Experiment 05

Experiment 5: Write an ALP program

1. Program to ADD,SUB 8bit and 16 bit number
2. Program for 8 bit and 16 bit BCD addition

Roll No.	17
Name	Manav Jawrani
Class	D10-A
Subject	Microprocessor Lab
LO Mapped	LO3: Build a program on a microprocessor using arithmetic & logical instruction set of 8086.

Aim: Write an ALP program

1. Program to ADD,SUB 8bit and 16 bit number
2. Program for 8 bit and 16 bit BCD addition

Introduction:

An assembly language is a type of low-level programming language that is intended to communicate directly with a computer's hardware. Unlike machine language, which consists of binary and hexadecimal characters, assembly languages are designed to be readable by humans.Low-level programming languages such as assembly language are a necessary bridge between the underlying hardware of a computer and the higher-level programming languages—such as Python or JavaScript—in which modern software programs are written.

Theory:

The 8086 Microprocessor is an enhanced version of 8085 Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20 address lines and16 data lines that provides up to 1MB storage. It consists of a powerful instruction set, which provides operations like multiplication and division easily.

It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for a system having multiple processors and Minimum mode is suitable for a system having a single processor.

8 bits can represent positive numbers from 0 to 255. While 16 bit: Signed Integers ranging from -32768 to +32767. The 16-bit data type is used for numerical tags where variables have the potential for negative or positive values.

A.

1. **Addition of two 8 bit numbers.**

Algorithm:

Step 1 - Initialize the data segment with input numbers and memory location for the answer.

Step 2 - Start the program by loading the first data into the Accumulator.

Step 3 - Move the first number to register ‘al’ and move the second number to register ‘bl’.

Step 4 - Add the two register contents.

Step 5 - Store the content of al to ans.

Step 6 - Terminate the program.

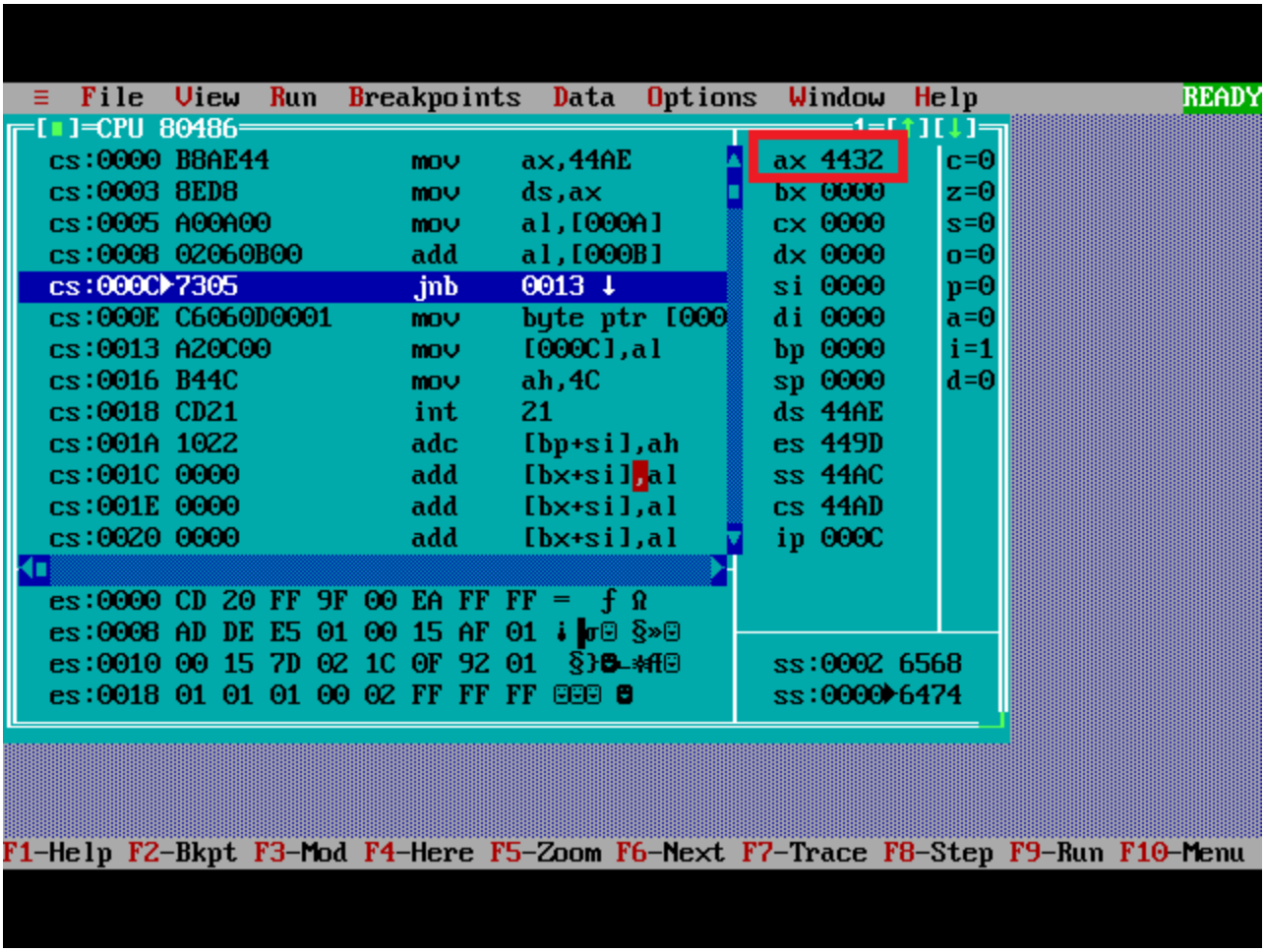
Code:

```
.model small
.data
n1 db 10h
n2 db 22h

sum db ?
carry db 00h
.code
start: mov ax, @data
       mov ds,ax
mov al, n1
add al, n2
jnc skip
mov carry, 01h
skip: mov sum, al
```

```
mov ah,4ch
int 21h
end start
end code
end
```

Output:



2. Subtraction of two 8 bit numbers.

Algorithm:

Step 1 - Initialize the data segment with input numbers and memory location for the answer.

Step 2 - Start the program by loading the first data into the Accumulator.

Step 3 - Move the first number to register ‘al’ and move the second number to register ‘bl’.

Step 4 - Subtract the two register contents.

Step 5 - Store the content of al to ans

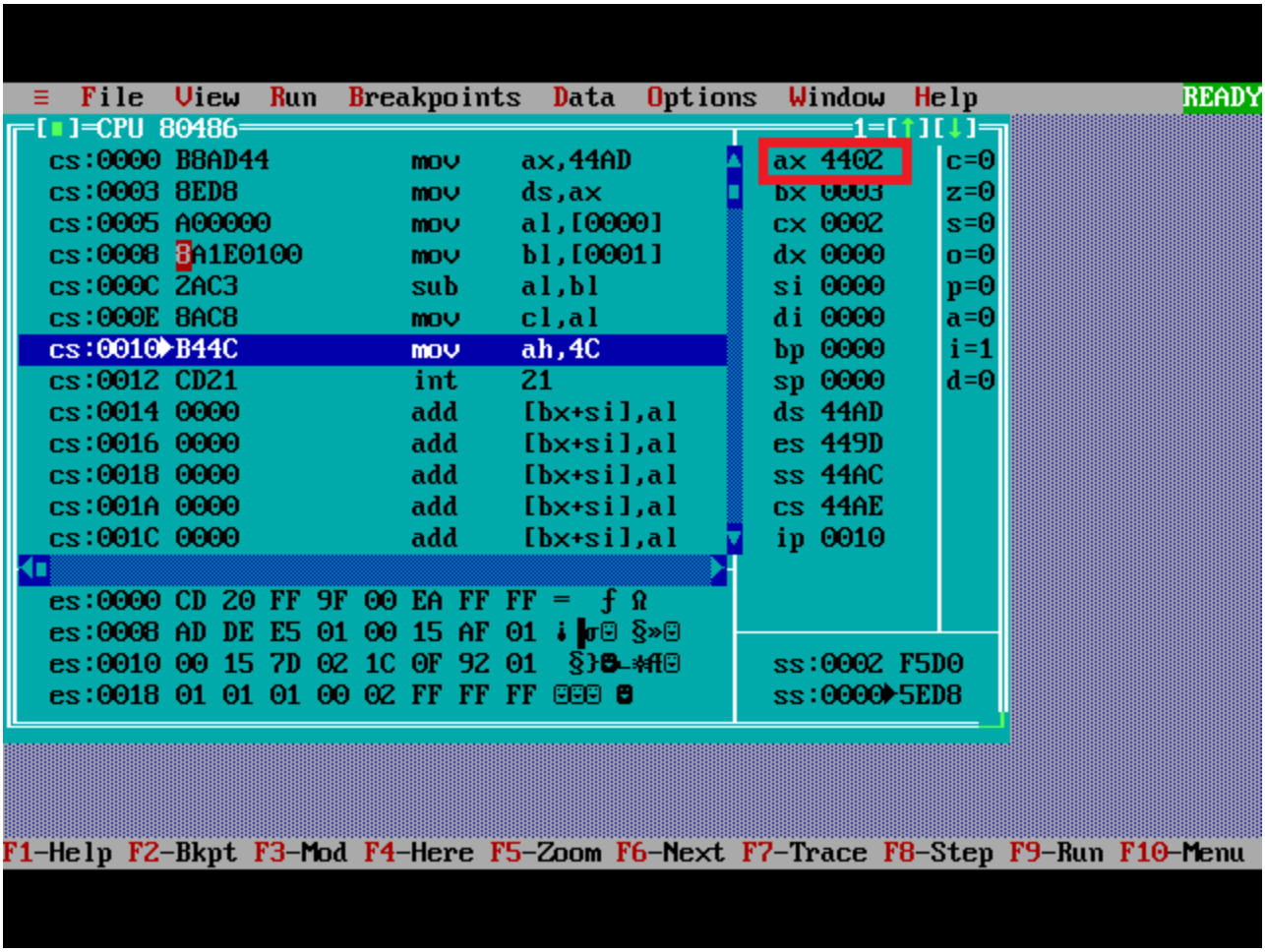
Step 6 - Terminate the program.

Code:

```
data segment
n1 db 05h
n2 db 03h
ans db ?
data ends
code segment
start : assume cs :code, ds :data
mov ax,data
mov ds,ax
mov al,n1
mov bl,n2
SUB al,bl
mov cl,al
mov ah, 4ch
int 21h
```

code ends
end start

Output:



3. Addition of two 16 bit numbers.

Algorithm:

Step 1 - Initialize the data segment with input numbers and memory location for the answer.

Step 2 - Start the program by loading the first data into the Accumulator.

Step 3 - Move the first number to register ‘ax’ and move the second number to register ‘bx’.

Step 4 - Add the two register contents.

Step 5 - Store the content of ‘ax’ to ‘c’.

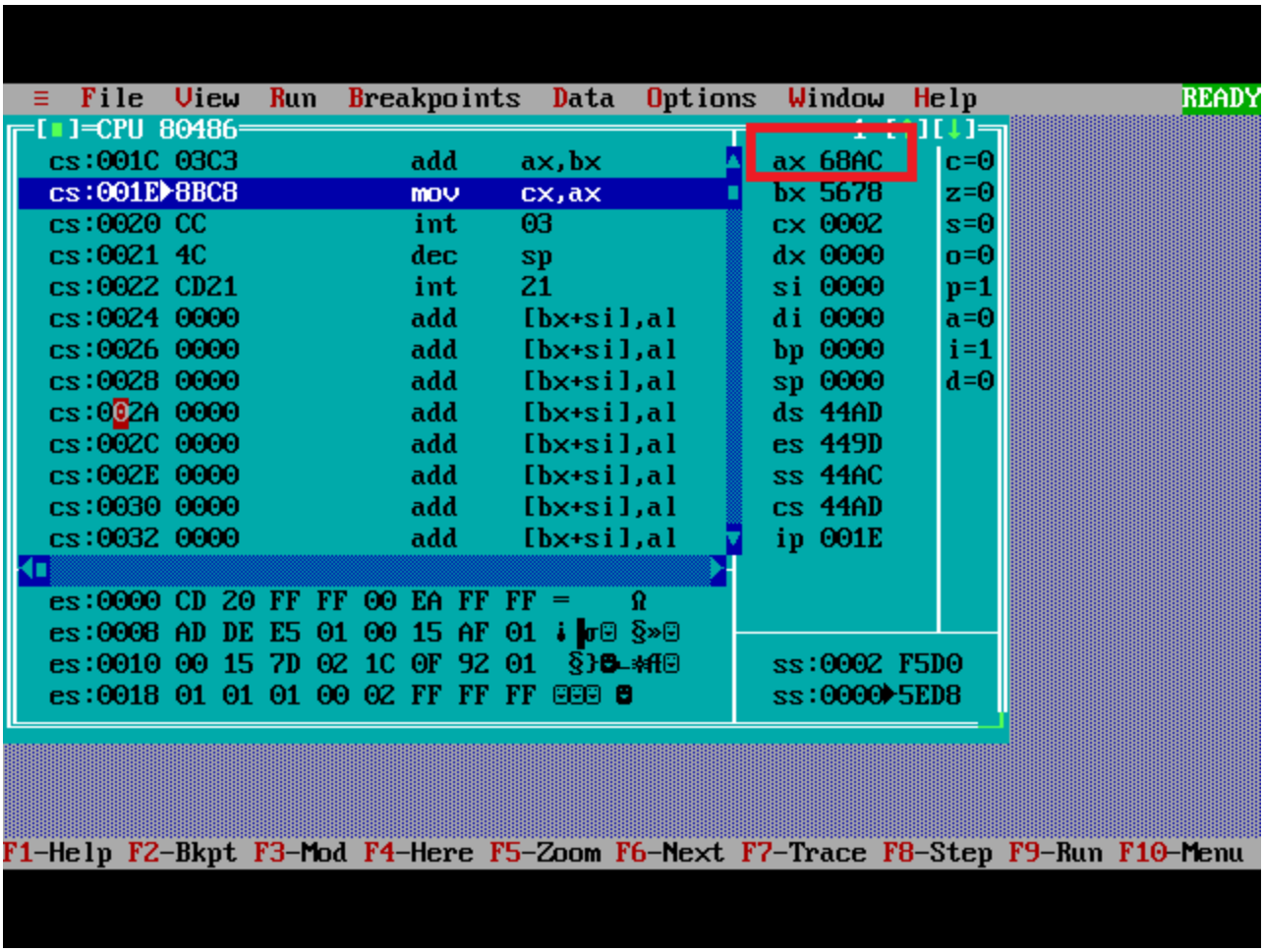
Step 6 - Terminate the program.

Code:

```
data segment
a dw 1234h
b dw 5678h
c dw ?
data ends
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov ax,a
mov bx,b
add ax,bx
mov cx,ax
int 3
code ends
```

end

Output:


File View Run Breakpoints Data Options Window Help
READY
CPU 80486
cs:001C 03C3 add ax,bx
cs:001E 8BC8 mov cx,ax
cs:0020 CC int 03
cs:0021 4C dec sp
cs:0022 CD21 int 21
cs:0024 0000 add [bx+si],al
cs:0026 0000 add [bx+si],al
cs:0028 0000 add [bx+si],al
cs:002A 0000 add [bx+si],al
cs:002C 0000 add [bx+si],al
cs:002E 0000 add [bx+si],al
cs:0030 0000 add [bx+si],al
cs:0032 0000 add [bx+si],al
ax 68AC
bx 5678
cx 0002
dx 0000
si 0000
di 0000
bp 0000
sp 0000
ds 44AD
es 449D
ss 44AC
cs 44AD
ip 001E
c=0
z=0
s=0
o=0
p=1
a=0
i=1
d=0
es:0000 CD 20 FF FF 00 EA FF FF
es:0008 AD DE E5 01 00 15 AF 01
es:0010 00 15 7D 02 1C 0F 92 01
es:0018 01 01 01 00 02 FF FF FF
ss:0002 F5D0
ss:0000 5ED8
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

4. Subtraction of two 16 bit numbers.

Algorithm:

Step 1 - Initialize the data segment with input numbers and memory location for the answer.

Step 2 - Start the program by loading the first data into the Accumulator.

Step 3 - Move the first number to register ‘ax’ and move the second number to register ‘bx’.

Step 4 - Subtract the two register contents.

Step 5 - Store the content of ‘ax’ to ‘c’.

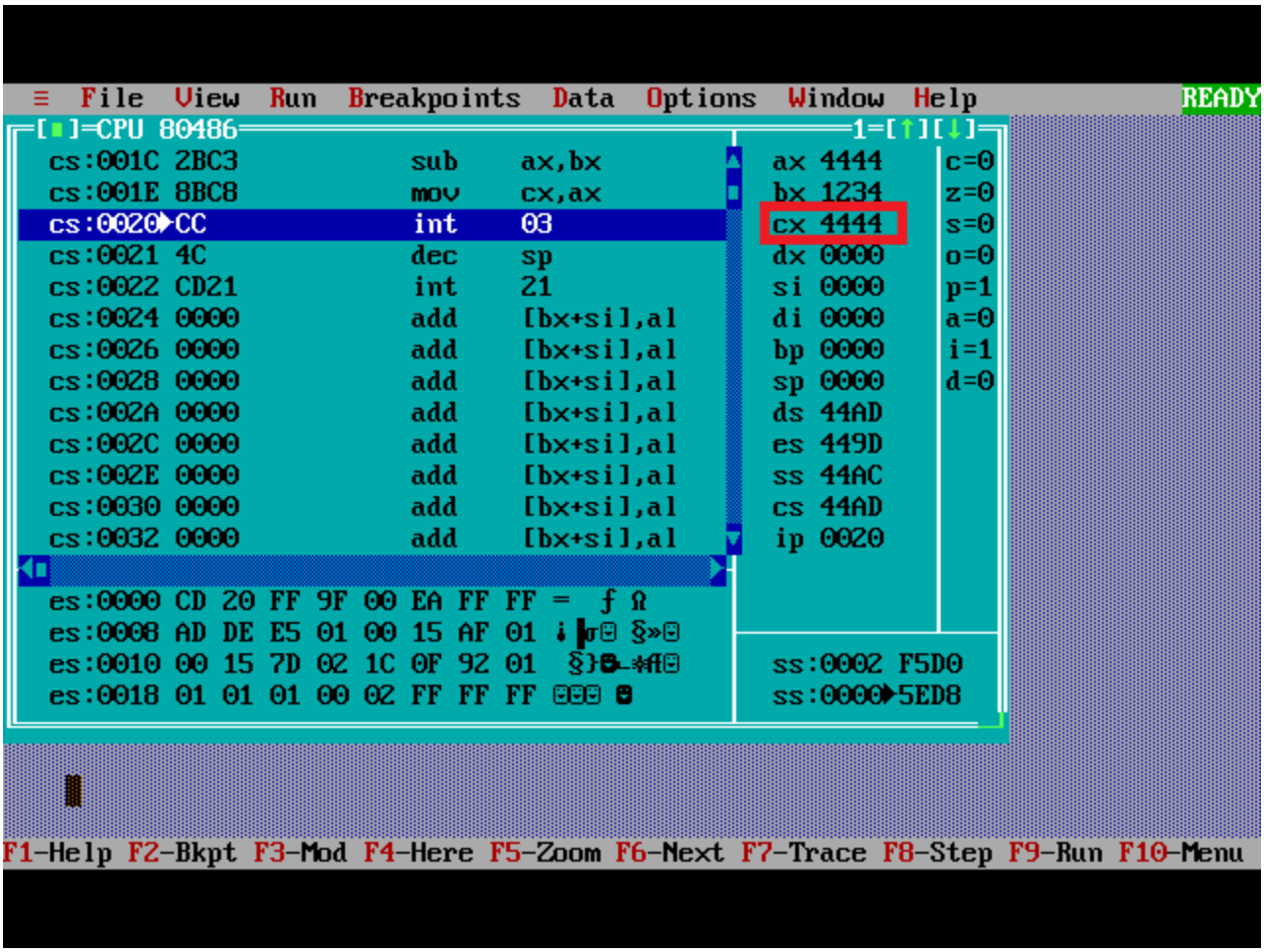
Step 6 - Terminate the program.

Code:

```
data segment
a dw 5678h
b dw 1234h
c dw ?
data ends
code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov ax,a
mov bx,b
sub ax,bx
mov cx,ax
int 3
code ends
```

end

Output:


File View Run Breakpoints Data Options Window Help
READY
CPU 80486
cs:001C 2BC3 sub ax,bx
cs:001E 8BC8 mov cx,ax
cs:0020 CC int 03
cs:0021 4C dec sp
cs:0022 CD21 int 21
cs:0024 0000 add [bx+si],al
cs:0026 0000 add [bx+si],al
cs:0028 0000 add [bx+si],al
cs:002A 0000 add [bx+si],al
cs:002C 0000 add [bx+si],al
cs:002E 0000 add [bx+si],al
cs:0030 0000 add [bx+si],al
cs:0032 0000 add [bx+si],al
ax 4444
bx 1234
cx 4444
dx 0000
si 0000
di 0000
bp 0000
sp 0000
ds 44AD
es 449D
ss 44AC
cs 44AD
ip 0020
c=0
z=0
s=0
o=0
p=1
a=0
i=1
d=0
es:0000 CD 20 FF 9F 00 EA FF FF
es:0008 AD DE E5 01 00 15 AF 01
es:0010 00 15 7D 02 1C 0F 92 01
es:0018 01 01 01 00 02 FF FF FF
ss:0002 F5D0
ss:0000 5ED8
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

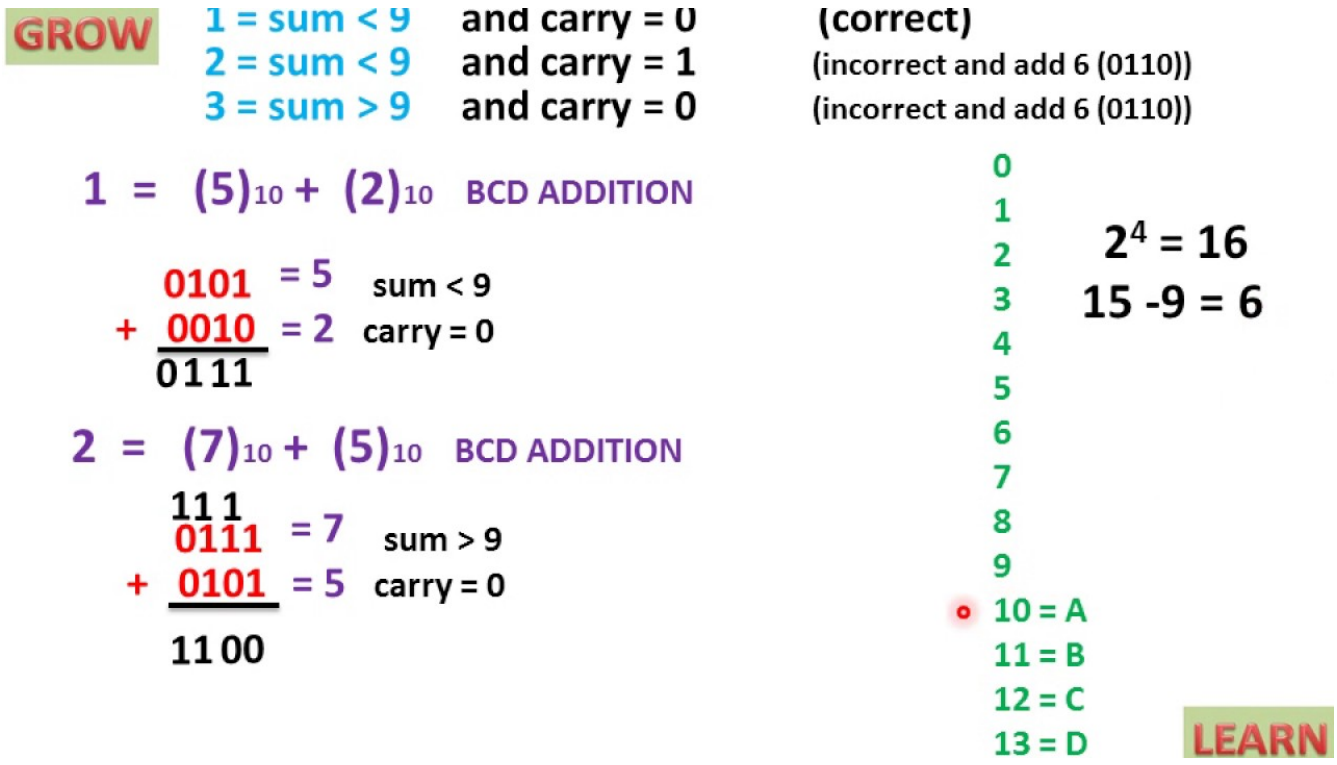
B.

BCD is a binary code of the ten decimal digits. It is not a binary equivalent.

To perform BCD addition

1. Add the BCD digits as regular binary numbers.
2. If the sum is 9 or less and no carry was generated, it is a valid BCD digit.
3. If the sum produces a carry, the sum is invalid and the number 6 (0110) must be added to the digit.
4. If the sum is greater than nine, the sum is invalid and the number 6 (0110) must be added to the digit.
5. Repeat for each of the BCD digits.

Ex.



1. BCD addition of two 8 bit numbers.

Algorithm:

Step 1 - Initialize the data section.

Step 2 - Load number1 in ‘al’ and load number2 in ‘bl’.

Step 3 - Add numbers and store the result in ‘al’.

Step 4 - Adjust result to valid BCD number. Also count the digits displayed.

Step 5 - Count to roll by 4 bits.And result in registered ‘bh’.

Step 6 - Roll ‘bl’ so that msb comes to lsb. Now load ‘dl’ with data to be displayed.

Step 7 - Get only lsb.

Step 8 - Check if digit is 0-9 or letter A-F. And if a letter is there then add 37H else only add 30H.

Step 9 - Function 2 under INT 21H (Display character).

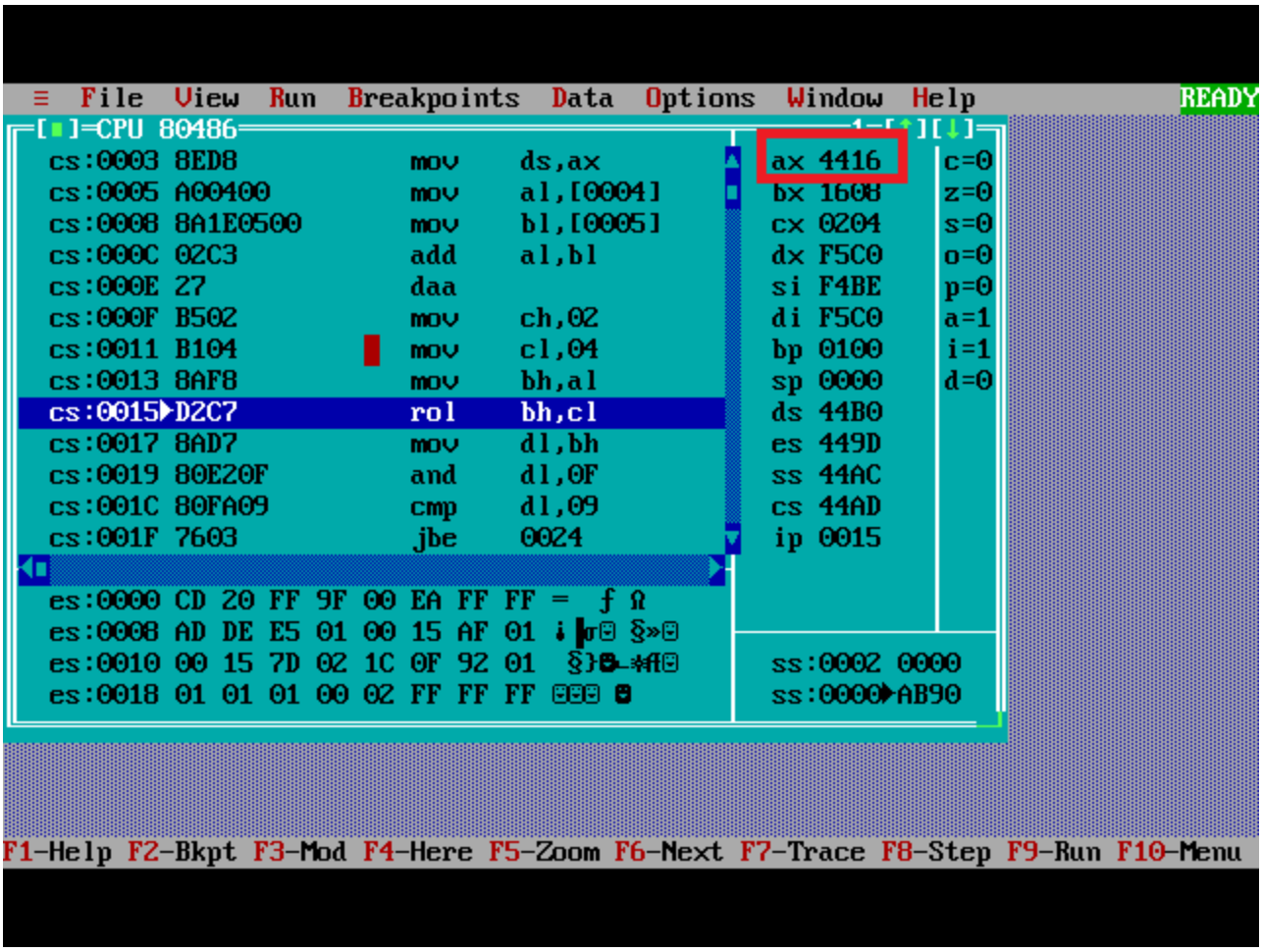
Step 10 - Decrement count and terminate the program.

Code:

```
.model small
.data
a db 08H
b db 08H
.code
    mov   ax, @data
    mov   ds, ax
    mov   al, a
    mov   bl, b
    add   al, bl
    daa
```

```
      mov     ch, 02h
      mov     cl, 04h
      mov     bh, al
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, 4cH
      int     21H
      end
```

Output:


File View Run Breakpoints Data Options Window Help READY
CPU 80486
cs:0003 8ED8 mov ds,ax
cs:0005 A00400 mov al,[0004]
cs:0008 8A1E0500 mov bl,[0005]
cs:000C 02C3 add al,bl
cs:000E 27 daa
cs:000F B502 mov ch,02
cs:0011 B104 mov cl,04
cs:0013 8AF8 mov bh,al
cs:0015 D2C7 rol bh,cl
cs:0017 8AD7 mov dl,bh
cs:0019 80E20F and dl,0F
cs:001C 80FA09 cmp dl,09
cs:001F 7603 jbe 0024
ax 4416 c=0
bx 1608 z=0
cx 0204 s=0
dx F5C0 o=0
si F4BE p=0
di F5C0 a=1
bp 0100 i=1
sp 0000 d=0
ds 44B0
es 449D
ss 44AC
cs 44AD
ip 0015
es:0000 CD 20 FF 9F 00 EA FF FF
es:0008 AD DE E5 01 00 15 AF 01
es:0010 00 15 7D 02 1C 0F 92 01
es:0018 01 01 01 00 02 FF FF FF
ss:0002 0000
ss:0000 AB90
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

2. BCD addition of two 16 bit numbers.

Algorithm:

Step 1 - Initialize data section. And load number1 in ‘ax’ and load number2 in ‘bx’.

Step 2 - Add lower two digits.And store the result in ‘al’.

Step 3 - Adjust result to valid bcd and store result in ‘bl’.

Step 4 - Add upper two digits. And store the result in ‘ah’.

Step 5 - ‘al’=’ah’ as daa works on ‘al’ only.

Step 6 - Adjust result to valid BCD. And store the result in ‘bh’.

Step 7 - Count the digits displayed.

Step 8 - Count to roll by 4 bits and roll ‘bl’ so that msb comes to lsb.

Step 9 - Now load ‘dl’ with data to be displayed and get only lsb.

Step 10 - Check if digit is 0-9 or letter A-F. If a letter is there then add 37H or else only add 30H.

Step 11 - Function 2 under INT 21H (Display character).

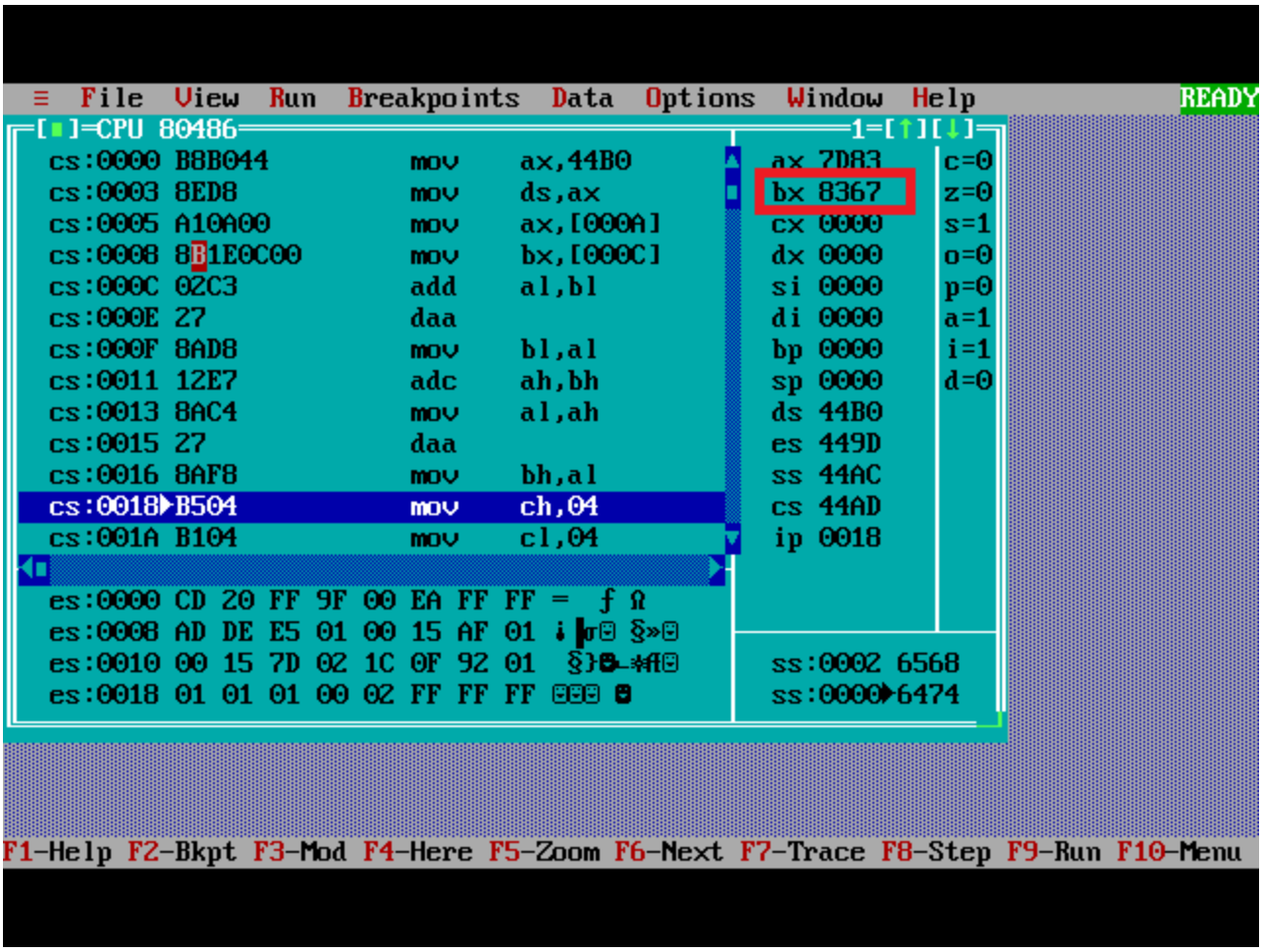
Step 12 - Decrement count and terminate the program.

Code:

```
.model small
 .data
 a dw 3629H
 b dw 4738H
 .code
    mov  ax, @data
```

```
    mov  ds, ax
    mov  ax, a
    mov  bx, b
    add   al, bl
    daa
    mov  bl, al
    adc   ah, bh
    mov  al, ah
    daa
    mov  bh, al
    mov  ch, 04h
    mov  cl, 04h
l2 : rol    bx, cl
    mov  dl, bl
    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, 4cH
    int    21H
    end
```

Output:



Conclusion:

We have studied and understood 8086 Assembly Language Programming.