**FLAGS IN PSW REGISTER**

ORG 0000H ; Start of code

; ----- Demonstrate Carry (CY) -----

MOV A, #0xF0 ; A = 240 (11110000)

ADD A, #0x20 ; A = A + 32 = 272 (0x110), CY = 1

; ----- Demonstrate Auxiliary Carry (AC) -----

MOV A, #0x0F ; A = 00001111

ADD A, #0x01 ; A = A + 1 = 0x10, AC = 1 (carry from bit 3 to bit 4)

; ----- Demonstrate Overflow (OV) -----

MOV A, #0x7F ; A = 01111111 (127)

ADD A, #0x01 ; A = 128 (10000000), OV = 1 (signed overflow)

; ----- Demonstrate Parity (P) -----

MOV A, #0x55 ; A = 01010101 (4 ones) → Even parity → P = 1

SJMP $ ; Infinite loop (stop here)

END

**LOGICAL OPERATIONS**

ORG 0000H ; Start of program

; ----- AND Operation -----

MOV A, #0F0h ; A = 11110000

ANL A, #0AAh ; A = A AND 10101010 → Result: 10100000

; ----- OR Operation -----

MOV A, #0F0h ; A = 11110000

ORL A, #0AAh ; A = A OR 10101010 → Result: 11111010

; ----- XOR Operation -----

MOV A, #0F0h ; A = 11110000

XRL A, #0AAh ; A = A XOR 10101010 → Result: 01011010

; ----- NOT Operation (1's Complement) -----

MOV A, #55h ; A = 01010101

CPL A ; A = ~A → 10101010

; ----- CLEAR BIT Example -----

CLR P1.0 ; Clear bit 0 of Port 1 (set to 0)

; ----- SET BIT Example -----

SETB P1.1 ; Set bit 1 of Port 1 (set to 1)

SJMP $ ; Infinite loop

END

**ARITHMETIC OPERATIONS**

ORG 0000H ; Start of program

; -------- UNSIGNED ADDITION --------

MOV A, #0x50 ; A = 80

ADD A, #0x40 ; A = A + 64 = 0x90 (144), CY = 0

; -------- SIGNED ADDITION --------

; Example: 50 + (-20) = 30

MOV A, #50 ; A = 50

ADD A, #-20 ; A = 50 + (-20) = 30 (1Eh), OV set if overflow

; -------- SUBTRACTION --------

MOV A, #0x40 ; A = 64

CLR C ; Clear carry before SUBB

SUBB A, #0x20 ; A = A - 32 = 32 (0x20)

; -------- 16-BIT ADDITION --------

; Add 0x1234 + 0x1111 → Result in R4:R5

MOV R0, #0x34 ; Low byte of first number

MOV R1, #0x12 ; High byte of first number

MOV R2, #0x11 ; Low byte of second number

MOV R3, #0x11 ; High byte of second number

MOV A, R0 ; A = Low1

ADD A, R2 ; A = A + Low2

MOV R4, A ; Store result low

MOV A, R1 ; A = High1

ADDC A, R3 ; Add with carry

MOV R5, A ; Store result high

; -------- BCD ADDITION --------

MOV A, #25h ; A = BCD 25

ADD A, #37h ; A = A + 37 = 5Ch

DA A ; Adjust to valid BCD → Result = 62h (BCD)

; -------- MULTIPLICATION --------

MOV A, #0x06 ; A = 6

MOV B, #0x04 ; B = 4

MUL AB ; A = Low byte = 24, B = High byte = 0

; -------- DIVISION --------

MOV A, #0x14 ; A = 20

MOV B, #0x05 ; B = 5

DIV AB ; A = Quotient = 4, B = Remainder = 0

; -------- INCREMENT --------

MOV A, #0x0A ; A = 10

INC A ; A = A + 1 = 11

; -------- DECREMENT --------

DEC A ; A = A - 1 = 10

SJMP $ ; Infinite loop

END

**ROTATE AND SWAP**

ORG 0000H

; -------- RL A (Rotate Left) --------

MOV A, #0x96 ; A = 1001 0110

RL A ; A = 0010 1101 (rotated left)

; Result: 0x2D

; -------- RLC A (Rotate Left Through Carry) --------

MOV A, #0x81 ; A = 1000 0001

CLR C ; Clear carry

RLC A ; A = 0000 0011, CY = 1

; Result: A = 0x03, CY = 1

; -------- RR A (Rotate Right) --------

MOV A, #0x96 ; A = 1001 0110

RR A ; A = 0100 1011

; Result: 0x4B

; -------- RRC A (Rotate Right Through Carry) --------

MOV A, #0x01 ; A = 0000 0001

SETB C ; Set Carry = 1

RRC A ; A = 1000 0000, CY = 1

; Result: A = 0x80, CY = 1

; -------- SWAP A (Swap Nibbles) --------

MOV A, #0x3C ; A = 0011 1100

SWAP A ; A = 1100 0011

; Result: 0xC3

SJMP $ ; Infinite loop

END

**50% DUTY CYCLE:**

ORG 0000H ; Set program start address

MAIN:

SETB P1.0 ; Set P1.0 High

ACALL DELAY ; Call delay

CLR P1.0 ; Set P1.0 Low

ACALL DELAY ; Call delay

SJMP MAIN ; Repeat forever

; --- Delay Subroutine (approx. 500ms for 12MHz)

DELAY:

MOV R2, #20

LOOP1: MOV R1, #255

LOOP2: MOV R0, #255

LOOP3: DJNZ R0, LOOP3

DJNZ R1, LOOP2

DJNZ R2, LOOP1

RET

END ; End of program

Open Keil µVision

Create a new project

Select device: AT89C51

Add new Assembly file (e.g., squarewave.a51)

Paste the code above

Build project (F7)

Load HEX into Proteus or microcontroller board for testing

**FACTORIAL OF 6:**

ORG 0000H

MAIN:

MOV R4, #01H ; Low byte = 1

MOV R5, #00H ; High byte = 0

MOV R3, #06H ; Counter = 6

LOOP:

MOV A, R4

MOV B, R3

MUL AB

MOV R4, A ; New low byte

MOV R6, B ; Temp high byte

MOV A, R5

MOV B, R3

MUL AB

ADD A, R6

MOV R5, A ; New high byte

DJNZ R3, LOOP

SJMP $

END

**FIBONACCI:**

ORG 0000H

MAIN:

MOV R0, #30H

MOV A, #00H

MOV @R0, A

INC R0

MOV A, #01H

MOV @R0, A

INC R0

MOV R1, #00H

MOV R2, #01H

MOV R3, #07H

FIB\_LOOP:

MOV A, R1

ADD A, R2

MOV @R0, A

INC R0

MOV R1, R2

MOV R2, A

DJNZ R3, FIB\_LOOP

SJMP $

END

**JUMPS:**  
ORG 0000h//Unconditional- Long Jump

MOV A, #05h

LJMP NEXT

MOV A, #00h

NEXT: MOV B, A

//Unconditional- Short Jump

MOV A, #0Ah

SJMP SKIP

MOV A, #00h

SKIP: MOV B, A

//Unconditional- Absolute Jump

MOV A, #0Fh

AJMP TARGET

MOV A, #00h

TARGET: MOV B, A

//Conditional- Jump if Zero

MOV A, #00h

JZ ZERO\_LABEL

MOV A, #0FFh

ZERO\_LABEL: MOV B, A

//Conditional- Jump if Not Zero

MOV A, #0Ah

SJMP NOT\_ZERO

MOV A, #00h

NOT\_ZERO: MOV B, A

//Conditional- Jump if Carry Set

MOV A, #0FFh

ADD A, #01h

JC CARRY\_LABEL

MOV B, A

MOV A, #00h

CARRY\_LABEL: MOV B, #00h

//Conditional- Jump if No Carry

MOV A, #0Ah

ADD A, #05h

JNC NO\_CARRY

MOV B, A

MOV A, #00h

NO\_CARRY: MOV B, #00h

//Conditional- Jump if Bit is Set

MOV A, #0FFh

JB ACC.7, BIT\_SET

MOV B, A

MOV A, #00h

BIT\_SET: MOV A, #0Fh

//Conditional- Jump if Bit is Not Set

MOV A, #0Fh

JNB ACC.7, BIT\_NOT\_SET

MOV B, A

MOV A, #00h

BIT\_NOT\_SET: MOV A, #0F0h

//Conditional- Jump if Bit is Set and Clear the Bit

SETB P1.0

JBC P1.0, BIT\_CLEAR

MOV A, #0FFh

BIT\_CLEAR: MOV A, #0Ch

END