WARREN FERNANDES SE COMPS B (8940)

Batch B

EXPERIMENT 5

OBJECTIVE: To use data transfer, Logical instructions and Rotate instructions, accepting input from keyboard and to understand basic concepts of ALP.

Aim: To perform code conversion

- Hex to BCD BCD to Hex
- ASCII to BCD
- BCD to ASCII

Theory:

Logical Instructions:

Logical Instructions are bit manipulation instructions. These instructions are used to perform logical operations like NOT, AND, OR, XOR They are also used to perform rotate or shift operations.

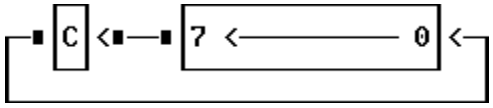
The **AND** operations is also called as a mask operation i.e. it can be used to mask a part of data without affecting the remaining part of data. For example, When a data 35H is ANDed with 0FH, the result is 05H. Hence upper nibble gets masked without affecting the lower nibble. Similarly any part of data can be masked using AND operation.

Rotate instructions:

RCL - Rotate Through Carry Left

Usage: RCL destination, count

Modifies Flags: CF OF

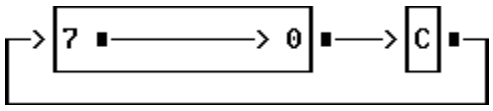


Rotates the bits in the destination to the left "count" times with all data pushed out the left side re-entering on the right. The Carry Flag holds the last bit rotated out.

RCR - Rotate Through Carry Right

Usage: RCR destination, count

Modifies Flags: CF OF

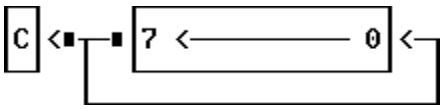

7
0
C

Rotates the bits in the destination to the right "count" times with all data pushed out the right side re-entering on the left. The Carry Flag holds the last bit rotated out.

ROL - Rotate Left

Usage: ROL destination ,count

Modifies Flags: CF OF

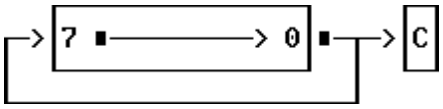


Rotates the bits in the destination to the left "count" times with all data pushed out the left side re-entering on the right. The Carry Flag will contain the value of the last bit rotated out.

ROR - Rotate Right

Usage: ROR destination ,count

Modifies Flags: CF OF



Rotates the bits in the destination to the right "count" times with all data pushed out the right side re-entering on the left. The Carry Flag will contain the value of the last bit rotated out.

Algorithm:

a. BCD to HEX

1. Initialize the data segment.
2. Move 8 bit data into any of the general purpose register
3. Perform Logical AND operation on it with F0h.
4. Rotate the resultant data four times to the right to get the upper nibble.
5. Multiply Upper nibble by 0A and store product in register dl.
6. Separate lower nibble by ANDing 0Fh and add it with the contents of dl register which holds the partial product.
7. Finally dl register will have the resultant HEX value.

b. Hex to BCD

1. Initialize the data segment.
2. Move 8 bit data into any of the general purpose register
3. Divide the 8 bit number by 0Ah
4. Rotate the quotient stored in AL register by 4 times to the right to make it upper nibble.
5. Add the rotated data with remainder stored in AH to get the resultant BCD equivalent.

c. BCD to ASCII

1. Initialize the data segment.
2. Move 8 bit data into any of the general purpose register
3. Perform Logical AND operation on it with 0Fh and store in AL
4. Perform Logical AND with original data with F0h and store the result in AH.
5. Add 3030h to get AX to get equivalent ASCII data.

d. ASCII to BCD

1. Initialize the data segment.
2. Move 8 bit data into any of the general purpose register
3. Subtract 30h from data to get equivalent BCD

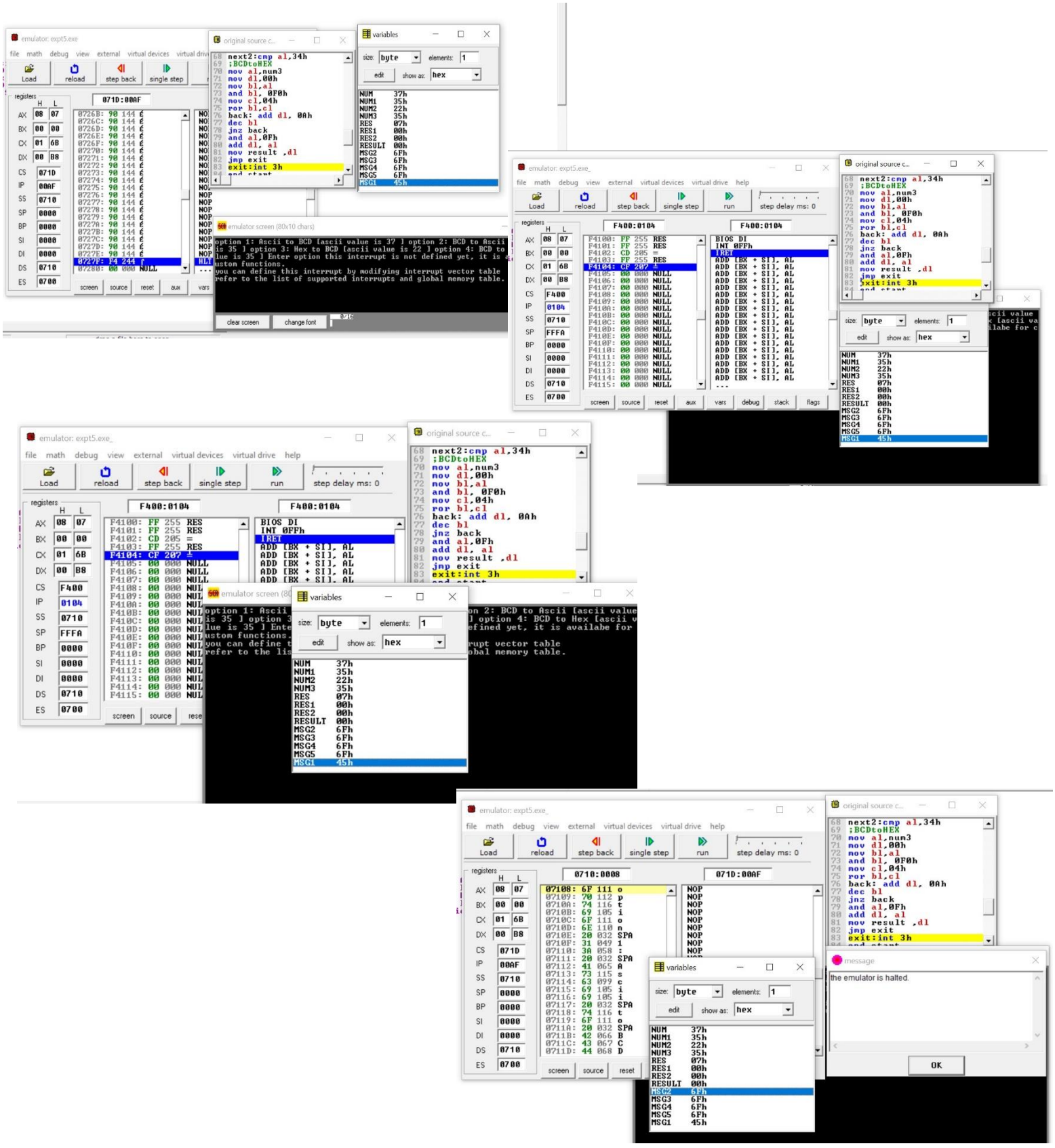
Post lab Questions:

1. Write any 10 Bit manipulation instructions with example
2. Explain 8086 in minimum and Maximum mode

```
.8086
.model  small

.data num   db
37h num1   db
35h num2   db
22h num3   db
35h

res    db    ?
res1   db   ?
res2  db    ?
result   db  ?

msg2  db  'option 1: Ascii  to BCD  [ascii  value is  37 ] $'
msg3  db  'option 2: BCD  to Ascii  [ascii  value is  35 ] $'
msg4 db  'option 3: Hex  to BCD  [ascii  value is  22 ] $'
msg5  db  'option 4: BCD  to Hex  [ascii  value is  35 ] $'
msg1 db  'Enter  option  $'

.code Start:
mov  ax,@data
mov  ds,  ax

lea  dx,msg2
mov  ah,09h
int 21h

lea  dx,msg3
mov  ah,09h
int 21h

lea  dx,msg4
mov  ah,09h
int 21h
```

```
lea dx,msg5
mov ah,09h
int 21h

lea dx,msg1
mov ah,09h
int 21h

mov ah,08h
int 21h

cmp al,31h jnz
next
;AsciiToBCD
mov al, num sub
al, 30h mov res
,al jmp exit next
:cmp al,32h jnz
next1

;BCDtoAscii
mov al,num1
and al,0F0h ror
al, 4 add al,30h
mov bl,num1 and
bl,0Fh add bl,30h
mov res1, al mov
res2, bl jmp exit
next1:cmp al,33h
jnz next2

;HextoDEC
mov al,num2
mov ah,00h
mov bl,0Ah
div bl ror
al,04h add
al,ah jmp
exit

next2:cmp al,34h
;BCDtoHEX mov
al,num3 mov
dl,00h mov bl,al
and bl, 0F0h
mov cl,04h
ror bl,cl
back: add dl, 0Ah
dec bl
jnz back and
al,0Fh add dl, al
mov result ,dl
jmp exit
```

```
exit:int  3h
end  start
```



Postlabs:

Q1.

Ans. Bit Manipulation Instructions

Name	Mnemonic	Explanation	Example
Clear	CLR	It will set the accumulator to 0 AC<-0	CLR
Complement	COM	It will complement the accumulator AC<-(AC)'	COM A
AND	AND	It will AND the contents of register B with the contents of accumulator and store it in the accumulator AC<-AC AND B	AND B
OR	OR	It will OR the contents of register B with the contents of accumulator and store it in the accumulator AC<-AC OR B	OR B
Exclusive-OR	XOR	It will XOR the contents of register B with the contents of the accumulator and store it in the accumulator AC<-AC XOR B	XOR B
Clear carry	CLRC	It will set the carry flag to 0 Carry flag<-0	CLRC
Set carry	SETC	It will set the carry flag to 1 Carry flag<-1	SETC
Complement carry	COMC	It will complement the carry flag Carry flag<- (Carry flag)'	COMC
Enable interrupt	EI	It will enable the interrupt	EI
Disable interrupt	DI	It will disable the interrupt	DI

Q2.

Ans.

Minimum Mode 8086 System

The microprocessor 8086 is operated in minimum mode by strapping its MN/MX pin to logic 1.

- In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system.
- The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices.
- Latches are generally buffered output D-type flip-flops like 74LS373 or 8282. They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086 Transreceivers are the bidirectional buffers and some times they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signals. They are controlled by two signals namely, DEN and DT/R.
- The DEN signal indicates the direction of data, i.e. from or to the processor.
- The system contains memory for the monitor and users program storage. Usually, EPROM are used for monitor storage, while RAM for users program storage. A system may contain I/O devices.
- The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle.
- The read cycle begins in T1 with the assertion of address latch enable (ALE) signal and also M / IO signal. During the negative going edge of this signal, the valid address is latched on the local bus.
- The BHE and A0 signals address low, high or both bytes. From T1 to T4 , the M/IO signal indicates a memory or I/O operation.
- At T2, the address is removed from the local bus and is sent to the output. The bus is then tristated. The read (RD) control signal is also activated in T2.

Maximum Mode 8086 System

In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.

- In this mode, the processor derives the status signal S2, S1, S0. Another chip called bus controller derives the control signal using this status information .
- In the maximum mode, there may be more than one microprocessor in the system configuration. The components in the system are same as in the minimum mode system.
- The basic function of the bus controller chip IC8288, is to derive control signals like RD and WR (for memory and I/O devices), DEN, DT/R, ALE etc. using the information by the processor on the status lines.
- The bus controller chip has input lines S2, S1, S0 and CLK. These inputs to 8288 are driven by CPU.
- It derives the outputs ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC and AIOWC. The AEN, IOB and CEN pins are specially useful for multiprocessor systems.
- AEN and IOB are generally grounded. CEN pin is usually tied to +5V. The significance of the MCE/PDEN output depends upon the status of the IOB pin.
- INTA pin used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.
- IORC, IOWC are I/O read command and I/O write command signals respectively . These signals enable an IO interface to read or write the data from or to the address port.
- The MRDC, MWTC are memory read command and memory write command signals respectively and may be used as memory read or write signals.
- All these command signals instructs the memory to accept or send data from or to the bus.