CSEN402: Computer Organization
Machine & Assembly II

Nada Sharaf

Spring Semester 2021

Implementation of Arithmetic and Logic Operations

- Hardware Implementation
  - An operation is implemented through one machine instruction
- **Software Implementation**
  - Long programs in size and execution time

Double Precision Addition

- Double precision numbers are placed in two memory locations: AL and AH

```
    LDA AL    / Load A low
    ADD BL    / Add B low, carry in E
    STA CL    / Store in C low
    CLA       / Clear AC
    CIL       / Circulate to bring carry into AC
    ADD AH    / Add A high and carry
    ADD BH    / Add B high
    STA CH    / Store in C high
    HLT
AL,
AH,
BL,
BH,
CL,
CH,
```

Example

Implement a Basic Computer (BC) assembly program that subtracts two double-precision numbers.

```
    LDA YL
    CMA
    INC
    ADD XL
    STA ZL
    CLA
    CIL
    STA TMP
    LDA YH
    CMA
    ADD XH
    ADD TMP
    STA ZH
    HLT
```


```
XL,     DEC 2505
XH,     DEC 1560
YL,     DEC 480
YH,     DEC 40
ZL,     DEC 0
ZH,     DEC 0
TMP,    DEC 0
        END
```

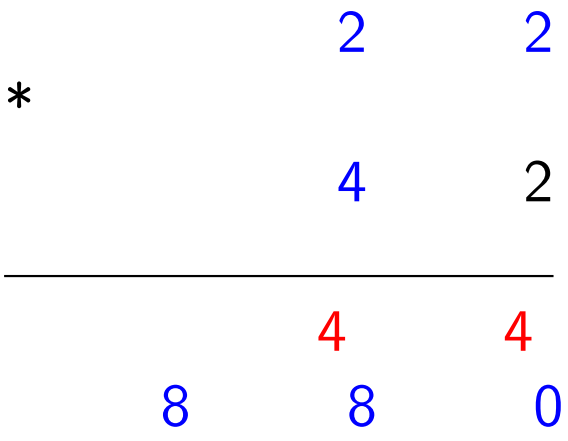
Multiplication

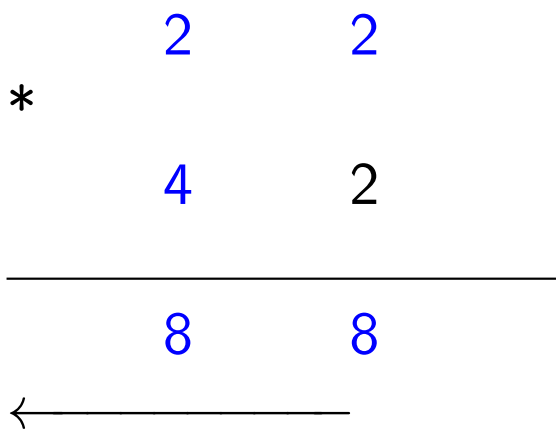
- Ignore sign bits assuming positive numbers only
- Each number is 8 bits
- Output is at most 16 bits (255×255)

Decimal Example

$$
\begin{array}{cccc}
 & 2 & 2 & \\
* & & & \\
 & 4 & 2 & \\
\hline
 & 4 & 4 &
\end{array}
$$

Decimal Example





Decimal Example

$$
\begin{array}{cccc}
 & & 2 & 2 \\
* & & & \\
 & & 4 & 2 \\
\hline
 & & 4 & 4 \\
+ & & & \\
 & 8 & 8 & 0 \\
\hline
 & \mathbf{9} & \mathbf{2} & \mathbf{4}
\end{array}
$$

Decimal Example II

$$
\begin{array}{cccc}
 & & 3 & 4 \\
* & & & \\
 & & 2 & 1 \\
\hline
 & & 3 & 4 \\
+ & & & \\
 & 6 & 8 & 0 \\
\hline
 & \mathbf{7} & \mathbf{1} & \mathbf{4}
\end{array}
$$

Shift-and-Add Multiplication

- Ignore sign bits assuming positive numbers only
- Each number is 8 bits
- Output is at most 16 bits (255×255)
- What if the two numbers were 16 bits? **We would need to use two words in memory**

Binary Example

Let us start with 4 bits' numbers

$$
\begin{array}{cccc}
 & 0 & 0 & 1 & 1 \\
* & & & & \\
 & 0 & 0 & 1 & 0 \\
\hline
 & 0 & 0 & 0 & 0 \\
+ & & & & \\
 & 0 & 1 & 1 & 0 \\
+ & & & & \\
 & 0 & 0 & 0 & 0 \\
+ & & & & \\
 & 0 & 0 & 0 & 0 \\
\hline
 & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0}
\end{array}
$$

Some Remarks

- How many numbers can we add at a time? only 2
- Do we need to keep the rows with zeros only? No, we will remove them for simplicity
- Binary numbers mean that the output of the multiplication is either 0 or the multiplicand as is
  - Afterwards shifting has to be done

Binary Example II

10×20

$$
\begin{array}{cccccccc}
 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
* & & & & & & & & \\
 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
\hline
 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
+ & & & & & & & & \\
 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
 & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{array}
$$

Binary Example III

$$15 \times 11$$

$$\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 & & & * & & & & \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\hline
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 & & & + & & & & \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
 & & & + & & & & \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0
\end{array}$$

Binary Example III: Addition Step

First Step:

```
   1111
00001111
+
00011110
--------
00101101
```

Second Step:

```
1111
00101101
+
01111000
--------
10100101
```

Binary Example III

$$15 \times 11$$

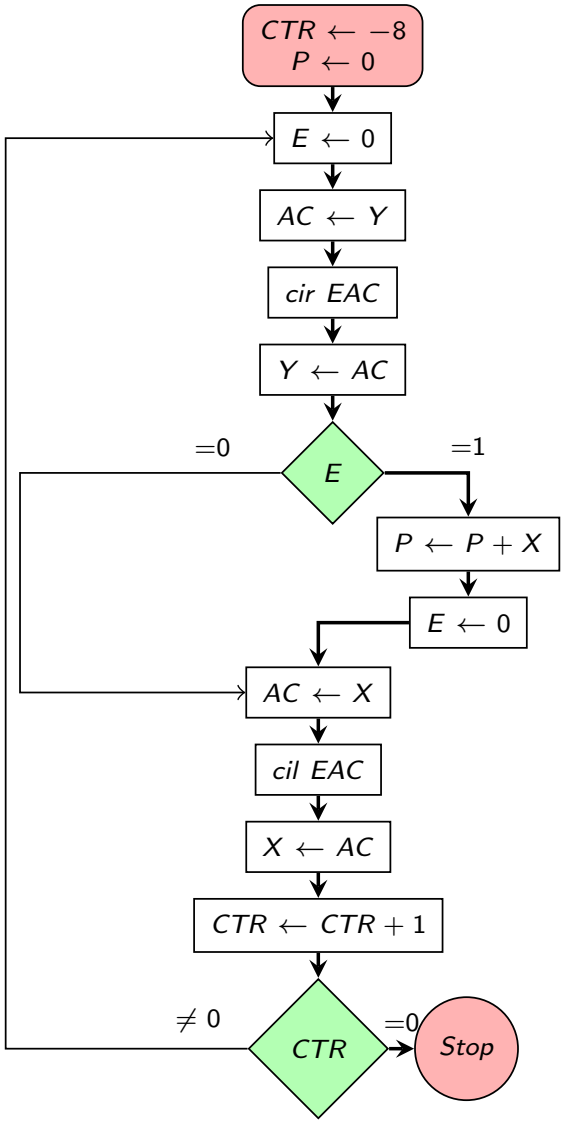
$$
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 & & & * & & & & \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\hline
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 & & & + & & & & \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
 & & & + & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & + & & & & \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
\hline
\mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1}
\end{array}
$$

Multiplication Flowchart

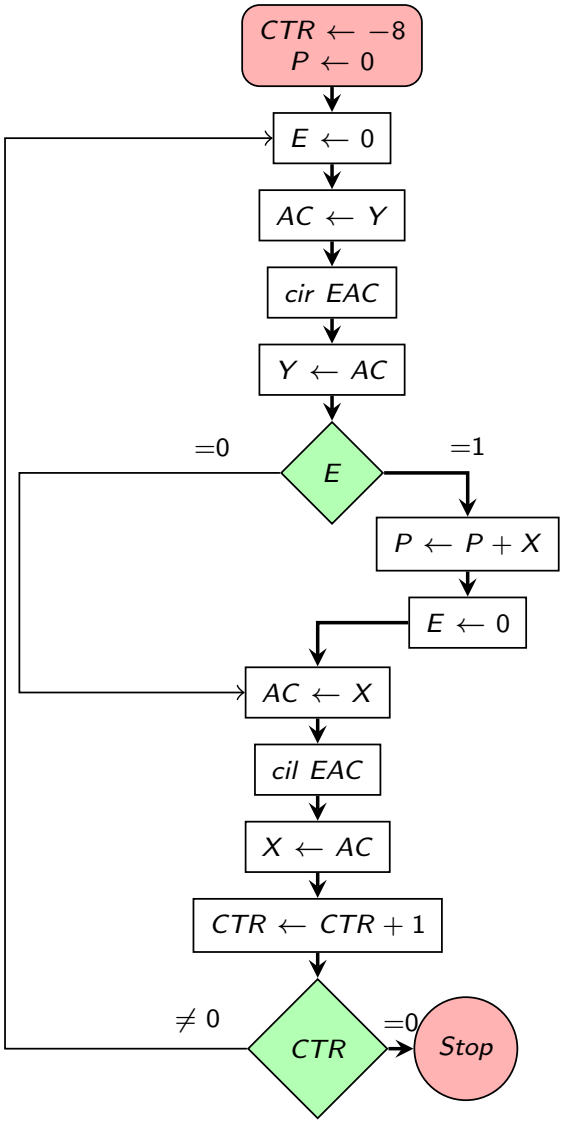
X: 00001111 E: 0 Y: 00001011 P:0

1. E: 1, Y: 00000101
   - Output = 00001111
   - P = P+Output = 00001111
   - X: 00011110
2. E: 1, Y: 00000010
   - Output = 00011110
   - P = P+Output = 00101101
   - X: 00111100
3. E: 0, Y: 00000001
   - Output = 00000000
   - P= 00101101
   - X: 01111000
4. E: 1, Y: 00000000
   - Output = 01111000
   - P= P+X = 10100101
   - X: 11110000
5. E: 0, Y: 0000000
   - Output = 00000000
   - P= 10100101
   - X: 11100000
6. E: 0, Y: 0000000
   - Output = 00000000
   - P= 10100101
   - X: 11000000
7. E: 0, Y: 0000000
   - Output = 00000000
   - P= 10100101
   - X: 10000000
8. E: 0, Y: 0000000
   - Output = 00000000
   - P= 10100101
   - X: 00000000

Multiplication Flowchart

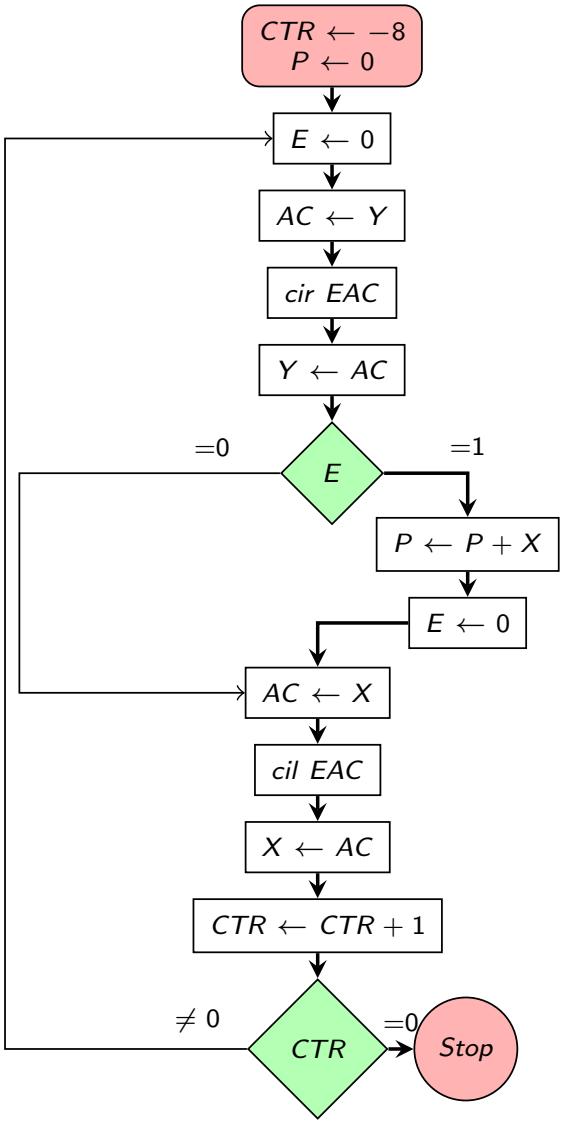


Tracing



- X: 00001111
- Y: 00001011
- E:0 AC:?
- CTR: -8

Assembly Program

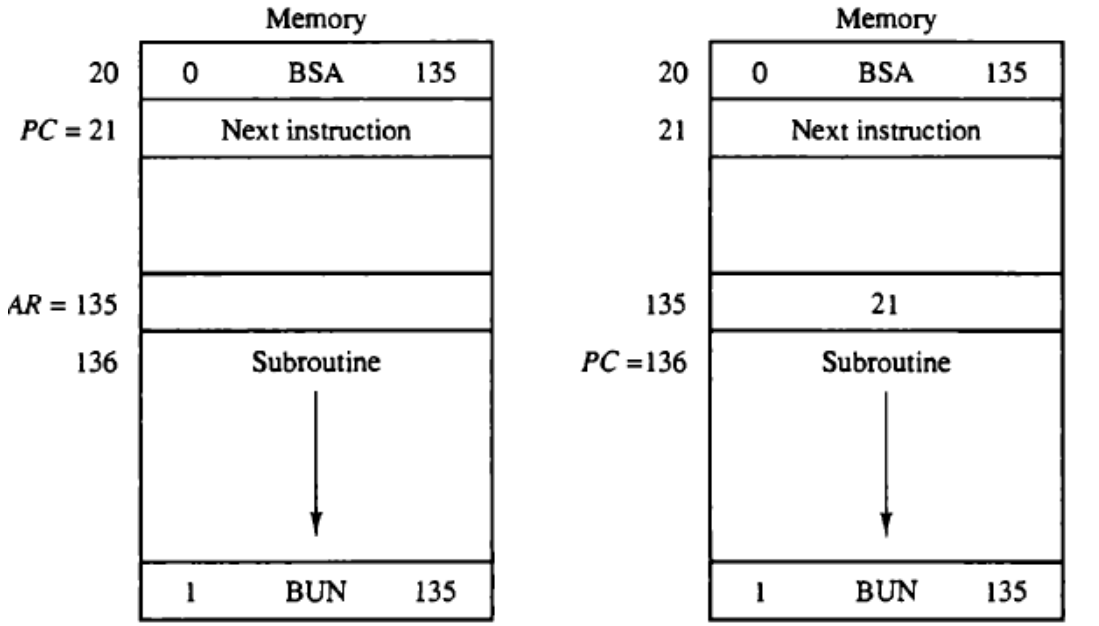



```
        ORG 100
LOP,    CLE         / Clear E
        LDA Y       / Load multiplier
        CIR         / Transfer multiplier bit to E
        STA Y       / Store shifted multiplier
        SZE         / Check if bit is zero
        BUN ONE     / Bit is one; goto ONE
        BUN ZRO     / Bit is 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      / a counter for the loop
X,      HEX 00F     /Multiplicand stored here
Y,      HEX 00B     / Multiplier stored here
P,      HEX 0       / Product formed here
```

BSA: Reminder

$$D_5 T_4 : M[AR] \leftarrow PC, AR \leftarrow AR + 1$$

$$D_5 T_5 : PC \leftarrow AR, SC \leftarrow 0$$



Memory, *PC*, and *AR* at time T_4 | Memory and *PC* after execution

© Text Book

Example: Shifting AC 4 times to the left

```
        LDA X       / Load X
        CIL         /Circulate left once
        CIL
        CIL
        CIL         /Circulate left fourth time
        AND MSK     /Set least significant bits to 0
        HLT
X,      HEX 1234
Y,      HEX 4321
MSK,    HEX FFF0    /Mask Operand
        END
```

Example: Shifting AC 4 times to the left

Location			
		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*
10A		CIL	*/Circulate left once*
10B		CIL	
10C		CIL	
10D		CIL	*/Circulate left fourth time*
10E		AND MSK	*/Set least significant bits to 0*
10F		BUN SH4 I	*/Return to main program*
110	MSK,	HEX FFF0	*/Mask Operand*
111		END	

- What is the input value? AC
- What is the output value? AC
- What if we need more?
  - Processor Registers
    - ★ How many processor registers does the basic computer have ?
  - Using memory

Subroutine Logical OR

- $A + B = (A'B')'$
- Two operands
- One is stored in AC
- The other is stored right after the BSA instruction

Location			
		ORG 200	*/ Main Program*
200		LDA X	*/ Load first operand into AC*
201		BSA OR	*/ Branch to subroutine OR*
202		HEX 3AF6	*/ Second operand is stored here*
203		STA Y	*/ Subroutine returns here*
204		HLT	*/*
205	X,	HEX 7B95	*/First Operand stored here*
206	Y,	HEX 0	*Result stored here*
207	OR,	HEX 0	*Subroutine OR*
208		CMA	*Complement first operand*
209		STA TMP	*Store in temporary location*
20A		LDA OR I	*/Load second operand*
20B		CMA	*Complement second operand*
20C		AND TMP	*And complemented first operand*
20D		CMA	*/omplement 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	

Transferring a Block of Data

```
1 void subroutine(int* src
      , int*dst, int n)
2 {
3     for(int i=0;i<n;i++)
4         dst[i] = src[i];
5 }
```

```
                    / Main Program
      BSA MVE       /Branch to subroutine
      HEX 100       /First address of source data
      HEX 200       /First address of destination data
      DEC -16       /number of items to move
      HLT
MVE,  HEX 0         / Subroutine to move
      LDA MVE I     / Bring address of source
      STA PT1       / Store in first pointer
      ISZ MVE       / Increment return address
      LDA MVE I     / Bring address of destination
      STA PT2       / Store in second 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,  —
```

References

- Chapter 6

Thank you :)