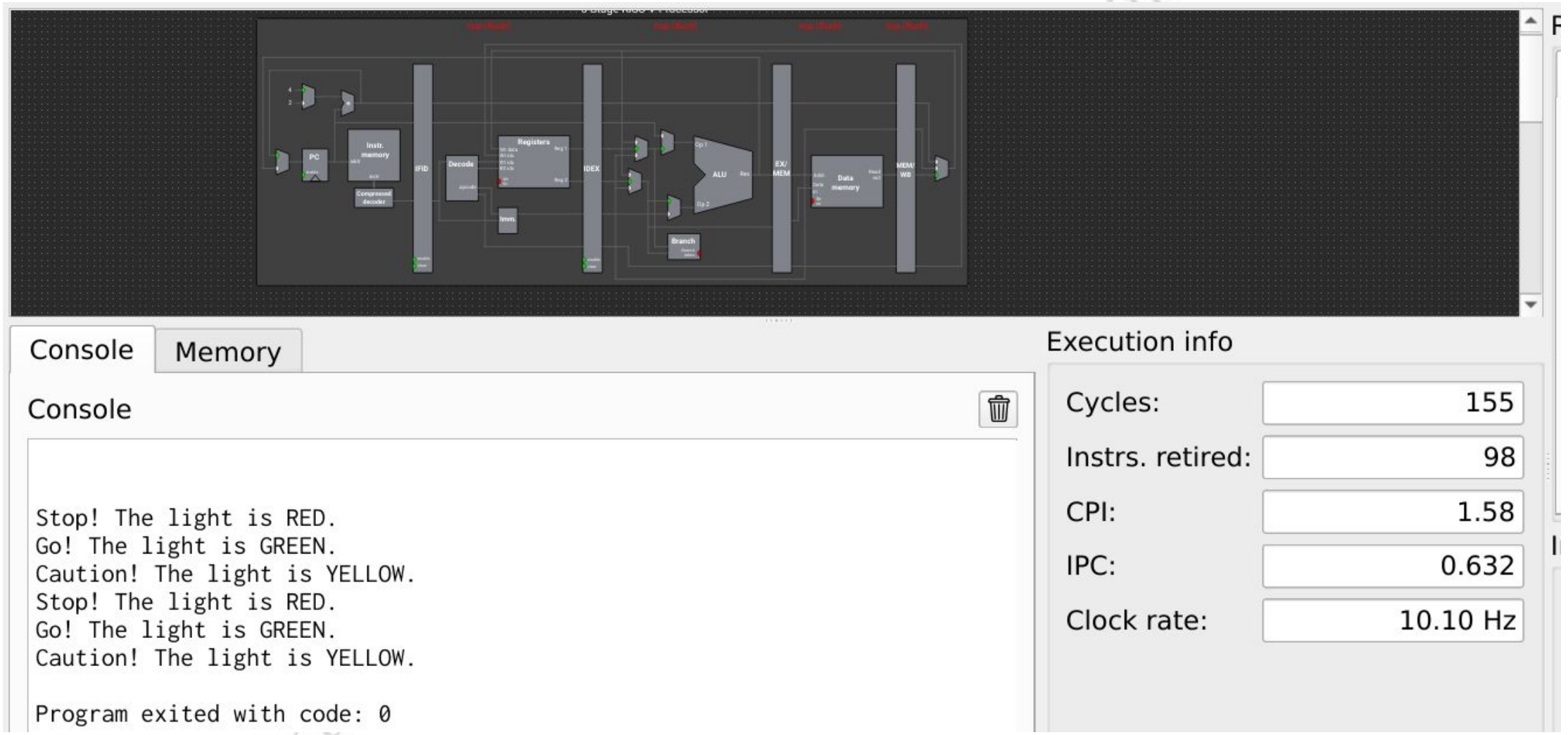
Command to view generated asm file: **cat main.s**

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

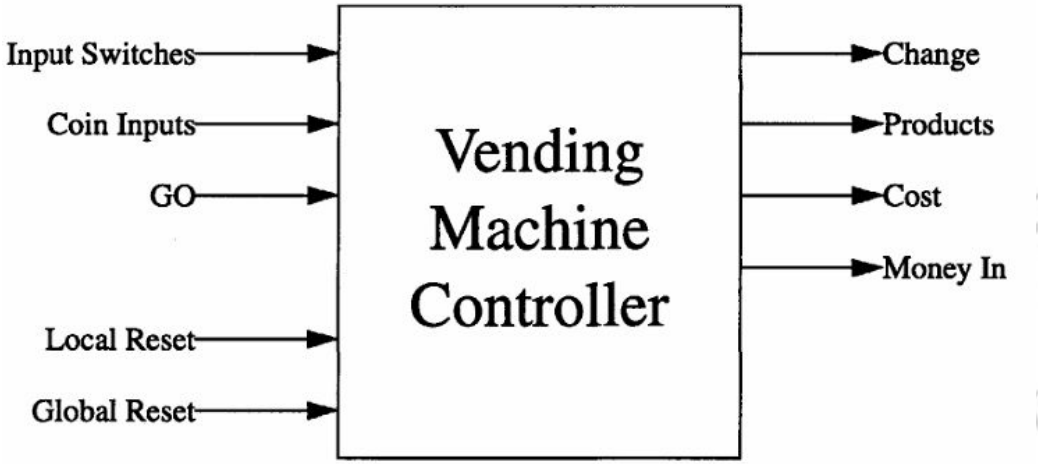
FSM RISC-V assembly for Traffic Light controller

```
.data
state: .word 0        # Initial state = RED (0)
loopCount: .word 6    # Loop 6 times

msgRed:    .asciz "Stop! The light is RED.\n"
msgGreen:  .asciz "Go! The light is GREEN.\n"
msgYellow: .asciz "Caution! The light is YELLOW.\n"
.text
.globl main
main:
   la t0, state       # t0 = &state
   la t1, loopCount   # t1 = &loopCount
   lw t2, 0(t1)       # t2 = loop counter i = 6

loop:
   beq t2, zero, exit  # if i == 0, exit
   lw t3, 0(t0)       # t3 = *state

   # Switch-case: RED = 0, YELLOW = 1, GREEN = 2
   li t4, 0           # RED
   beq t3, t4, case_red

   li t4, 2           # GREEN
   beq t3, t4, case_green

   li t4, 1           # YELLOW
   beq t3, t4, case_yellow

   j end_switch

case_red:
   la a0, msgRed
   li a7, 4           # syscall print_string
   ecall
   li t3, 2           # next state = GREEN (2)
   sw t3, 0(t0)
   j end_switch

case_green:
   la a0, msgGreen
   li a7, 4
   ecall
   li t3, 1           # next state = YELLOW (1)
   sw t3, 0(t0)
   j end_switch

case_yellow:
   la a0, msgYellow
   li a7, 4
   ecall
   li t3, 0           # next state = RED (0)
   sw t3, 0(t0)
   j end_switch

end_switch:
   addi t2, t2, -1    # i--
   j loop
exit:
   li a7, 10          # syscall exit
   ecall
```

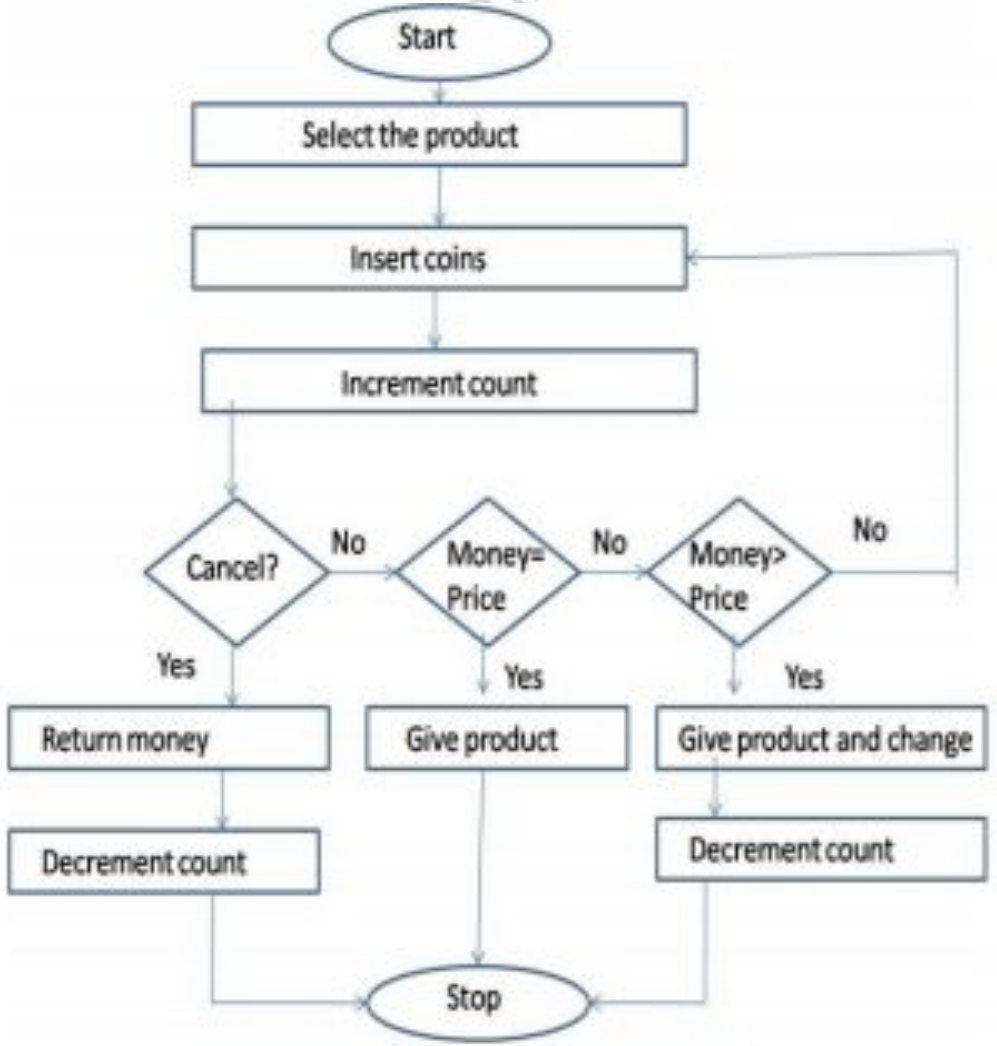
ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT


Console
Memory
Console
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
Stop! The light is RED.
Go! The light is GREEN.
Caution! The light is YELLOW.
Program exited with code: 0
Execution info
Cycles: 155
Instrs. retired: 98
CPI: 1.58
IPC: 0.632
Clock rate: 10.10 Hz

Finite State Machine for Vending Machine



Vending Machine Top Level Controller Diagram



Flow chart of Vending Machine

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

FSM for Coin vending machine

```
#include <stdio.h>
typedef enum {
   S0, S1, S2, S3, S4, S5
} State;
State current_state = S0;
void insert_coin(int coin) {
   switch (current_state) {
      case S0:
         if (coin == 1) current_state = S1;
    else if (coin == 2) current_state = S2;
         break;
      case S1:
         if (coin == 1) current_state = S2;
   else if (coin == 2) current_state = S3;
         break;
      case S2:
         if (coin == 1) current_state = S3;
 else if (coin == 2) current_state = S4;
         break;
      case S3:
         if (coin == 1) current_state = S4;
 else if (coin == 2) current_state = S5;
         break;
      case S4:
         if (coin == 1 || coin == 2) current_state = S5;
         break;
         default:
         break;
   }
   if (current_state == S5) {
      printf("Total ₹5 collected. Product dispensed.\n");
      current_state = S0; // Reset
   } else {
      printf("Current total: ₹%d\n", current_state);
   } }
int main() {
   printf("Simulating coin insertions...\n");
   // Hardcoded sequence: ₹1, ₹2, ₹1, ₹1 (total ₹5)
   int coins[ ] = {1, 2, 1, 1};
   int n = sizeof(coins) / sizeof(coins[0]);
   for (int i = 0; i < n; ++i) {
      printf("Inserted: ₹%d\n", coins[i]);
      insert_coin(coins[i]);
   }
   return 0;
}
```

ngstromers Engineering Solutions Private Limited
Incubatee of C-CIR, CIT

Simulation using spike

```
copi-003@copi003-OptiPlex-7040:~/Music/spike/vend$ spike pk -s a.out
Simulating coin insertions...
Inserted: ₹1
Current total: ₹1
Inserted: ₹2
Current total: ₹3
Inserted: ₹1
Current total: ₹4
Inserted: ₹1
Total ₹5 collected. Product dispensed.
2200 ticks
105041 cycles
105041 instructions
1.00 CPI
```

Assembly Code generation using following command

riscv64-unknown-elf-gcc –S main.c

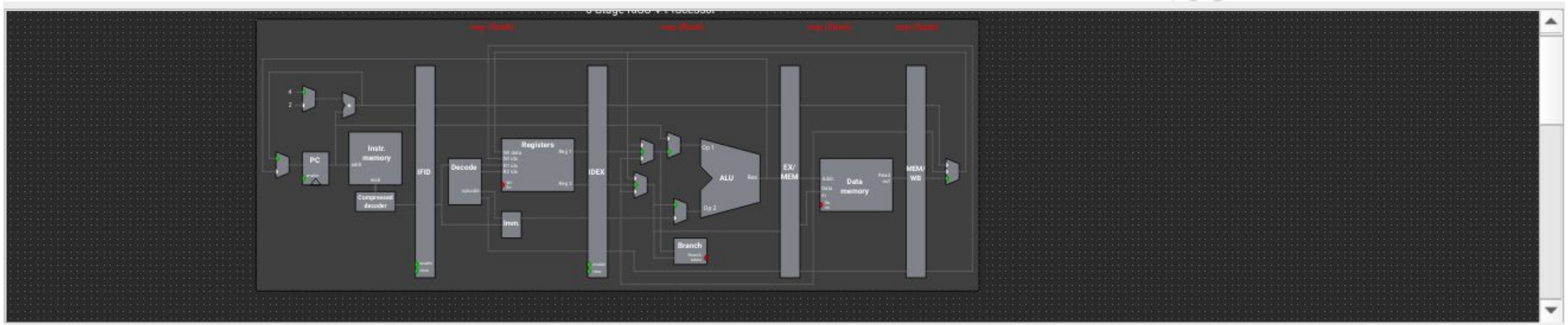
Limited
ngstromers Engi
Incubatee

FSM RISC-V Assembly Coin vending machine

```
.data
promptWelcome:  .asciz "Welcome to the Coin Vending
Machine\n"
promptInsert:   .asciz "Insert ₹1 or ₹2 coins. Product costs ₹5.\n"
msgDispense:    .asciz "Total ₹5 collected. Product dispensed.\n"
msgTotal:       .asciz "Current total: ₹"
current_state:  .word 0       # Initial state = S0
coins:          .word 1, 2, 1, 1, 2, 2   # Hardcoded coin inputs
num_coins:      .word 6
.text
.globl main
main:
    # Print welcome
    la a0, promptWelcome
    li a7, 4
    ecall
    la a0, promptInsert
    li a7, 4
    ecall
 # Load base address of coins array
    la t0, coins
    li t1, 0          # index = 0
    la t6, num_coins
    lw t6, 0(t6)      # number of coins
loop:
    bge t1, t6, end
    slli t2, t1, 2    # offset = index * 4
    add t3, t0, t2
    lw t4, 0(t3)      # t4 = coin value
    # Load current_state
    la t5, current_state
    lw t2, 0(t5)
    # current_state += coin
    add t2, t2, t4
    sw t2, 0(t5)
    # Check if current_state == 5
    li t3, 5
    beq t2, t3, dispense
    # Else, print current total
    la a0, msgTotal
    li a7, 4
    ecall
    mv a0, t2         # current total
    li a7, 1
    ecall
    li a0, 10         # newline
    li a7, 11
    ecall
    addi t1, t1, 1    # index++
    j loop
dispense:
    la a0, msgDispense
    li a7, 4
    ecall
    # Reset state to 0
    li t2, 0
    la t5, current_state
    sw t2, 0(t5)
    addi t1, t1, 1    # move to next coin
    j loop
end:
    li a0, 10     # Exit
    li a7, 93
    ecall
```

ngstromers Engineering Solutions Private Limited
Incubated @ CCIR, CIT

RIPES RISC-V Simulator



Console | Memory

Console

```
Welcome to the Coin Vending Machine
Insert ₹1 or ₹2 coins. Product costs ₹5.
Current total: ₹1
Current total: ₹3
Current total: ₹4
Total ₹5 collected. Product dispensed.
Current total: ₹2
Current total: ₹4
```

Execution info

Cycles:	219
Instrs. retired:	154
CPI:	1.42
IPC:	0.703
Clock rate:	10.75 Hz

FSM code for Binary Sequence detector

```
#include <stdio.h>
#include <string.h>
typedef enum {
    S0, S1, S2, S3
} State;
void detect_sequence(const char *input) {
    State current_state = S0;
    int i = 0;
    printf("Input sequence: %s\n", input);
    while (input[i] != '\0') {
        char bit = input[i];
        switch (current_state) {
            case S0:
                if (bit == '1') current_state = S1;
                else current_state = S0;
                break
case S1:
                if (bit == '0') current_state = S2;
                else current_state = S1;
                break
case S2:
                if (bit == '1') current_state = S3;
                else current_state = S0;
                break;
            case S3:
                if (bit == '1') {
        printf("Sequence '1011' detected at position %d\n", i - 3);
                    current_state = S1;  // Overlapping allowed
                } else {
                    current_state = S2;
                }
                break;
        }

        i++;
    }
}
int main() {
    char input[100];
    printf("Enter a binary string (e.g., 1101011011): ");
    scanf("%s", input);
    detect_sequence(input);
    return 0;
}
```

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

Result

Simulation using spike

```
copi-003@copi003-OptiPlex-7040:~/Music/spike$ riscv64-unknown-elf-gcc main.c
copi-003@copi003-OptiPlex-7040:~/Music/spike$ spike pk -s a.out
Input sequence: 1101011011
Sequence '1011' detected at position 3
Sequence '1011' detected at position 6
1650 ticks
71925 cycles
71925 instructions
1.00 CPI
```

Assembly Code generation using following command

riscv64-unknown-elf-gcc –S main.c

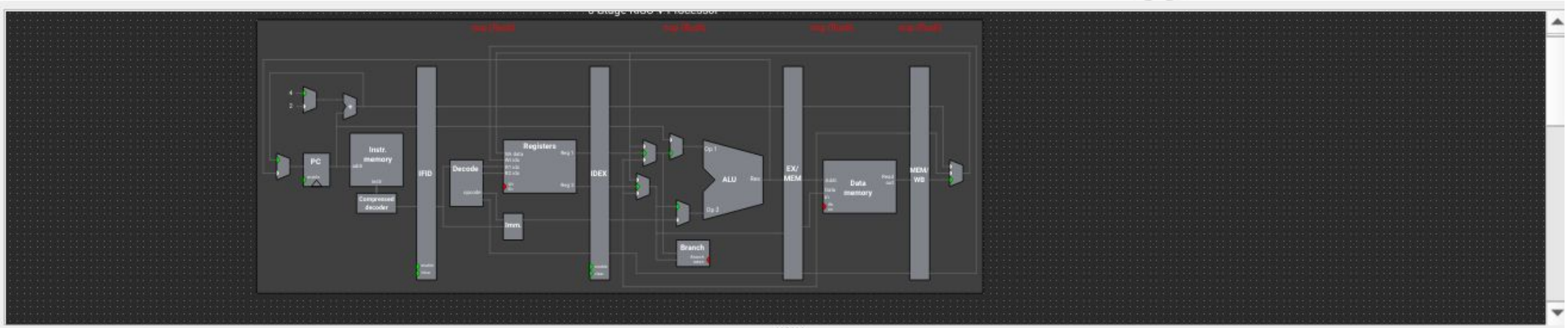
ANGSTROMERS
CCCIR CAMBRIAN CONSULTANCY CENTER AND INDUSTRIAL RESEARCH
CAMBRIDGE INSTITUTE OF TECHNOLOGY An Autonomous Institution
Chips to Startup Programme

FSM RISC-V Assembly

```
.data
input:      .asciz "1101011011"
prompt1:    .asciz "Input sequence: "
found:      .asciz "Sequence '1011' detected at position "
newline:    .asciz "\n"
.text
.globl main
main:
    # Print "Input sequence: "
    la a0, prompt1
    li a7, 4
    ecall
    # Print the input string
    la a0, input
    li a7, 4
    ecall
    li a0, 10     # Newline
    li a7, 11
    ecall
    # Initialize pointers and state
    la t0, input    # t0 = pointer to current char
    li t1, 0        # t1 = index
    li t2, 0        # t2 = current_state (S0)
loop:
    lb t3, 0(t0)     # Load current char
    beq t3, zero, end  # If null terminator, end
 li t4, 49        # ASCII '1'
 li t5, 48        # ASCII '0'
beq t2, zero, state_S0
    li a3, 1
    beq t2, a3, state_S1
    li a3, 2
    beq t2, a3, state_S2
    li a3, 3
    beq t2, a3, state_S3
    j next
state_S0:
    beq t3, t4, set_S1
    j set_S0
state_S1:
    beq t3, t5, set_S2
    j set_S1
state_S2:
    beq t3, t4, set_S3
    j set_S0
state_S3:
    beq t3, t4, match_found
    j set_S2
set_S0:
    li t2, 0
    j next
set_S1:
    li t2, 1
j next
set_S2:
    li t2, 2
    j next
set_S3:
    li t2, 3
    j next
match_found:
la a0, found
    li a7, 4
    ecall
    li a3, 3
    sub a0, t1, a3
    li a7, 1
    ecall
    # Newline
    li a0, 10
    li a7, 11
    ecall
    li t2, 1
    j next
next:
    addi t1, t1, 1
    addi t0, t0, 1
    j loop
end:
    li a7, 93     # Exit
    li a0, 0
    ecall
```

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT

RIPES RISC-V Simulator



Console | Memory

Console

```
Input sequence: 1101011011
Sequence '1011' detected at position 3
Sequence '1011' detected at position 6

Program exited with code: 0
```

Execution info

Cycles:	307
Instrs. retired:	190
CPI:	1.62
IPC:	0.619
Clock rate:	9.26 Hz

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

Thank you

ngstromers Engineering Solutions Private Limited
Incubatee of CCCIR, CIT