

# Arithmetic Instructions:

---

**; 8-bit addition**

**MOV A,#01EH**

**ADD A,#127 ; A = ?**

**MOV 30H,#0BBH**

**ADD A,30H ; A = ?**

**MOV R0,#30H**

**ADD A,@R0 ; A = ?**

# Arithmetic Instructions:

---

**; 16-bit addition**

**; 13EFH + 16FBH = ?**

**; immediate addition**

**MOV A,#0EFH**

**ADD A,#0FBH**

**MOV 30H,A**

**MOV A,#13H**

**ADDC A,#16H**

**MOV 31H,A**

# Define Data Segment Area

---

**DSEG AT 30H**

**num1: ds 2 ; 16-bit variable**

**num2: ds 2 ; 16-bit variable**

**CSEG AT 8100H**

**MOV num1, #0EFH**

**MOV num1+1, #13H**

**MOV num2, #0FBH**

**MOV num2+1, #16H**

# Define Data Segment Area

---

MOV A,num1

ADD A,num2

MOV num2,A

MOV A,num1+1

ADDC A,num2+1

MOV num2+1,A ;

; num2 = ?

# Define Data Segment Area

---

```
0030          DSEG AT 30H
0030 num1: ds 2; 16-bit variable
0032 num2: ds 2; 16-bit variable
8100          CSEG AT 8100H
8100 7530EF MOV num1,#0EFH
8103 753113 MOV num1+1,#13H
8106 7532FB MOV num2,#0FBH
8109 753316 MOV num2+1,#16H
```

# Multiple precision: 32-bit addition

---

**DSEG AT 30H**

**num1: ds 4 ; 32-bit variable**

**num2: ds 4 ; 32-bit variable**

**; num1 = 12345678H**

**; num2 = 1EFEECDAH**

**; num2 = num1+num2**

**; num2 and carry = ?**

# Multiple precision: 32-bit addition

---

MOV R7,#4 ; 8 x 4 = 32-bit

CLR C ; clear carry

MOV R0,#num1; pointer to num1

MOV R1,#num2; pointer to num2

loop:

MOV A,@R0

ADDC A,@R1 ; add with carry

MOV @R1,A ; save result

INC R0 ; next location

INC R1

DJNZ R7,loop

---

# Multiple precision: 32-bit subtraction

---

MOV R7,#4 ; 8 x 4 = 32-bit

CLR C ; clear borrow

MOV R0,#num1; pointer to num1

MOV R1,#num2; pointer to num2

loop: MOV A,@R0

SUBB A,@R1 ; sub with borrow

MOV @R1,A ; save result

INC R0 ; next location

INC R1

DJNZ R7,loop

---

# Multiplication and Division

---

**MOV A,#16**

**MOV B,#255**

**MUL AB**

**; result is saved in B:A**

**MOV DPL,A**

**MOV DPH,B**

**CALL PINT16U ; print 16-bit**

**; result to screen**

# Multiplication and Division

---

**MOV A,#255 ; 255/16 = 15.9375**

**MOV B,#16**

**DIV AB ; A = Int[A/B]**

**; B = Mod[A/B]**

**; test print result**

**CALL PINT8U**

**MOV A,B**

**CALL PINT8U; result to screen**

# Let's compute 255/16

---

$$255/16 = 15.9375$$

Can we print result like above  
with simple multiply and divide  
instructions?

$$; 255/16 \rightarrow A = 15 \quad B = 15$$

$$; 15 \times 10 / 16 \rightarrow A = 9 \quad B = 6$$

$$; 6 \times 10 / 16 \rightarrow A = 3 \quad B = 12$$

$$; 12 \times 10 / 16 \rightarrow A = 7 \quad B = 8 \dots$$

# BCD Arithmetic

---

BCD is decimal number coded with  
4-bit binary number.

Sample BCD number:

1234H

1991H

; see sample code

MOV A,#19H

ADD A,#21H ;binary addition

# BCD Arithmetic

---

**; A = 19H + 21H = 3AH**

**; now put DA A instruction  
followed ADD instruction**

**DA A**

**; the result will be 40H!**

# BCD Arithmetic

---

```
; BCD1 = 12345678H  
; BCD2 = 12345678H  
; BCD1 = BCD1+BCD2  
; result must be 24691356H  
; suppose such both BCD number  
; are stored at external memory  
; location 9000H-9006H  
; use edit to enter the number
```

# BCD Arithmetic

---

**DSEG AT 30H**

**BCD1 DS 4 ; provide internal**

**BCD2 DS 4 ; memory for two BCD**

**CSEG**

**COPY: MOV R7,#8**

**MOV R0,#BCD1**

**MOV DPTR,#9000H**

**LOOP: MOVX A,@DPTR**

**MOV @R0,A**

# BCD Arithmetic

---

**INC DPTR**

**INC R0**

**DJNZ R7,loop**

**RET**

**; check location 30H-37H of the**

**; internal memory**

**; 30H = 78563412H**

**; 34H = 78563412H**

# BCD Arithmetic

---

MOV R7, #4

MOV R0, #BCD1

MOV R1, #BCD2

CLR C

loop2: MOV A, @R0

ADDC A, @R1

DA A

MOV @R1, A

# BCD Arithmetic

---

INC R0

INC R1

DJNZ R7,loop2

RET

; run single step and study  
; the result after instruction  
; ADDC and DA A