

## Multi byte addition

Note : program running and giving the result but moving to infinite loop just check it once if possible.

## Program

; Multi Byte addition

CLC

MOV SI,2000H

MOV DI,3000H

MOV CL,[1050]

L1:MOV AL,[SI]

ADC AL,[DI]

MOV [SI],AL

INC SI

INC DI

DEC CL

JNZ L1

INT 03

### Data for input

INPUT		OUTPUT	
ADDRESS	DATA	ADDRESS	DATA
1050	04	2000	29
2000	A7	2001	9B
2001	37	2002	D9
2002	48	2003	A8
2003	23		
3000	82		
3001	63		
3002	91		
3003	85		

CL = 04 ← [ 1050 ]


23	48	37	A7 ← [ SI ], 2000
(+) 85	91	63	82 ← [ DI ], 3000
<hr/>			
A8	D9	9B	29 → [ SI ], 2000

0100:1050

update

☒ table
 ☐ list

0100:1050	04	00	00	00	00	00	00	00	00-00	00	00	00	00
0100:1060	00	00	00	00	00	00	00	00	00-00	00	00	00	00
0100:1070	00	00	00	00	00	00	00	00	00-00	00	00	00	00
0100:1080	00	00	00	00	00	00	00	00	00-00	00	00	00	00
0100:1090	00	00	00	00	00	00	00	00	00-00	00	00	00	00
0100:10A0	00	00	00	00	00	00	00	00	00-00	00	00	00	00
0100:10B0	00	00	00	00	00	00	00	00	00-00	00	00	00	00
0100:10C0	00	00	00	00	00	00	00	00	00-00	00	00	00	00

 Random

0100:2000	update				<input checked="" type="radio"/> table	<input type="radio"/> list
0100:2000	A7	37	48	23	00	00-00
0100:2010	00	00	00	00	00	00-00
0100:2020	00	00	00	00	00	00-00
0100:2030	00	00	00	00	00	00-00
0100:2040	00	00	00	00	00	00-00
0100:2050	00	00	00	00	00	00-00
0100:2060	00	00	00	00	00	00-00
0100:2070	00	00	00	00	00	00-00

Random Access Memory												
0100:3000		update		table		list						
0100:3000	82	63	91	85	00	00	00	00	00-00	00	00	0
0100:3010	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:3020	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:3030	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:3040	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:3050	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:3060	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:3070	00	00	00	00	00	00	00	00	00-00	00	00	0

Random Access Memory												
0100:2000		update		table		list						
0100:2000	29	9B	D9	A8	00	00	00	00	00-00	00	00	0
0100:2010	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:2020	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:2030	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:2040	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:2050	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:2060	00	00	00	00	00	00	00	00	00-00	00	00	0
0100:2070	00	00	00	00	00	00	00	00	00-00	00	00	0

Program: Greatest for 3 numbers AND Taking input from the keyboard

.MODEL SMALL

.DATA

MSG1 DB 10,13,"ENTER 1ST NUM:\$"

MSG2 DB 10,13,"ENTER 2ND NUM:\$"

MSG3 DB 10,13,"ENTER 3RD NUM:\$"

MSG4 DB 10,13,"LARGEST NUM:\$"

NUM1 DB ?

NUM2 DB ?

NUM3 DB ?

.CODE

MAIN PROC

MOV AX,@DATA

MOV DS,AX

LEA DX,MSG1

MOV AH,9

INT 21H

MOV AH,1

INT 21H

MOV NUM2,AL

LEA DX,MSG2

MOV AH,9

INT 21H

MOV AH,1

INT 21H

MOV NUM1,AL

```

LEA DX,MSG3
MOV AH,9
INT 21H
MOV AH,1
INT 21H
MOV NUM3,AL
LEA DX,MSG4
MOV AH,9
INT 21H
MOV BL,NUM1
CMP BL,NUM2
JNG NUMBER2 ;JUMP NOT GREAT BL
CMP BL,NUM3
JNG NUMBER3
MOV DL,NUM1
JMP DISPLAY
NUMBER2:
MOV BL,NUM2
CMP BL,NUM3
JNG NUMBER3
MOV DL,NUM1
JMP DISPLAY
NUMBER3:
MOV DL,NUM3
DISPLAY:
MOV AH,2
INT 21H

```

56h

emulator screen (80x25 chars)

ENTER 1ST NUM:3  
ENTER 2ND NUM:4  
ENTER 3RD NUM:5  
LARGEST NUM:5

Output:

```

Searching an element in the Array

; FINDING AN ELEMENT IN THE GIVEN ARRAY

MOV SI,1100H
MOV DI,1200H
MOV DL,[DI]
MOV BL,01H

```

```
MOV AL,[SI]
AGAIN:
CMP AL,DL ; CMP BOTH ARRAY ELEMENTS
JZ AVAIL ; BOTH ARE EQUA JUMP TO AVAIL
INC SI ; IF DATA ARE NOT EQUAL, INC SI
INC BL ; INC POS COUNT
MOV AL,[SI] ; GET NEXT ELEMENT OF ARRAY
CMP AL,20H ; CHECK FOR END OF THE ARRAY
JNZ AGAIN ; IF NOT END REPEAT THE SEARCH
NODATA:
MOV CX,0000H ; IF SEARCH ELEMENT NOT FOUND
MOV [DI+1],CX
MOV [DI+2],CX
JMP OVER
AVAIL:
MOV BH,0FFH ; STORE FFH TO INDICATE ELEMENT FOUND
MOV [DI+1],BH
MOV [DI+2],BL ;STORE THE POSITION OF DATA ELEMENT
MOV [DI+3],SI ; STORE THE ADDRESS OF DATA ELEMENT
OVER:
HLT
```

Source reg data

Random Access Memory															
0100:1100		update		table		list		02106							
0100:1100	1A	23	44	56	45	20	00	00-00	00	00	00	00	00	00	00
0100:1110	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1120	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1130	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1140	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1150	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1160	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1170	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00

Search eleemt in destination reg

Random Access Memory															
0100:1200		update		table		list		02205							
0100:1200	56	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1210	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1220	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1230	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1240	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1250	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1260	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00
0100:1270	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00

Final search result including element position and address

Random Access Memory																
0100:1200		update		table		list										
0100:1200	56	FF	04	03	11	00	00	00-00	00	00	00	00	00	00	00	00
0100:1210	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1220	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1230	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1240	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1250	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1260	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00
0100:1270	00	00	00	00	00	00	00	00-00	00	00	00	00	00	00	00	00

Average of 5 numbers

; average of 5 numbers

DATA SEGMENT

SUM DB 01 DUP(?) ; BYTE MEMORY RESERVED

AVG DB 01 DUP(?) ; BYTE MEMORY RESERVED

DATA ENDS

CODE SEGMENT

ASSUME CS: CODE DS:DATA

START:

MOV AX,DATA ; INI DATA SEGMENT REG

MOV DS,AX

MOV AX,00 ; SET AX AS 00

MOV AL,04 ; MOV 4 IN AL REG

ADD AL,02 ; ADD 02 WITH VAL OF AL REG

ADD AL,08 ; ADD 08 WITH VAL OF AL REG

ADD AL,03 ; ADD 03 WITH VAL OF AL REG

ADD AL,03 ; ADD 03 WITH VAL OF AL REG

MOV SUM,AL ; AL IS MOVED TO SUM

MOV BL,05 ; MOV 05 TO BL

DIV BL ; DIVISION PERFORMED

MOV AVG,AL ; STORE THE AVG CAL IN AVG

CODE ENDS ; END CODE SEGMENT

END START ; END PROG

variables

size: byte

elements: 1

edit

show as: signed

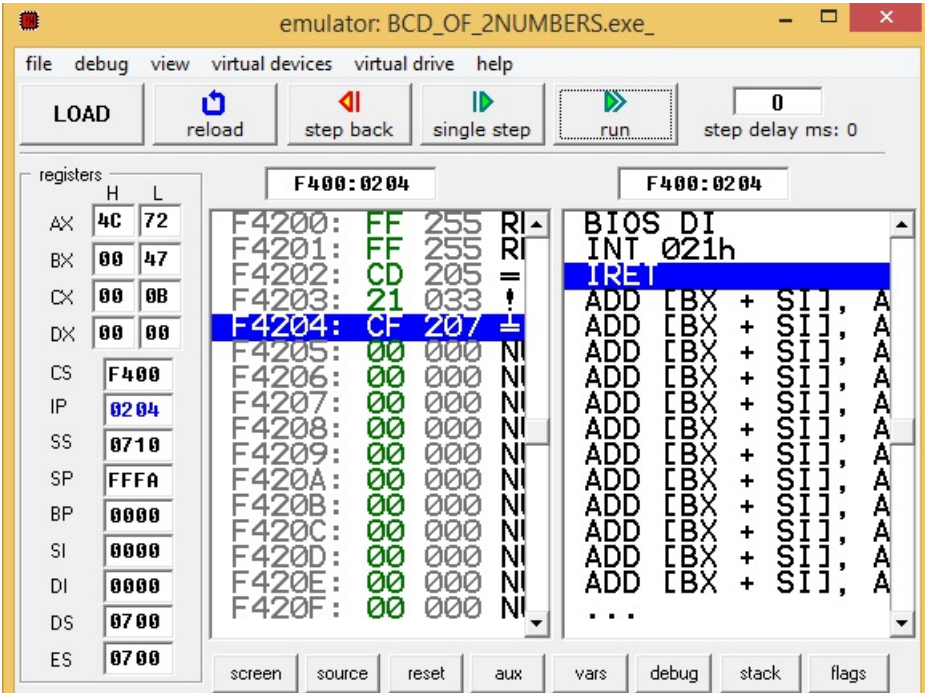
SUM	14h
AVG	4

BCD of 2 numbers:

.MODEL SMALL

```
.CODE
MOV AL,25H
MOV BL,47H
;MOV AL,99H
;MOV BL,99H
ADD AL,BL
DAA
MOV AH,4CH
INT 21H
END
```

Output:



FACORIAL OF A NUMBER

```
.DATA
ANS DB ?

.CODE

MAIN PROC

MOV AX,@DATA
MOV DS,AX

MOV AL,5
MOV CL,4

MOV BL,AL
SUB BL,1

L1:

MUL BL

SUB BL,1

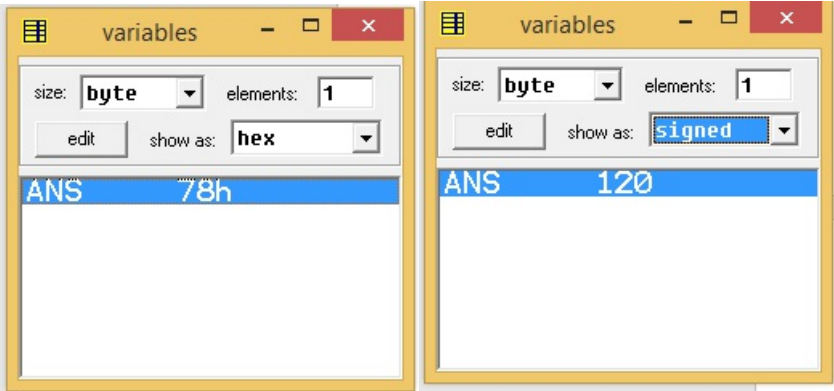
LOOP L1

MOV ANS,AL
```

END MAIN

RET

OUTPUT



HCF AND LCM

DATA SEGMENT

NUM1 DW 1500

NUM2 DW 2500

HCF DW ?

LCM DW ?

ENDS

CODE SEGMENT

ASSUME DS:DATA CS:CODE

START:

MOV AX,DATA

MOV DS,AX

MOV AX,NUM1

MOV BX,NUM2

WHILE:MOV DX,0

MOV CX,BX

DIV BX

MOV BX,DX

MOV AX,CX

CMP BX,0

JNE WHILE

MOV HCF,AX

MOV CX,AX

MOV AX,NUM1

MOV BX,NUM2

MUL BX

DIV CX

MOV LCM,AX

MOV AH,4CH

INT 21H

ENDS

END START

OUTPUT:

variables

size: word elements: 1

edit show as: signed

NUM1	1500
NUM2	2500
HCF	500
LCM	7500

variables

size: word elements: 1

edit show as: hex

NUM1	05DCh
NUM2	09C4h
HCF	01F4h
LCM	1D4Ch