1, 8086 Assembly Language program:
ASCII Addition

Software :- 8086 emulator

Program code:-

```
CODE  SEGMENT
ASSUME CS: CODE
START : MOV AL, 05H
        MOV BL, 09H
        ADD AL, BL
        AAA
        OR AX, 3030H
        INT 21H
CODE ENDS
END START.
```

Result:-

AL : 05H
BL : 09H

After AAA AL : 3134H

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[28-08-2020 -- 17:44:55]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	CODE SEGMENT
[2]	:	ASSUME CS: CODE
[3]	0000: B0 05	START: MOV AL,05H
[4]	0002: B3 09	MOV BL,09H
[5]	0004: 02 C3	ADD AL, BL
[6]	0006: 37	AAA
[7]	0007: 0D 30 30	OR AX, 3030H
[8]	000A: CD 21	INT 21H
[9]	:	CODE ENDS
[10]	:	END START
[11]	:	
[12]	:	

02, 8086 Assembly Language Program:
ASCII Subtraction

Software:- 8086 emulator

Program Code:-

```
CODE SEGMENT
ASSUME CS: CODE
START: MOV AL, 09H
       MOV BL, 05H
       SUB AL, BL
       AAS
       OR AX, 3030H
       INT 21H
CODE ENDS
END START
```

Result:-

AL : 09H
BL : 05H

After AAS AL : 3034H

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[28-08-2020 -- 17:50:34]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	CODE SEGMENT
[2]	:	ASSUME CS: CODE
[3]	0000: B0 09	START: MOV AL,09H
[4]	0002: B3 05	MOV BL,05H
[5]	0004: 2A C3	SUB AL, BL
[6]	0006: 3F	AAS
[7]	0007: 0D 30 30	OR AX, 3030H
[8]	000A: CD 21	INT 21H
[9]	:	CODE ENDS
[10]	:	END START
[11]	:	
[12]	:	

03, 8086 Assembly Language Program:
ASCII Multiplication.

Software:- 8086 emulator

Program Code:-

```
CODE SEGMENT
ASSUME CS: CODE
START : MOV AL, 05H
        MOV BL, 09H
        MUL BL
        AAM
        OR AX, 3030H
        INT 21H
CODE ENDS
END START
```

Result:-

AL : 05H

BL : 09H

After AAM AL : 3435H

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.bin_ -- emu8086 assembler version: 4.08

[28-08-2020 -- 17:55:04]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	CODE SEGMENT
[2]	:	ASSUME CS: CODE
[3]	0000: B0 05	START: MOV AL, 05H
[4]	0002: B3 09	MOV BL, 09H
[5]	0004: F6 E3	MUL BL
[6]	0006: D4 0A	AAM
[7]	0008: 0D 30 30	OR AX, 3030H
[8]	000B: CD 21	INT 21H
[9]	:	CODE ENDS
[10]	:	
[11]	:	

04, 8086 Assembly Language program:

ASCII Division

Software:- 8086 emulator

Program Code:-

```
CODE SEGMENT
  ASSUME CS: CODE
  START: MOV AX, 0607H
         MOV CH, 09H
         AAD
         DIV CH
         OR AX, 3030H
         INT 21H
CODE ENDS
END START
```

Result:-

AX: 0607H

CH: 09H

After AAD AX: 3437H

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[28-08-2020 -- 17:57:38]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	CODE SEGMENT
[2]	:	ASSUME CS: CODE
[3]	0000: B8 07 06	START: MOV AX, 0607H
[4]	0003: B5 09	MOV CH, 09H
[5]	0005: D5 0A	AAD
[6]	0007: F6 F5	DIV CH
[7]	0009: 0D 30 30	OR AX, 3030H
[8]	000C: CD 21	INT 21H
[9]	:	CODE ENDS
[10]	:	END START
[11]	:	
[12]	:	

05) Logic Operations 8086 Assembly Language

Program: ~~AXXII~~ LOGICAL AND

Software:- 8086 emulator

Program Code:-

```
CODE SEGMENT
ASSUME CS: CODE
START : MOV AL, 85H
        MOV BL, 99H
        AND AL, BL
        INT 21H
CODE ENDS
END START
```

Result:-

AL - 85H
BL - 99H

After AND AL - 81H

Theoretical Verification:-

AL - 85H - 1000 0101
BL - 99H - 1001 1001

AND -1000 0001

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe -- emu8086 assembler version: 4.08

[28-08-2020 -- 18:02:08]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	CODE SEGMENT
[2]	:	ASSUME CS: CODE
[3]	0000: B0 85	START: MOV AL, 85H
[4]	0002: B3 99	MOV BL, 99H
[5]	0004: 22 C3	AND AL, BL
[6]	0006: CD 21	INT 21H
[7]	:	CODE ENDS
[8]	:	END START
[9]	:	
[10]	:	

06) Logic Operations 8086 Assembly Language Program: LOGICAL OR

Software:- 8086 emulator

Program Code:-

```
CODE SEGMENT
ASSUME CS: CODE
START: MOV AL, 85H
       MOV BL, 99H
       OR AL, BL
       INT 21H
CODE ENDS
END START
```

Result:-

AL - 85H
BL - 99H

After OR AL - 9DH

Theoretical Verification:-

AL - 85H	-	1000 0101
BL - 99H	-	1001 1001
OR		1001 1101

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[28-08-2020 -- 18:05:16]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	CODE SEGMENT
[2]	:	ASSUME CS: CODE
[3]	0000: B0 85	START: MOV AL, 85H
[4]	0002: B3 99	MOV BL, 99H
[5]	0004: 0A C3	OR AL, BL
[6]	0006: CD 21	INT 21H
[7]	:	CODE ENDS
[8]	:	END START
[9]	:	
[10]	:	

07) Logical operations 8086 Assembly Language

Program: Logical XOR

Software:- 8086 emulator

Program Code:-

```
CODE SEGMENT
ASSUME CS: CODE
START: MOV AL, 85H
       MOV BL, 99H
       XOR AL, BL
       INT 21H
CODE ENDS
END START
```

Result:-

AL : 85H

BL : 99H

After XOR|AL:- 1CH

Theoretical Verification:-

AL - 85H - 1000 0101

BL - 99H - 1001 1001

XOR - 0001 1100

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[28-08-2020 -- 18:07:58]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	CODE SEGMENT
[2]	:	ASSUME CS: CODE
[3]	0000: B0 85	START: MOV AL, 85H
[4]	0002: B3 99	MOV BL, 99H
[5]	0004: 32 C3	XOR AL, BL
[6]	0006: CD 21	INT 21H
[7]	:	CODE ENDS
[8]	:	END START
[9]	:	
[10]	:	

08) Logical Operations 8086 Assembly Language program: NOT operation

Software:- 8086 emulator

Program Code:-

```
CODE SEGMENT
ASSUME CS: CODE
START: MOV AL, 85H
       NOT AL
       INT 21H
CODE ENDS
END START
```

Result:-

AL - 85H

After NOT AL - 7AH

Theoretical verification:-

AL - 85H - 1000 0101

NOT - 0111 1010

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[28-08-2020 -- 18:09:52]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	CODE SEGMENT
[2]	:	ASSUME CS: CODE
[3]	0000: B0 85	START: MOV AL, 85H
[4]	0002: F6 D0	NOT AL
[5]	0004: CD 21	INT 21H
[6]	:	CODE ENDS
[7]	:	END START
[8]	:	
[9]	:	

09, Logical Operations 8086 Assembly Language program: NAND operation

Software:- 8086 emulator

Program Code:-

```
CODE SEGMENT
ASSUME CS: CODE
START: MOV AL, 85H
       MOV BL, 99H
       AND AL, BL
       NOT AL
       INT 21H
CODE ENDS
END START
```

Result:-

AL - 85H
BL - 99H
After NAND AL - 7EH

Theoretical Verification:-

AL - 85H - 1000 0101
BL - 99H - 1001 1001

NAND — 0111 1110

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[28-08-2020 -- 18:12:40]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	CODE SEGMENT
[2]	:	ASSUME CS: CODE
[3]	0000: B0 85	START: MOV AL, 85H
[4]	0002: B3 99	MOV BL, 99H
[5]	0004: 22 C3	AND AL, BL
[6]	0006: F6 D0	NOT AL
[7]	0008: CD 21	INT 21H
[8]	:	CODE ENDS
[9]	:	END START
[10]	:	
[11]	:	

10. Write and Execute an assembly language program for adding two 8-bit numbers.

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
OPR1  DB 50H
OPR2  DB 51H
SUM   DB ?
DATA  ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
MOV   DS, AX
XOR   AX, AX
MOV   AL, OPR1
MOV   BL, OPR2
ADD   AL, BL
MOV   SUM, AL
INT   21H
CODE  ENDS
END   START
```

Result:-

OPR1 :	0710:0000	50H
OPR2 :	0710:0001	51H
SUM :	0710:0002	A1H

8-bit ADDITION

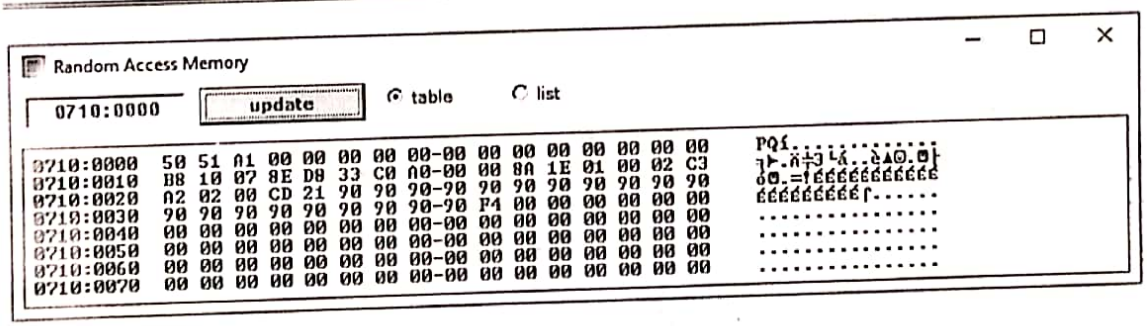
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[07-09-2020 -- 18:00:54]

```
[LINE]        LOC: MACHINE CODE                SOURCE

[  1]                :                         DATA SEGMENT
[  2]                     0000: 50    OPR1 DB 50H
[  3]                     0001: 51            OPR2 DB 51H
[  4]                     0002: 00            SUM DB  ?
[  5]                     :                   DATA ENDS
[  6]                     :                   CODE SEGMENT
[  7]                     :                   ASSUME CS: CODE, DS:DATA
[  8]                     0010: B8 00 00      START: MOV AX, DATA
[  9]                     0013: 8E D8         MOV DS, AX
[ 10]                     0015: 33 C0         XOR AX, AX
[ 11]                     0017: A0 00 00      MOV AL, OPR1
[ 12]                     001A: 8A 1E 01 00   MOV BL, OPR2
[ 13]                     001E: 02 C3         ADD AL, BL
[ 14]                     0020: A2 02 00      MOV SUM, AL
[ 15]                     0023: CD 21         INT 21H
[ 16]                     :                   CODE ENDS
[ 17]                     :                   END START
[ 18]                     :
```



11, Write and execute an assembly language program for subtracting two 8-bit numbers.

Software :- 8086 emulator

Program Code :-

```
DATA SEGMENT
OPR 1   DB 50H
OPR 2   DB 51H
 SB     DB ?
DATA    ENDS
CODE    SEGMENT
ASSUME  CS: CODE, DS: DATA
 START: MOV AX, DATA
 MOV    DS, AX
 XOR    AX, AX
 MOV    AL , OPR1
 MOV    BL , OPR2
 SUB    AL, BL
 MOV    SB, AL
 INT    21H
 CODE   ENDS
 END    START
```

Result :-

OPR1 : 0710 : 0000 50H

OPR2 : 0710 : 0001 51H

SUB : 0710 : 0002 FFH

8-bit SUBTRACTION

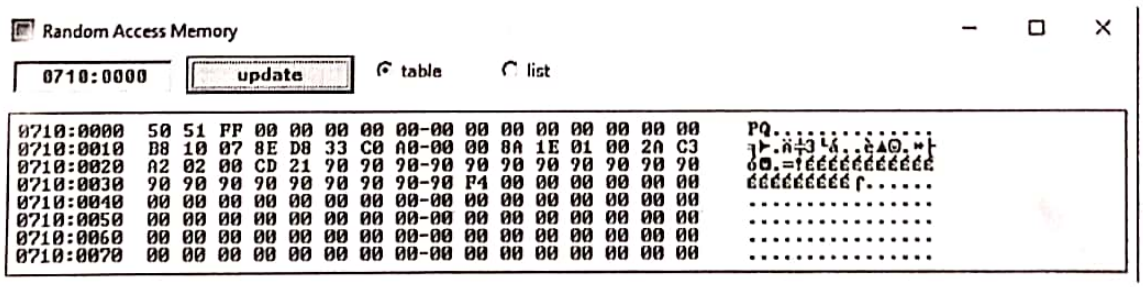
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[07-09-2020 -- 18:23:22]

```
[LINE]         LOC: MACHINE CODE                  SOURCE

[  1]                :                              DATA SEGMENT
[  2]                0000: 50    OPR1 DB 50H
[  3]                0001: 51    OPR2 DB 51H
[  4]                0002: 00    SB DB  ?
[  5]                :                          DATA ENDS
[  6]                :                          CODE SEGMENT
[  7]                :                          ASSUME CS: CODE, DS:DATA
[  8]                0010: B8 00 00                START: MOV AX, DATA
[  9]                0013: 8E D8                   MOV DS, AX
[ 10]                0015: 33 C0                   XOR AX, AX
[ 11]                0017: A0 00 00                MOV AL, OPR1
[ 12]                001A: 8A 1E 01 00              MOV BL, OPR2
[ 13]                001E: 2A C3                   SUB AL, BL
[ 14]                0020: A2 02 00                MOV SB, AL
[ 15]                0023: CD 21                   INT 21H
[ 16]                :                          CODE ENDS
[ 17]                :                          END START
[ 18]                :
[ 19]                :
```



12, Write and Execute an assembly language program for multiplying two 8-bit numbers.

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
OPR1  DB  50H
OPR2  DB  51H
MLP   DW  ?
DATA  ENDS
CODE  SEGMENT
ASSUME CS: CODE, DS: DATA
 MOV  DS, AX
 XOR  AX, AX
 MOV  AL, OPR1
 MOV  BL, OPR2
 MUL  BL
 LEA  SI, MLP
 MOV  [SI], AX
 INT  21H
CODE ENDS
END  START
```

Result:-

```
OPR1 : 0710 : 0000   50H
OPR2 : 0710 : 0001   51H
MLP  : 0710 : 0002   1950H
```

8-bit MULTIPLICATION

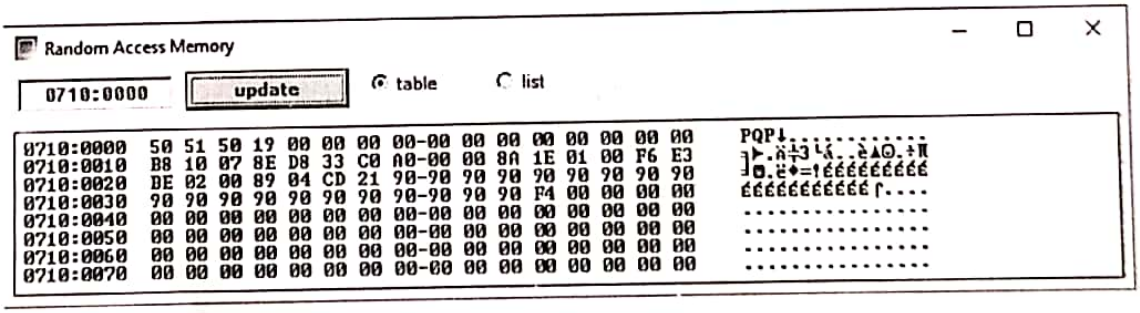
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[07-09-2020 -- 18:29:59]

```
[LINE]        LOC: MACHINE CODE                SOURCE
```

```
[  1]                :
[  2]         0000: 50                        DATA SEGMENT
[  3]         0001: 51                        OPR1 DB 50H
[  4]         0002: 00 00                     OPR2 DB 51H
[  5]                :                        MLP DW  ?
[  6]                :                        DATA ENDS
[  7]                :                        CODE SEGMENT
[  8]         0010: B8 00 00                  ASSUME CS: CODE, DS:DATA
[  9]         0013: 8E D8                     START: MOV AX, DATA
[ 10]         0015: 33 C0                     MOV DS, AX
[ 11]         0017: A0 00 00                   XOR AX, AX
[ 12]         001A: 8A 1E 01 00                MOV AL, OPR1
[ 13]         001E: F6 E3                      MOV BL, OPR2
[ 14]         0020: BE 02 00                   MUL BL
[ 15]         0023: 89 04                      LEA SI, MLP
[ 16]         0025: CD 21                      MOV [SI], AX
[ 17]                :                        INT 21H
[ 18]                :                        CODE ENDS
[ 19]                :                        END START
[ 20]                :
```



13, Write and execute an assembly language program for dividing two 8-bit numbers.

Software :- 8086 emulator

Program Code :-

```
DATA SEGMENT
OPR1 DB 40H
OPR2 DB 02H
RES  DB ?
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START : MOV, AX, DATA
MOV DS, AX
XOR AX, AX
MOV AL, OPR1
MOV BL, OPR2
DIV BL
LEA SI, RES
MOV [SI], AX
INT 21H
CODE ENDS
END START
```

Result :-

OPR 1 :	0710 : 0000	40H
OPR 2 :	0710 : 0001	02H
RES :	0710 : 0002	20H

8-bit DIVISION

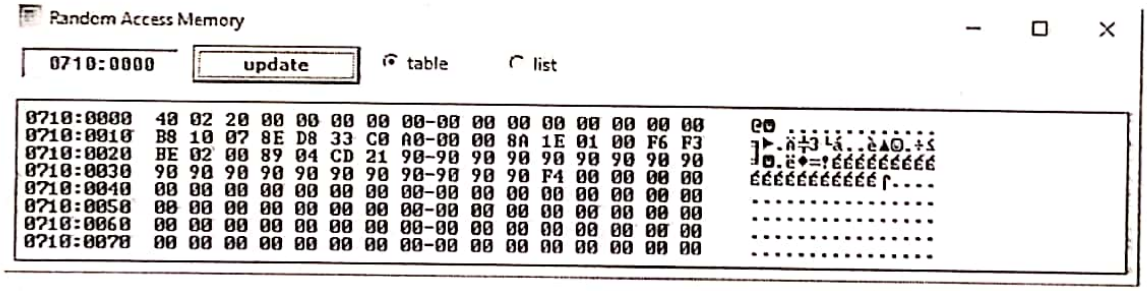
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[07-09-2020 -- 18:39:10]

```
[LINE]        LOC: MACHINE CODE          SOURCE

[  1]         :                          DATA SEGMENT
[  2]         0000: 40                   OPR1 DB 40H
[  3]         0001: 02                   OPR2 DB 02H
[  4]         0002: 00                   RES DB  ?
[  5]         :                          DATA ENDS
[  6]         :                          CODE SEGMENT
[  7]         :                          ASSUME CS: CODE, DS:DATA
[  8]         0010: B8 00 00             START: MOV AX, DATA
[  9]         0013: 8E D8                MOV DS, AX
[ 10]         0015: 33 C0                XOR AX, AX
[ 11]         0017: A0 00 00             MOV AL, OPR1
[ 12]         001A: 8A 1E 01 00          MOV BL, OPR2
[ 13]         001E: F6 F3                DIV BL
[ 14]         0020: BE 02 00             LEA SI, RES
[ 15]         0023: 89 04                MOV [SI], AX
[ 16]         0025: CD 21                INT 21H
[ 17]         :                          CODE ENDS
[ 18]         :                          END START
[ 19]         :
[ 20]         :
```



14) Write and execute an assembly language program for two 16-BIT numbers addition.

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
OPR1 DW 1234H
OPR2 DW 5678H
SUM DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
MOV DS, AX
MOV AX, OPR1
MOV BX, OPR2
ADD AX, BX
LEA SI, SUM
MOV [AX], SI
INT 21H
CODE ENDS
END START
```

Result:-

OPR1 : 1234H

OPR2 : 5678H

SUM : 68ACH

16-bit ADDITION

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[08-09-2020 -- 19:12:16]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	DATA SEGMENT
[2]	0000: 34 12	OPR1 DW 1234H
[3]	0002: 78 56	OPR2 DW 5678H
[4]	0004: 00 00	SUM DW ?
[5]	:	DATA ENDS
[6]	:	CODE SEGMENT
[7]	:	ASSUME CS: CODE, DS:DATA
[8]	0010: B8 00 00	START: MOV AX, DATA
[9]	0013: 8E D8	MOV DS, AX
[10]	0015: A1 00 00	MOV AX, OPR1
[11]	0018: 8B 1E 02 00	MOV BX, OPR2
[12]	001C: 03 C3	ADD AX, BX
[13]	001E: BE 04 00	LEA SI, SUM
[14]	0021: 89 04	MOV [SI], AX
[15]	0023: CD 21	INT 21H
[16]	:	CODE ENDS
[17]	:	END START
[18]	:	
[19]	:	

Random Access Memory

0710:0000 | update | table | list

```
0710:0000  34 12 78 56 AC 68 00 00-00 00 00 00 00 00 00 00   4↕xV¼h..........
0710:0010  B8 10 07 8E D8 A1 00 00-8B 1E 02 00 03 C3 BE 04   ╕►.Ä╪í..ï▲☻.♥├╛♦
0710:0020  00 89 04 CD 21 90 90 90-90 90 90 90 90 90 90 90   .ë♦=!ÉÉÉÉÉÉÉÉÉÉÉ
0710:0030  90 90 90 90 90 90 90 90-F4 00 00 00 00 00 00 00   ÉÉÉÉÉÉÉÉ⌠.......
0710:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
```

15. Write and execute an assembly language program for two 16-bit subtraction

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
OPR1 DW 1234H
OPR2 DW 5678H
SB DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
MOV DS, AX
MOV AX, OPR1
MOV BX, OPR2
SUB AX, BX
LEA SI, SB
MOV [SI], AX
INT 21H
CODE ENDS
END START
```

Result:-

OPR1 : 1234H

OPR2 : 5678H

SB :

16-bit SUBTRACTION

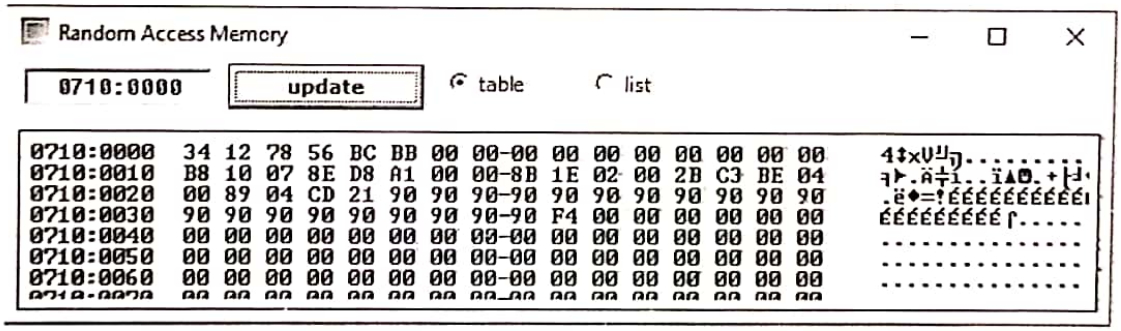
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[08-09-2020 -- 19:32:14]

```
[LINE]        LOC: MACHINE CODE          SOURCE

[  1]         :                          DATA SEGMENT
[  2]         0000: 34 12                OPR1 DW 1234H
[  3]         0002: 78 56                OPR2 DW 5678H
[  4]         0004: 00 00                SB DW ?
[  5]         :                          DATA ENDS
[  6]         :                          CODE SEGMENT
[  7]         :                          ASSUME CS: CODE, DS:DATA
[  8]         0010: B8 00 00             START: MOV AX, DATA
[  9]         0013: 8E D8                MOV DS, AX
[ 10]         0015: A1 00 00             MOV AX, OPR1
[ 11]         0018: 8B 1E 02 00          MOV BX, OPR2
[ 12]         001C: 2B C3                SUB AX, BX
[ 13]         001E: BE 04 00       LEA SI, SB
[ 14]         0021: 89 04                MOV [SI], AX
[ 15]         0023: CD 21                INT 21H
[ 16]         :                          CODE ENDS
[ 17]         :                          END START
[ 18]         :
[ 19]         :
```



16. Write and execute an assembly language program for two 16-bit numbers multiplication

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
OPR 1 DW 1234H
OPR 2 DW 5678H
MLP DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START : MOV AX, DATA
MOV DS, AX
MOV AX, OPR1
MOV BX, OPR2
MUL BX
LEA SI, MLP
MOV [SI], DX
MOV [SI+2], AX
INT 21H
CODE ENDS
END START
```

Result:-

```
OPR 1 : 1234H
OPR 2 : 5678H
MLP   :
```

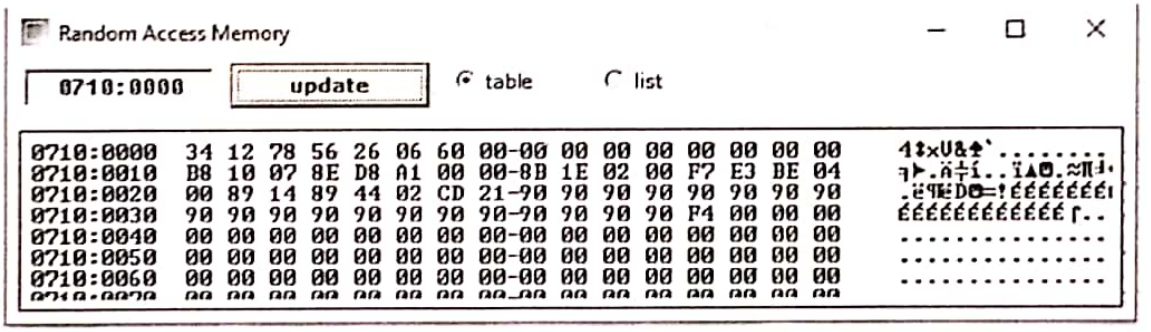
16-bit MULTIPLICATION

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[08-09-2020 -- 19:37:36]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	DATA SEGMENT
[2]	0000: 34 12	OPR1 DW 1234H
[3]	0002: 78 56	OPR2 DW 5678H
[4]	0004: 00 00	MLP DW ?
[5]	:	DATA ENDS
[6]	:	CODE SEGMENT
[7]	:	ASSUME CS: CODE, DS:DATA
[8]	0010: B8 00 00	START: MOV AX, DATA
[9]	0013: 8E D8	MOV DS, AX
[10]	0015: A1 00 00	MOV AX, OPR1
[11]	0018: 8B 1E 02 00	MOV BX, OPR2
[12]	001C: F7 E3	MUL BX
[13]	001E: BE 04 00	LEA SI, MLP
[14]	0021: 89 14	MOV [SI], DX
[15]	0023: 89 44 02	MOV [SI+2],AX
[16]	0026: CD 21	INT 21H
[17]	:	CODE ENDS
[18]	:	END START
[19]	:	
[20]	:	



17. Write and Execute an assembly language program for two 16-bit division

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
OPR1  DW 5673H
OPR2  DW 1278H
RES   DW ?
DATA  ENDS
CODE  SEGMENT
ASSUME CS: CODE, DS: DATA
START : MOV AX, DATA
        MOV DS, AX
        MOV AX, OPR1
        MOV BX, OPR2
        DIV BX
        LEA SI, RES
        MOV [SI], DX
        MOV [SI+2], AX
        INT 21H
CODE ENDS
END START
```

Result:-

OPR1 : 5673H
OPR2 : 1278H

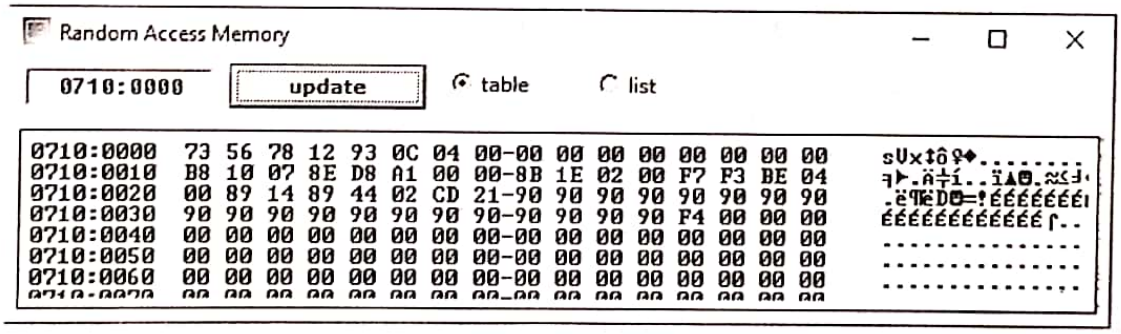
16-bit DIVISION

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[08-09-2020 -- 19:43:24]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	DATA SEGMENT
[2]	0000: 73 56	OPR1 DW 5673H
[3]	0002: 78 12	OPR2 DW 1278H
[4]	0004: 00 00	RES DW ?
[5]	:	DATA ENDS
[6]	:	CODE SEGMENT
[7]	:	ASSUME CS: CODE, DS:DATA
[8]	0010: B8 00 00	START: MOV AX, DATA
[9]	0013: 8E D8	MOV DS, AX
[10]	0015: A1 00 00	MOV AX, OPR1
[11]	0018: 8B 1E 02 00	MOV BX, OPR2
[12]	001C: F7 F3	DIV BX
[13]	001E: BE 04 00	LEA SI, RES
[14]	0021: 89 14	MOV [SI], DX
[15]	0023: 89 44 02	MOV [SI+2], AX
[16]	0026: CD 21	INT 21H
[17]	:	CODE ENDS
[18]	:	END START
[19]	:	
[20]	:	



18. Program of adding two multibyte numbers and store the result as the third.

Software: 8086 emulator

Program Code:-

```
DATA SEGMENT
BYTES EQU 08H
NUM1 DB 05H, 5AH, 6CH, 55H, 66H, 77H, 34H, 12H
NUM2 DB 04H, 56H, 04H, 57H, 32H, 12H, 19H, 15H
NUM3 DB 0AH DUP(00)
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
       MOV DS, AX
       MOV CX, BYTES
       LEA SI, NUM1
       LEA DI, NUM2
       LEA BX, NUM3
       MOV AX, 00
NEXT:  MOV AL, [SI]
       ADC AL, [DI]
       MOV [BX], AL
       INC SI
       INC DI
       INC BX
       DEC CX
       JNZ NEXT
```

```
INT 21 H
CODE ENDS
END START
```

Result:-

NUM 1: 05H, 5AH, 6CH, 55H, 66H, 77H, 34H, 12H

NUM2: 04H, 56H, 04H, 57H, 32H, 12H, 19H, 13H

NUM 3: 09H, B0H, 70H, ACH, 9CH, 89H, 4DH, 25H

ADDITION OF TWO MULTIBYTE NUMBERS

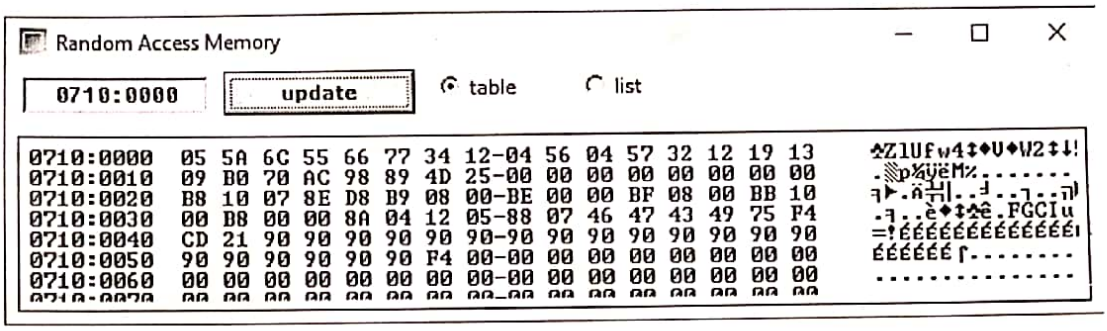
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 22:29:42]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]        :                               DATA SEGMENT
[  2]        :                               BYTES EQU 08H
[  3]    0000: 05 5A 6C 55 66 77 34 12       NUM1 DB 05H, 5AH, 6CH,
55H, 66H, 77H, 34H, 12H
[  4]    0008: 04 56 04 57 32 12 19 13       NUM2 DB 04H, 56H, 04H,
57H, 32H, 12H, 19H, 13H
[  5]    0010: 00 00 00 00 00 00 00 00 00 00 NUM3 DB 0AH DUP (00)
[  6]        :                               DATA ENDS
[  7]        :                               CODE SEGMENT
[  8]        :                               ASSUME CS: CODE, DS: DATA
[  9]    0020: B8 00 00                          START: MOV AX, DATA
[ 10]    0023: 8E D8                             MOV DS, AX
[ 11]    0025: B9 08 00                           MOV CX, BYTES
[ 12]    0028: BE 00 00                           LEA SI, NUM1
[ 13]    002B: BF 08 00                           LEA DI, NUM2
[ 14]    002E: BB 10 00                           LEA BX, NUM3
[ 15]    0031: B8 00 00                           MOV AX, 00
[ 16]    0034: 8A 04                             NEXT: MOV AL, [SI]
[ 17]    0036: 12 05                            ADC AL, [DI]
[ 18]    0038: 88 07                            MOV [BX], AL
[ 19]    003A: 46                              INC SI
[ 20]    003B: 47                              INC DI
[ 21]    003C: 43                              INC BX
[ 22]    003D: 49                              DEC CX
[ 23]    003E: 75 F4                            JNZ NEXT
[ 24]    0040: CD 21                            INT 21H
[ 25]        :                               CODE ENDS
[ 26]        :                               END START
[ 27]        :
[ 28]        :
```



19. Program of subtraction of two multibyte numbers and store the result as third number

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
BYTES EQU 08H
NUM2 DB 05H, 5AH, 6CH, 55H, 66H, 77H, 34H, 12H
NUM1 DB 04H, 56H, 04H, 57H, 32H, 12H, 19H, 13H
NUM3 DB 0AH DUP(00)
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
       MOV DS, AX
       MOV CS, BYTES
       LEA SI, NUM1
       LEA DI, NUM2
       LEA BX, NUM3
       MOV AX, 00
NEXT:  MOV AL, [SI]
       SBB AL, [DI]
       MOV [BX], AL
       INC SI
       INC DI
       INC BX
       DEC CX
       JNZ NEXT
```

```
INT 21H
CODE ENDS
END START.
```

Result:

NUM2: 05H, 5AH, 6CH, 55H, 66H, 77H, 34H, 12H

NUM1: 04H, 56H, 04H, 57H, 32H, 12H, 19H, 13H

NUM3: FFH, FBH, 97H, 01H, CCH, 9AH, E4H, 00H

SUBTRACTION OF TWO MULTIBYTE NUMBERS

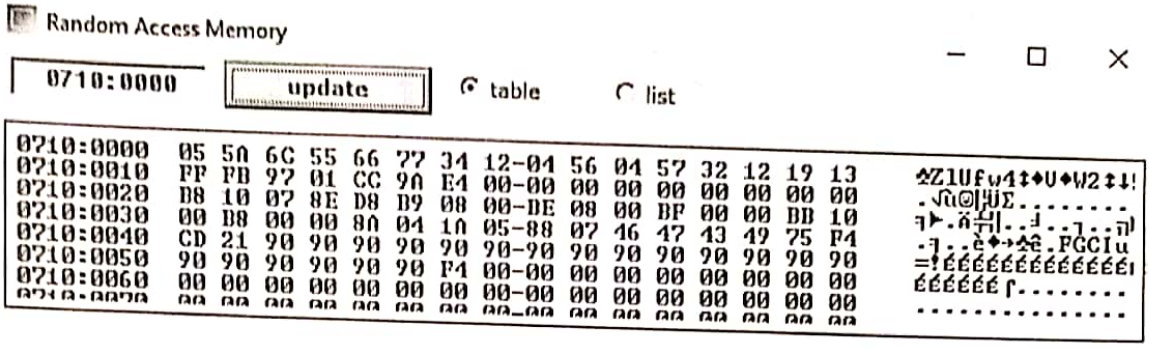
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 22:34:44]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]        :                          DATA SEGMENT
[  2]        :                          BYTES EQU 08H
[  3]   0000: 05 5A 6C 55 66 77 34 12       NUM2 DB 05H, 5AH, 6CH,
55H, 66H, 77H, 34H, 12H
[  4]   0008: 04 56 04 57 32 12 19 13       NUM1 DB 04H, 56H, 04H,
57H, 32H, 12H, 19H, 13H
[  5]   0010: 00 00 00 00 00 00 00 00 00 00    NUM3 DB 0AH DUP (00)
[  6]        :                          DATA ENDS
[  7]        :                          CODE SEGMENT
[  8]        :                          ASSUME CS: CODE, DS: DATA
[  9]   0020: B8 00 00                      START: MOV AX, DATA
[ 10]   0023: 8E D8                         MOV DS, AX
[ 11]   0025: B9 08 00                       MOV CX, BYTES
[ 12]   0028: BE 08 00                       LEA SI, NUM1
[ 13]   002B: BF 00 00                       LEA DI, NUM2
[ 14]   002E: BB 10 00                       LEA BX, NUM3
[ 15]   0031: B8 00 00                      MOV AX, 00
[ 16]   0034: 8A 04                        NEXT: MOV AL, [SI]
[ 17]   0036: 1A 05                        SBB AL, [DI]
[ 18]   0038: 88 07                       MOV [BX], AL
[ 19]   003A: 46                         INC SI
[ 20]   003B: 47                         INC DI
[ 21]   003C: 43                         INC BX
[ 22]   003D: 49                         DEC CX
[ 23]   003E: 75 F4                       JNZ NEXT
[ 24]   0040: CD 21                        INT 21H
[ 25]      :                          CODE ENDS
[ 26]      :                          END START
[ 27]      :
[ 28]      :
```



20. 8086 Assembly Language program: multiplication of two multibyte numbers and store the result as the third number.

Software :- 8086 emulator:

Program Code:-

```
DATA SEGMENT
        BYTES EQU 08H
NUM1 DB 05H, 5AH, 6CH, 55H, 66H, 77H, 34H, 12H
NUM2 DB 04H, 56H, 04H, 57H, 32H, 12H, 19H, 13H
NUM3 DB 0AH DUP(00)
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START : MOV AX, DATA
        MOV DS, AX
        MOV CX, BYTES
        LEA SI, NUM1
        LEA DI, NUM2
        LEA BX, NUM3
        MOV AX, 00
NEXT:   MOV AL, [SI]
        MOV DL, [DI]
        MUL DL
        MOV [BX], AX
        INC SI
        INC DI
```

```
INC BX
DEC CX
JNZ NEXT
INT 21H
CODE ENDS
END START
```

Result:-

NUM1 : 05H, 5AH, 6CH, 55H, 66H, 77H, 34H, 12H

NUM 2 : 04H, 56H, 04H, 57H, 32H, 12H, 19H, 13H

NUM 3 : 14H, 3CH, B0H, E3H, ECH, 5EH, 14H, 56H

MULTIPLICATION OF TWO MULTIBYTE NUMBERS

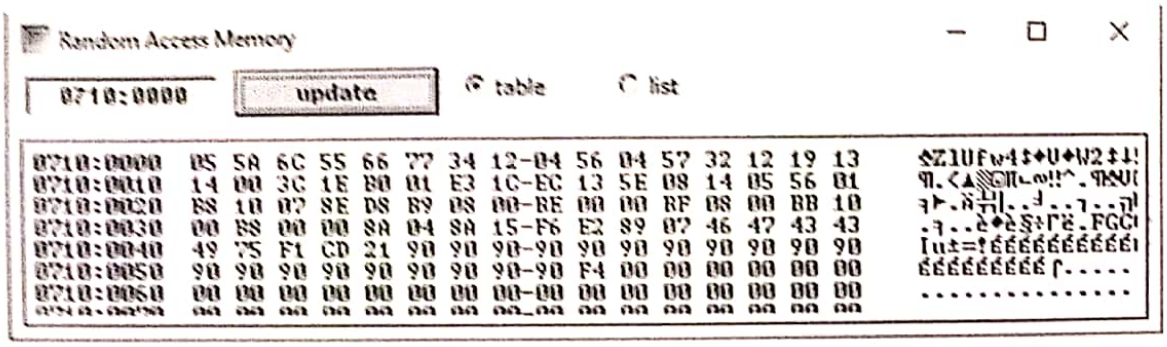
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 22:45:44]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]       :                                DATA SEGMENT
[  2]       :                                BYTES EQU 08H
[  3]    0000: 05 5A 6C 55 66 77 34 12          NUM1 DB 05H, 5AH, 6CH,
55H, 66H, 77H, 34H, 12H
[  4]    0008: 04 56 04 57 32 12 19 13          NUM2 DB 04H, 56H, 04H,
57H, 32H, 12H, 19H, 13H
[  5]    0010: 00 00 00 00 00 00 00 00 00 00     NUM3 DB 0AH DUP (00)
[  6]       :                                DATA ENDS
[  7]       :                                CODE SEGMENT
[  8]       :                                ASSUME CS: CODE, DS: DATA
[  9]    0020: B8 00 00                            START: MOV AX, DATA
[ 10]    0023: 8E D8                               MOV DS, AX
[ 11]    0025: B9 08 00                             MOV CX, BYTES
[ 12]    0028: BE 00 00                             LEA SI, NUM1
[ 13]    002B: BF 08 00                              LEA DI, NUM2
[ 14]    002E: BB 10 00                              LEA BX, NUM3
[ 15]    0031: B8 00 00                             MOV AX, 00
[ 16]    0034: 8A 04                               NEXT: MOV AL, [SI]
[ 17]    0036: 8A 15                               MOV DL, [DI]
[ 18]    0038: F6 E2                               MUL DL
[ 19]    003A: 89 07                               MOV [BX], AX
[ 20]    003C: 46                                 INC SI
[ 21]    003D: 47                                 INC DI
[ 22]    003E: 43                                 INC BX
[ 23]    003F: 43                                 INC BX
[ 24]    0040: 49                                 DEC CX
[ 25]    0041: 75 F1                               JNZ NEXT
[ 26]    0043: CD 21                                INT 21H
[ 27]       :                                   CODE ENDS
[ 28]       :                                   END START
[ 29]       :
[ 30]       :
```



21. 8086 Assembly language program: division of two multibyte numbers and store the result as the third number.

Software:- 8086 emulator

Program Code:-

```
        DATA SEGMENT
    BYTES EQU 08H
NUM1 DB 05H, 08H, 6CH, 55H, 66H, 77H, 34H, 12H
NUM2 DB 04H, 02H, 04H, 57H, 32H, 12H, 19H, 13H
NUM3 DB 0AX DUP(00)
        DATA ENDS
    CODE SEGMENT
        ASSUME CS: CODE, DS: DATA
        START: MOV AX, DATA
               MOV DS, AX
               MOV CX, BYTES
               LEA SI, NUM1
               LEA DI, NUM2
               LEA BX, NUM3
        NEXT: MOV AX, 00
              MOV AL, [SI]
              MOV DL, [DI]
              DIV DL
              MOV [BX], AX
              INC SI
```

```
        INC DI
        INC BX
        INC BX
        DEC CX
      JNZ NEXT
        INT 21H
      CODE ENDS
      END START.
```

Result:-

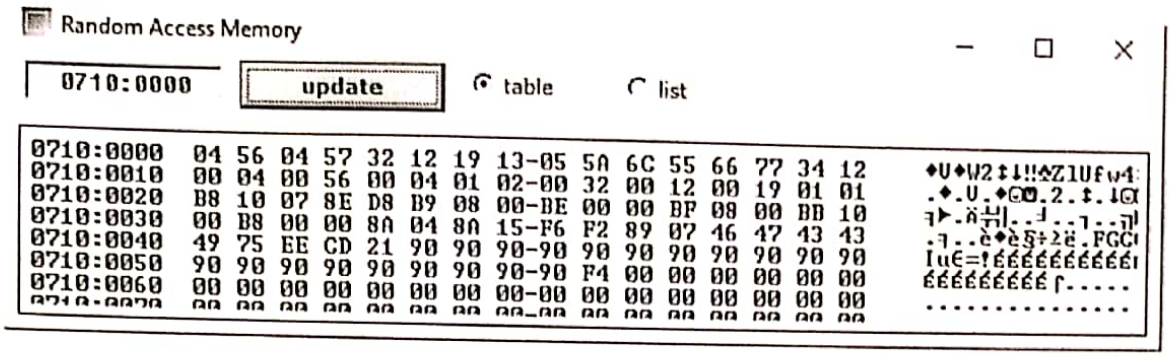
NUM2: 05H, 5AH, 6CH, 55H, 66H, 77H, 34H, 12H
NUM1: 04H, 56H, 04H, 57H, 32H, 12H, 19H, 18H
NUM3: 00H, 04H, 00H, 56H, 00H, 04H, 01H, 02H

DIVISION OF TWO MULTIBYTE NUMBERS

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.
noname.exe_ -- emu8086 assembler version: 4.08
[09-10-2020 -- 22:50:47]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]      :                                 DATA SEGMENT
[  2]      :                                 BYTES EQU 08H
[  3]    0000: 04 56 04 57 32 12 19 13       NUM1 DB 04H, 56H, 04H,
57H, 32H, 12H, 19H, 13H
[  4]    0008: 05 5A 6C 55 66 77 34 12       NUM2 DB 05H, 5AH, 6CH,
55H, 66H, 77H, 34H, 12H
[  5]    0010: 00 00 00 00 00 00 00 00 00 00  NUM3 DB 0AH DUP (00)
[  6]      :                                 DATA ENDS
[  7]      :                                 CODE SEGMENT
[  8]      :                                 ASSUME CS: CODE, DS: DATA
[  9]    0020: B8 00 00                          START: MOV AX, DATA
[ 10]    0023: 8E D8                             MOV DS, AX
[ 11]    0025: B9 08 00                           MOV CX, BYTES
[ 12]    0028: BE 00 00                           LEA SI, NUM1
[ 13]    002B: BF 08 00                           LEA DI, NUM2
[ 14]    002E: BB 10 00                           LEA BX, NUM3
[ 15]    0031: B8 00 00                          NEXT: MOV AX, 00
[ 16]    0034: 8A 04                            MOV AL, [SI]
[ 17]    0036: 8A 15                            MOV DL, [DI]
[ 18]    0038: F6 F2                            DIV DL
[ 19]    003A: 89 07                            MOV [BX], AX
[ 20]    003C: 46                              INC SI
[ 21]    003D: 47                              INC DI
[ 22]    003E: 43                              INC BX
[ 23]    003F: 43                              INC BX
[ 24]    0040: 49                              DEC CX
[ 25]    0041: 75 EE                             JNZ NEXT
[ 26]    0043: CD 21                             INT 21H
[ 27]      :                                 CODE ENDS
[ 28]      :                                 END START
[ 29]      :
[ 30]      :
```



22) 8086 Assembly language program: Converting packed BCD to unpacked BCD

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
    NUM DB 45H
DATA ENDS
CODE SEGMENT
  ASSUME CS:CODE,DS:DATA
   START: MOV AX, DATA
          MOV DS, AX
          MOV AL, NUM
          MOV AH, AL
          MOV CL, 4
          SHR AH, CL
          AND AX, 0F0FH
          INT 21H
  CODE ENDS
  END START
```

Result:-

Input: 45H: NUM [packed BCD]

Output: AX: 0405H [unpacked BCD]

PACKED TO UNPACKED BCD

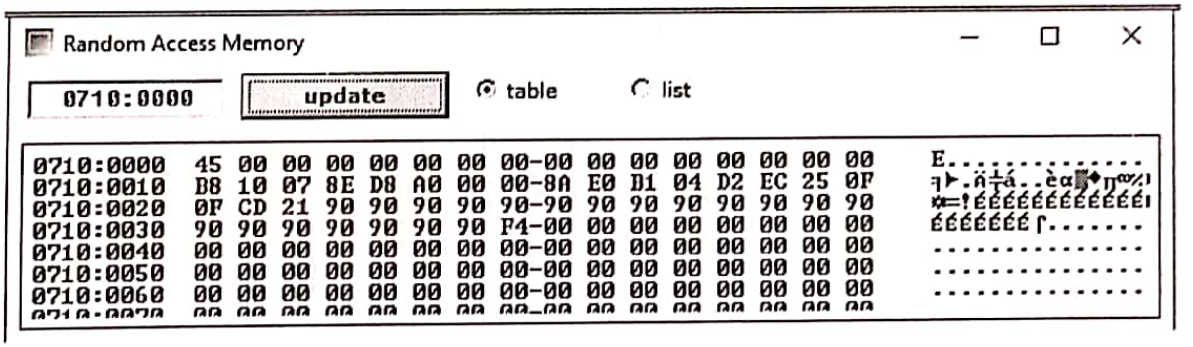
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 22:55:14]

```
[LINE]    LOC: MACHINE CODE                SOURCE

[  1]      :                               DATA SEGMENT
[  2]    0000: 45                            NUM DB 45H
[  3]      :                               DATA ENDS
[  4]      :                               CODE SEGMENT
[  5]      :                               ASSUME CS: CODE, DS: DATA
[  6]    0010: B8 00 00                        START: MOV AX, DATA
[  7]    0013: 8E D8                          MOV DS, AX
[  8]    0015: A0 00 00                        MOV AL, NUM
[  9]    0018: 8A E0                          MOV AH, AL
[ 10]    001A: B1 04                           MOV CL, 4
[ 11]    001C: D2 EC                           SHR AH, CL
[ 12]    001E: 25 0F 0F                        AND AX, 0F0FH
[ 13]    0021: CD 21                          INT 21H
[ 14]      :                               CODE ENDS
[ 15]      :                               END START
[ 16]      :
[ 17]      :
```



23) 8086 Assembly language program:- Converting BCD to ASCII

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
NUM DB 45H
RES DW ?
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
       MOV DS, AX
       MOV AL, NUM
       MOV AH, AL
       MOV CL, 4
       SHR AH, CL
       AND AX, 0F0FH
       OR AX, 3030H
       MOV RES, AX
       MOV AH, 4CH
       INT 21H
CODE ENDS
END START
```

Result:-

Input:- NUM: 45H [BCD]

output: RES : 3435H [ASCII]

BCD TO ASCII

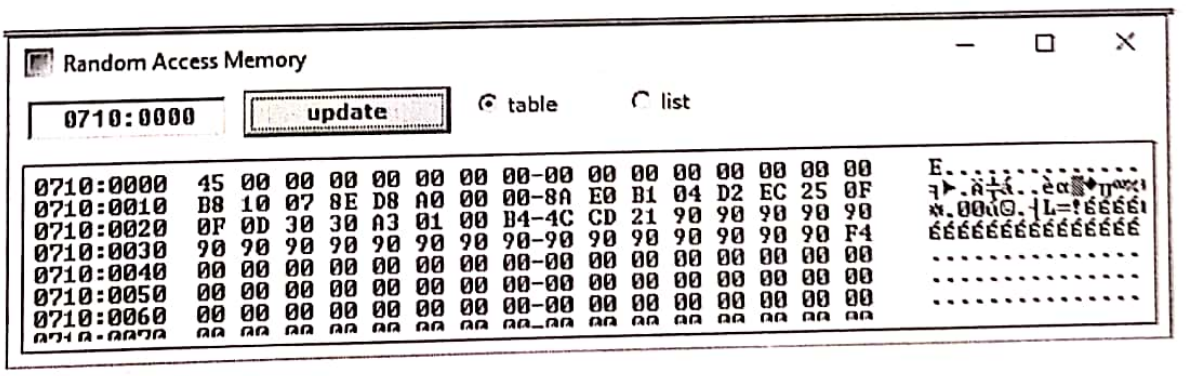
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 22:58:18]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]      :                                 DATA SEGMENT
[  2]    0000: 45                              NUM DB 45H
[  3]    0001: 00 00                           RES DW ?
[  4]      :                                 DATA ENDS
[  5]      :                                 CODE SEGMENT
[  6]      :                                 ASSUME CS: CODE, DS: DATA
[  7]    0010: B8 00 00                          START: MOV AX, DATA
[  8]    0013: 8E D8                           MOV DS, AX
[  9]    0015: A0 00 00                         MOV AL, NUM
[ 10]    0018: 8A E0                           MOV AH, AL
[ 11]    001A: B1 04                           MOV CL, 4
[ 12]    001C: D2 EC                            SHR AH, CL
[ 13]    001E: 25 0F 0F                         AND AX, 0F0FH
[ 14]    0021: 0D 30 30                         OR AX, 3030H
[ 15]    0024: A3 01 00                         MOV RES, AX
[ 16]    0027: B4 4C                           MOV AH,4CH
[ 17]    0029: CD 21                           INT 21H
[ 18]      :                                 CODE ENDS
[ 19]      :                                 END START
[ 20]      :
[ 21]      :
```



24). 8086 Assembly Language program: Find out the number of Even and Odd numbers from a given series of 16-bit hexadecimal numbers

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
N1 DW 2356H, 5749H, 8633H, 7655H, 9266H
COUNT EQU 05H
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
       MOV DS, AX
       XOR AX, AX
       MOV DX, 0000H
       MOV BX, 0000H
       MOV CL, COUNT
       MOV SI, OFFSET N1
AGAIN: MOV [AX], [SI]
       ROR AX, 01H
       JC ODD
       INC BX
       JMP NEXT
ODD:   INC DX
NEXT:  ADD SI, 02H
       DEC CL
       JNZ AGAIN
```

```
MOV AH, 4CH
INT 21H
CODE ENDS
END START.
```

Result:-

Input – N_1 : 2356H, 5749H, 8633H, 7855H, 9266H

output – BX : 0002H (even count)

DX : 0003H (odd count)

FINDING EVEN AND ODD NUMBERS IN 16-BIT HEXADECIMAL NUMBERS

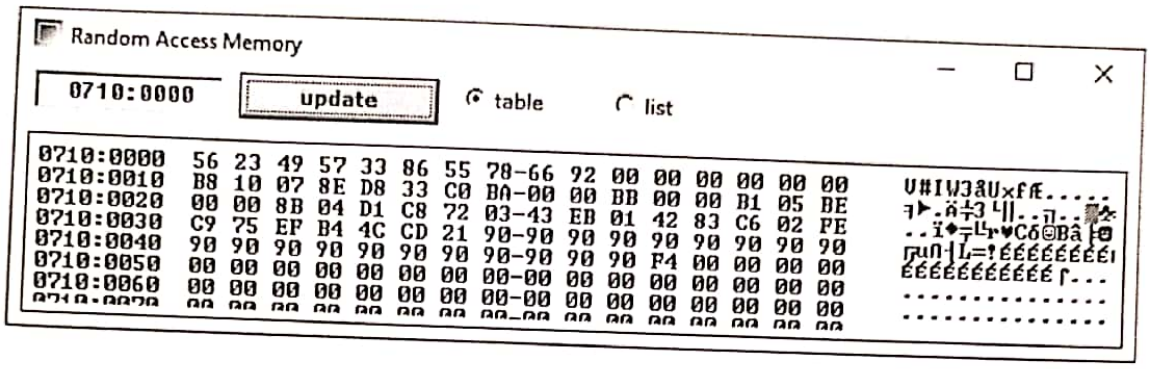
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:02:19]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]      :                                 DATA SEGMENT
[  2]    0000: 56 23 49 57 33 86 55 78 66 92      N1 DW 2356H, 5749H,
8633H, 7855H, 9266H
[  3]      :                                 COUNT EQU 05H
[  4]      :                                 DATA ENDS
[  5]      :                                 CODE SEGMENT
[  6]      :                                 ASSUME CS: CODE, DS:DATA
[  7]    0010: B8 00 00                          START: MOV AX, DATA
[  8]    0013: 8E D8                           MOV DS, AX
[  9]    0015: 33 C0                           XOR AX, AX
[ 10]    0017: BA 00 00                          MOV DX, 0000H
[ 11]    001A: BB 00 00                          MOV BX, 0000H
[ 12]    001D: B1 05                           MOV CL, COUNT
[ 13]    001F: BE 00 00                          MOV SI, OFFSET N1
[ 14]    0022: 8B 04                           AGAIN: MOV AX, [SI]
[ 15]    0024: D1 C8                           ROR AX, 01H
[ 16]    0026: 72 03                         JC ODD
[ 17]    0028: 43                          INC BX
[ 18]    0029: EB 01                           JMP NEXT
[ 19]    002B: 42                          ODD: INC DX
[ 20]    002C: 83 C6 02                          NEXT: ADD SI, 02H
[ 21]    002F: FE C9                           DEC CL
[ 22]    0031: 75 EF                         JNZ AGAIN
[ 23]    0033: B4 4C                           MOV AH, 4CH
[ 24]    0035: CD 21                           INT 21H
[ 25]      :                                 CODE ENDS
[ 26]      :                                 END START
[ 27]      :
[ 28]      :
```



25. 8086 Assembly Language Program: find out the number of positive and negative numbers from a given series of signed numbers.

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
N1 DW 2556H, 3749H, 5633H, 6755H, 9986H
COUNT EQU 05H
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START: MOV AX, DATA
       MOV DS, AX
       XOR AX, AX
       MOV DX, 0000H
       MOV BX, 0000H
       MOV CX, COUNT
       MOV SI, OFFSET N1
AGAIN: MOV AX, [SI]
       ROL AX, 01H
       JC NEG
       INC BX
       JMP NEXT
NEG:   INC DX
NEXT:  ADD SI, 02H
       DEC CX
```

```
    JNZ AGAIN
    MOV AH, 4CH
    INT 21H
CODE ENDS
    END START
```

Result:-

N1: 56H, 49H, 33H, 55H, 86H

Bx: 0002H [Positive count]

Dx: 0003H [Negative count]

FINDING POSITIVE AND NEGATIVE NUMBERS IN GIVEN 16-BIT HEXADECIMAL NUMBERS

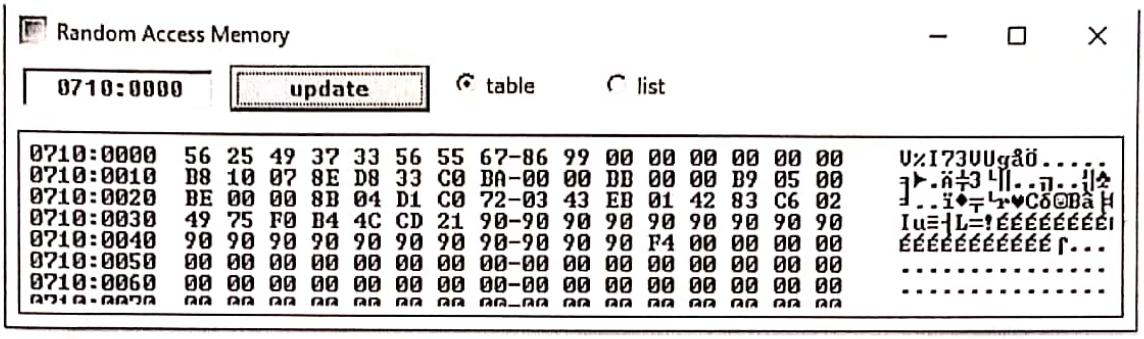
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:05:26]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]      :                                 DATA SEGMENT
[  2]    0000: 56 25 49 37 33 56 55 67 86 99       N1 DW 2556H, 3749H,
5633H, 6755H, 9986H
[  3]      :                                 COUNT EQU 05H
[  4]      :                                 DATA ENDS
[  5]      :                                 CODE SEGMENT
[  6]      :                                 ASSUME CS: CODE, DS:DATA
[  7]    0010: B8 00 00                          START: MOV AX, DATA
[  8]    0013: 8E D8                           MOV DS, AX
[  9]    0015: 33 C0                           XOR AX, AX
[ 10]    0017: BA 00 00                          MOV DX, 0000H
[ 11]    001A: BB 00 00                          MOV BX, 0000H
[ 12]    001D: B9 05 00                          MOV CX, COUNT
[ 13]    0020: BE 00 00                          MOV SI, OFFSET N1
[ 14]    0023: 8B 04                           AGAIN: MOV AX, [SI]
[ 15]    0025: D1 C0                           ROL AX, 01H
[ 16]    0027: 72 03                          JC NEG
[ 17]    0029: 43                            INC BX
[ 18]    002A: EB 01                           JMP NEXT
[ 19]    002C: 42                           NEG: INC DX
[ 20]    002D: 83 C6 02                          NEXT: ADD SI,02H
[ 21]    0030: 49                            DEC CX
[ 22]    0031: 75 F0                          JNZ  AGAIN
[ 23]    0033: B4 4C                           MOV AH, 4CH
[ 24]    0035: CD 21                           INT 21H
[ 25]      :                                 CODE ENDS
[ 26]      :                                 END START
[ 27]      :
[ 28]      :
```



26. Write a program to perform a one-byte BCD Addition

Software:- 8086 emulator

Program Code:-

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
        OPR1 EQU 92H
        OPR2 EQU 52H
      RESULT DB 02 DUP(00)
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
       MOV DS, AX
       MOV BL, OPR1
       XOR AL, AL
       MOV AL, OPR2
       ADD AL, BL
       DAA
       MOV RESULT, AL
       JNC MSB0
       INC [RESULT+1]
MSB0: MOV AH, 4C
      INT 21H
CODE ENDS
END START
```

Result:- Input: OPR1 - 92H
OPR2 - 52H
Output:- RESULT: 0144H

ONE BYTE BCD ADDITION

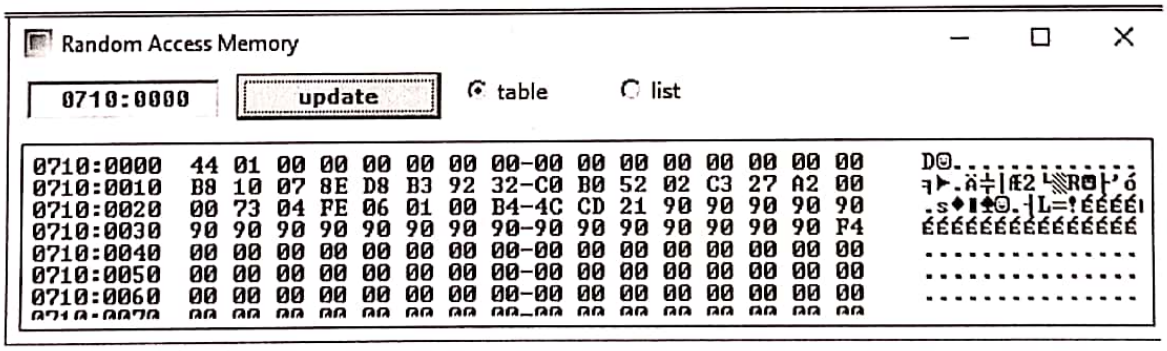
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:09:40]

```
[LINE]    LOC: MACHINE CODE                SOURCE

[  1]       :                              ASSUME CS: CODE, DS:DATA
[  2]       :                              DATA SEGMENT
[  3]       :                              OPR1 EQU 92H
[  4]       :                              OPR2 EQU 52H
[  5]    0000: 00 00                           RESULT DB 02 DUP (00)
[  6]       :                              DATA ENDS
[  7]       :                              CODE SEGMENT
[  8]    0010: B8 00 00                        START:  MOV AX, DATA
[  9]    0013: 8E D8                                   MOV DS, AX
[ 10]    0015: B3 92                           MOV BL, OPR1
[ 11]    0017: 32 C0                           XOR AL, AL
[ 12]    0019: B0 52                           MOV AL, OPR2
[ 13]    001B: 02 C3                           ADD AL, BL
[ 14]    001D: 27                             DAA
[ 15]    001E: A2 00 00                          MOV RESULT, AL
[ 16]    0021: 73 04                          JNC MSB0
[ 17]    0023: FE 06 01 00                       INC [RESULT+1]
[ 18]    0027: B4 4C                          MSB0:MOV AH,4CH
[ 19]    0029: CD 21                          INT 21H
[ 20]       :                              CODE ENDS
[ 21]       :                              END START
[ 22]       :
[ 23]       :
```



27) Find square root of a two-digit number. Assume that number is a perfect square

Software:- 8086 emulator

Program Code:-

```
ASSUME CS: CODE, DS: DATA
DATA SEGMENT
    NUM EQU 36
    RESULT DB (?)
DATA ENDS
CODE SEGMENT
START: MOV AX, DATA
       MOV DS, AX
       MOV CL, NUM
       MOV BL, 1
       MOV AL, 0
UP:    CMP CL, 0
       ZRESULT
       SUB CL, BL
       INC AL
       ADD BL, 02
       JMP UP
ZRESULT: MOV RESULT, AL
       MOV AH, 4CH
       INT 21H
CODE ENDS
END START
```

Result:-

Input - NUM: 36

Output - RESULT : 06

(28) Program to arrange a given series of hexadecimal bytes in Ascending Order.

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT.
N1 DW 56H, 49H, 33H, 05H, 12H, 17H
COUNT EQU 06H
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
       MOV DS, AX
       XOR AX, AX
       MOV DX, COUNT-1
AGAIN0: MOV CX, DX
        MOV SI, OFFSET N1
AGAIN1: MOV AX, [SI]
        CMP AX, [SI+2]
        JL PR1
        XCHG [SI+2], AX
        XCHG [SI], AX
PR1:    ADD SI, 02
        LOOP AGAIN1
        DEC DX
        JNZ AGAIN0
        MOV AH, 4CH
        INT 21H
CODE ENDS
END START
```

Result:-

Given series N1: 56 H, 49H, 33H, 05H, 12H, 17 H

Ascending order : 05H, 12H, 17H, 33H, 49H, 56H

ARRANGING ELEMENTS IN ASCENDING ORDER

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

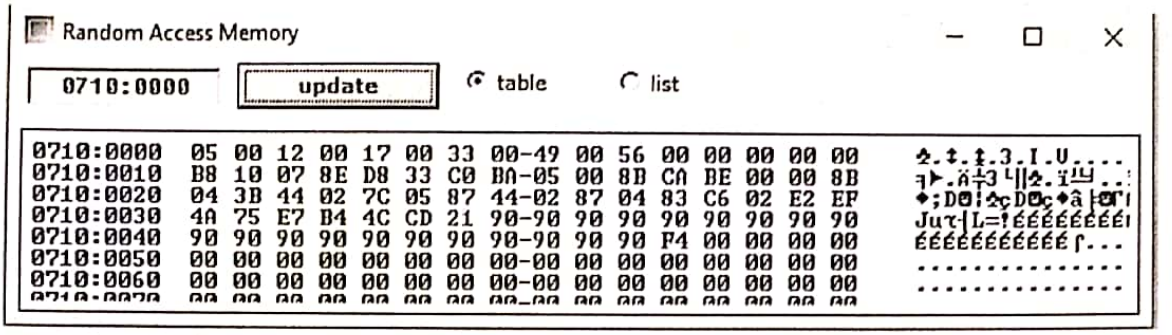
noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:22:47]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]      :                                 DATA SEGMENT
[  2]   0000: 56 00 49 00 33 00 05 00 12 00 17 00   N1 DW
56H,49H,33H,05H,12H,17H

[  3]      :                                 COUNT EQU 06H
[  4]      :                                 DATA ENDS
[  5]      :                                 CODE SEGMENT
[  6]      :                                 ASSUME CS: CODE, DS:DATA
[  7]   0010: B8 00 00                          START: MOV AX, DATA
[  8]   0013: 8E D8                            MOV DS, AX
[  9]   0015: 33 C0                            XOR AX, AX
[ 10]   0017: BA 05 00                           MOV DX, COUNT-1
[ 11]   001A: 8B CA                              AGAIN0: MOV CX, DX
[ 12]   001C: BE 00 00                           MOV SI, OFFSET N1
[ 13]   001F: 8B 04                             AGAIN1: MOV AX, [SI]
[ 14]   0021: 3B 44 02                           CMP AX, [SI+2]
[ 15]   0024: 7C 05                             JL PR1
[ 16]   0026: 87 44 02                           XCHG [SI+2], AX
[ 17]   0029: 87 04                             XCHG [SI], AX
[ 18]   002B: 83 C6 02                           PR1: ADD SI, 02
[ 19]   002E: E2 EF                             LOOP AGAIN1
[ 20]   0030: 4A                               DEC  DX
[ 21]   0031: 75 E7                             JNZ AGAIN0
[ 22]   0033: B4 4C                             MOV AH, 4CH
[ 23]   0035: CD 21                             INT 21H
[ 24]      :                                 CODE ENDS
[ 25]      :                                 END START
[ 26]      :
[ 27]      :
```



29, Program to arrange a given series of hexadecimal bytes in Descending order.

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
N1 DW 56H, 49H, 33H, 05H, 12H, 17H
COUNT EQU 06H
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
  START : MOV AX, DATA
          MOV DS, AX
          XOR, AX, AX
          MOV DX, COUNT-1
  AGAIN0: MOV CX, DX
          MOV SI, OFFSET N1
  AGAIN1: MOV AX, [SI]
          CMP AX, [SI+2]
          JG PR1
          XCHG [SI+2], AX
          XCHG [SI], AX
  PR1:    ADD SI, 02
          LOOP AGAIN1
          DEC DX
          JNZ AGAIN0
          MOV AH, 4CH
          INT 21H
CODE ENDS
END START
```

Result:- Given Series N1: 56H, 49H, 33H, 05H, 12H, 17H

Descending Order: 56H, 49H, 33H, 17H, 12H, 05H

ARRANGING ELEMENTS IN DESCENDING ORDER

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

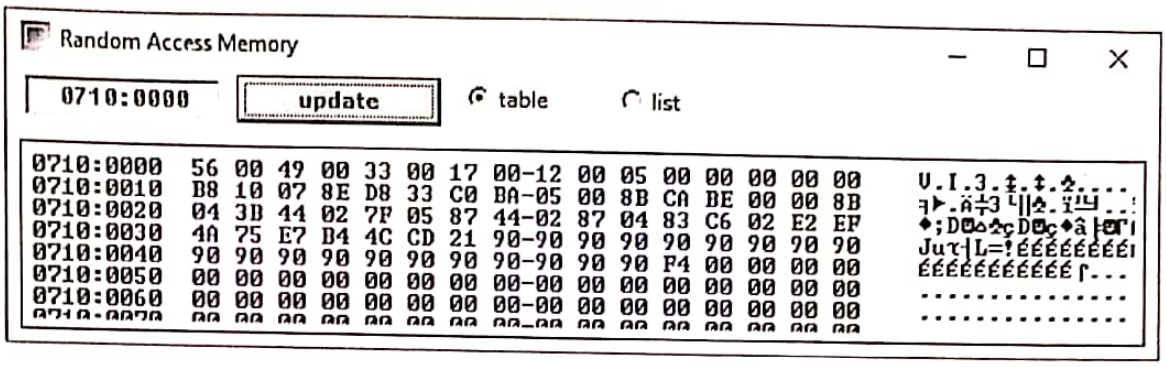
noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:25:58]

```
[LINE]    LOC: MACHINE CODE                SOURCE

[  1]      :                               DATA SEGMENT
[  2]    0000: 56 00 49 00 33 00 05 00 12 00 17 00   N1 DW
56H,49H,33H,05H,12H,17H

[  3]      :                               COUNT EQU 06H
[  4]      :                               DATA ENDS
[  5]      :                               CODE SEGMENT
[  6]      :                               ASSUME CS: CODE, DS:DATA
[  7]    0010: B8 00 00                        START: MOV AX, DATA
[  8]    0013: 8E D8                          MOV DS, AX
[  9]    0015: 33 C0                          XOR AX, AX
[ 10]    0017: BA 05 00                         MOV DX, COUNT-1
[ 11]    001A: 8B CA                            AGAIN0: MOV CX, DX
[ 12]    001C: BE 00 00                         MOV SI, OFFSET N1
[ 13]    001F: 8B 04                           AGAIN1: MOV AX, [SI]
[ 14]    0021: 3B 44 02                         CMP AX, [SI+2]
[ 15]    0024: 7F 05                          JG PR1
[ 16]    0026: 87 44 02                        XCHG [SI+2], AX
[ 17]    0029: 87 04                          XCHG [SI], AX
[ 18]    002B: 83 C6 02                         PR1: ADD SI, 02
[ 19]    002E: E2 EF                          LOOP AGAIN1
[ 20]    0030: 4A                            DEC  DX
[ 21]    0031: 75 E7                          JNZ AGAIN0
[ 22]    0033: B4 4C                          MOV AH, 4CH
[ 23]    0035: CD 21                          INT 21H
[ 24]      :                               CODE ENDS
[ 25]      :                               END START
[ 26]      :
[ 27]      :
```



30. Program of moving a block using strings.

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
  SRC DB 'MICROPROCESSOR'
  DB 18 DUP ?
  DST DB 20 DUP (0)
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA, ES:DATA
START: MOV AX, DATA
       MOV DS, AX
       MOV ES, AX
       LEA SI, SRC
       LEA DI, DST
       MOV CX, 20
       CLD
       REP MOVSB
       MOV AH, 4CH
       INT 21H
CODE ENDS
END START
```

Result:-

0710:0000 — MICROPROCESSOR
0710:0020 — MICROPROCESSOR

MOVING A BLOCK USING STRINGS

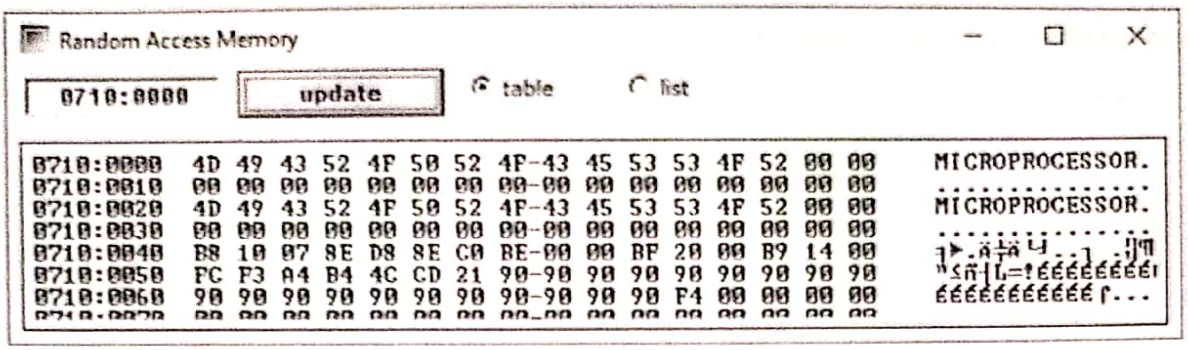
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:28:34]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]      :                                 DATA SEGMENT
[  2]   0000: 4D 49 43 52 4F 50 52 4F 43 45 53 53   SRC DB
'MICROPROCESSOR'
            4F 52
[  3]   000E: 00 00 00 00 00 00 00 00 00 00 00 00   DB 18 DUP (?)
            00 00 00 00 00 00
[  4]   0020: 00 00 00 00 00 00 00 00 00 00 00 00   DST DB 20 DUP (0)
            00 00 00 00 00 00 00 00
[  5]      :                                 DATA ENDS
[  6]      :                                 CODE SEGMENT
[  7]      :                                 ASSUME CS: CODE, DS: DATA, ES:
DATA
[  8]   0040: B8 00 00                          START: MOV AX, DATA
[  9]   0043: 8E D8                            MOV DS, AX
[ 10]   0045: 8E C0                            MOV ES, AX
[ 11]   0047: BE 00 00                          LEA SI, SRC
[ 12]   004A: BF 20 00                          LEA DI, DST
[ 13]   004D: B9 14 00                          MOV CX, 20
[ 14]   0050: FC                               CLD
[ 15]   0051: F3 A4                             REP MOVSB
[ 16]   0053: B4 4C                             MOV AH,4CH
[ 17]   0055: CD 21                             INT 21H
[ 18]      :                                 CODE ENDS
[ 19]      :                                 END START
[ 20]      :
[ 21]      :
```



3). 8086 Assembly Language program: String reversal

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
SRC DB 'MICROPROCESSOR'
COUNT EQU ($-SRC)
DEST DB 20 DUP ?
DATA ENDS
CODE SEGMENT
    ASSUME CS:CODE, DS:DATA
      START: MOV AX, DATA
             MOV DS, AX
             MOV CX, COUNT
             LEA SI, SRC
             LEA DI, DEST
             ADD SI, CX
             INC CX
      BACK:  MOV AL, [SI]
             MOV [DI], AL
               DEC SI
               INC DI
               DEC CX
             JNZ BACK
             MOV AH, 4CH
              INT 21H
          CODE ENDS
          END START
```

Result:-

Given string: MICROPROCESSOR$

Reverse of string: $ROSSECORPORCIM

REVERSING A STRING

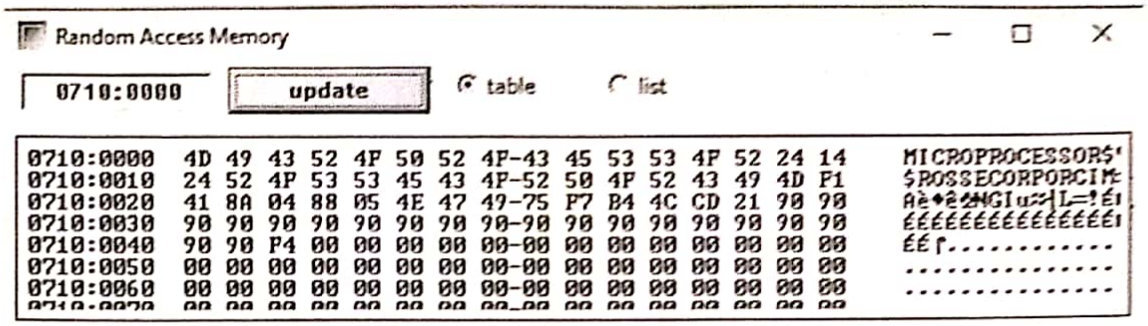
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:31:54]

```
[LINE]    LOC: MACHINE CODE                  SOURCE

[  1]       :                                DATA SEGMENT
[  2]    0000: 4D 49 43 52 4F 50 52 4F 43 45 53 53   SRC DB
'MICROPROCESSOR$'
             4F 52 24
[  3]    000F:                                 COUNT EQU ($-SRC)
[  4]    000F: 14                               DEST DB 20 DUP ?
[  5]       :                                DATA ENDS
[  6]       :                                CODE SEGMENT
[  7]       :                                ASSUME CS: CODE, DS: DATA
[  8]    0010: B8 00 00                          START: MOV AX, DATA
[  9]    0013: 8E D8                            MOV DS, AX
[ 10]    0015: B9 0F 00                          MOV CX, COUNT
[ 11]    0018: BE 00 00                          LEA SI, SRC
[ 12]    001B: BF 0F 00                          LEA DI, DEST
[ 13]    001E: 03 F1                            ADD SI, CX
[ 14]    0020: 41                              INC CX
[ 15]    0021: 8A 04                            BACK: MOV AL, [SI]
[ 16]    0023: 88 05                           MOV [DI], AL
[ 17]    0025: 4E                              DEC SI
[ 18]    0026: 47                              INC DI
[ 19]    0027: 49                              DEC CX
[ 20]    0028: 75 F7                            JNZ BACK
[ 21]    002A: B4 4C                             MOV AH,4CH
[ 22]    002C: CD 21                             INT 21H
[ 23]       :                                CODE ENDS
[ 24]       :                                END START
[ 25]       :
[ 26]       :
```



32, 8086 Assembly Language program: Comparison of two strings.

Program Software:- 8086 emulator

Program Code:-

```
PRINTSTRING MACRO MSG
        MOV AH,09H
        LEA DX, MSG
        INT 21H
        ENDM
DATA SEGMENT
    STR1 DB 'MICROPROCESSOR'
    LEN  EQU ($-STR1)
    STR2 DB 'MICROPROCESSOR'
    M1   DB "STRINGS ARE EQUAL$"
    M2   DB "STRINGS ARE NOT EQUAL$"
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA, ES: DATA
START: MOV AX, DATA
       MOV DS, AX
       MOV ES, AX
       LEA SI, STR1
       LEA DI, STR2
       MOV CX, LEN
       CLD
       REPE CMPSB
       JNE FAIL
       PRINTSTRING M1
```

```
        MOV AH, 4CH
        INT 21H
FAIL:   PRINTSTRING M2
        MOV AH, 4CH
        INT 21H
CODE    ENDS
END     START
```

Result:-

Input:- STR1: MICROPROCESSOR
STR2: MICROPROCESSOR

output: STRINGS ARE EQUAL

COMPARISON OF TWO STRINGS

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

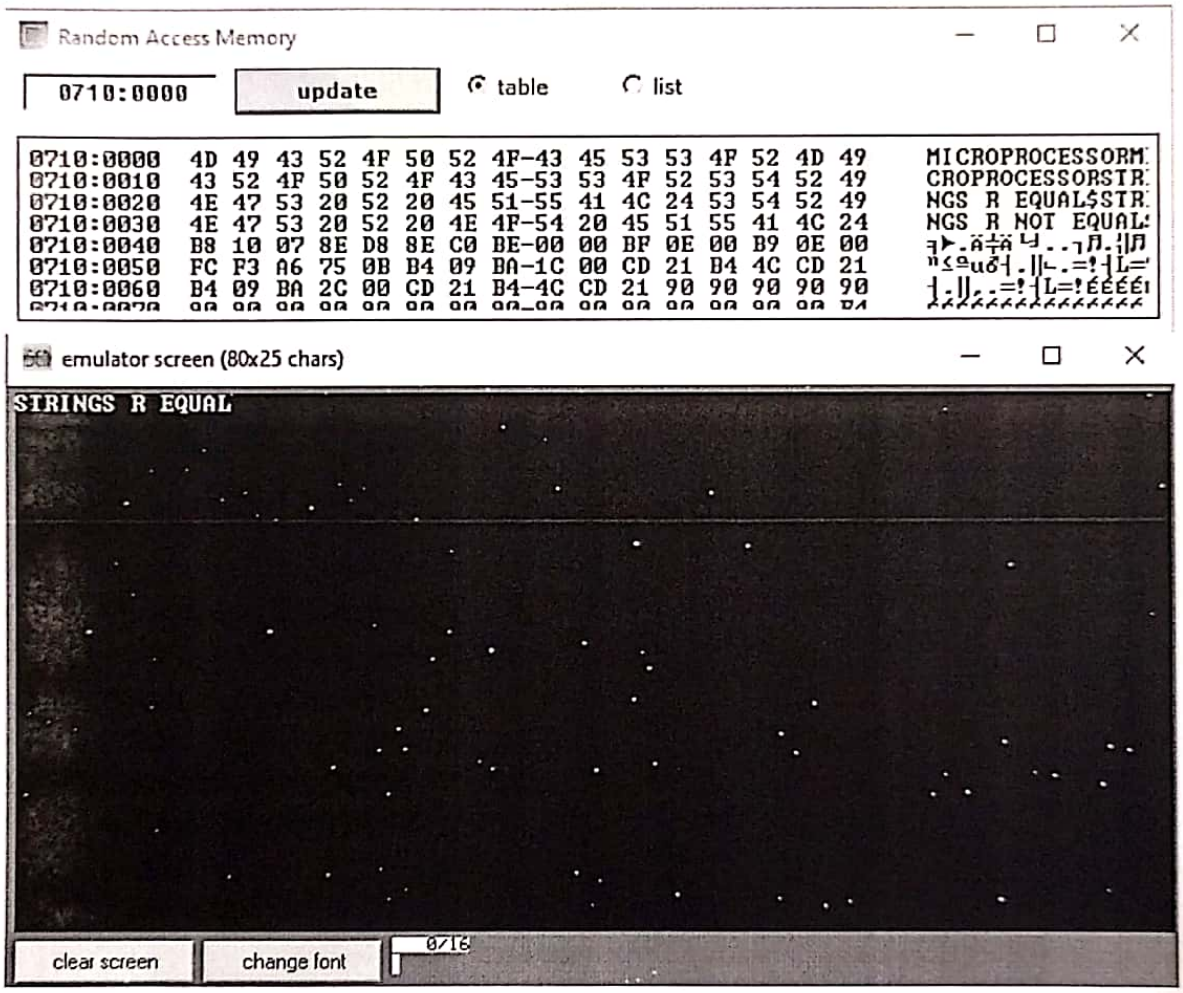
noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:35:59]

```
[LINE]    LOC: MACHINE CODE                    SOURCE

[  1]      :                                   PRINTSTRING MACRO MSG
[  2]      :                                   MOV AH, 09H
[  3]      :                                   LEA DX, MSG
[  4]      :                                   INT 21H
[  5]      :                                   ENDM
[  6]      :                                   DATA SEGMENT
[  7]   0000: 4D 49 43 52 4F 50 52 4F 43 45 53 53   STR1 DB
'MICROPROCESSOR'
           4F 52
[  8]   000E:                                    LEN EQU ($-STR1)
[  9]   000E: 4D 49 43 52 4F 50 52 4F 43 45 53 53   STR2 DB
'MICROPROCESSOR'
           4F 52
[ 10]   001C: 53 54 52 49 4E 47 53 20 52 20 45 51   M1 DB "STRINGS R
EQUAL$"
           55 41 4C 24
[ 11]   002C: 53 54 52 49 4E 47 53 20 52 20 4E 4F   M2 DB "STRINGS R
NOT EQUAL$"
           54 20 45 51 55 41 4C 24
[ 12]      :                                   DATA ENDS
[ 13]      :                                   CODE SEGMENT
[ 14]      :                                   ASSUME CS: CODE, DS: DATA, ES:
DATA
[ 15]   0040: B8 00 00                          START: MOV AX, DATA
[ 16]   0043: 8E D8                             MOV DS, AX
[ 17]   0045: 8E C0                             MOV ES, AX
[ 18]   0047: BE 00 00                           LEA SI, STR1
[ 19]   004A: BF 0E 00                           LEA DI, STR2
[ 20]   004D: B9 0E 00                           MOV CX, LEN
[ 21]   0050: FC                               CLD
```

```
[ 22]   0051: F3 A6                     REPE CMPSB
[ 23]   0053: 75 0B                     JNE FAIL
[ 24]   0055: B4 09 BA 1C 00 CD 21              PRINTSTRING M1
[ 25]   005C: B4 4C                     MOV AH,4CH
[ 26]   005E: CD 21                     INT 21H
[ 27]   0060: B4 09 BA 2C 00 CD 21              FAIL: PRINTSTRING M2
[ 28]   0067: B4 4C                     MOV AH,4CH
[ 29]   0069: CD 21                     INT 21H
[ 30]      :                         CODE ENDS
[ 31]      :                         END START
[ 32]      :
[ 33]      :
```



(33) Program to find median of given array of element

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
  ARRAY DB 99H, 35H, 56H, 02H, 63H, 21H
  COUNT EQU 06H
  MEDIAN DB ?
DATA ENDS
CODE SEGMENT.
ASSUME CS: CODE, DS: DATA
  START: MOV AX, DATA
         MOV DS, AX
         SUB AX, AX
        MOV SI, OFFSET ARRAY
        MOV DI, OFFSET MEDIAN
         MOV AL, COUNT
         MOV BL, 02H
         DIV BL.
        CMP AH, 00H
    JE EVEN
          DEC AH
          ADD SI, AX
          MOV AL, [SI]
          MOV [DI], AX
    JMP EXIT
EVEN: ADD SI, AX
      MOV AL, [SI]
      MOV AH, [SI-1]
```

```
        MOV [DI], AX
EXIT:   MOV AH, 4CH
        INT 21H
CODE    ENDS
        END START
```

Result:-

Input:- 99H, 35H, 56H, 02H, 63H, 21H

output:- 02H, 56H.

MEDIAN OF GIVEN ARRAY OF NUMBERS

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

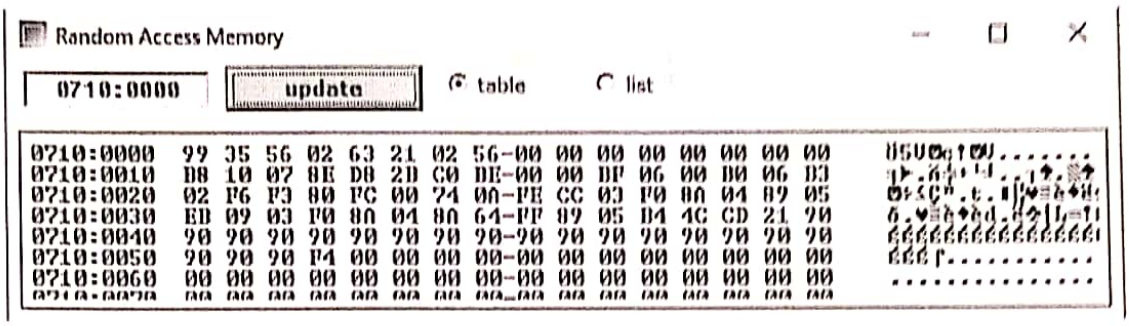
[09-10-2020 -- 23:41:56]

[LINE]	LOC: MACHINE CODE	SOURCE
[1]	:	
[2]	0000: 99 35 56 02 63 21	99H,35H,56H,02H,63H,21H
[3]	:	
[4]	0006: 00	
[5]	:	
[6]	:	
[7]	:	
[8]	0010: B8 00 00	
[9]	0013: 8E D8	
[10]	0015: 2B C0	
[11]	0017: BE 00 00	
[12]	001A: BF 06 00	
[13]	001D: B0 06	
[14]	001F: B3 02	
[15]	0021: F6 F3	
[16]	0023: 80 FC 00	
[17]	0026: 74 0A	
[18]	0028: FE CC	
[19]	002A: 03 F0	
[20]	002C: 8A 04	
[21]	002E: 89 05	
[22]	0030: EB 09	
[23]	0032: 03 F0	
[24]	0034: 8A 04	
[25]	0036: 8A 64 FF	
[26]	0039: 89 05	
[27]	003B: B4 4C	
[28]	003D: CD 21	
[29]	:	

```
DATA SEGMENT
        ARRAY DB

COUNT EQU 06H
   MEDIAN DB ?
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS:DATA
      START:MOV AX, DATA
      MOV DS, AX
      SUB AX, AX
       MOV SI, OFFSET ARRAY
          MOV DI, OFFSET MEDIAN
      MOV AL, COUNT
      MOV BL,02H
      DIV BL
       CMP AH,00H
      JE EVEN
          DEC AH
      ADD SI, AX
       MOV AL, [SI]
      MOV [DI], AX
       JMP EXIT
      EVEN: ADD SI, AX
       MOV AL, [SI]
        MOV AH, [SI-1]
      MOV [DI], AX
        EXIT: MOV AH,4CH
        INT 21H
CODE ENDS
```

[30] : END START
[31] :
[32] :



34. Program to find string length

Software: 8086 emulator

Program Code:-

```
DATA SEGMENT
    STR DB 'INDIA $'
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
       MOV DS, AX
       MOV AX, 0000H
       MOV CX, 0000H
       MOV SI, OFFSET STR
X:     MOV AL, [SI]
       CMP AL, '$'
       JE Y
       INC SI
       INC CX
       JMP X
Y:     MOV AH, 4CH
       INT 21H
CODE ENDS
END START
```

Result:-

Input: INDIA

output: CX - 05

STRING LENGTH

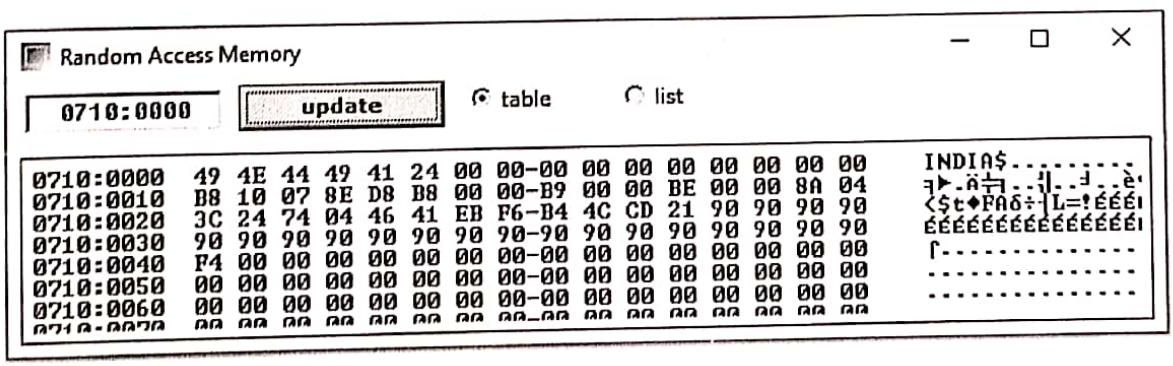
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:45:30]

```
[LINE]    LOC: MACHINE CODE                  SOURCE
==========================================================

[  1]      :                                 DATA SEGMENT
[  2]    0000: 49 4E 44 49 41 24                 STR DB "INDIA$"
[  3]      :                                 DATA ENDS
[  4]      :                                 ASSUME CS: CODE, DS: DATA
[  5]      :                                 CODE SEGMENT
[  6]    0010: B8 00 00                          START: MOV AX, DATA
[  7]    0013: 8E D8                           MOV DS, AX
[  8]    0015: B8 00 00                         MOV AX, 0000H
[  9]    0018: B9 00 00                         MOV CX, 0000H
[ 10]    001B: BE 00 00                          MOV SI, OFFSET STR
[ 11]    001E: 8A 04                           X: MOV AL, [SI]
[ 12]    0020: 3C 24                           CMP AL, '$'
[ 13]    0022: 74 04                           JE Y
[ 14]    0024: 46                             INC SI
[ 15]    0025: 41                             INC CX
[ 16]    0026: EB F6                            JMP X
[ 17]    0028: B4 4C                            Y: MOV AH,4CH
[ 18]    002A: CD 21                             INT 21H
[ 19]      :                                 CODE ENDS
[ 20]      :                                 END START
[ 21]      :
[ 22]      :
```



35) Program of string reversal and palindrome.

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
    STRING DB 'SRIIT'
    COUNT EQU ($- STRING1)
RSTRING2 DB 10 DUP(0)
MES1 DB 'THE STRING IS PALINDROME $'
MES2 DB 'THE STRING IS NOT PALINDROME $'
DATA ENDS
CODE SEGMENT
ASSUME CS: CODE, DS: DATA
START: MOV AX, DATA
       MOV DS, AX
       MOV SI, OFFSET STRING1
       MOV SI, OFFSET RSTRING2
       MOV CX, COUNT
       MOV BX, COUNT-1
       ADD SI, BX
LO:
       MOV AL, [SI]
       MOV [DI], AL
       INC DI
       DEC SI
       DEC CX
       JNZ LO ; REVERSING STRING
       MOV CX, BX
```

```
        MOV SI, OFFSET STRING1
        MOV DI, OFFSET RSTRING2
L3:
        MOV AL, [SI]
        CMP AL, [DI]
        JZ  L1
        JNZ L2 ; CHECKING FOR PALINDROME
L1:
        INC SI
        INC DI
        DEC CX
        JNZ L3
        JL  L4
L2:
        MOV AH, 09H
        LEA DX, MES2 ; DISPLAY FOR NOT
        INT 21H                PALINDROME
        JMP L5
L4:
        MOV AH, 09H
        LEA DX, MES1: DISPLAY FOR PALINDROME
        INT 21H
L5:
        MOV AH, 4CH
        INT 21H
CODE ENDS
END START
```

Result:-

Input - SRIIT

output - THE STRING IS NOT PALINDROME

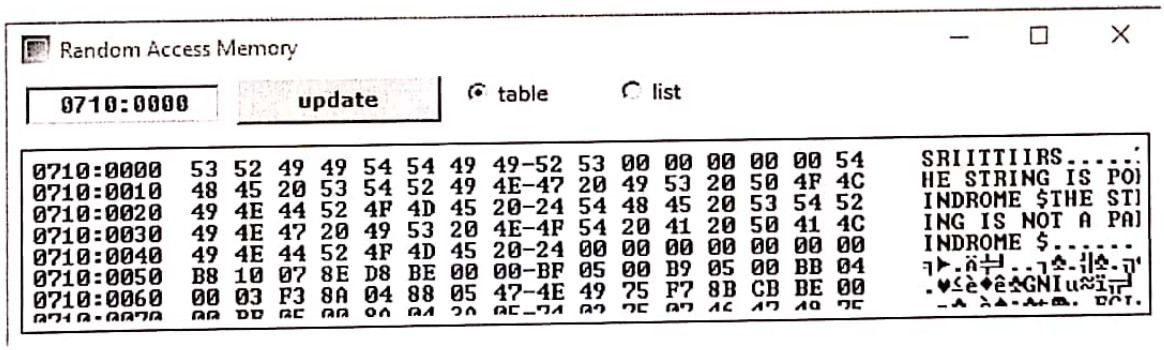
STRING REVERSAL AND PALINDROME

```
EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.
noname.exe_ -- emu8086 assembler version: 4.08
[ 09-10-2020  --  23:53:20 ]
```

```
[LINE]    LOC: MACHINE CODE                    SOURCE

[  1]      :                                   DATA SEGMENT
[  2]    0000: 53 52 49 49 54                      STRING1 DB 'SRIIT'
[  3]    0005:                                  COUNT EQU ($-STRING1)
[  4]    0005: 00 00 00 00 00 00 00 00 00 00     RSTRING2 DB 10 DUP(0)
[  5]    000F: 54 48 45 20 53 54 52 49 4E 47 20 49   MES1 DB 'THE STRING
IS POLINDROME $'
              53 20 50 4F 4C 49 4E 44 52 4F 4D 45
              20 24
[  6]    0029: 54 48 45 20 53 54 52 49 4E 47 20 49   MES2 DB 'THE STRING
IS NOT A PALINDROME $'
              53 20 4E 4F 54 20 41 20 50 41 4C 49
              4E 44 52 4F 4D 45 20 24
[  7]      :                                   DATA ENDS
[  8]      :                                   ASSUME CS: CODE, DS:DATA
[  9]      :                                   CODE SEGMENT
[ 10]    0050: B8 00 00                          START: MOV AX, DATA
[ 11]    0053: 8E D8                            MOV DS, AX
[ 12]    0055: BE 00 00                          MOV SI, OFFSET STRING1
[ 13]    0058: BF 05 00                              MOV DI, OFFSET RSTRING2
[ 14]    005B: B9 05 00                          MOV CX, COUNT
[ 15]    005E: BB 04 00                          MOV BX, COUNT-1
[ 16]    0061: 03 F3                           ADD SI, BX
[ 17]    0063:                                L0:
[ 18]    0063: 8A 04                            MOV AL, [SI]
[ 19]    0065: 88 05                           MOV [DI], AL
[ 20]    0067: 47                             INC DI
[ 21]    0068: 4E                             DEC SI
[ 22]    0069: 49                             DEC CX
[ 23]    006A: 75 F7                            JNZ L0 ; REVERSING STRING
[ 24]    006C: 8B CB                             MOV CX,BX
[ 25]    006E: BE 00 00                          MOV SI,OFFSET STRING1
[ 26]    0071: BF 05 00                          MOV DI,OFFSET RSTRING2
[ 27]    0074: 8A 04                           L3:         MOV AL,[SI]
[ 28]    0076: 3A 05                           CMP AL,[DI]
```

```
[ 29]   0078: 74 02                    JZ  L1
[ 30]   007A: 75 07                    JNZ  L2 ;CHECKING FOR
PALINDROME
[ 31]   007C:                        L1:
[ 32]   007C: 46                      INC SI
[ 33]   007D: 47                      INC DI
[ 34]   007E: 49                      DEC CX
[ 35]   007F: 75 F3                    JNZ L3
[ 36]   0081: 74 09                    JZ L4
[ 37]   0083:                        L2:
[ 38]   0083: B4 09                      MOV AH,09H
[ 39]   0085: BA 29 00                    LEA DX, MES2 ; DISPLAY FOR
NOT A PALINDROME
[ 40]   0088: CD 21                      INT 21H
[ 41]   008A: EB 07                      JMP L5
[ 42]   008C:                         L4:
[ 43]   008C: B4 09                      MOV AH,09H
[ 44]   008E: BA 0F 00                    LEA DX, MES1 ; DISPLAY FOR A
PALINDROME
[ 45]   0091: CD 21                      INT 21H
[ 46]   0093: B4 4C                      L5:       MOV AH,4CH
[ 47]   0095: CD 21                      INT 21H
[ 48]      :                       CODE ENDS
[ 49]      :                       END START
[ 50]      :
```



36. Assembly language program to insert a string

Software:- 8086 emulator

Program Code:-

```
DATA SEGMENT
   STRING1 DB 'EMPTY VESSELS MOR NOISE$'
   STRLEN  EQU ($-STRING1)
   STRING2 DB 'MAKE$'
DATA ENDS
EXTRA SEGMENT
     STRING3 DB STRLEN+5 DUP(0)
EXTRA ENDS
 CODE SEGMENT
ASCUME CS: CODE, DS: DATA, ES: DATA
   START: MOV AX, DATA
          MOV DS, AX
          MOV AX, EXTRA
          MOV ES, AX
         MOV SI, OFFSET STRING1
         MOV DI, OFFSET STRING3
         CLD
          MOV CX, 14
          REP MOVSB
          JZ X
   X:    MOV CX, 5
         MOV SI, OFFSET STRING2
         REP MOVSB
          JZ Y
```

```
Y:  MOV   SI , OFFSET STRING1
    ADD   SI, 14
    MOV   CX, 11
    REP   MOVSB
    JZ Z
Z:  MOV AH, 4CH
    INT 21H
CODE ENDS
END START
```

Result :-

Input:- STRING 1 - EMPTY VESSELS MORE NOISE

STRING 2 - MAKE

output:- STRING 3 - EMPTY VESSELS MAKE MORE NOISE

INSERTING A STRING

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

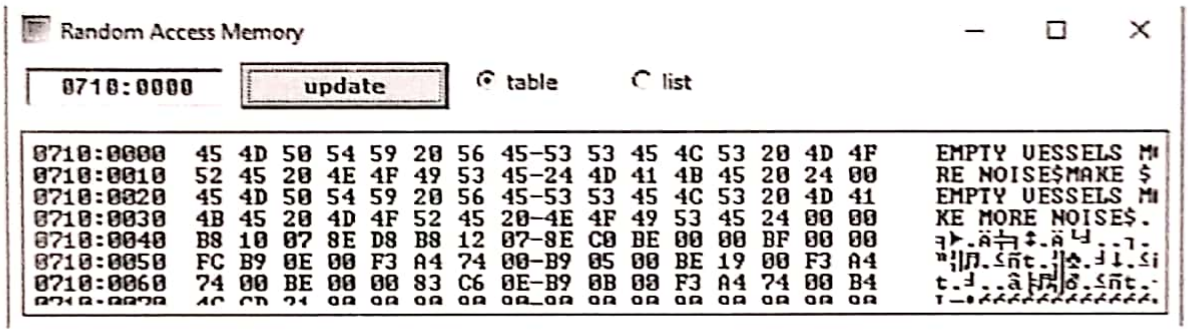
noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:56:22]

```
[LINE]    LOC: MACHINE CODE                    SOURCE

[  1]      :                                   DATA SEGMENT
[  2]    0000: 45 4D 50 54 59 20 56 45 53 53 45 4C   STRING1 DB 'EMPTY
VESSELS MORE NOISE$'
             53 20 4D 4F 52 45 20 4E 4F 49 53 45
             24
[  3]    0019:                                 STRLEN EQU ($-STRING1)
[  4]    0019: 4D 41 4B 45 20 24                     STRING2 DB 'MAKE $'
[  5]      :                                   DATA ENDS
[  6]      :                                   EXTRA SEGMENT
[  7]    0020: 00 00 00 00 00 00 00 00 00 00 00 00   STRING3 DB STRLEN+5
DUP (0)
             00 00 00 00 00 00 00 00 00 00 00 00
             00 00 00 00 00 00
[  8]      :                                   EXTRA ENDS
[  9]      :                                   CODE SEGMENT
[ 10]      :                                    ASSUME CS: CODE, DS: DATA, ES:
EXTRA
[ 11]    0040: B8 00 00                          START: MOV AX, DATA
[ 12]    0043: 8E D8                            MOV DS, AX
[ 13]    0045: B8 02 00                          MOV AX, EXTRA
[ 14]    0048: 8E C0                            MOV ES, AX
[ 15]    004A: BE 00 00                           MOV SI, OFFSET STRING1
[ 16]    004D: BF 00 00                           MOV DI, OFFSET STRING3
[ 17]    0050: FC                               CLD
[ 18]    0051: B9 0E 00                           MOV CX, 14
[ 19]    0054: F3 A4                             REP MOVSB
[ 20]    0056: 74 00                            JZ X
[ 21]    0058: B9 05 00                           X:              MOV CX, 5
[ 22]    005B: BE 19 00                                       MOV SI, OFFSET
STRING2
```

```
[ 23]   005E: F3 A4                         REP MOVSB
[ 24]   0060: 74 00                         JZ Y
[ 25]   0062: BE 00 00              Y:      MOV SI, OFFSET
STRING1
[ 26]   0065: 83 C6 0E                      ADD SI, 14
[ 27]   0068: B9 0B 00                      MOV CX, 11
[ 28]   006B: F3 A4                         REP MOVSB
[ 29]   006D: 74 00                         JZ Z
[ 30]   006F: B4 4C                 Z:      MOV AH, 4CH
[ 31]   0071: CD 21                         INT 21H
[ 32]      :                        CODE ENDS
[ 33]      :                        END START
[ 34]      :
[ 35]      :
```



37, Assembly language program to delete a string.

Software :- 8086 emulator

Program Code :-

```
DATA SEGMENT
  STRING1 DB 'EMPTY VESSELS MAKE MORE NOISE$'
  STRLEN EQU ($-STRING1)
DATA ENDS
EXTRA SEGMENT
   STRING3 DB STRLEN-5 DUP(0)
EXTRA ENDS
CODE SEGMENT
 ASSUME CS: CODE, DS: DATA, ES: DATA
   START: MOV AX, DATA
          MOV DS, AX
          MOV AX, EXTRA
          MOV ES, AX
       MOV SI, OFFSET STRING1
       MOV DI, OFFSET STRING3
        CLD
        MOV CX, 14
         REP MOVSB
        MOV CX, 12
        MOV SI, OFFSET STRING1
          ADD SI, 18
          REP MOVSB
          MOV AH, 4CH
```

INT 21H
CODE ENDS
END START

Result:-

Input:- STRING1 - EMPTY VESSELS MAKE MORE NOISE
STRING2 - MAKE

Output:- STRING3- EMPTY VESSELS MORE NOISE.

DELETING A STRING

EMU8086 GENERATED LISTING. MACHINE CODE <- SOURCE.

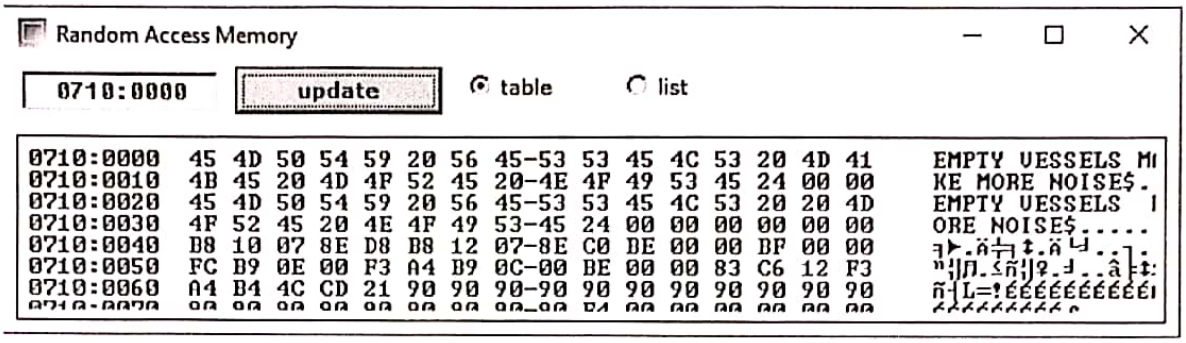
noname.exe_ -- emu8086 assembler version: 4.08

[09-10-2020 -- 23:58:48]

```
[LINE]    LOC: MACHINE CODE                    SOURCE

[  1]      :                                   DATA SEGMENT
[  2]   0000: 45 4D 50 54 59 20 56 45 53 53 45 4C   STRING1 DB 'EMPTY
VESSELS MAKE MORE NOISE$'
           53 20 4D 41 4B 45 20 4D 4F 52 45 20
           4E 4F 49 53 45 24
[  3]   001E:                                  STRLEN EQU ($-STRING1)
[  4]      :                                   DATA ENDS
[  5]      :                                   EXTRA SEGMENT
[  6]   0020: 00 00 00 00 00 00 00 00 00 00 00 00   STRING3 DB STRLEN-5
DUP (0)
           00 00 00 00 00 00 00 00 00 00 00 00
           00
[  7]      :                                   EXTRA ENDS
[  8]      :                                   CODE SEGMENT
[  9]      :                                   ASSUME CS: CODE, DS: DATA, ES:
EXTRA
[ 10]   0040: B8 00 00                          START: MOV AX, DATA
[ 11]   0043: 8E D8                            MOV DS, AX
[ 12]   0045: B8 02 00                          MOV AX, EXTRA
[ 13]   0048: 8E C0                            MOV ES, AX
[ 14]   004A: BE 00 00                           MOV SI, OFFSET STRING1
[ 15]   004D: BF 00 00                           MOV DI, OFFSET STRING3
[ 16]   0050: FC                               CLD
[ 17]   0051: B9 0E 00                           MOV CX, 14
[ 18]   0054: F3 A4                            REP MOVSB
[ 19]   0056: B9 0C 00                           MOV CX, 12
[ 20]   0059: BE 00 00                              MOV SI, OFFSET STRING1
[ 21]   005C: 83 C6 12                               ADD SI, 18
[ 22]   005F: F3 A4                                 REP MOVSB
[ 23]   0061: B4 4C                            MOV AH, 4CH
```

```
[ 24]   0063: CD 21                INT 21H
[ 25]      :                       CODE ENDS
[ 26]      :                       END START
[ 27]      :
[ 28]      :
```



1. Program for 8-bit addition in immediate, direct and indirect addressing mode

Software:- Keil uVision5

Program Code:-

(a) Immediate addressing mode:-

```
MOV  a,#45H
MOV  b,#86H
ADD  a,b
MOV  30H, a
end
```

(b) Direct addressing mode:-

```
MOV 20H, #45H
MOV 21H, #86H
MOV  a, 20H
MOV  b, 21H
ADD  a, b
MOV  30H, a
end
```

(b) Indirect addressing mode:-

```
MOV 20H, #45H
MOV 21H, #86H
MOV  a,  20H
MOV  R0  21H
ADD  a, @R0
MOV  30H, a
end.
```

Result:-

Input numbers: 45H, 86H

output:- CB

2. Program to do BCD addition of two numbers.

Software :- Keil uVision 5

Program Code :-

```
MOV  a, #47H
MOV  b, #25H
ADD  a, b
DA   a
end.
```

Result :-

Input : a = 47H
b = 25H

output : a : 72H

3. Program for 16-bit addition

Software:- Keil uVision 5

Program Code:-

```
MOV  a, #86H
MOV  b, #95H
ADD  a, b
MOV  R0, a
MOV  a, #4AH
MOV  b, #38H
ADDC a, b
MOV  R1, a
end
```

Result:-

Input a: 4A86H

b: 3895H

Output: 83 1BH

R1 R0

4. Program to convert ASCII to packed BCD

Software:- Keil uVision5

Program Code:-

```
MOV  a, #'8'
MOV  b, #'6'
ADD  a, #0FH
ADD  b, #0FH
SWAP  a
ORL   a, b
end
```

Result:-

Input a: 8
b: 6

output : 86

5. Timer program [Generating wave form of 50% duty cycle in Timer 0, mode 1]

Software:- Keil uVision5

Program code:-

```
        MOV  P0, #0FFH
        ACALL  Delay
        MOV  P0, #00H
        ACALL  Delay
        SJMP  Main
Delay:  MOV  TMOD, #01H
        MOV  TH0, #0DCH
        MOV  TL0, #00H
        SETB  TR0
L:      JNB  TF0, L
        CLR  TR0
        CLR  TF0
        RET
END
```

Result:- Square wave is generated at port 0.

6. Load 45H (data) into external memory location 8000H

Software:- keil uVision 5

Program Code:-

```
MOV DPTR, #8000H
MOV a, #45H
MOVX @DPTR, a
end.
```

Result:-

45 is appeared in 8000H memory location.

7. Program to find largest number for given series of 10 numbers

Software:- Keil uVision 5

Program Code:-

```
        ORG   0000H
        MOV   R0, #50H
        MOV   R1, #10
        MOV   B, #00H
BACK :  MOV   A, @R0
        CJNE  A, B, LOOP
LOOP :  JC LOOP 1
        MOV   B, A
        INC   R0
        DJNZ  R1, BACK
LOOP 1: INC   R0
        DJNZ  R1, BACK
        end.
```

Result:-

INPUT: 23, 02, 43, 74, 27, 18, 20, 29, 31, 72

OUTPUT: B : 74

8. Counter Program:-

Software:- keil uVision5

Program Code:-

```
MOV TMOD , #60H
MOV TH1 , #00H
SETB P3.5
AGAIN : SETB TR1
BACK : MOV A, TL1
       MOV P2, A
       JNB, TF1, BACK
       CLR TF1
       CLR TR1
       SJMP AGAIN
END
```

Result:-

Count from 00H to FFH is observed in port 2.

9. Counter program:-

Software:- Keil uVision 5

Program Code:-

```
        MOV  TMOD, #06H
        MOV  TL0, #00H
BACK:   SETB TR0
        MOV  P1, TL0
        MOV  A, P1
        CJNE A, #10, BACK
        CLR  TR0
        END
```

Result:-

Count from 00H to 0AH is observed in port 1.

10. Program to transfer 10 bytes memory block from location 12H to memory location starting from address 32H.

Software:- Keil uVision5

Program Code:-

```
        MOV R7, #0AH
        MOV R0, #12H
        MOV R1, #32H
BACK:
        MOV A, @R0
        MOV @R1, A
        INC R0
        INC R1
        DJNZ R7, BACK
        END
```

Result:-

Input:-

In memory location 12H

01, 02, 03, 04, 05, 06, 07, 08, 09, 0A

Output:-

In memory location 32H

01, 02, 03, 04, 05, 06, 07, 08, 09, 0A

11. Program to arrange numbers in Ascending Order.

Software:- Keil uVision 5

Program Code:-

```
        ORG , 0000H
        MOV R7, #04
LOOP1 : MOV R0, #40H
        MOV R6, #03
LOOP :  MOV A, @R0
        INC R0
        MOV 50H, @R0
        CJNE A, 50H, NEXT
        SJMP DOWN
DOWN :  DJNZ R6, LOOP
        DJNZ R7, LOOP1
        END
```

Result:-

Input:- 78, 45 32 12

Output:- 12 32 45 78

12. Program to count number of 0's and 1's in 8-bit number

Software :- keil uVision 5

Program Code :-

```
MOV, R1, #00H
MOV  R2, #00H
MOV  R7, #08H
MOV  A, #97H
AGAIN :  RLC A
         JC  NEXT
         INC R1
         SJMP HERE
NEXT :   INC R2
HERE:    DJNZ R7, AGAIN
         END
```

Result :-

Input: A: 97H

Output: R_1: 03

R_2: 05

13. Program to find whether the given 8-bit number is odd or even.

Software:- Keil uVision 5

Program Code:-

```
ORG  0000H
MOV  A, #23H
 RRC , A
JC   ODD
MOV  A, # OFF H
ODD:  MOV A, # 00H
 END
```

Result:-

Input:- A: 23H

Output:- A: 00H {odd}

19. Program to perform Logical AND, OR, XOR for two 8-bit number stored in internal RAM location, 21H, 22H. store the result in 30H, 31H and 32H

Software:- Keil uVision 5

Program Code:-

```
MOV  A, 21H
MOV  A, 22H
ANL  A, B
MOV  30H, A
MOV  A, 21H
ORL  A, B
MOV  31H, A
MOV  A, 21H
XRL  A, B
MOV  32H, A
END
```

Result:-

Input A: 12 {in 21H}
B: 39 {in 22H}

Output 30H: 10H {AND}
31H: 3BH {OR}
32H: 2BH {XOR}

Theoretically:-

ANL :

0001	0010
0011	1001
0001	0000
1	0

ORL :

0001	0010
0011	1001
0011	1011
3	B

XRL :

0001	0010
0011	1001
0010	~~1010~~ 1011
2	B

15. Program to convert binary to decimal

Software:- Keil uVision 5

Program Code:-

```
MOV A, #0FEH
MOV B, #0AH
DIV AB
MOV R0, B
MOV B, #0AH
DIV AB
MOV 30H, A
MOV A, B
SINAP A
ORL A, R0
MOV 31H, A
END
```

Result:-

Input : A → 0FEH
B → 0AH

Output : 30H → 02
31H → 5H

16. Program to convert Decimal to Hexadecimal.

Software:- Keil uVision5

Program Code:-

```
MOV A, #95H
MOV B, #10H
DIV AB
MOV R1, B
MOV B, #0AH
MUL AB
ADD A, R1
MOV 30H, A
END
```

Result:-

Input:

A: 95H

B: 10H

Output:

30H : 5F

17. Program to transfer letter 'A' serially at 4800 band continuously

Software:- Keil uVision 5

Program Code:-

```
        MOV TMOD, #20H
        MOV TH1, #
        MOV SCON, #50H
        SET TR1
AGAIN:  MOV SBUF, #'A'
HERE :  JNB TI, HERE
        CLR T1
        SJMP AGAIN
        END
```

Result:-

A is displayed continuously.

18. Program for 8051 communication between 8051 kit and pc.

Software:- Keil uVision5

Program Code:-

```
PRESS 'E'
 ↓
PRESS 'ENTER'
 ↓
PRESS 'DOUBLEENTER'
 ↓
PRESS 'A'
 ↓
TYPE
6000   MOV  A, #33
6002   MOV  R0, #33
6004   ADD  A, R0
6005   RET
6006
 ↓
PRESS 'DOUBLE ENTER'
 ↓
PRESS 'Q'
 ↓
PRESS 'G'
 ↓
PRESS 'DOUBLE ENTER'
 ↓
SRC . SEG .6000
 ↓
PRESS 'S'
 ↓
PRESS 'ENTER'
```

PRESS 'ANY ONE KEY'

PRESS 'ENTER'

PRESS 'ENTER'

output:-

A 66

19. Interfacing keyboard to 8051 processor

Software:- Keil uVision 5

Program Code:-

```
PRESS 'I'
└→
└→ PRTY → N
     NOPR
     HEX
└→ STRT
└→ 7000
└→ 7FFF
└→ WAIT
     [GOLD Transfer open send her filter]
       X BOXPO. HEX
└→ G
  └→
  └→ 7000
  └→
```

Output:-

Interfacing done between key-board and 8051 microcontroller.

20. Program for interfacing 7-segment to 8051 processor

Software:- Keil uVision5

Program Code:-

```
PRESS 'I'
↳
↳ PRTY→N
   NO OPR
    HEX
↳ STRT
↳ 7120
↳ 7FFF
↳ WAIT
        [SEND TEXT FILE] Transfer option
   AND IS. HEX
↳ G.
↳
↳ 7120
↳
```

output:-

Interfacing is done between 7 segment display and 8051 microcontroller.

21) Interfacing between LED-display and 8051 processor

Software:- Keil uVision 5

Program Code:-

```
PRESS 'I'
↓
PRESS 'ENTER'
↳ PRTY→N
   NOPR
   HEX
↳ STRT
↳ 7180
↳ 7FFF
↳ WAIT
   (Go to transfer
   open send hex file)
   MATRIX.HEX
↳ G
↳
↳ 7180
↳
```

output:-

Interfacing is done between LED display and 8051 processor

22) Interfacing of 8086 and programming to control stepper motor

Software:- Keil uVision 5

Program Code:-

PRESS 'I'
↓
PRESS 'DOUBLE ENTER'
↓
DST - SEG 1000
↓
PRESS 'ENTER
↓
WAIT
↓
PRESS 'G'
↓
PRESS 'DOUBLE ENTER'
↓
SRC - SEGM.1000
↓
ADRR - 0100
↓
PRESS 'ENTER'
↓
DONE

Output:-

The stepper motor rotates in anti'clockwise direction

23, Parallel communication between two microprocessor bits using 8255

Software:- Keil uVision5

Program Code:-

PRESS 'S'
↳
PRESS ANY KEY
↓
INPUT → 'PRESS 'ENTER'
↓
PRESS ANY ONE KEY
OUTPUT
↓
PRESS 'ENTER'
↓
PRESS 'ENTER'
↓
PORT ADDR 8807
↓
PRESS 'ENTER'
↓
ENTER (80)
↓
PRESS 'ENTER'
↓
OFF
↓
ESCAPE
↳ ANY ONE KEY
↳ ANY ONE KEY

OUTPUT

↳ PORT ADD 8801

DATA 50

ENTER

Input (Read):

PRESS S

↓

PRESS ANY ONE KEY

↓

INPUT

↓

PRESS ENTER

↓

PRESS ANY ONE KEY

OUTPUT

↓

PRESS 'DOUBLE ENTER'

↓

PORT ADD 8807

↓

PRESS 'ENTER'

INPUT

↓

PRESS 'ENTER'

↓

PORT ADD 8801