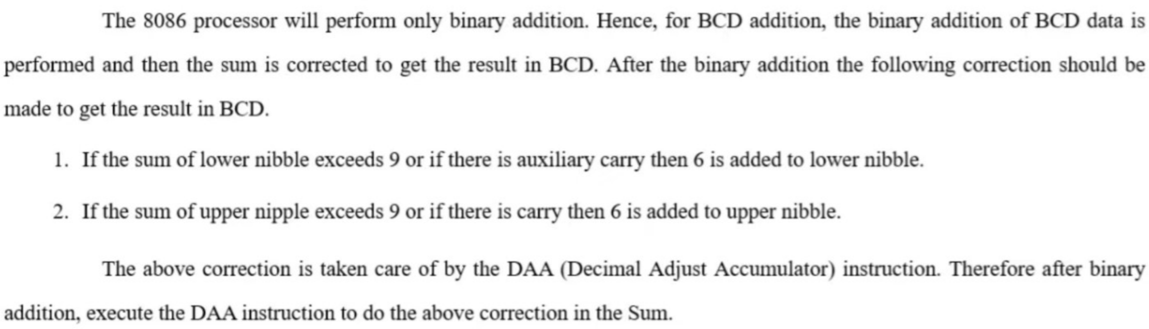
	Marwadi University **Department of Computer Engineering**	
Subject: Fundamental of Processors (01CE0509)	**Aim:To perform BCD operation in 8086.**	
Experiment No: 09	**Date:**	**Enrolment No:92201703058**

Experiment-9

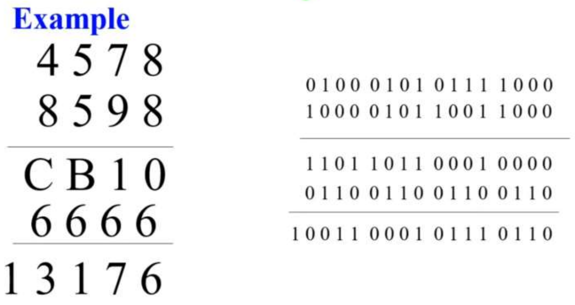
AIM:To perform Add Two Numbers of BCD Data in 8086.



The 8086 processor will perform only binary addition. Hence, for BCD addition, the binary addition of BCD data is performed and then the sum is corrected to get the result in BCD. After the binary addition the following correction should be made to get the result in BCD.

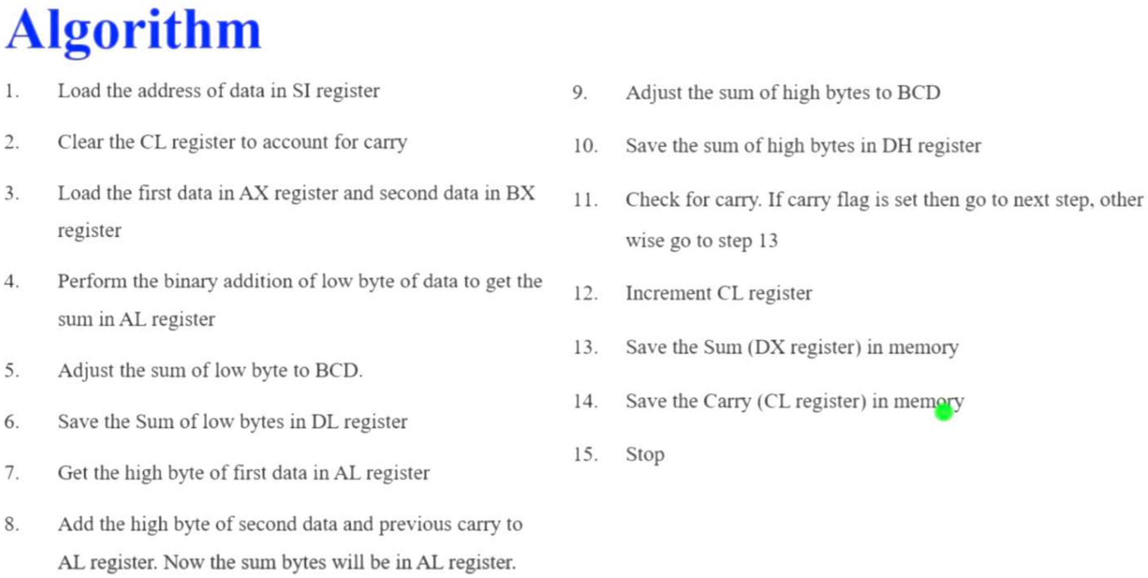
1. If the sum of lower nibble exceeds 9 or if there is auxiliary carry then 6 is added to lower nibble.
2. If the sum of upper nipple exceeds 9 or if there is carry then 6 is added to upper nibble.

The above correction is taken care of by the DAA (Decimal Adjust Accumulator) instruction. Therefore after binary addition, execute the DAA instruction to do the above correction in the Sum.



Example

```
 4578        0100 0101 0111 1000
 8598        1000 0101 1001 1000
 ----        -------------------
 CB10        1101 1011 0001 0000
 6666        0110 0110 0110 0110
 ----       --------------------
13176       10011 0001 0111 0110
```

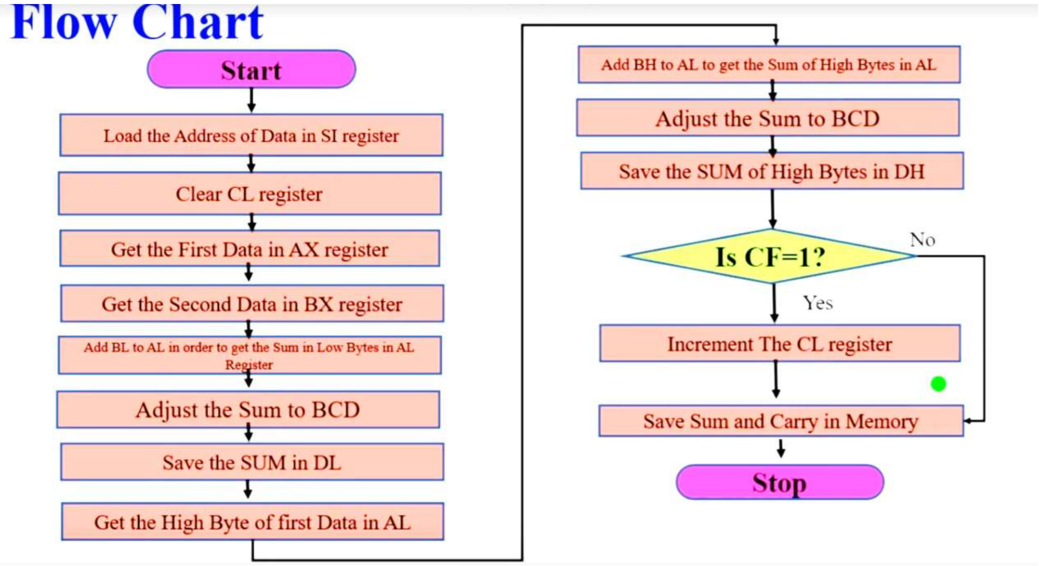


Algorithm

1. Load the address of data in SI register
2. Clear the CL register to account for carry
3. Load the first data in AX register and second data in BX register
4. Perform the binary addition of low byte of data to get the sum in AL register
5. Adjust the sum of low byte to BCD.
6. Save the Sum of low bytes in DL register
7. Get the high byte of first data in AL register
8. Add the high byte of second data and previous carry to AL register. Now the sum bytes will be in AL register.
9. Adjust the sum of high bytes to BCD
10. Save the sum of high bytes in DH register
11. Check for carry. If carry flag is set then go to next step, other wise go to step 13
12. Increment CL register
13. Save the Sum (DX register) in memory
14. Save the Carry (CL register) in memory
15. Stop

	Marwadi University **Department of Computer Engineering**
Subject: Fundamental of Processors (01CE0509)	**Aim:To perform BCD operation in 8086.**
Experiment No: 09	**Date:** | **Enrolment No:92201703058**

Flow Chart



Start

Load the Address of Data in SI register

Clear CL register

Get the First Data in AX register

Get the Second Data in BX register

Add BL to AL in order to get the Sum in Low Bytes in AL Register

Adjust the Sum to BCD

Save the SUM in DL

Get the High Byte of first Data in AL

Add BH to AL to get the Sum of High Bytes in AL

Adjust the Sum to BCD

Save the SUM of High Bytes in DH

Is CF=1?

No

Yes

Increment The CL register

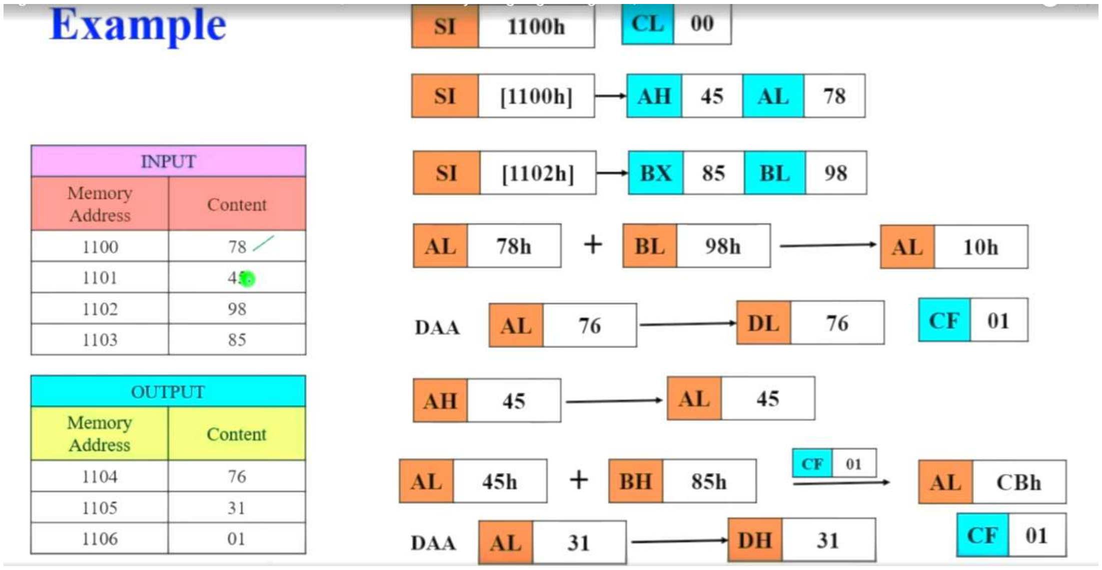
Save Sum and Carry in Memory

Stop

Program

```
MOV SI,1100H
MOV CL,00H
MOV AX,[SI]
MOV BX,[SI+2]
ADD AL,BL
DAA
MOV DL,AL
MOV AL,AH
ADC AL,BH
DAA
MOV DH,AL
JNC jump
INC CL
jump:
MOV [SI+4],DX
MOV [SI+6],CL
HLT
```

	Marwadi University Department of Computer Engineering	
Subject: Fundamental of Processors (01CE0509)	**Aim:To perform BCD operation in 8086.**	
Experiment No: 09	**Date:**	**Enrolment No:92201703058**



Example

SI 1100h   CL 00

SI [1100h] → AH 45 AL 78

SI [1102h] → BX 85 BL 98

AL 78h + BL 98h → AL 10h

DAA AL 76 → DL 76   CF 01

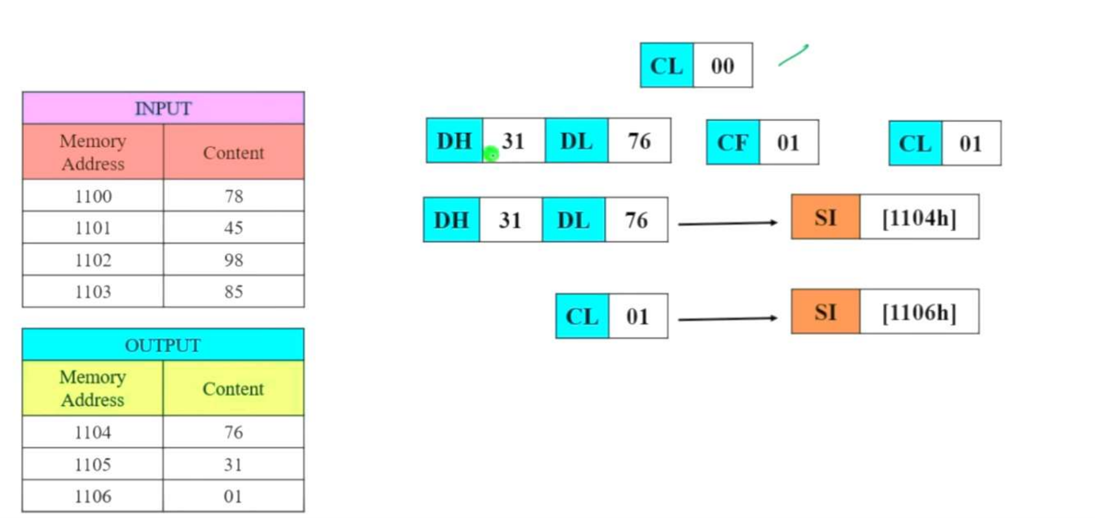
AH 45 → AL 45

AL 45h + BH 85h CF 01 → AL CBh

DAA AL 31 → DH 31   CF 01

INPUT	
Memory Address	Content
1100	78
1101	45
1102	98
1103	85

OUTPUT	
Memory Address	Content
1104	76
1105	31
1106	01



CL 00

DH 31 DL 76   CF 01   CL 01

DH 31 DL 76 → SI [1104h]

CL 01 → SI [1106h]

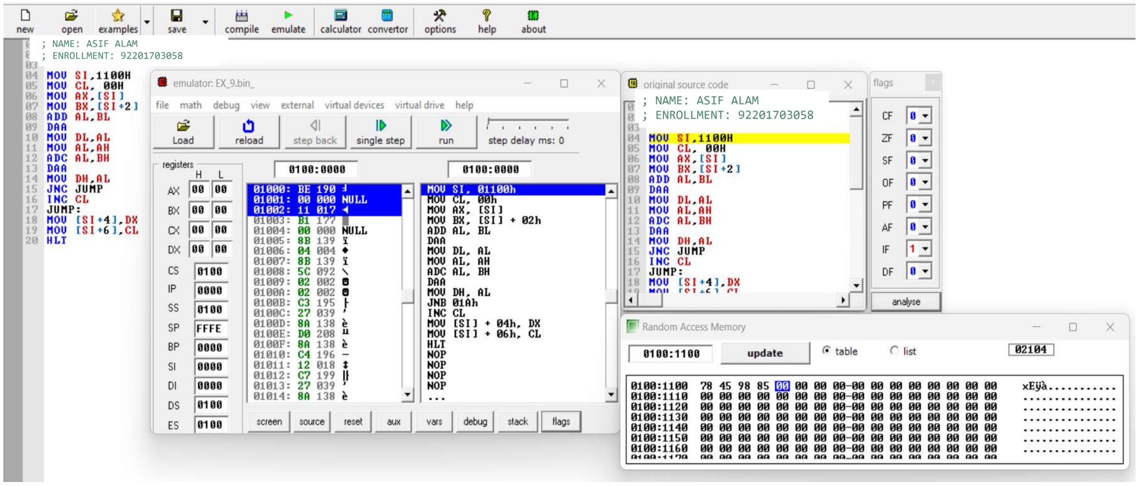
INPUT	
Memory Address	Content
1100	78
1101	45
1102	98
1103	85

OUTPUT	
Memory Address	Content
1104	76
1105	31
1106	01

Program :

	Marwadi University **Department of Computer Engineering**
Subject: Fundamental of Processors (01CE0509)	**Aim:To perform BCD operation in 8086.**
Experiment No: 09	**Date:** / **Enrolment No:92201703058**

Update memory provide data :



```
; NAME: ASIF ALAM
; ENROLLMENT: 92201703058

MOV SI,1100H
MOV CL, 00H
MOV AX,[SI]
MOV BX,[SI+2]
ADD AL,BL
DAA
MOV DL,AL
MOV AL,AH
ADC AL,BH
DAA
MOV DH,AL
JNC JUMP
INC CL
JUMP:
MOV [SI+4],DX
MOV [SI+6],CL
HLT
```

emulator: EX_9.bin_

```
registers  H  L
AX        00 00
BX        00 00
CX        00 00
DX        00 00
CS        0100
IP        0000
SS        0100
SP        FFFE
BP        0000
SI        0000
DI        0000
DS        0100
ES        0100
```

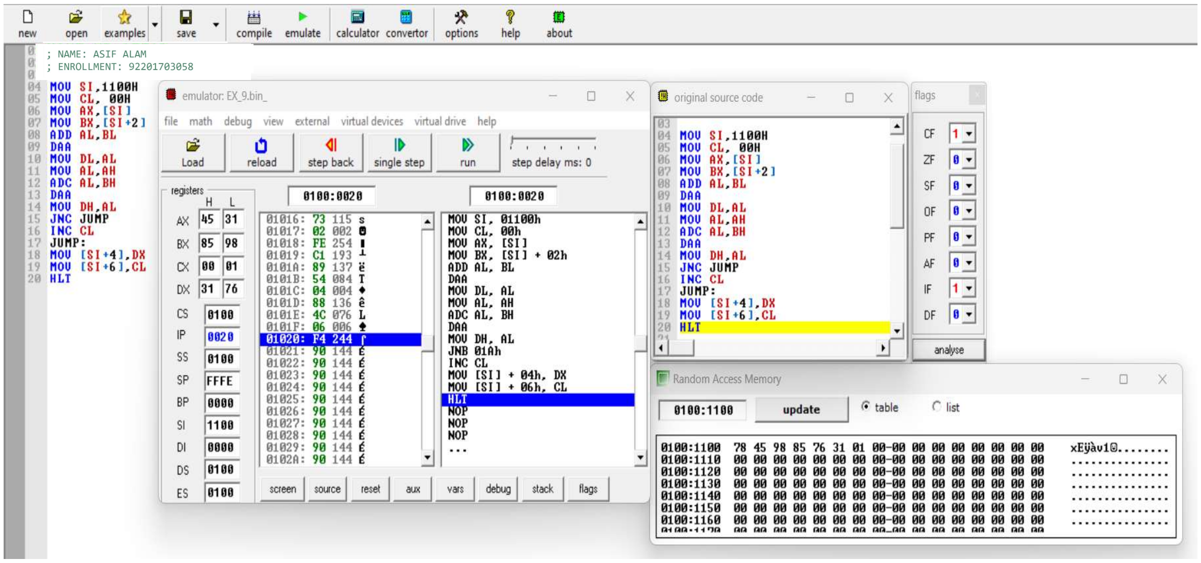
```
0100:0000                      0100:0000
01000: BE 190 ╛                MOV SI, 01100h
01001: 00 000 NULL             MOV CL, 00h
01002: 11 017 ◄                MOV AX, [SI]
01003: B1 177 ▒                MOV BX, [SI] + 02h
01004: 00 000 NULL             ADD AL, BL
01005: 8B 139 ï                DAA
01006: 04 004 ♦                MOV DL, AL
01007: 8B 139 ï                MOV AL, AH
01008: 5C 092 \                ADC AL, BH
01009: 02 002 ☻                DAA
0100A: 02 002 ☻                MOV DH, AL
0100B: C3 195 ├                JNB 01Ah
0100C: 27 039 '                INC CL
0100D: 8A 138 è                MOV [SI] + 04h, DX
0100E: D0 208 ╨                MOV [SI] + 06h, CL
0100F: 8A 138 è                HLT
01010: C4 196 ─                NOP
01011: 12 018 ↕                NOP
01012: C7 199 ╟                NOP
01013: 27 039 '                NOP
01014: 8A 138 è                ...
```

```
flags
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
```

Random Access Memory

```
0100:1100   update   ● table   ○ list   02104

0100:1100  78 45 98 85 00 00 00 00-00 00 00 00 00 00 00 00   xEÿà............
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   ................
```

Check the output and Flag :



emulator: EX_9.bin_

```
registers  H  L
AX        45 31
BX        85 98
CX        00 01
DX        31 76
CS        0100
IP        0020
SS        0100
SP        FFFE
BP        0000
SI        1100
DI        0000
DS        0100
ES        0100
```

```
0100:0020                      0100:0020
01016: 73 115 s                MOV SI, 01100h
01017: 02 002 ☻                MOV CL, 00h
01018: FE 254 ■                MOV AX, [SI]
01019: C1 193 ┴                MOV BX, [SI] + 02h
0101A: 89 137 ë                ADD AL, BL
0101B: 54 084 T                DAA
0101C: 04 004 ♦                MOV DL, AL
0101D: 88 136 ê                MOV AL, AH
0101E: 4C 076 L                ADC AL, BH
0101F: 06 006 ♠                DAA
01020: F4 244 ⌠                MOV DH, AL
01021: 90 144 É                JNB 01Ah
01022: 90 144 É                INC CL
01023: 90 144 É                MOV [SI] + 04h, DX
01024: 90 144 É                MOV [SI] + 06h, CL
01025: 90 144 É                HLT
01026: 90 144 É                NOP
01027: 90 144 É                NOP
01028: 90 144 É                NOP
01029: 90 144 É                ...
0102A: 90 144 É
```

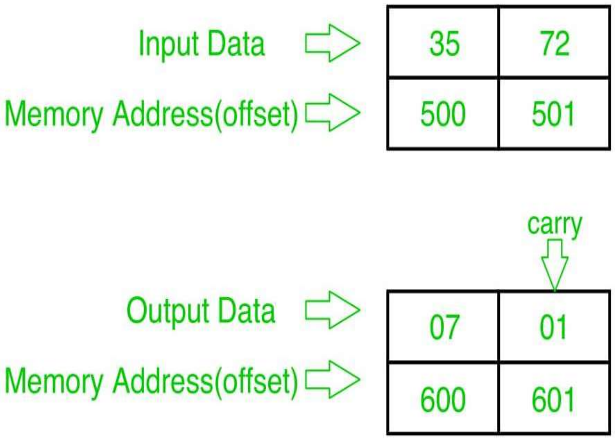
```
flags
CF 1
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
```

Random Access Memory

```
0100:1100   update   ● table   ○ list

0100:1100  78 45 98 85 76 31 01 00-00 00 00 00 00 00 00 00   xEÿàv1☺.........
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   ................
```

Program 2 :Write a program in 8086 microprocessor to find out the addition of two 8-bit BCD numbers, where numbers are stored from starting memory address 2000 : 500 and store the result into memory address 2000 : 600 and carry at 2000 : 601.

Example –



Algorithm –

1. Load data from offset 500 to register AL (first number)
2. Load data from offset 501 to register BL (second number)
3. Add these two numbers (contents of register AL and register BL)
4. Apply DAA instruction (decimal adjust)
5. Store the result (content of register AL) to offset 600
6. Set register AL to 00
7. Add contents of register AL to itself with carry
8. Store the result (content of register AL) to offset 601
9. Stop

Program –

MEMORY ADDRESS	MNEMONICS	COMMENT
400	MOV AL, [500]	AL<-[500]
404	MOV BL, [501]	BL<-[501]
408	ADD AL, BL	AL<-AL+BL
40A	DAA	DECIMAL ADJUST AL
40B	MOV [600], AL	AL->[600]
40F	MOV AL, 00	AL<-00

411	ADC AL, AL	AL<-AL+AL+cy(prev)
413	MOV [601], AL	AL->[601]
417	HLT	END

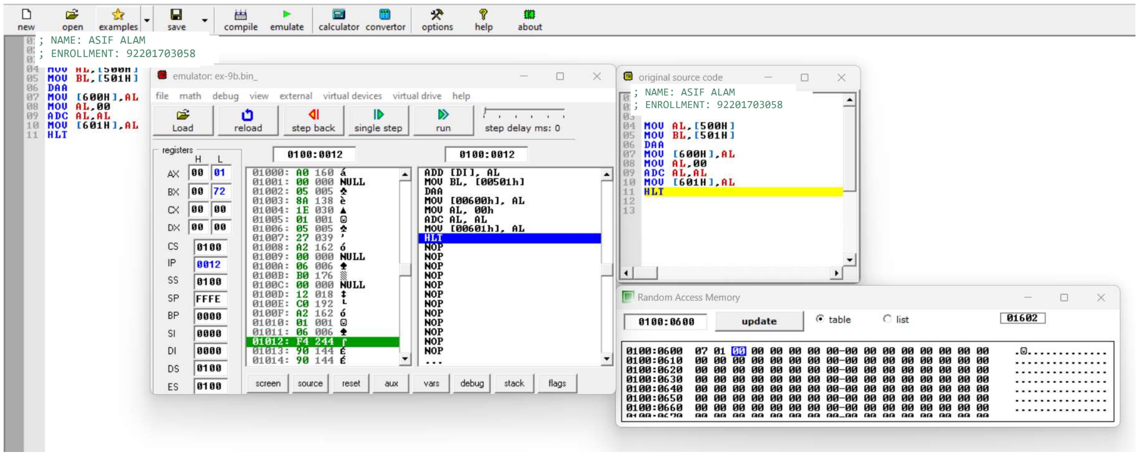
Explanation –

1. **MOV AL, [500]**: load data from offset 500 to register AL
2. **MOV BL, [501]**: load data from offset 501 to register BL
3. **ADD AL, BL**: ADD contents of registers AL AND BL
4. **DAA**: decimal adjust AL
5. **MOV [600], AL**: store data from register AL to offset 600
6. **MOV AL, 00**: set value of register AL to 00
7. **ADC AL, AL**: add contents of register AL to AL with carry
8. **MOV [601], AL**: store data from register AL to offset 601
9. **HLT**: stop

Program :

Marwadi University Marwadi Chandarana Group NAAC A+	Marwadi University Department of Computer Engineering	
Subject: Fundamental of Processors (01CE0509)	Aim:To perform BCD operation in 8086.	
Experiment No: 09	Date:	Enrolment No:92201703058

Update memory provide data :

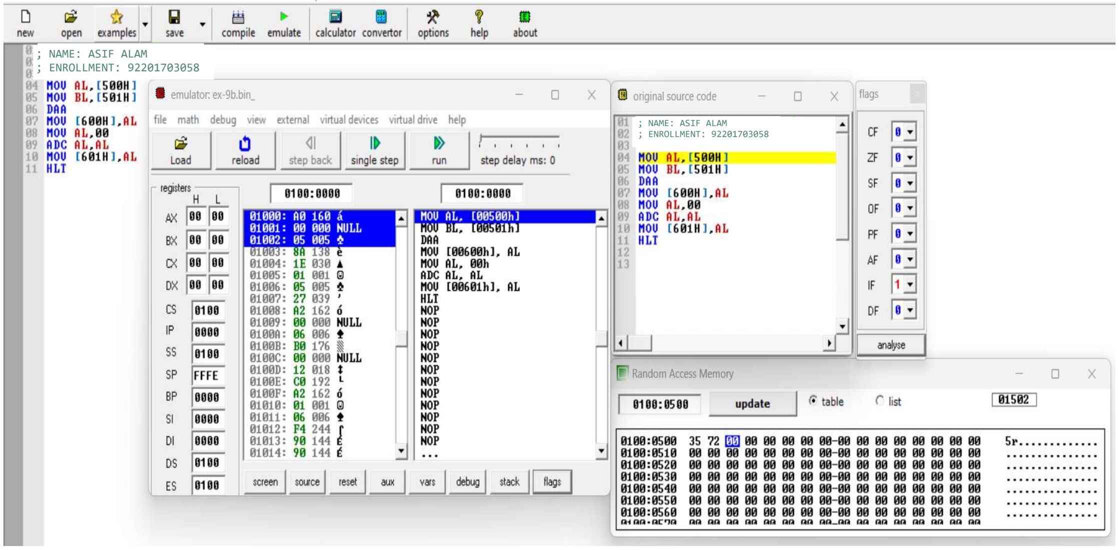


```
; NAME: ASIF ALAM
; ENROLLMENT: 92201703058

MOV AL,[500H]
MOV BL,[501H]
DAA
MOV [600H],AL
MOV AL,00
ADC AL,AL
MOV [601H],AL
HLT
```

```
registers
     H   L
AX  00  01
BX  00  72
CX  00  00
DX  00  00
CS  0100
IP  0012
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0012                      0100:0012
01000: A0 160 á                ADD [DI], AL
01001: 00 000 NULL             MOV BL, [00501h]
01002: 05 005 ♣                DAA
01003: 8A 138 è                MOV [00600h], AL
01004: 1E 030 ▲                MOV AL, 00h
01005: 01 001 ☺                ADC AL, AL
01006: 05 005 ♣                MOV [00601h], AL
01007: 27 039 '                HLT
01008: A2 162 ó                NOP
01009: 00 000 NULL             NOP
0100A: 06 006 ♠                NOP
0100B: B0 176 ░                NOP
0100C: 00 000 NULL             NOP
0100D: 12 018 ↕                NOP
0100E: C0 192 └                NOP
0100F: A2 162 ó                NOP
01010: 01 001 ☺                NOP
01011: 06 006 ♠                NOP
01012: F4 244 ⌠                NOP
01013: 90 144 É                NOP
01014: 90 144 É                ...
```

```
0100:0600    ○ table   ○ list    01602

0100:0600   07 01 00 00 00 00 00 00-00 00 00 00 00 00 00 00   .☺..............
0100:0610   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0620   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0630   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0640   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0650   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0660   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```



```
; NAME: ASIF ALAM
; ENROLLMENT: 92201703058

MOV AL,[500H]
MOV BL,[501H]
DAA
MOV [600H],AL
MOV AL,00
ADC AL,AL
MOV [601H],AL
HLT
```

```
registers
     H   L
AX  00  00
BX  00  00
CX  00  00
DX  00  00
CS  0100
IP  0000
SS  0100
SP  FFFE
BP  0000
SI  0000
DI  0000
DS  0100
ES  0100
```

```
0100:0000                      0100:0000
01000: A0 160 á                MOV AL, [00500h]
01001: 00 000 NULL             MOV BL, [00501h]
01002: 05 005 ♣                DAA
01003: 8A 138 è                MOV [00600h], AL
01004: 1E 030 ▲                MOV AL, 00h
01005: 01 001 ☺                ADC AL, AL
01006: 05 005 ♣                MOV [00601h], AL
01007: 27 039 '                HLT
01008: A2 162 ó                NOP
01009: 00 000 NULL             NOP
0100A: 06 006 ♠                NOP
0100B: B0 176 ░                NOP
0100C: 00 000 NULL             NOP
0100D: 12 018 ↕                NOP
0100E: C0 192 └                NOP
0100F: A2 162 ó                NOP
01010: 01 001 ☺                NOP
01011: 06 006 ♠                NOP
01012: F4 244 ⌠                NOP
01013: 90 144 É                NOP
01014: 90 144 É                ...
```

```
flags
CF 0
ZF 0
SF 0
OF 0
PF 0
AF 0
IF 1
DF 0
```

```
0100:0500    ● table   ○ list    01502

0100:0500   35 72 00 00 00 00 00 00-00 00 00 00 00 00 00 00   5r..............
0100:0510   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0520   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0530   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0540   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0550   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0100:0560   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

Conclusion :