Experiment 06

Experiment 6: Write an ALP program
1. Convert two digit Packed BCD to Unpacked BCD.
2. Program to evaluate given arithmetic expressions.
3. Program to evaluate given any logical expression of your choice.

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. Convert two digit Packed BCD to Unpacked BCD.
2. Program to evaluate given arithmetic expressions.
3. Program to evaluate given any logical expression of your choice.

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:
A. Convert two digit packed BCD to unpacked BCD.
Binary coded decimal (BCD) is a way to express each of the decimal digits with a binary code. This means that each decimal digit, 0 through 9, is represented by a binary code of four bits.
Eg: 98 => 10011000
Unpacking the BCD number is separating each BCD digit.
Eg: 98 can be separated as 09 and 08. So we can say 10011000 [98] is packed and 00001001 [09] & 00001000 [08] are unpacked.

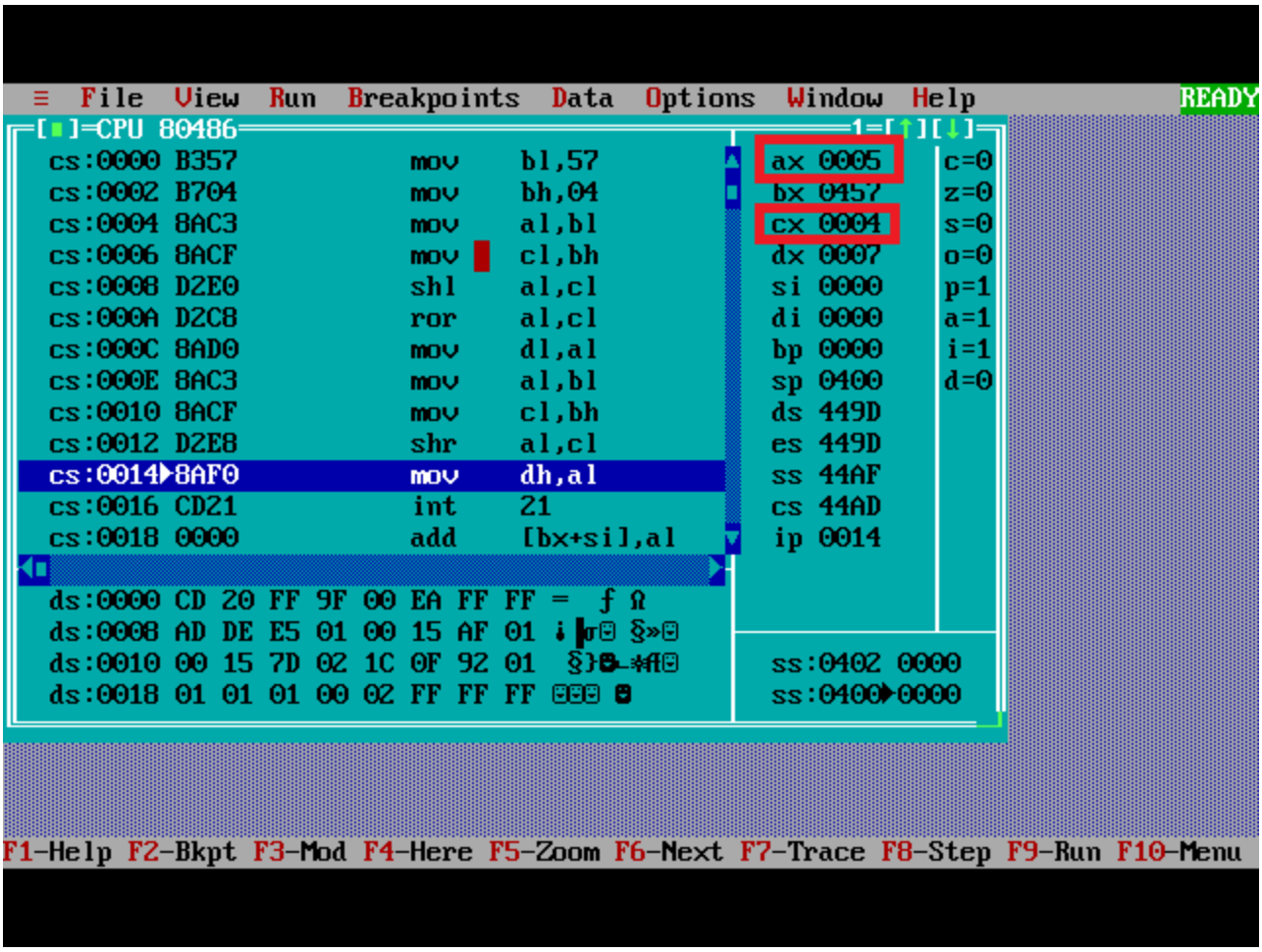
Algorithm:
Step 1 - Start and move packed data into the 'bl' register and count the value into the 'bh' register temporarily.
Step 2 - Copy packed BCD data from 'bl' to 'al' , also register count value from 'bh' to 'cl' register.
Step 3 - Perform shift left and rotate right operations by the 'al' with count number of times specified.

Step 4 - Copy lower byte of unpacked BCD data to ‘dl’ register and copy packed BCD data from ‘bl’ to ‘al’ register and count value from ‘bh’ to ‘cl’ registers.
Step 5 - Perform shift right operations by the ‘al’ with count number of times specified.
Step 6 - Copy higher byte of unpacked BCD data to ‘dh’ register.
Step 7 - Terminate the program.

Code:

```
.MODEL SMALL
.STACK
.DATA
.CODE
MOV BL, 57H
MOV BH, 04H
MOV AL, BL
MOV CL, BH
SHL AL, CL
ROR AL, CL
MOV DL, AL
MOV AL, BL
MOV CL, BH
SHR AL, CL
MOV DH, AL
INT 21H
END
```

Output:



B. Program to evaluate given arithmetic expression ((a+b) (b+c) (c+d))/(a+b+c+d):

Algorithm:

Step 1 - Start the program and declare data for variables A, B, C, D.

Step 2 - Move content of DATA to register ‘ax’ and move content of register ‘ax’ to ‘ds’.

Step 3 - Move the first number (ie A) to ‘al’ register and the second number (i.e B) to ‘bl’ register.

Step 4 - Add contents of ‘al’ and ‘bl’ (i.e A+B) and make the source index point towards memory location 1200H.

Step 5 - Move the content of AL (i.e A+B) to source index pointing towards location 1200H.

Step 6 - Move B to register ‘al’ and C to register ‘bl’.

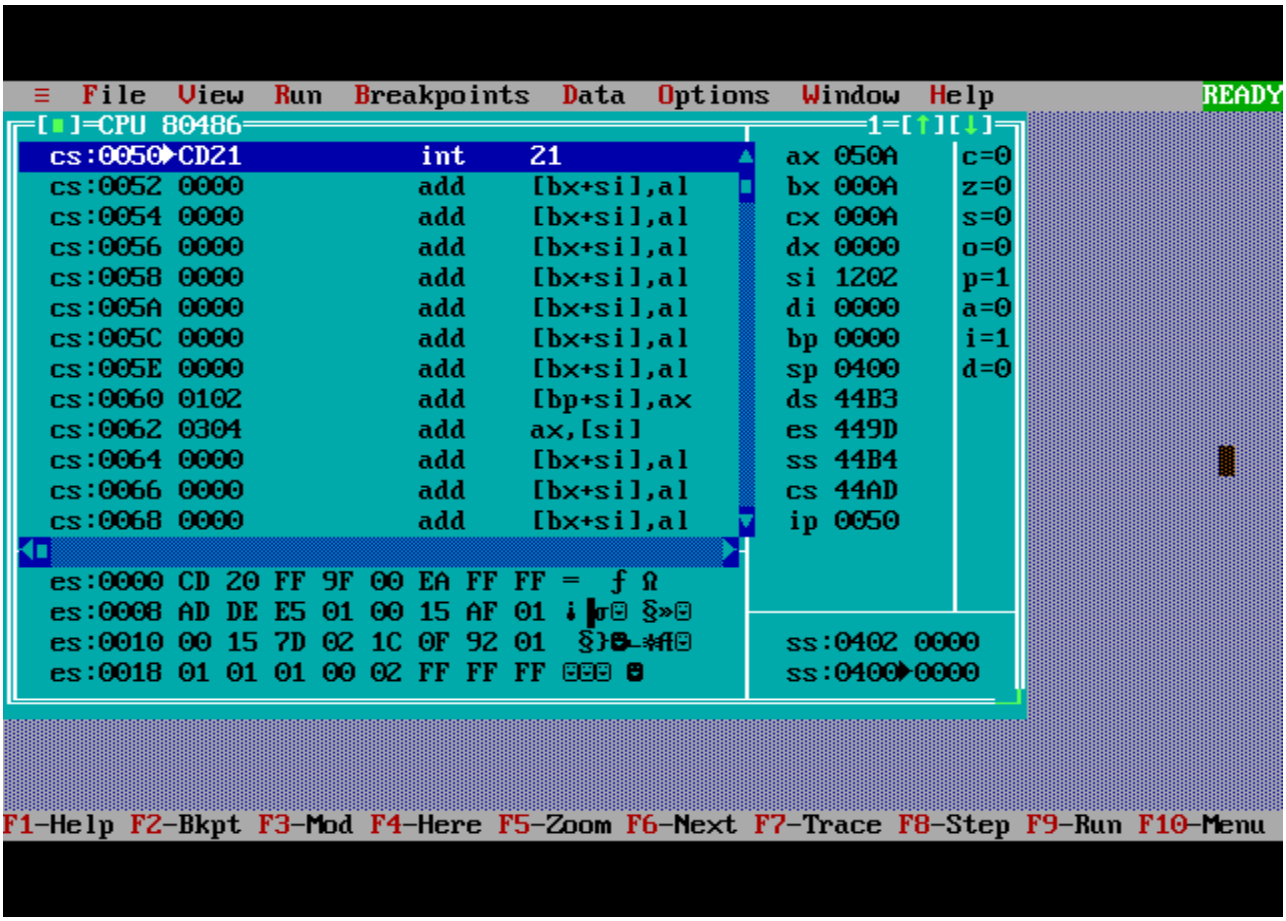
Step 7 - Similarly , do for (B+C) and (C+D) and store it in different SI locations i.e 1201H and 1202H respectively.
Step 8 - Move A to ‘al’ register and ADD ‘al’ and B (ie A+B) and store it back to ‘al’.
Step 9 - ADD ‘al’ and C (i.e A+B+C) and store it back to ‘al’.Similarly ADD ‘al’ and D (ie A+B+C+D) and store it back to ‘al’.
Step 10 - Move ‘al’ (ie A+B+C+D) to ‘cl’ , clear register ‘ax’
Step 11 - Move SI (ie C+D) to ‘al’ and move [SI-1] (i.e B+C) to ‘bl’.
Step 12 - Multiply ‘al’ and ‘bl’ and store it back to ‘al’ {i.e (C+D)*(B+C)}.
Step 13 - Move [SI-2] (i.e A+B) to ‘bl’ and multiply ‘al’ and ‘bl’ and store it back to ‘al’ {i.e (A+B)*(B+C)*(C+D)}
Step 14 - Move ‘bl’ to ‘cl’ and divide ‘al’,’bl’ {i.e (a+b) (b+c) (c+d) / (a+b+c+d)}
Step 15 - Terminate the program.

Code:

```
.MODEL SMALL
.STACK
.DATA
A DB 01H
B DB 02H
C DB 03H
D DB 04H
.CODE
MOV AX,@DATA
MOV DS,AX
XOR AX,AX
MOV AL,A
MOV BL,B
ADD AL,BL
MOV SI,1200H
MOV [SI],AL
MOV AL,B
```

```
MOV BL,C
ADD AL,BL
INC SI
MOV [SI],AL
MOV AL,C
MOV BL,D
ADD AL,BL
INC SI
MOV [SI], AL
MOV AL,A
ADD AL,B
ADD AL,C
ADD AL,D
MOV CL,AL
XOR AX, AX
MOV AL,[SI]
MOV BL,[SI-1]
MUL BL
MOV BL,[SI-2]
MUL BL
MOV BL,CL
DIV BL
INT 21H
END
```

Output:



C. Program to evaluate given any logical expression of your choice: (A⊕B) | (B&C) | (C⊕D).

Algorithm:

Step 1 - Start the program and declare data for variables A,B, C, D.

Step 2 - Move content of DATA to register ‘ax’ and content of register ‘ax’ to ‘ds’.

Step 3 - Clear register ‘ax’ and move first number (ie A) to ‘al’.

Step 4 - Move the second number (ie B) to ‘bl’ and perform XOR operation on ‘al’ and ‘bl’ (ie A&B).

Step 5 - Make source index point towards memory location 1200H and move the content of AL (ie A&B) to source index pointing towards location 1200H.

Step 6 - Move B to register ‘al’ and C to register ‘bl’.

Step 7 - Perform AND operation on ‘al’ and ‘bl’ (i.e B&C).

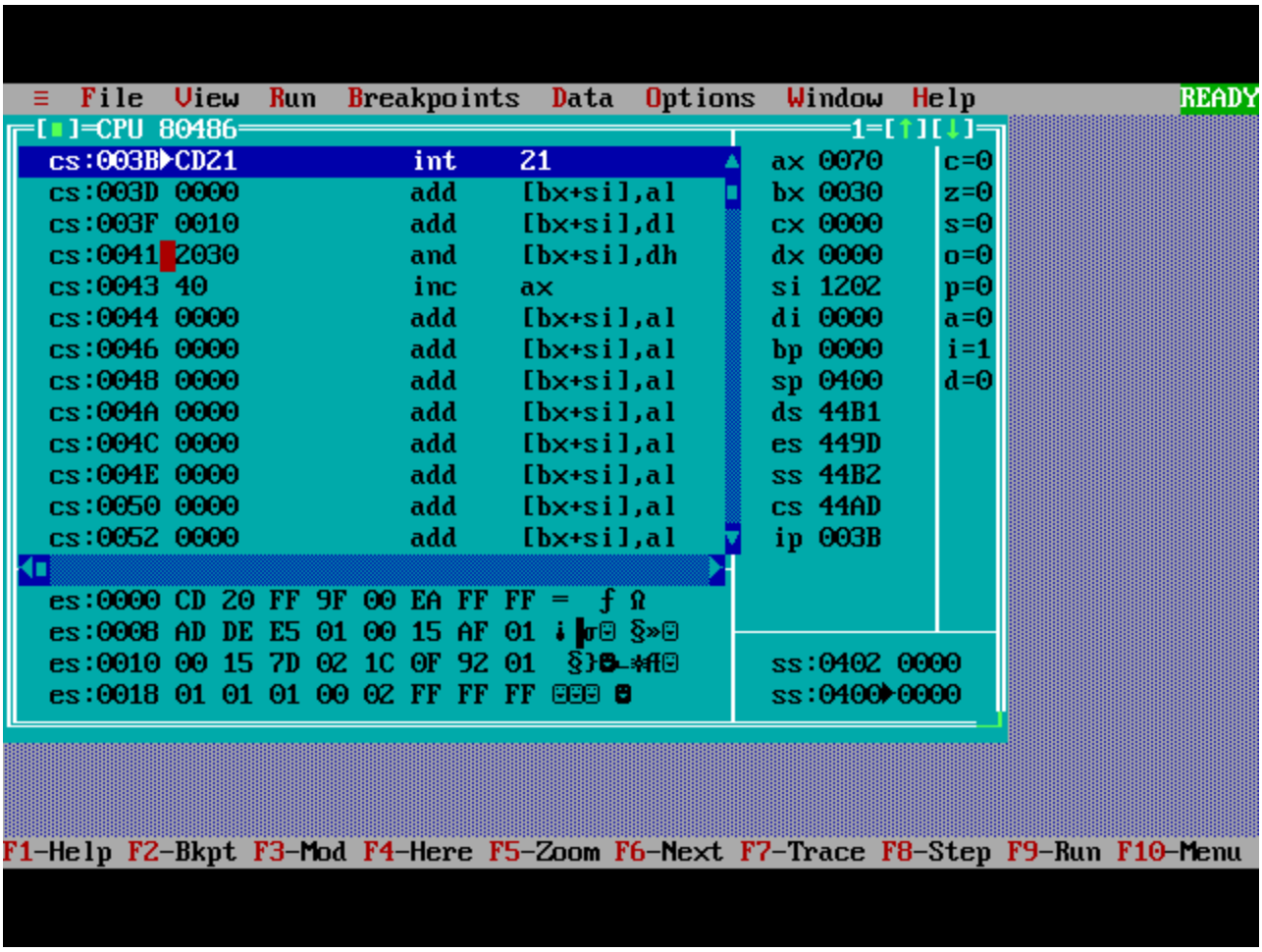
Step 8 - Increment Sl to location 1201H and move the content of ‘al’ (i.e B&C) to source index pointing towards location 1201H.
Step 9 - Move C to register ‘al’ and D to register ‘al’
Step 10 - Perform XOR operation on ‘al’ and ‘bl’ and move the content of ‘al’ to source index pointing towards location 1202H.
Step 10 - Move A to ‘al’ register and Clear ‘ax’ register.
Step 11 - Move SI to ‘al’. and [SI-1] to ‘bl’.
Step 12 - Perform OR operation on ‘al’ and ‘bl’.
Step 13 - Move [SI-2] to ‘bl’ and similarly perform OR operation on ‘al’ and ‘bl’ again.
Step 14 - Terminate the program.

Code:

```
.MODEL SMALL
.STACK
.DATA
A DB 10H
B DB 20H
C DB 30H
D DB 40H
.CODE
MOV AX,@DATA
MOV DS,AX
XOR AX,AX
MOV AL,A
MOV BL,B
XOR AL,BL
MOV SI,1200H
MOV [SI],AL
MOV AL,B
MOV BL,C
AND AL,BL
INC SI
```

```
MOV [SI],AL
MOV AL,C
MOV BL,D
XOR AL,BL
INC SI
MOV [SI], AL
XOR AX, AX
MOV AL, [SI]
MOV BL, [SI-1]
OR AL, BL
MOV BL,[SI-2]
OR AL,BL
INT 21H
END
```

Output:



Conclusion:

We have understood how to evaluate any logical expression and how to convert packed BCD to unpacked BCD using Assembly Language Programming.