EXPERIMENT 7

EE22B129

AIM=>
 to implement arithmetic and logical manipulation programs using the Atmel Atmega8 microcontroller in assembly program emu- lation.

PROBLEM STATEMENTS

1) Common 8-bit Mathematical Operations

```
.CSEG
.ORG 0
.EQU NUM1 = 0x02 .EQU NUM2 = 0XFF
LDI R16, NUM1
LDI R17, NUM2
MOV R18 , R16
ADD R18,R17; R18 contains Sum MUL R16, R17
LDS R19,0x00
LDS R20, 0x01; R19, R20 is product IJMP
```

2) Implementation of Division on AVR

```
.CSEG
.ORG 0
.EQU NUM1 =0x65
 .EQU NUM2 =0x05
LDI R30, NUM1
 LDI R31, NUM2
 STS 0x0E, R30
 STS 0 x10 , R31
 LDS R18 , 0 x0E
 LDS R19,0x10
LDI R20, 0x00
LOOP :
CPR18, R19
BRLO FINISH;  Carry clear  (If R17>R16) that is, divisor>
Dividend  (division  over)
SUB R18 , R19
INC R20 RJMP LOOP
 FINISH:
STS 0xF0, R20; Quotient
 STS 0xFF,R18; Remainder
```

3)Parity Detection=>

```
.CSEG
.ORG 0
LDS R16,0xFF
LDI R17,0x00;stores the parity
LDI R18,0x01; to extrat each bit
LDI R19,0x00; Extracted bit
LDI R20,0x08; looping variable
LOOP: MOV R19,R18
AND R19,R16
EOR R17 , R19
LSR R16
LSL R18
DEC R20
BRNE LOOP
STS 0xFF,R17 ; store parity bit at 0xFF
```

4) Largest and Smallest of a Number Set=>

```
.include "m8def.inc"
LDI R25, 0xff
LDI R20, 0x00
LDI ZL,LOW(2 * Words)
 LDI ZH,HIGH(2 * Words)
Number:
CP R17 , R20
 BRSH Update
 Loop :
LPM R17, Z+1
 CP R17, R25
 BRNE Number LPM R18, Z
 CP R18 , R25
 BRNE Number
 RJMP EXIT
Update :
MOV R20, R17 RJMP Loop
EXIT: RJMP EXIT
; add numbers here
Words: .db 0x01, 0xf1, 0x05, 0xf0, 0xff, 0x12
```

5)Fibonacci Sequence

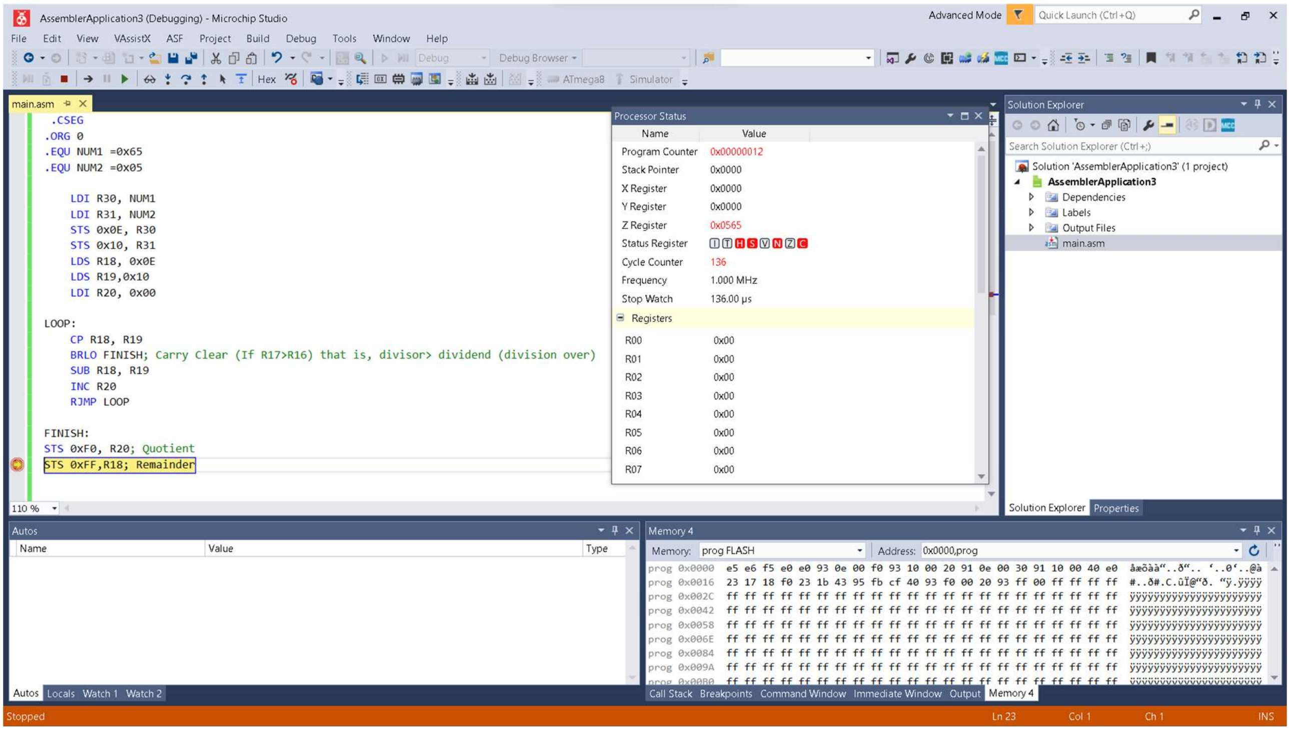
```
.include "m8def.inc"
LDI R25, 0xff
LDI R20, 0x00
LDI ZL,LOW(2 * Words) LDI ZH,HIGH(2 * Words)
Number:
CP R17 , R20
BRSH Update
 Loop :
LPM R17, Z+1
 CP R17, R25
 BRNE Number LPM R18, Z
 CP R18 , R25
 BRNE Number
 RJMP EXIT
Update :
MOV R20, R17 RJMP Loop
EXIT: RJMP EXIT
; add numbers here
Words: .db 0x01, 0xf1, 0x05, 0xf0, 0xff, 0x12
```

Results=>

Problem 1→

(3) WhatsApp

Problem 2→


AssemblerApplication3 (Debugging) - Microchip Studio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Advanced Mode
Quick Launch (Ctrl+Q)
Debug Browser
ATmega8
Simulator
main.asm
Solution Explorer
Solution 'AssemblerApplication3' (1 project)
AssemblerApplication3
Dependencies
Labels
Output Files
main.asm
Properties
Autos
Locals Watch 1 Watch 2
Call Stack Breakpoints Command Window Immediate Window Output Memory 4
Stopped
Ln 23 Col 1 Ch 1 INS

```
 .CSEG
.ORG 0
.EQU NUM1 =0x65
.EQU NUM2 =0x05

    LDI R30, NUM1
    LDI R31, NUM2
    STS 0x0E, R30
    STS 0x10, R31
    LDS R18, 0x0E
    LDS R19,0x10
    LDI R20, 0x00

LOOP:
    CP R18, R19
    BRLO FINISH; Carry Clear (If R17>R16) that is, divisor> dividend (division over)
    SUB R18, R19
    INC R20
    RJMP LOOP

FINISH:
STS 0xF0, R20; Quotient
STS 0xFF,R18; Remainder
```

Processor Status

Name	Value
Program Counter	0x00000012
Stack Pointer	0x0000
X Register	0x0000
Y Register	0x0000
Z Register	0x0565
Status Register	I T H S V N Z C
Cycle Counter	136
Frequency	1.000 MHz
Stop Watch	136.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00

Autos

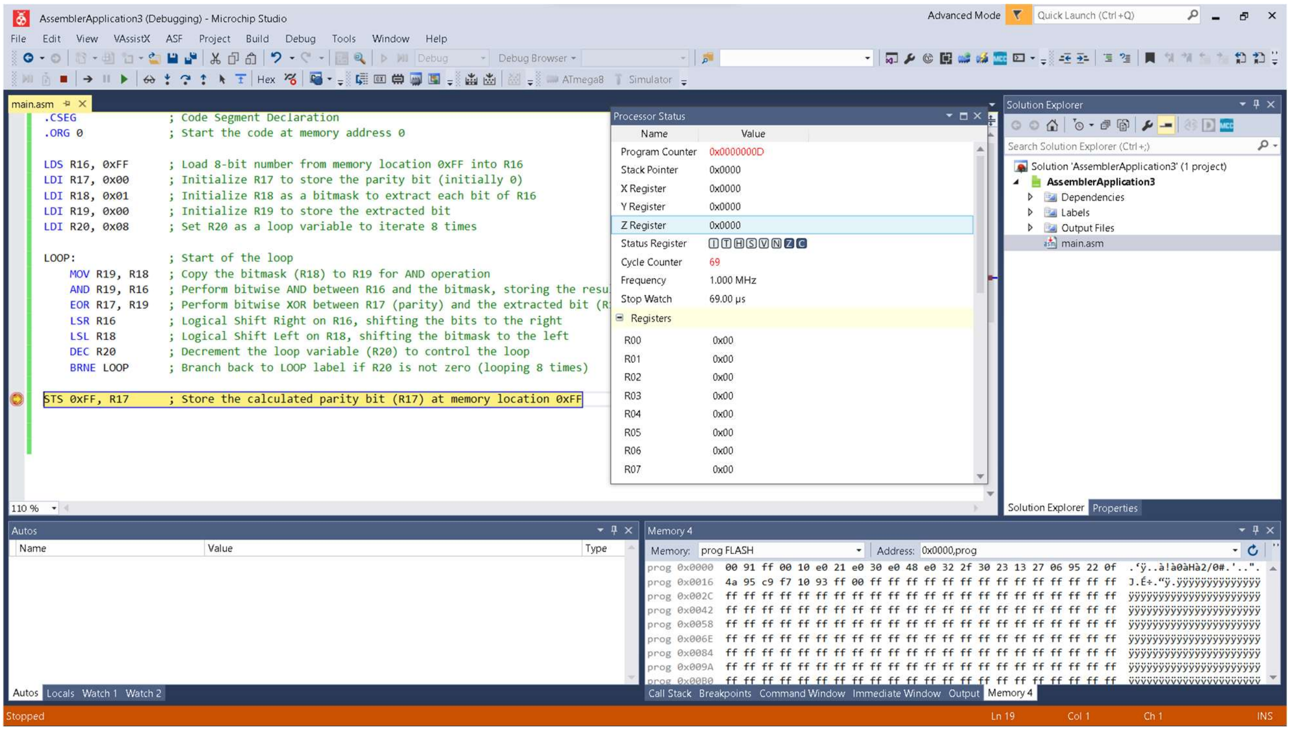
Name	Value	Type

Memory 4

Memory: prog FLASH Address: 0x0000,prog

```
prog 0x0000  e5 e6 f5 e0 e0 93 0e 00 f0 93 10 00 20 91 0e 00 30 91 10 00 40 e0  åæõàà“..ð“.. ‘..0‘..@à
prog 0x0016  23 17 18 f0 23 1b 43 95 fb cf 40 93 f0 00 20 93 ff 00 ff ff ff ff  #..ð#.C.ûÏ@“ð. “ÿ.ÿÿÿÿ
prog 0x002C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0042  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0058  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x006E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0084  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x009A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00B0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Problem 3→


AssemblerApplication3 (Debugging) - Microchip Studio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Advanced Mode
Quick Launch (Ctrl+Q)
Debug Browser
ATmega8
Simulator
main.asm
Solution Explorer
Solution 'AssemblerApplication3' (1 project)
AssemblerApplication3
Dependencies
Labels
Output Files
main.asm
Properties
Autos
Locals Watch 1 Watch 2
Call Stack Breakpoints Command Window Immediate Window Output Memory 4
Stopped
Ln 19 Col 1 Ch 1 INS

```
.CSEG                   ; Code Segment Declaration
.ORG 0                  ; Start the code at memory address 0

LDS R16, 0xFF           ; Load 8-bit number from memory location 0xFF into R16
LDI R17, 0x00           ; Initialize R17 to store the parity bit (initially 0)
LDI R18, 0x01           ; Initialize R18 as a bitmask to extract each bit of R16
LDI R19, 0x00           ; Initialize R19 to store the extracted bit
LDI R20, 0x08           ; Set R20 as a loop variable to iterate 8 times

LOOP:                   ; Start of the loop
    MOV R19, R18        ; Copy the bitmask (R18) to R19 for AND operation
    AND R19, R16        ; Perform bitwise AND between R16 and the bitmask, storing the resu
    EOR R17, R19        ; Perform bitwise XOR between R17 (parity) and the extracted bit (R
    LSR R16             ; Logical Shift Right on R16, shifting the bits to the right
    LSL R18             ; Logical Shift Left on R18, shifting the bitmask to the left
    DEC R20             ; Decrement the loop variable (R20) to control the loop
    BRNE LOOP           ; Branch back to LOOP label if R20 is not zero (looping 8 times)

STS 0xFF, R17           ; Store the calculated parity bit (R17) at memory location 0xFF
```

Processor Status

Name	Value
Program Counter	0x0000000D
Stack Pointer	0x0000
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	69
Frequency	1.000 MHz
Stop Watch	69.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00

Autos

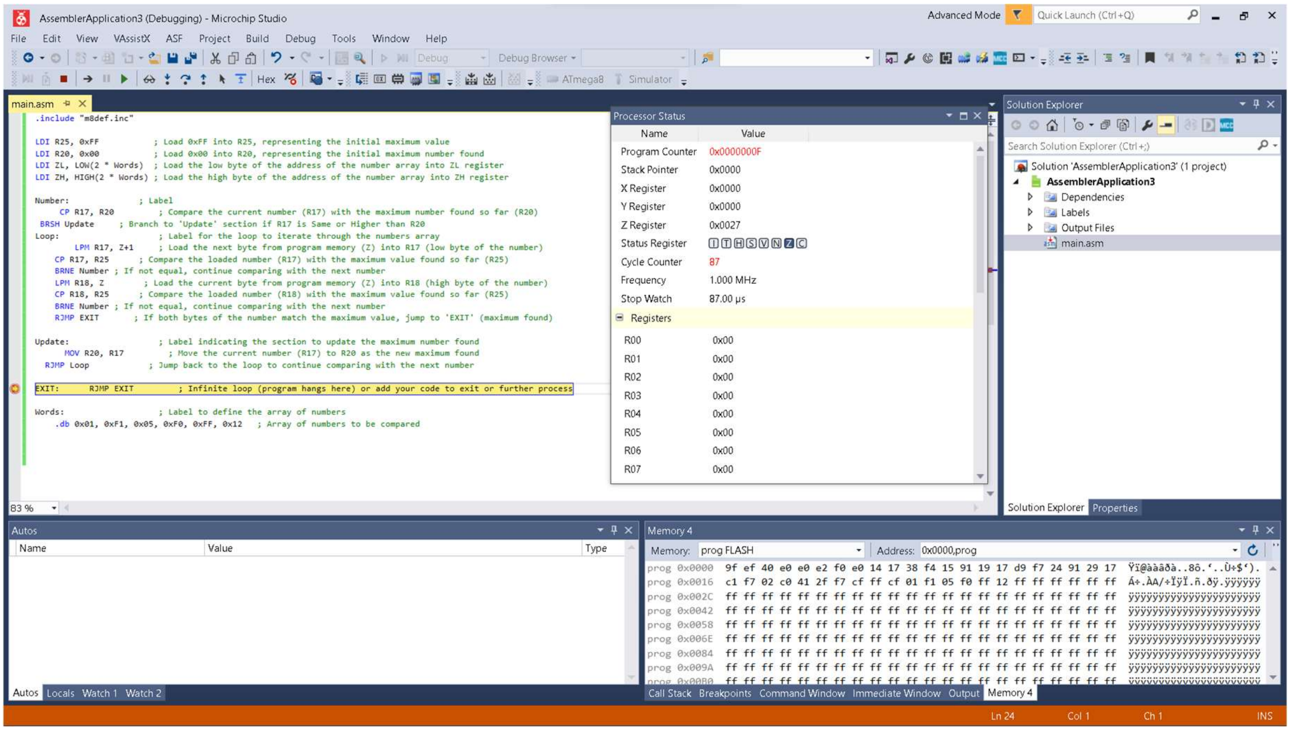
Name	Value	Type

Memory 4

Memory: prog FLASH Address: 0x0000,prog

```
prog 0x0000  00 91 ff 00 10 e0 21 e0 30 e0 48 e0 32 2f 30 23 13 27 06 95 22 0f  .‘ÿ..à!à0àHà2/0#.'..".
prog 0x0016  4a 95 c9 f7 10 93 ff 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff  J.É÷.“ÿ.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x002C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0042  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0058  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x006E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0084  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x009A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00B0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Problem 4→



```
.include "m8def.inc"

LDI R25, 0xFF           ; Load 0xFF into R25, representing the initial maximum value
LDI R20, 0x00           ; Load 0x00 into R20, representing the initial maximum number found
LDI ZL, LOW(2 * Words)  ; Load the low byte of the address of the number array into ZL register
LDI ZH, HIGH(2 * Words) ; Load the high byte of the address of the number array into ZH register

Number:                ; Label
     CP R17, R20          ; Compare the current number (R17) with the maximum number found so far (R20)
 BRSH Update     ; Branch to 'Update' section if R17 is Same or Higher than R20
Loop:                     ; Label for the loop to iterate through the numbers array
        LPM R17, Z+1      ; Load the next byte from program memory (Z) into R17 (low byte of the number)
    CP R17, R25       ; Compare the loaded number (R17) with the maximum value found so far (R25)
    BRNE Number ; If not equal, continue comparing with the next number
    LPM R18, Z         ; Load the current byte from program memory (Z) into R18 (high byte of the number)
    CP R18, R25       ; Compare the loaded number (R18) with the maximum value found so far (R25)
    BRNE Number ; If not equal, continue comparing with the next number
    RJMP EXIT       ; If both bytes of the number match the maximum value, jump to 'EXIT' (maximum found)

Update:                   ; Label indicating the section to update the maximum number found
      MOV R20, R17          ; Move the current number (R17) to R20 as the new maximum found
  RJMP Loop            ; Jump back to the loop to continue comparing with the next number

EXIT:      RJMP EXIT          ; Infinite loop (program hangs here) or add your code to exit or further process

Words:                    ; Label to define the array of numbers
    .db 0x01, 0xF1, 0x05, 0xF0, 0xFF, 0x12   ; Array of numbers to be compared
```

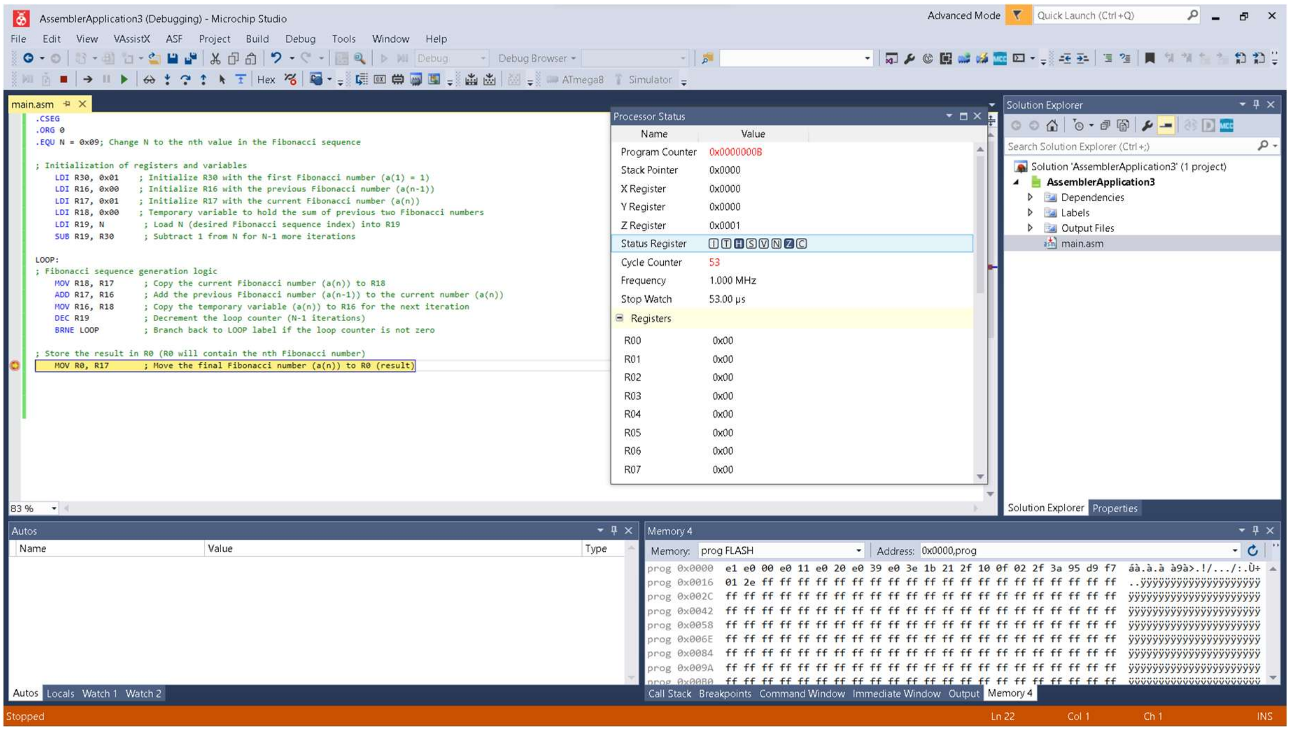
Name	Value
Program Counter	0x0000000F
Stack Pointer	0x0000
X Register	0x0000
Y Register	0x0000
Z Register	0x0027
Status Register	I T H S V N Z C
Cycle Counter	87
Frequency	1.000 MHz
Stop Watch	87.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00

Memory: prog FLASH   Address: 0x0000,prog

```
prog 0x0000  9f ef 40 e0 e0 e2 f0 e0 14 17 38 f4 15 91 19 17 d9 f7 24 91 29 17  Ÿï@àààðà..8ô.'..Ù÷$').
prog 0x0016  c1 f7 02 c0 41 2f f7 cf ff cf 01 f1 05 f0 ff 12 ff ff ff ff ff ff  Á÷.ÀA/÷ÏÿÏ.ñ.ðÿ.ÿÿÿÿÿÿ
prog 0x002C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0042  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0058  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x006E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0084  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x009A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00B0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

AssemblerApplication3 (Debugging) - Microchip Studio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Solution Explorer
Solution 'AssemblerApplication3' (1 project)
AssemblerApplication3
Dependencies
Labels
Output Files
main.asm
Autos Locals Watch 1 Watch 2
Call Stack Breakpoints Command Window Immediate Window Output Memory 4
Ln 24 Col 1 Ch 1 INS

Problem 5→



```
.CSEG
.ORG 0
.EQU N = 0x09; Change N to the nth value in the Fibonacci sequence

; Initialization of registers and variables
    LDI R30, 0x01    ; Initialize R30 with the first Fibonacci number (a(1) = 1)
    LDI R16, 0x00    ; Initialize R16 with the previous Fibonacci number (a(n-1))
    LDI R17, 0x01    ; Initialize R17 with the current Fibonacci number (a(n))
    LDI R18, 0x00    ; Temporary variable to hold the sum of previous two Fibonacci numbers
    LDI R19, N        ; Load N (desired Fibonacci sequence index) into R19
    SUB R19, R30      ; Subtract 1 from N for N-1 more iterations

LOOP:
; Fibonacci sequence generation logic
    MOV R18, R17      ; Copy the current Fibonacci number (a(n)) to R18
    ADD R17, R16      ; Add the previous Fibonacci number (a(n-1)) to the current number (a(n))
    MOV R16, R18      ; Copy the temporary variable (a(n)) to R16 for the next iteration
    DEC R19           ; Decrement the loop counter (N-1 iterations)
    BRNE LOOP         ; Branch back to LOOP label if the loop counter is not zero

; Store the result in R0 (R0 will contain the nth Fibonacci number)
    MOV R0, R17       ; Move the final Fibonacci number (a(n)) to R0 (result)
```

Name	Value
Program Counter	0x0000000B
Stack Pointer	0x0000
X Register	0x0000
Y Register	0x0000
Z Register	0x0001
Status Register	I T H S V N Z C
Cycle Counter	53
Frequency	1.000 MHz
Stop Watch	53.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00

Memory: prog FLASH   Address: 0x0000,prog

```
prog 0x0000  e1 e0 00 e0 11 e0 20 e0 39 e0 3e 1b 21 2f 10 0f 02 2f 3a 95 d9 f7  áà.à.à à9à>.!/.../:.Ù÷
prog 0x0016  01 2e ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ..ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x002C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0042  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0058  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x006E  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0084  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x009A  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00B0  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

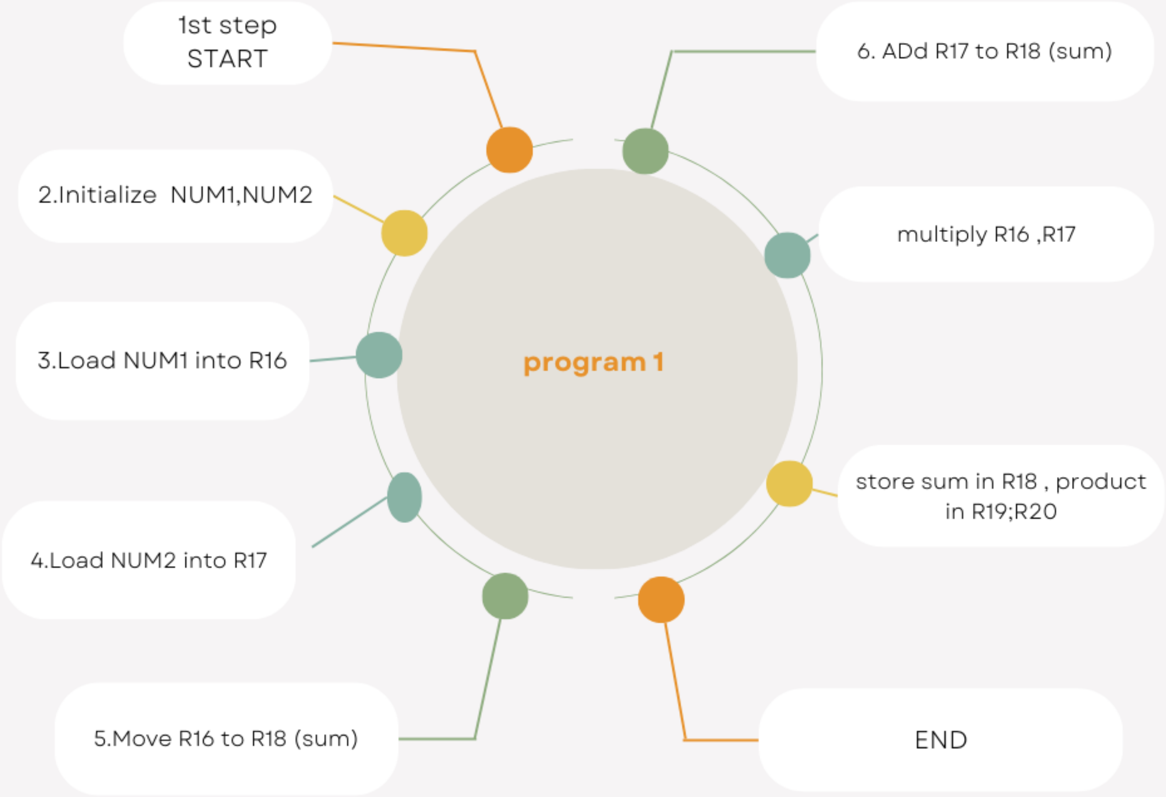
AssemblerApplication3 (Debugging) - Microchip Studio
File Edit View VAssistX ASF Project Build Debug Tools Window Help
Solution Explorer
Solution 'AssemblerApplication3' (1 project)
AssemblerApplication3
Dependencies
Labels
Output Files
main.asm
Autos Locals Watch 1 Watch 2
Call Stack Breakpoints Command Window Immediate Window Output Memory 4
Stopped
Ln 22 Col 1 Ch 1 INS

4) Conclusion

The ATMEga -8 can help in solving complex functions .

5. FLOW charts=>

ARITHAMETIC OPERATIONS



program 1

1st step
START

2.Initialize NUM1,NUM2

3.Load NUM1 into R16

4.Load NUM2 into R17

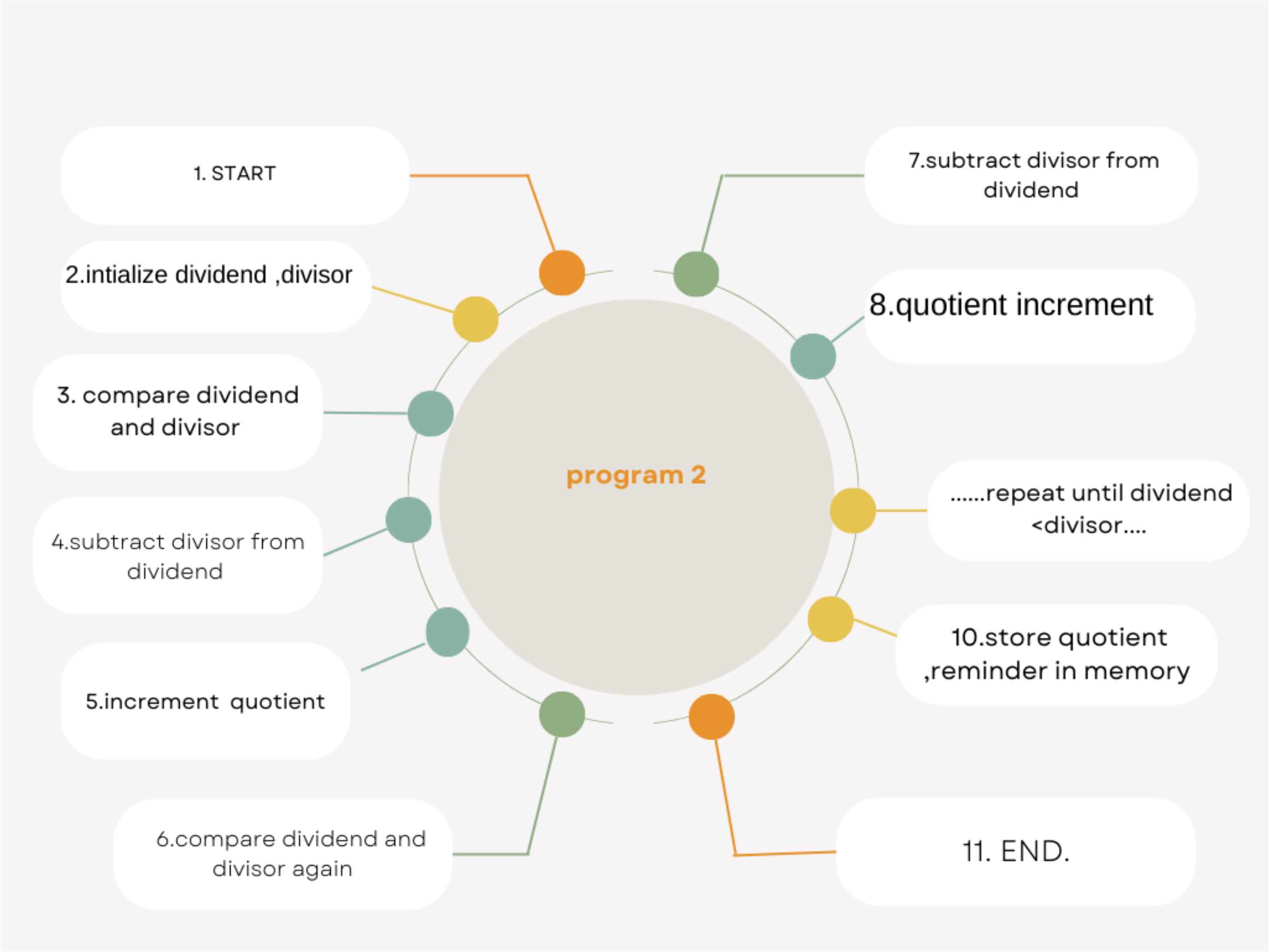
5.Move R16 to R18 (sum)

6. ADd R17 to R18 (sum)

multiply R16 ,R17

store sum in R18 , product
in R19;R20

END



program 2

1. START

2.intialize dividend ,divisor

3. compare dividend and divisor

4.subtract divisor from dividend

5.increment quotient

6.compare dividend and divisor again

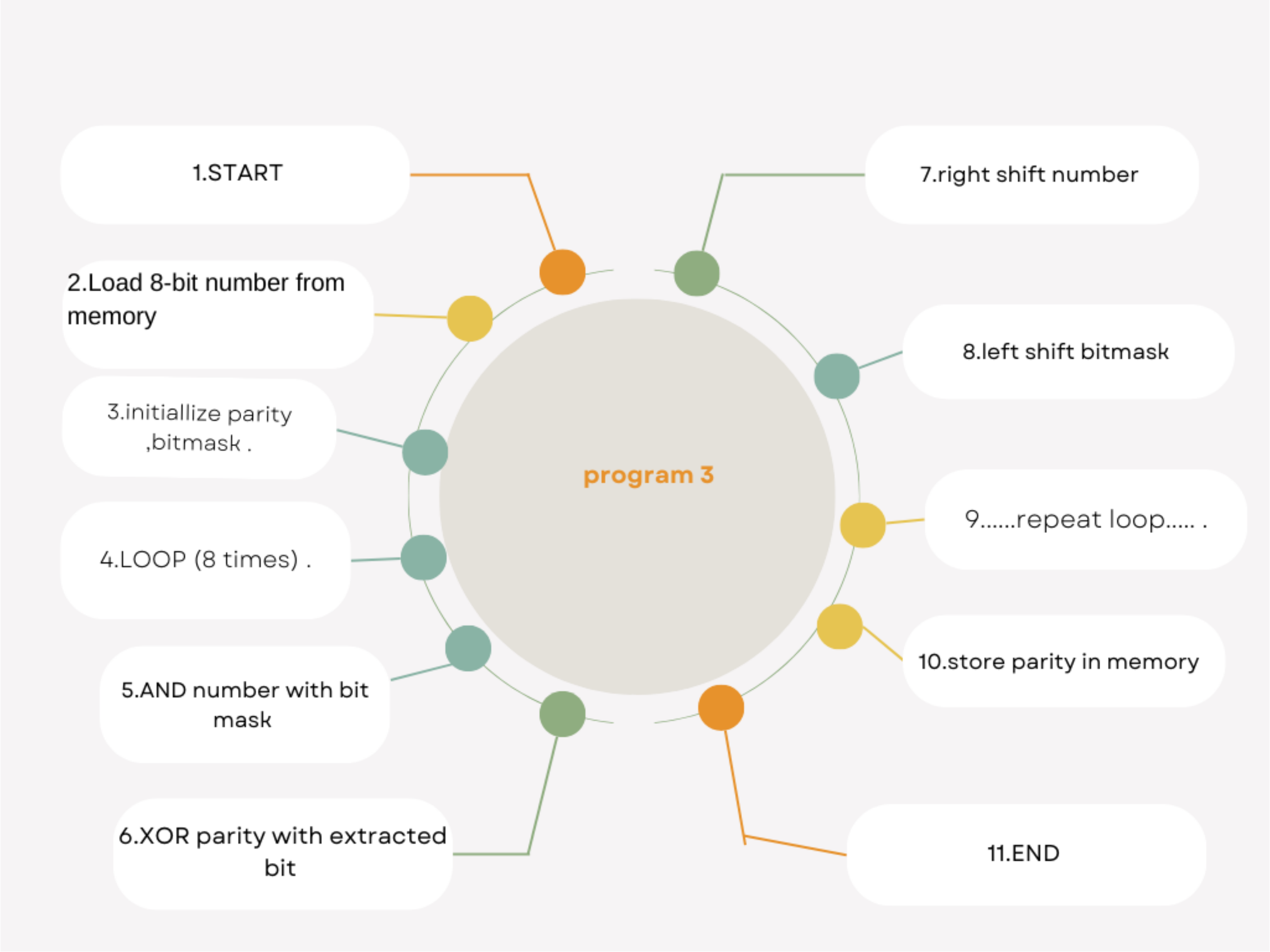
7.subtract divisor from dividend

8.quotient increment

......repeat until dividend <divisor....

10.store quotient ,reminder in memory

11. END.



program 3

1.START

2.Load 8-bit number from memory

3.initiallize parity ,bitmask .

4.LOOP (8 times) .

5.AND number with bit mask

6.XOR parity with extracted bit

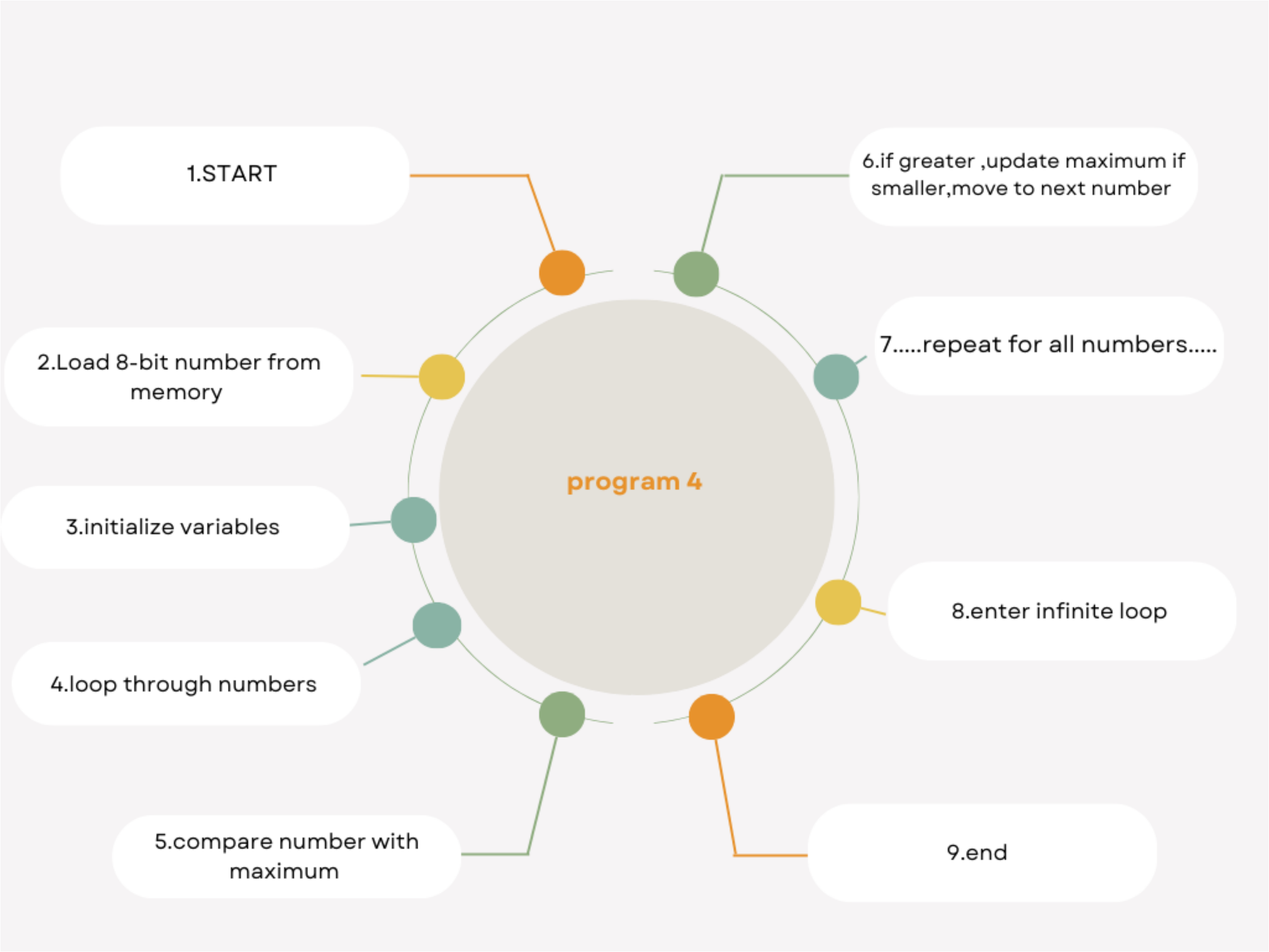
7.right shift number

8.left shift bitmask

9......repeat loop..... .

10.store parity in memory

11.END



program 4

1.START

2.Load 8-bit number from memory

3.initialize variables

4.loop through numbers

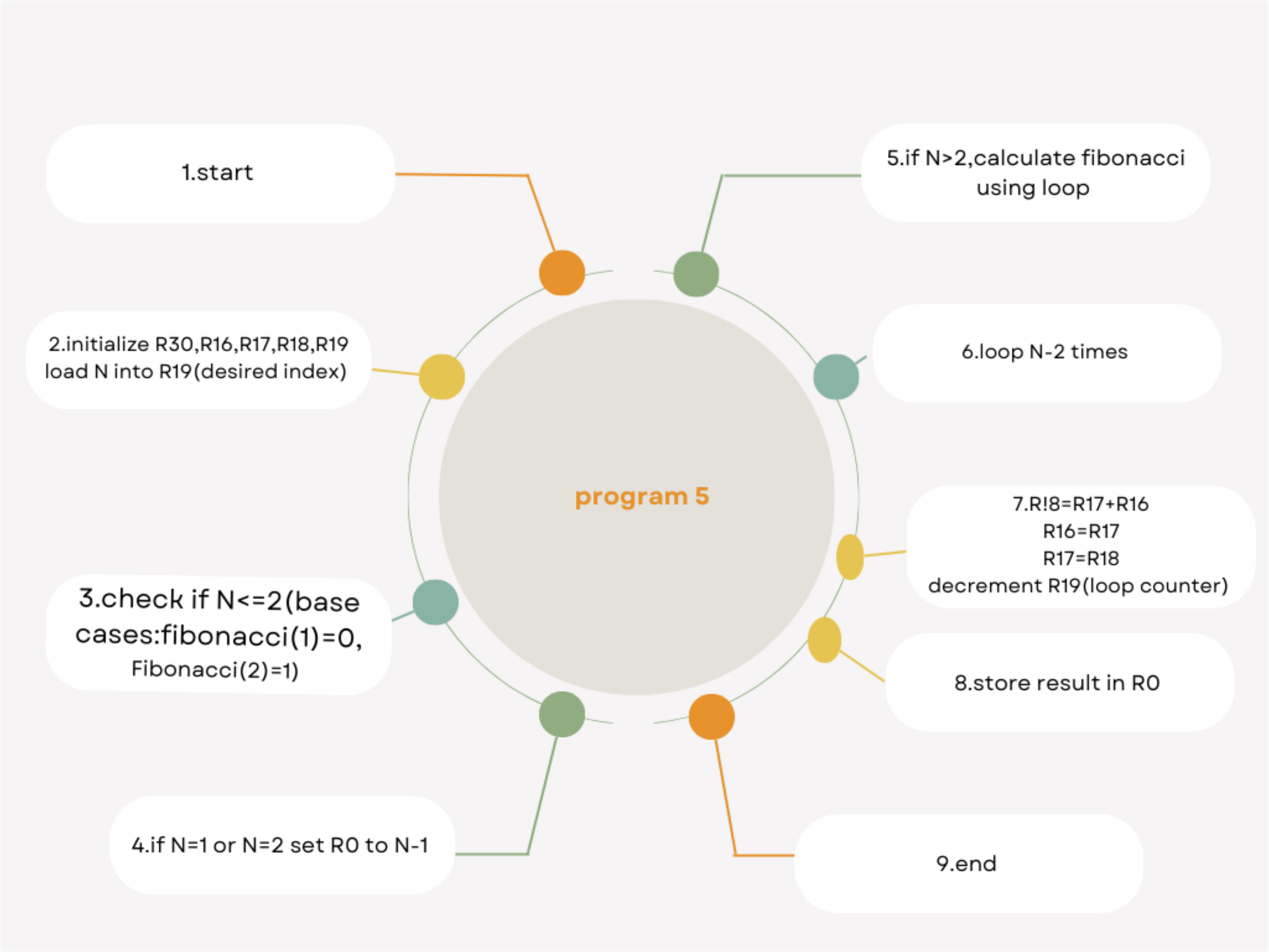
5.compare number with maximum

6.if greater ,update maximum if smaller,move to next number

7.....repeat for all numbers.....

8.enter infinite loop

9.end



program 5

1.start

2.initialize R30,R16,R17,R18,R19
load N into R19(desired index)

3.check if N<=2(base cases:fibonacci(1)=0, Fibonacci(2)=1)

4.if N=1 or N=2 set R0 to N-1

5.if N>2,calculate fibonacci using loop

6.loop N-2 times

7.R!8=R17+R16
R16=R17
R17=R18
decrement R19(loop counter)

8.store result in R0

9.end