PROGRAMMING ARITHMETIC AND LOGIC OPERATIONS

Implementation of Arithmetic and Logic Operations

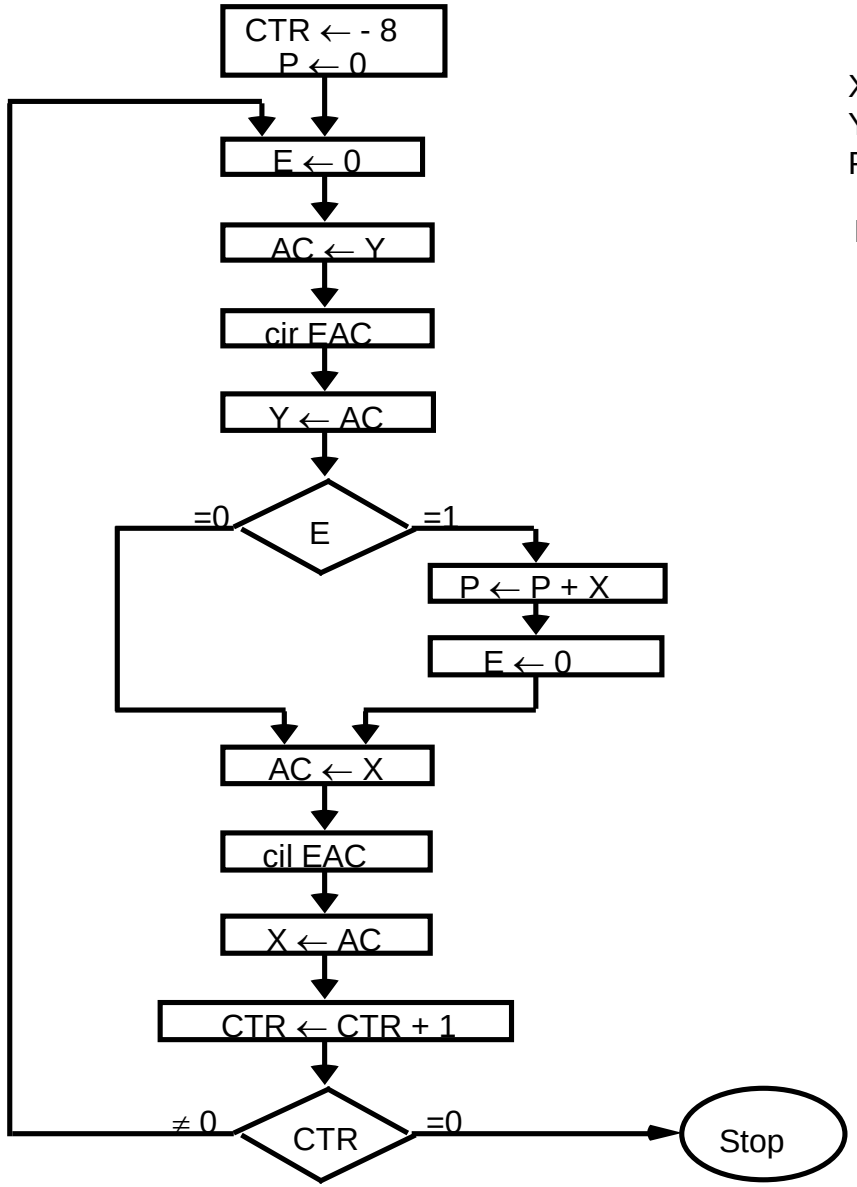
- Software Implementation
  - Implementation of an operation with a program using machine instruction set
  - Usually when the operation is not included in the instruction set

- Hardware Implementation
  - Implementation of an operation in a computer with one machine instruction

Software Implementation example:

* Multiplication
  - For simplicity, unsigned positive numbers
  - 8-bit numbers -> 16-bit product

FLOWCHART OF A PROGRAM - Multiplication -



X holds the multiplicand
Y holds the multiplier
P holds the product

Example with four significant digits

		P
X =	0000 1111	
Y =	0000 1011	0000 0000
	0000 1111	0000 1111
	0001 1110	0010 1101
	0000 0000	0010 1101
	0111 1000	1010 0101
	1010 0101	

ASSEMBLY LANGUAGE PROGRAM - Multiplication -

```
        ORG  100
LOP,    CLE          / Clear E
        LDA  Y       / Load multiplier
        CIR          / Transfer multiplier bit to E
        STA  Y       / Store shifted multiplier
        SZE          / Check if bit is zero
        BUN  ONE     / Bit is one; goto ONE
        BUN  ZRO     / Bit is zero; goto ZRO
ONE,    LDA  X       / Load multiplicand
        ADD  P       / Add to partial product
        STA  P       / Store partial product
        CLE          / Clear E
ZRO,    LDA  X       / Load multiplicand
        CIL          / Shift left
        STA  X       / Store shifted multiplicand
        ISZ  CTR     / Increment counter
        BUN  LOP     / Counter not zero; repeat loop
        HLT          / Counter is zero; halt
CTR,    DEC  -8      / This location serves as a counter
X,      HEX  000F    / Multiplicand stored here
Y,      HEX  000B    / Multiplier stored here
P,      HEX  0       / Product formed here
        END
```

ASSEMBLY LANGUAGE PROGRAM
- Logic and Shift Operations -

- Logic operations

  - BC instructions : AND, CMA, CLA
  - Program for OR operation (A+B) = (A\B\)\

```
LDA   A      / Load 1st operand
CMA          / Complement to get A'
STA   TMP    / Store in a temporary location
LDA   B      / Load 2nd operand B
CMA          / Complement to get B'
AND   TMP    / AND with A' to get A' AND B'
CMA          / Complement again to get A OR B
```

- Shift operations - BC has *Circular Shift* only
  - Logical shift-right operation - Logical shift-left operation

```
CLE
CIR
```

```
CLE
CIL
```

  - Arithmetic right-shift operation

```
CLE     / Clear E to 0
SPA     / Skip if AC is positive
CME     / AC is negative
CIR     / Circulate E and AC
```

SUBROUTINES

Subroutine

- A set of common instructions that can be used in a program many times.
- Subroutine *linkage* : a procedure for branching to a subroutine and returning to the main program

Example : Sub Routine that makes Logical Shift Left 4 times

Loc.			
		ORG 100	/ Main program
100		LDA X	/ Load X
101		BSA SH4	/ Branch to subroutine
102		STA X	/ Store shifted number
103		LDA Y	/ Load Y
104		BSA SH4	/ Branch to subroutine again
105		STA Y	/ Store shifted number
106		HLT	
107	X,	HEX 1234	
108	Y,	HEX 4321	
			/ Subroutine to shift left 4 times
109	SH4,	HEX 0	/ Store return address here
10A		CIL	/ Circulate left once
10B		CIL	
10C		CIL	
10D		CIL	/ Circulate left fourth time
10E		AND MSK	/ Set AC(0-3) to zero
10F		BUN SH4 I	/ Return to main program
110	MSK,	HEX FFF0	/ Mask operand
		END	

SUBROUTINE PARAMETERS AND DATA LINKAGE

Linkage of Parameters and Data between the Main Program and a Subroutine

- via Registers
- via Memory locations
- ….

Example: Subroutine performing *LOGICAL OR operation*; Need two parameters

```
Loc.
200         ORG  200
200         LDA  X      / Load 1st operand into AC
201         BSA  OR     / Branch to subroutine OR
202         HEX  3AF6   / 2nd operand stored here
203         STA  Y      / Subroutine returns here
204         HLT
205   X,    HEX  7B95   / 1st operand stored here
206   Y,    HEX  0      / Result stored here
207   OR,   HEX  0      / Subroutine OR
208         CMA         / Complement 1st operand
209         STA  TMP    / Store in temporary location
20A         LDA  OR I   / Load 2nd operand
20B         CMA         / Complement 2nd operand
20C         AND  TMP    / AND complemented 1st operand
20D         CMA         / Complement again to get OR
20E         ISZ  OR     / Increment return address
20F         BUN  OR I   / Return to main program
210   TMP,  HEX  0      / Temporary storage
            END
```

SUBROUTINE - Moving a Block of Data -

```
                        / Main program
        BSA  MVE        / Branch to subroutine
        HEX  100        / 1st address of source data
        HEX  200        / 1st address of destination data
        DEC  -16        / Number of items to move
        HLT
MVE,    HEX  0          / Subroutine MVE
        LDA  MVE  I     / Bring address of source
        STA  PT1        / Store in 1st pointer
        ISZ  MVE        / Increment return address
        LDA  MVE  I     / Bring address of destination
        STA  PT2        / Store in 2nd pointer
        ISZ  MVE        / Increment return address
        LDA  MVE  I     / Bring number of items
        STA  CTR        / Store in counter
        ISZ  MVE        / Increment return address
LOP,    LDA  PT1  I     / Load source item
        STA  PT2  I     / Store in destination
        ISZ  PT1        / Increment source pointer
        ISZ  PT2        / Increment destination pointer
        ISZ  CTR        / Increment counter
        BUN  LOP        / Repeat 16 times
        BUN  MVE  I     / Return to main program
PT1,    --
PT2,    --
CTR,    --
```

- Fortran subroutine

```
      SUBROUTINE  MVE (SOURCE, DEST, N)
      DIMENSION  SOURCE(N), DEST(N)
      DO  20  I = 1, N
20    DEST(I) = SOURCE(I)
      RETURN
      END
```

INPUT OUTPUT PROGRAM

Program to Input one Character(Byte)

```
CIF,    SKI          / Check input flag
        BUN  CIF     / Flag=0, branch to check again
        INP          / Flag=1, input character
        OUT          / Display to ensure correctness
        STA  CHR     / Store character
        HLT
CHR,    --           / Store character here
```

Program to Output a Character

```
        LDA  CHR     / Load character into AC
COF,    SKO          / Check output flag
        BUN  COF     / Flag=0, branch to check again
        OUT          / Flag=1, output character
        HLT
CHR,    HEX  0057    / Character is "W"
```

CHARACTER MANIPULATION

Subroutine to Input 2 Characters and pack into a word

```
IN2,    --          / Subroutine entry
FST,    SKI
        BUN  FST
        INP         / Input 1st character
        OUT
        BSA  SH4    / Logical Shift left 4 bits
        BSA  SH4    / 4 more bits
SCD,    SKI
        BUN  SCD
        INP         / Input 2nd character
        OUT
        BUN  IN2  I / Return
```