VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Fall Semester 2021-22

Microprocessors and Interfacing LAB

CSE2006

Slot – L43+L44

Digital Assignment 4

Name: Ishan Sagar Jogalekar

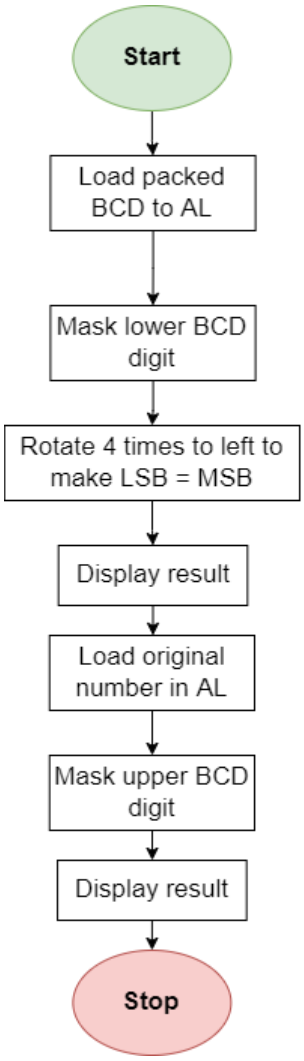
Reg. No: 19BCE2250

1. Packed to Unpacked number conversion -

Input:

Consider input as packed number for this program so that its equivalent unpacked number will be resulted. Here input is 35H.

Flowchart:



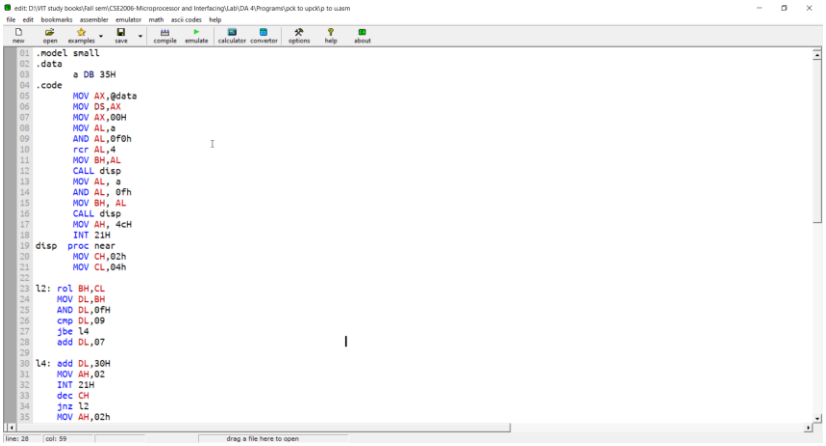
Algorithm:

1. Initialize the data memory and load input BCD to AL.
2. Mask the lower nibble using 0FH, use ADD instruction.
3. Rotate resultant value 4 times right in order to make MSB digit = LSB.
4. Display, store the partial result and load the input in AL again.
5. Mask the upper nibble using 0FH.
6. Combine the resultant values and display final Unpacked number.
7. Stop program.

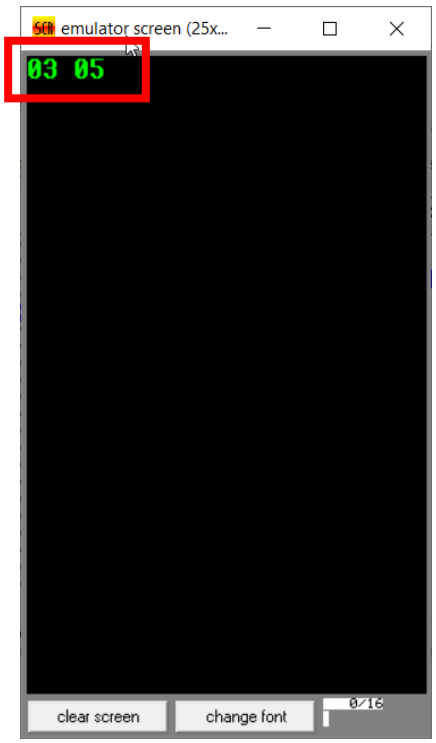
Program:

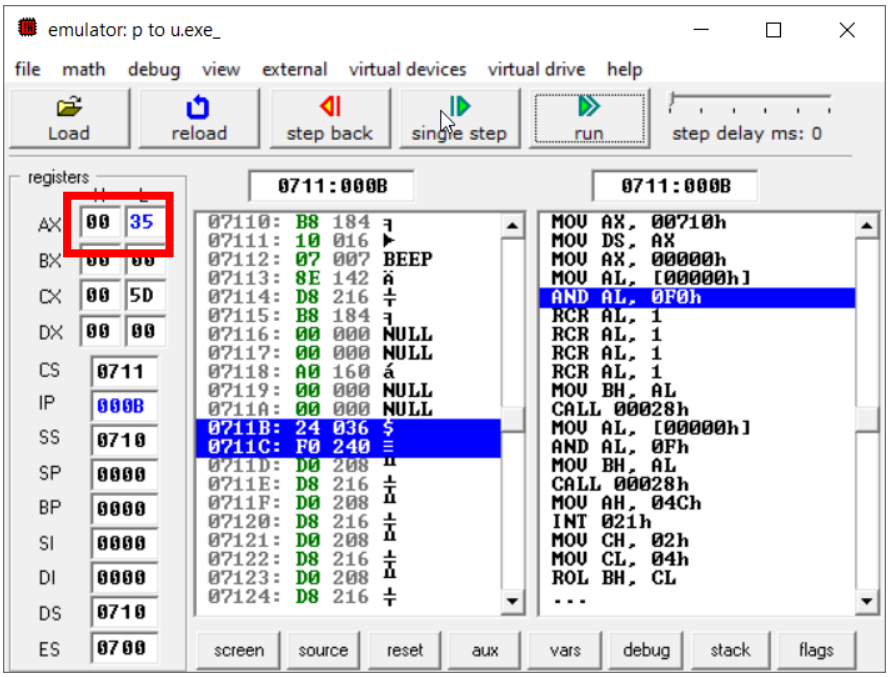
```
.model small
.data
        a DB 35H
.code
        MOV AX,@data
        MOV DS,AX
        MOV AX,00H
        MOV AL,a
        AND AL,0f0h
        rcr AL,4
        MOV BH,AL
        CALL disp
        MOV AL, a
        AND AL, 0fh
        MOV BH, AL
        CALL disp
        MOV AH, 4cH
        INT 21H
disp  proc near
        MOV CH,02h
        MOV CL,04h
l2: rol BH,CL
    MOV DL,BH
    AND DL,0fH
    cmp DL,09
    jbe l4
    add DL,07
l4: add DL,30H
    MOV AH,02
    INT 21H
```

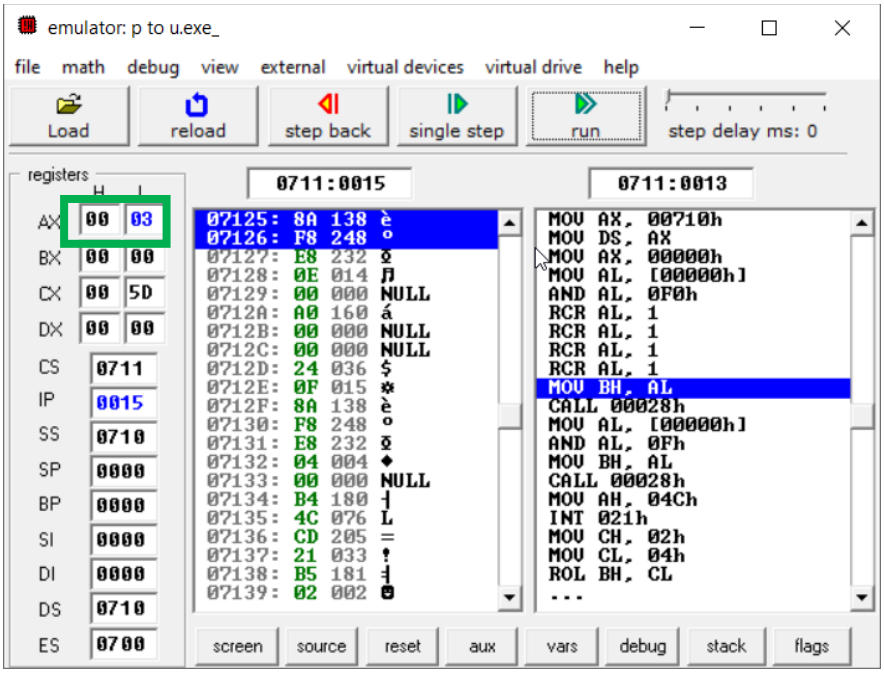
```
        dec CH
        jnz l2
        MOV AH,02h
        MOV DL,' '
        INT 21h
        ENDP
        RET
    END
```

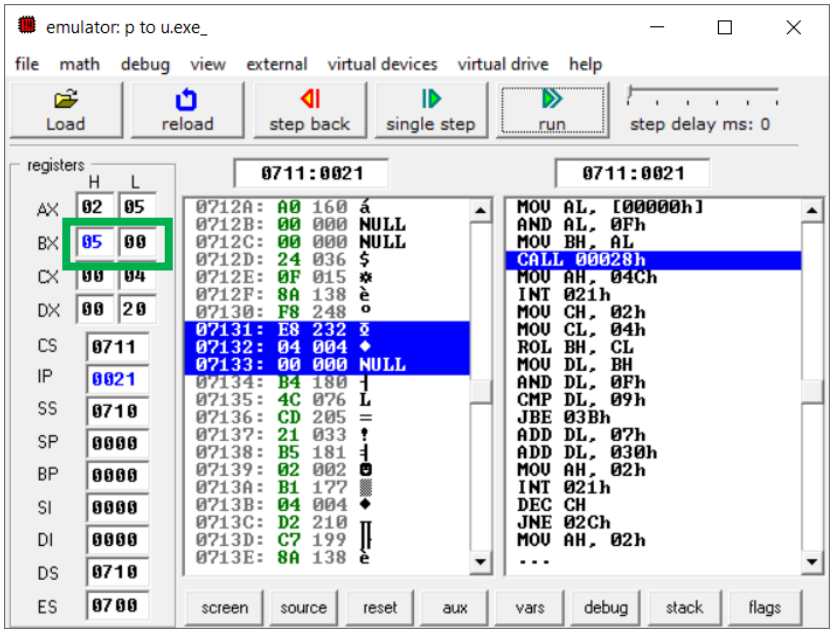


Output:




emulator: p to u.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
AX 00 35
BX 00 00
CX 00 5D
DX 00 00
CS 0711
IP 000B
SS 0710
SP 0000
BP 0000
SI 0000
DI 0000
DS 0710
ES 0700
0711:000B
07110: B8 184
07111: 10 016
07112: 07 007 BEEP
07113: 8E 142
07114: D8 216
07115: B8 184
07116: 00 000 NULL
07117: 00 000 NULL
07118: A0 160
07119: 00 000 NULL
0711A: 00 000 NULL
0711B: 24 036 $
0711C: F0 240
0711D: D0 208
0711E: D8 216
0711F: D0 208
07120: D8 216
07121: D0 208
07122: D8 216
07123: D0 208
07124: D8 216
0711:000B
MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV AL, [00000h]
AND AL, 0F0h
RCR AL, 1
RCR AL, 1
RCR AL, 1
RCR AL, 1
MOV BH, AL
CALL 00028h
MOV AL, [00000h]
AND AL, 0Fh
MOV BH, AL
CALL 00028h
MOV AH, 04Ch
INT 021h
MOV CH, 02h
MOV CL, 04h
ROL BH, CL
...
screen source reset aux vars debug stack flags


emulator: p to u.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
AX 00 03
BX 00 00
CX 00 5D
DX 00 00
CS 0711
IP 0015
SS 0710
SP 0000
BP 0000
SI 0000
DI 0000
DS 0710
ES 0700
0711:0015
07125: 8A 138
07126: F8 248
07127: E8 232
07128: 0E 014
07129: 00 000 NULL
0712A: A0 160
0712B: 00 000 NULL
0712C: 00 000 NULL
0712D: 24 036 $
0712E: 0F 015
0712F: 8A 138
07130: F8 248
07131: E8 232
07132: 04 004
07133: 00 000 NULL
07134: B4 180
07135: 4C 076 L
07136: CD 205
07137: 21 033 !
07138: B5 181
07139: 02 002
0711:0013
MOV AX, 00710h
MOV DS, AX
MOV AX, 00000h
MOV AL, [00000h]
AND AL, 0F0h
RCR AL, 1
RCR AL, 1
RCR AL, 1
RCR AL, 1
MOV BH, AL
CALL 00028h
MOV AL, [00000h]
AND AL, 0Fh
MOV BH, AL
CALL 00028h
MOV AH, 04Ch
INT 021h
MOV CH, 02h
MOV CL, 04h
ROL BH, CL
...
screen source reset aux vars debug stack flags

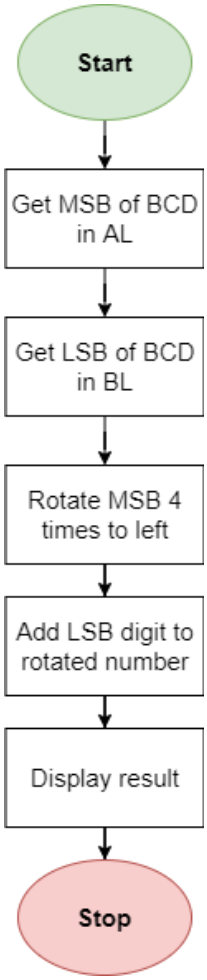

emulator: p to u.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
AX 02 05
BX 05 00
CX 00 04
DX 00 20
CS 0711
IP 0021
SS 0710
SP 0000
BP 0000
SI 0000
DI 0000
DS 0710
ES 0700
0711:0021
0712A: A0 160
0712B: 00 000 NULL
0712C: 00 000 NULL
0712D: 24 036 $
0712E: 0F 015
0712F: 8A 138
07130: F8 248
07131: E8 232
07132: 04 004
07133: 00 000 NULL
07134: B4 180
07135: 4C 076 L
07136: CD 205
07137: 21 033 !
07138: B5 181
07139: 02 002
0713A: B1 177
0713B: 04 004
0713C: D2 210
0713D: C7 199
0713E: 8A 138
0711:0021
MOV AL, [00000h]
AND AL, 0Fh
MOV BH, AL
CALL 00028h
MOV AH, 04Ch
INT 021h
MOV CH, 02h
MOV CL, 04h
ROL BH, CL
MOV DL, BH
AND DL, 0Fh
CMP DL, 09h
JBE 03Bh
ADD DL, 07h
ADD DL, 030h
MOV AH, 02h
INT 021h
DEC CH
JNE 02Ch
MOV AH, 02h
...
screen source reset aux vars debug stack flags

2. Unpacked to Packed number conversion –

Input:

Consider input of 2 unpacked numbers as U1 and U2, also create variable as P to store resultant packed number as output of program. Here input is 08H and 04H.

Flowchart:



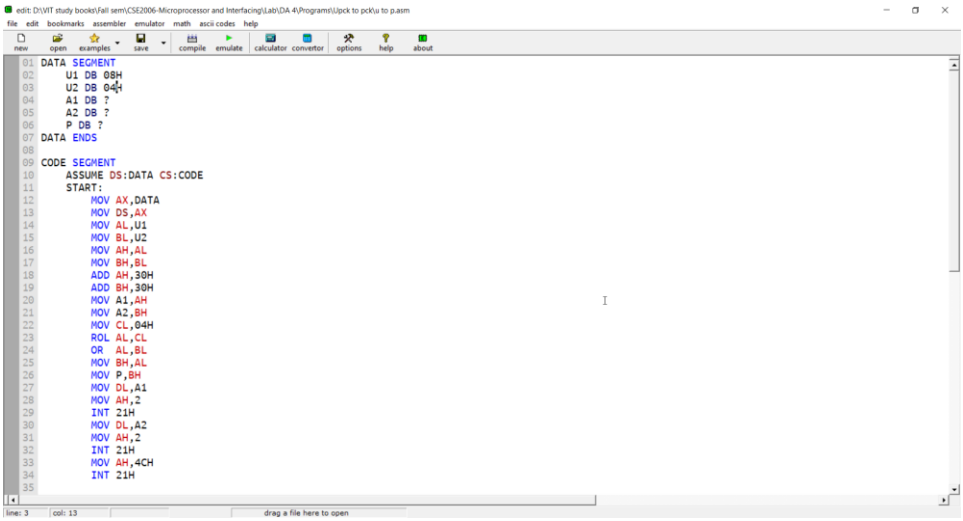
Algorithm:

1. Store 2 unpacked numbers in 2 separate variables.
2. Consider U1 as MSB and U2 as LSB.
3. Load MSB in AL and LSB in BL.
4. Rotate AL contents 4 times to left and store in AL itself as partial result.
5. ADD LSB digit to rotated number in AL.
6. Display Packed number as result.
7. Stop program.

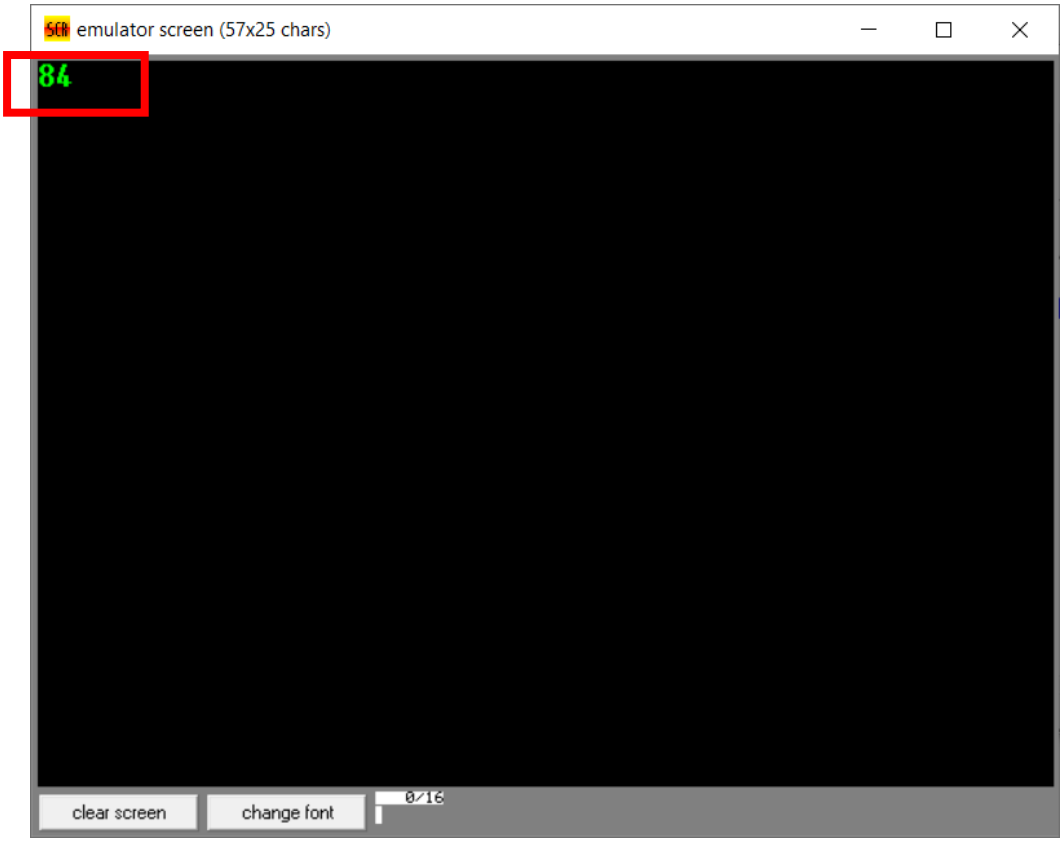
Program:

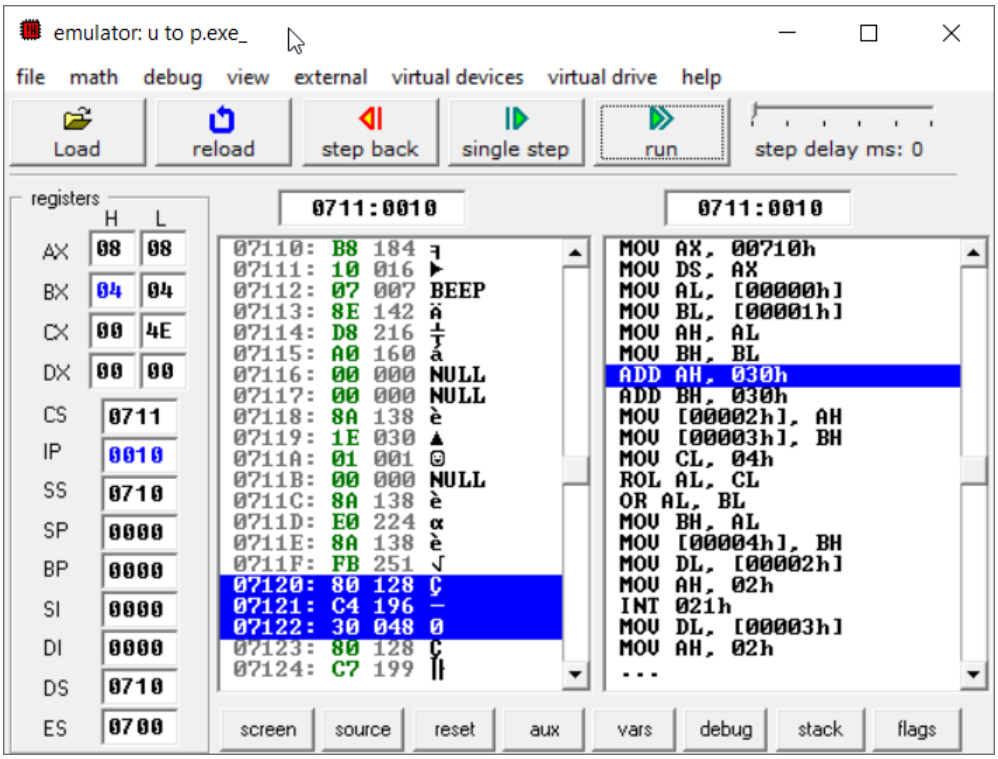
```
DATA SEGMENT
    U1 DB 08H
    U2 DB 04H
    A1 DB ?
    A2 DB ?
    P DB ?
DATA ENDS
CODE SEGMENT
    ASSUME DS:DATA CS:CODE
    START:
        MOV AX,DATA
        MOV DS,AX
        MOV AL,U1
        MOV BL,U2
        MOV AH,AL
        MOV BH,BL
        ADD AH,30H
        ADD BH,30H
        MOV A1,AH
        MOV A2,BH
        MOV CL,04H
        ROL AL,CL
        OR  AL,BL
        MOV BH,AL
        MOV P,BH
        MOV DL,A1
        MOV AH,2
        INT 21H
        MOV DL,A2
        MOV AH,2
```

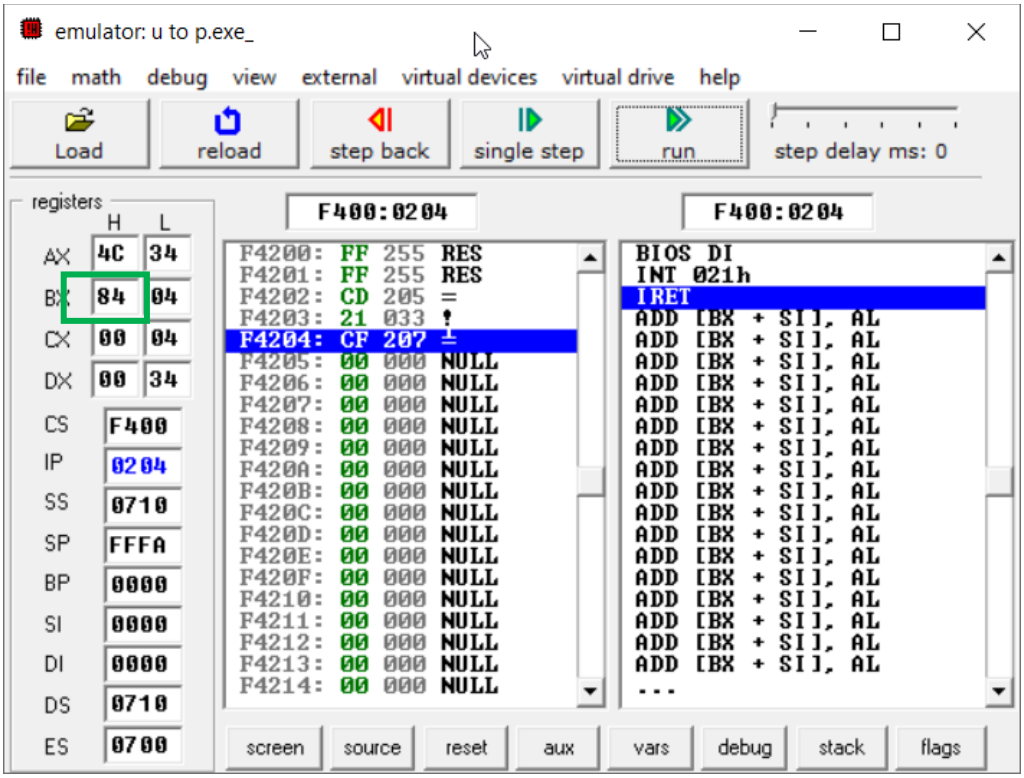
```
        INT 21H
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```



Output:




emulator: u to p.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
H L
AX 08 08
BX 04 04
CX 00 4E
DX 00 00
CS 0711
IP 0010
SS 0710
SP 0000
BP 0000
SI 0000
DI 0000
DS 0710
ES 0700
0711:0010
07110: B8 184
07111: 10 016
07112: 07 007 BEEP
07113: 8E 142
07114: D8 216
07115: A0 160
07116: 00 000 NULL
07117: 00 000 NULL
07118: 8A 138
07119: 1E 030
0711A: 01 001
0711B: 00 000 NULL
0711C: 8A 138
0711D: E0 224
0711E: 8A 138
0711F: FB 251
07120: 80 128
07121: C4 196
07122: 30 048
07123: 80 128
07124: C7 199
0711:0010
MOV AX, 00710h
MOV DS, AX
MOV AL, [00000h]
MOV BL, [00001h]
MOV AH, AL
MOV BH, BL
ADD AH, 030h
ADD BH, 030h
MOV [00002h], AH
MOV [00003h], BH
MOV CL, 04h
ROL AL, CL
OR AL, BL
MOV BH, AL
MOV [00004h], BH
MOV DL, [00002h]
MOV AH, 02h
INT 021h
MOV DL, [00003h]
MOV AH, 02h
...
screen source reset aux vars debug stack flags

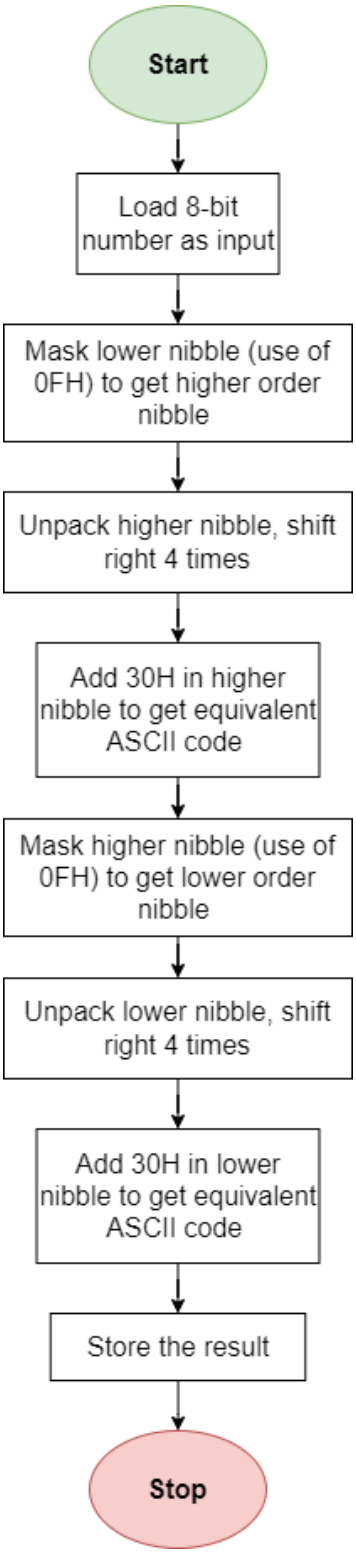

emulator: u to p.exe_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
H L
AX 4C 34
BX 84 04
CX 00 04
DX 00 34
CS F400
IP 0204
SS 0710
SP FFFA
BP 0000
SI 0000
DI 0000
DS 0710
ES 0700
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ╧
F4205: 00 000 NULL
F4206: 00 000 NULL
F4207: 00 000 NULL
F4208: 00 000 NULL
F4209: 00 000 NULL
F420A: 00 000 NULL
F420B: 00 000 NULL
F420C: 00 000 NULL
F420D: 00 000 NULL
F420E: 00 000 NULL
F420F: 00 000 NULL
F4210: 00 000 NULL
F4211: 00 000 NULL
F4212: 00 000 NULL
F4213: 00 000 NULL
F4214: 00 000 NULL
F400:0204
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
screen source reset aux vars debug stack flags

3. BCD to ASCII number conversion –

Input:

Input BCD number in specific memory location using EMU8086 emulator feature to insert BCD number at specific location. Here for example 45 as input.

Flowchart:

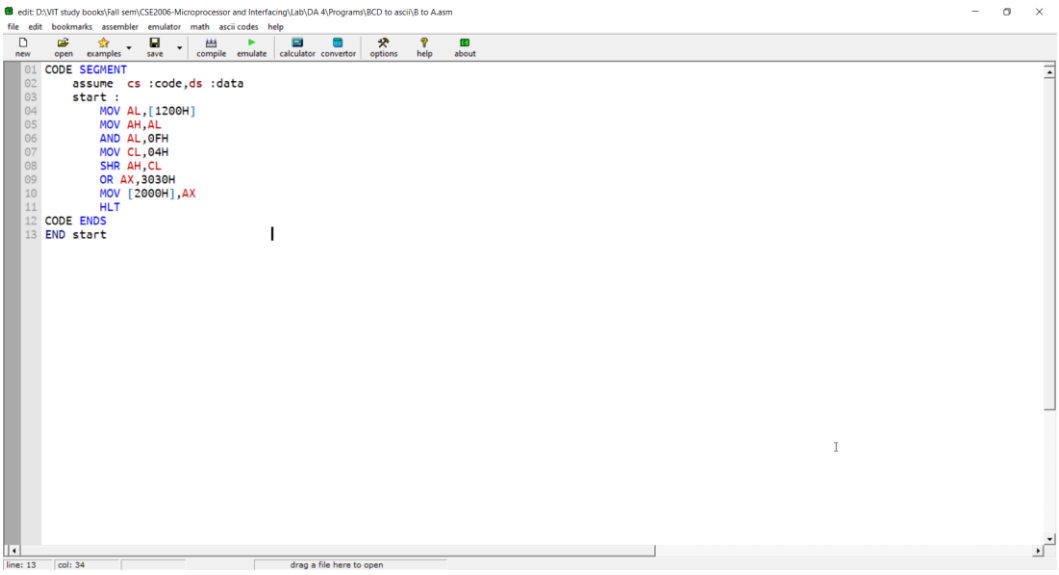


Algorithm:

1. Load 8-bit BCD number as input. Mask lower nibble using 0FH (ADD instruction) to get higher order nibble.
2. Unpack that higher order nibble and shift right resultant value 4 times.
3. Add 30H in higher nibble and store it partially, it is partial ASCII code.
4. Mask higher nibble using 0FH and get lower order nibble.
5. Unpack the lower order nibble and shit right 4 times.
6. Add 30H in lower nibble and get its ASCII code.
7. Display result of ASCII codes.
8. Stop the program.

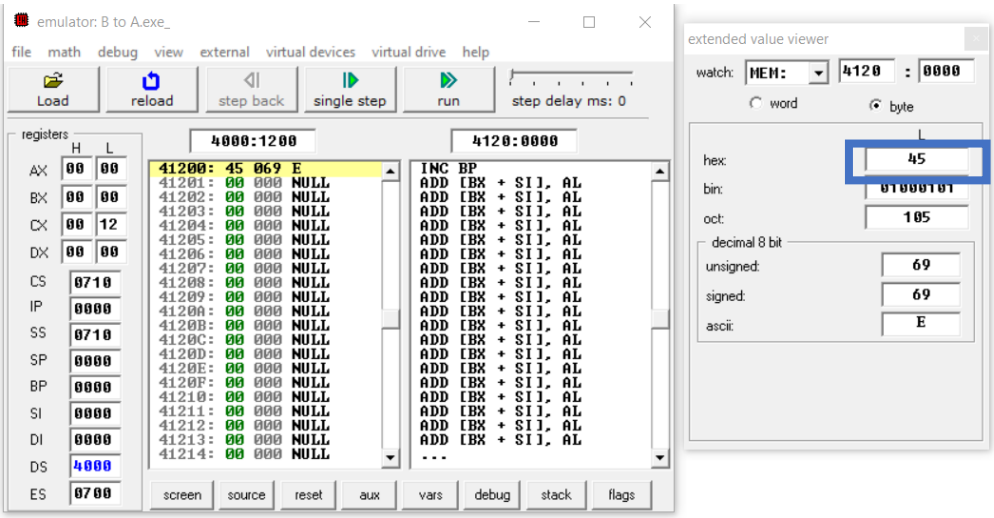
Program:

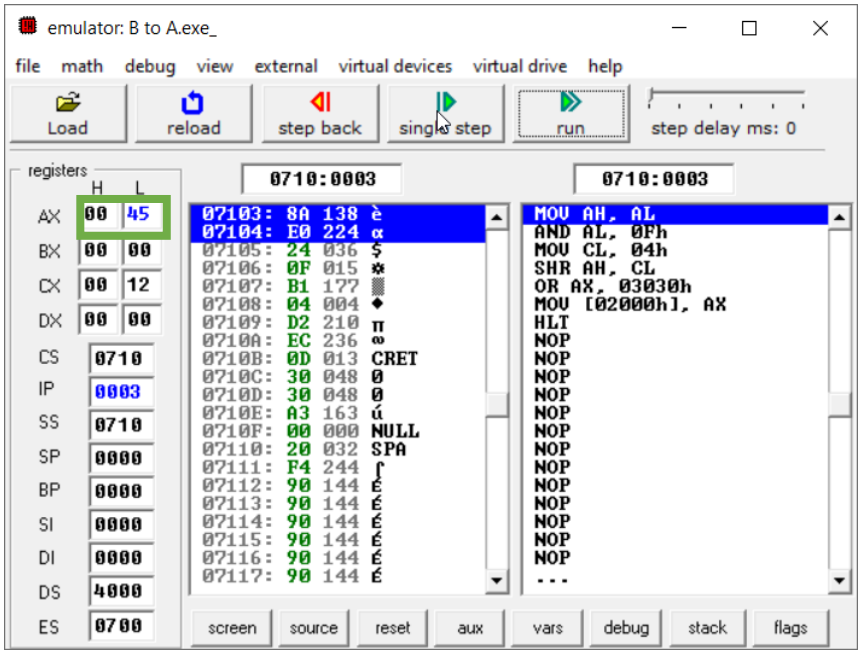
```
CODE SEGMENT
  assume  cs :code,ds :data
  start :
     MOV AL,[1200H]
     MOV AH,AL
     AND AL,0FH
     MOV CL,04H
     SHR AH,CL
     OR AX,3030H
     MOV [2000H],AX
     HLT
CODE ENDS
END start
```

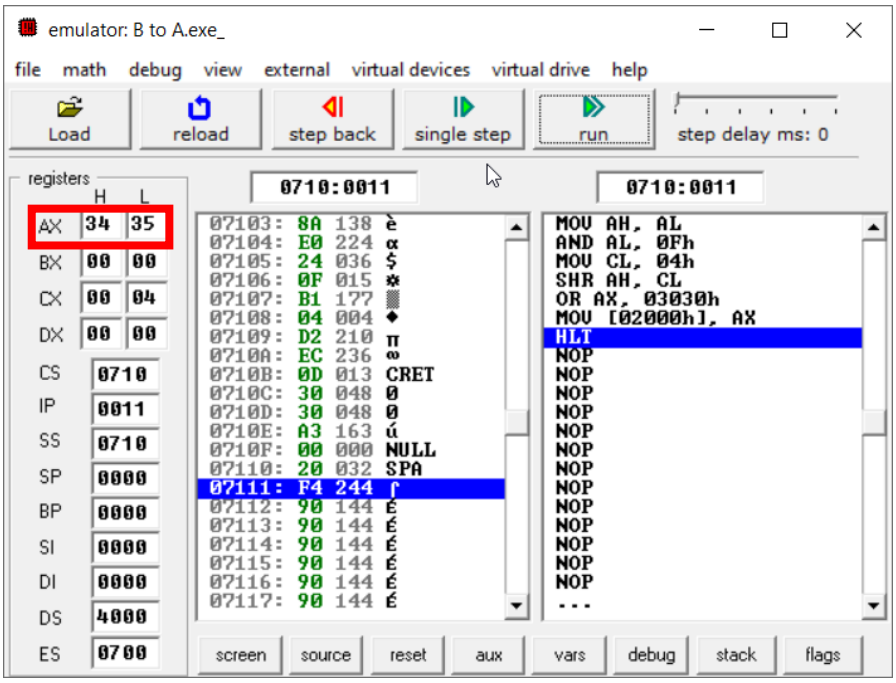


Output:

Load BCD value - Output for 45 should be 34 and 35.





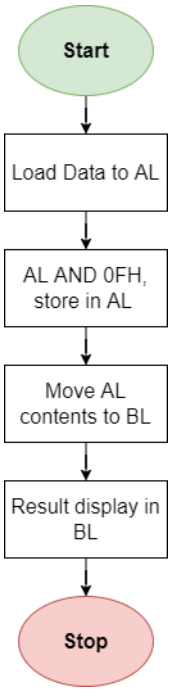


4. ASCII code to BCD number conversion –

Input:

Input ASCII code in variable in order to find its BCD number. Here consider 38H as example.

Flowchart:



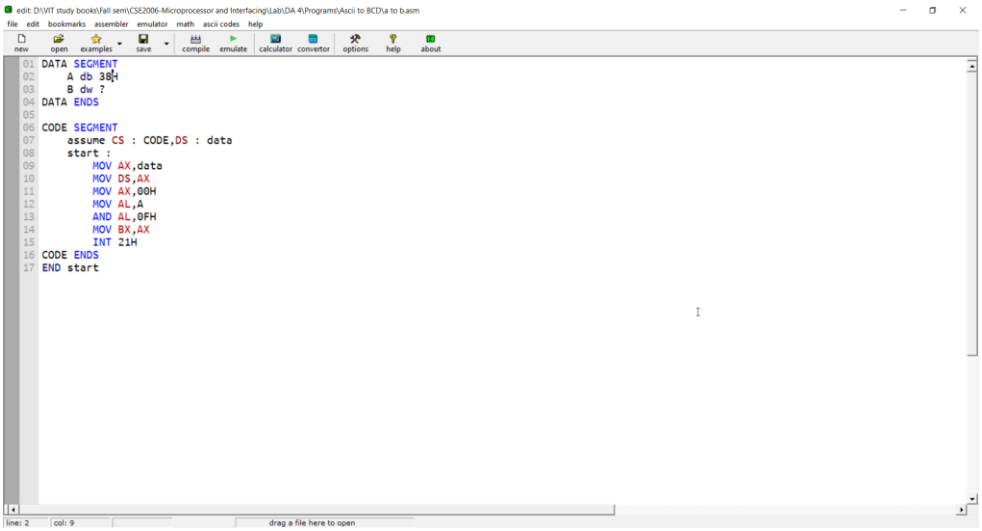
Algorithm:

1. Move value of variable into AL
2. Perform AND operation on AL with 0F
3. Move content of accumulator AL into BL or AH.
4. Display output.
5. Stop program.

Program:

```
DATA SEGMENT
        A db 38H
        B dw ?
DATA ENDS
CODE SEGMENT
        assume CS : CODE,DS : data
        start :
                MOV AX,data
                MOV DS,AX
```

```
        MOV AX,00H
        MOV AL,A
        AND   AL,0FH
        MOV BX,AX
        INT 21H
CODE ENDS
END start
```



For input 38H as ASCII code output BCD should be 08H.

Output: