Fall Semester 2021-2022

Microprocessor and Interfacing

LAB FAT

Course Code: CSE2006

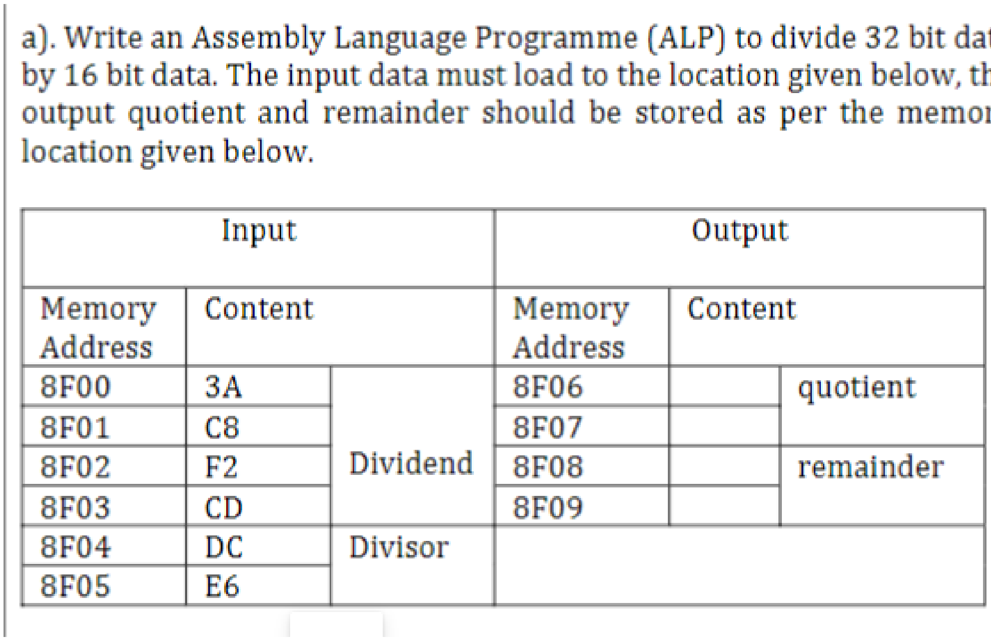
Slot: L7+L8

Submitted By: Alokam Nikhitha

Reg. Numb: 19BCE2555

Submitted To: Dr. Abdul Majed KK

SNO . 36



a). Write an Assembly Language Programme (ALP) to divide 32 bit data by 16 bit data. The input data must load to the location given below, the output quotient and remainder should be stored as per the memory location given below.

Input			Output		
Memory Address	Content		Memory Address	Content	
8F00	3A	Dividend	8F06		quotient
8F01	C8		8F07		
8F02	F2		8F08		remainder
8F03	CD		8F09		
8F04	DC	Divisor			
8F05	E6				

Aim

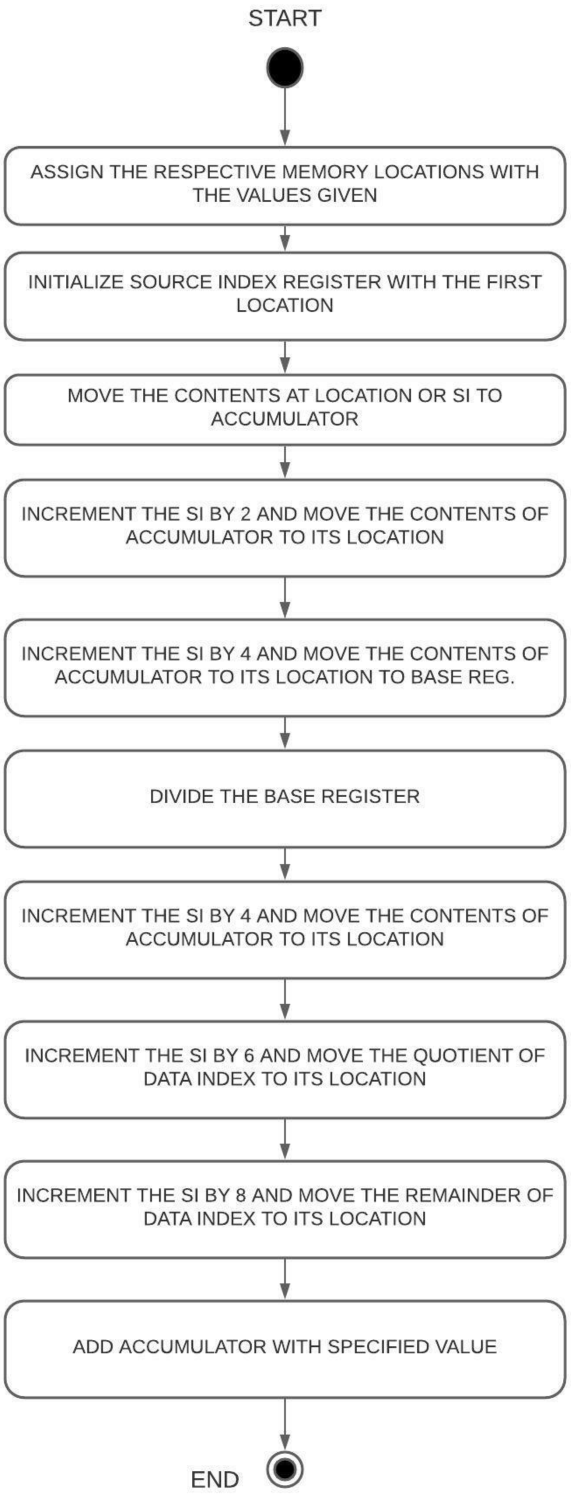
To write an Assembly code for Divison of 32 bit number with 16 bit number and store vaklues in given memory location

Algorithm:

1) **Move all the values in the specified memory locations.**
2) **Move the starting memory to SI register for reference.**
3) **Move the contents at location SI to accumulator register (AX).**
4) **Increment the SI value by 2 in order to point to the next memory location.**
5) **Increment the SI value by 4 and move the next contents to base register (BX).**
6) **Divide the base register. This will store the quotient in AX.**
7) **Move the contents of accumulator register to specified memory location**

8)The remainder is stored in Data register and we move it to the specified location

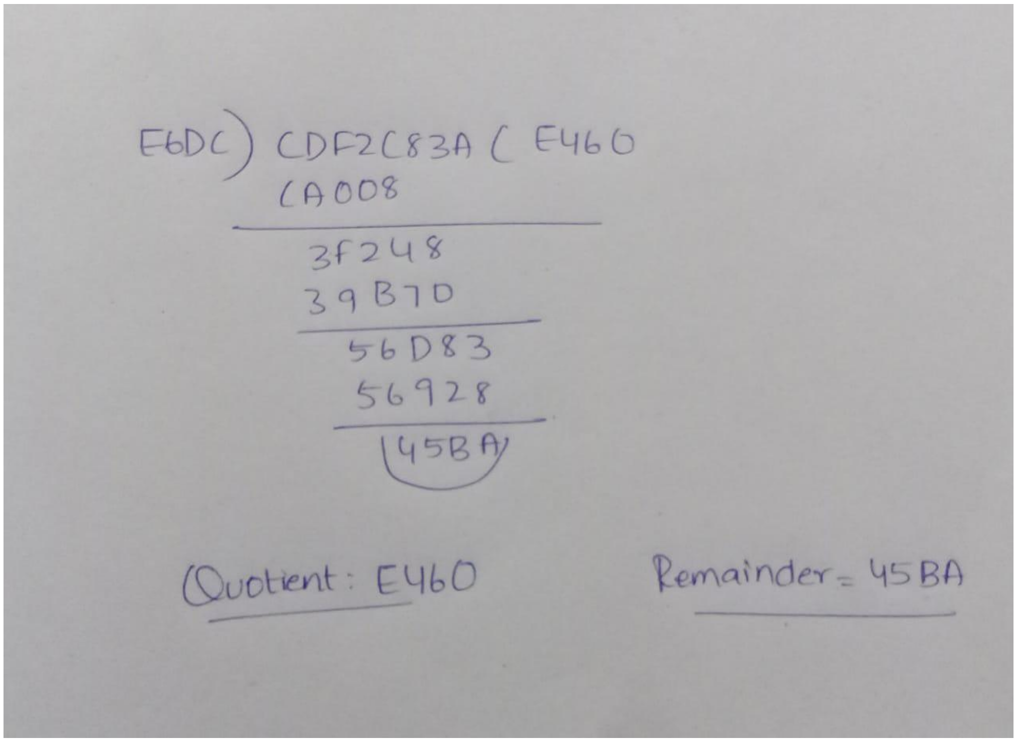
Flow Chart:


START
ASSIGN THE RESPECTIVE MEMORY LOCATIONS WITH THE VALUES GIVEN
INITIALIZE SOURCE INDEX REGISTER WITH THE FIRST LOCATION
MOVE THE CONTENTS AT LOCATION OR SI TO ACCUMULATOR
INCREMENT THE SI BY 2 AND MOVE THE CONTENTS OF ACCUMULATOR TO ITS LOCATION
INCREMENT THE SI BY 4 AND MOVE THE CONTENTS OF ACCUMULATOR TO ITS LOCATION TO BASE REG.
DIVIDE THE BASE REGISTER
INCREMENT THE SI BY 4 AND MOVE THE CONTENTS OF ACCUMULATOR TO ITS LOCATION
INCREMENT THE SI BY 6 AND MOVE THE QUOTIENT OF DATA INDEX TO ITS LOCATION
INCREMENT THE SI BY 8 AND MOVE THE REMAINDER OF DATA INDEX TO ITS LOCATION
ADD ACCUMULATOR WITH SPECIFIED VALUE
END

Design and Calculations:

Here we are going to need Source index register, accumulator, base register and the data register. The source index register is used as reference for the location to point at and stores the memory location of 8F00H. We then store the values mentioned to the specific locations of 8F00H, 8F01H, 8F02H, 8F03H, 8F04H, 8F05H. We then move the data to accumulator

We then divide the base register which stores the quotient in accumulator and the remainder in data register. Hence, we move the contents of accumulator and data register to the specified location of 8F06H and 8F08H.



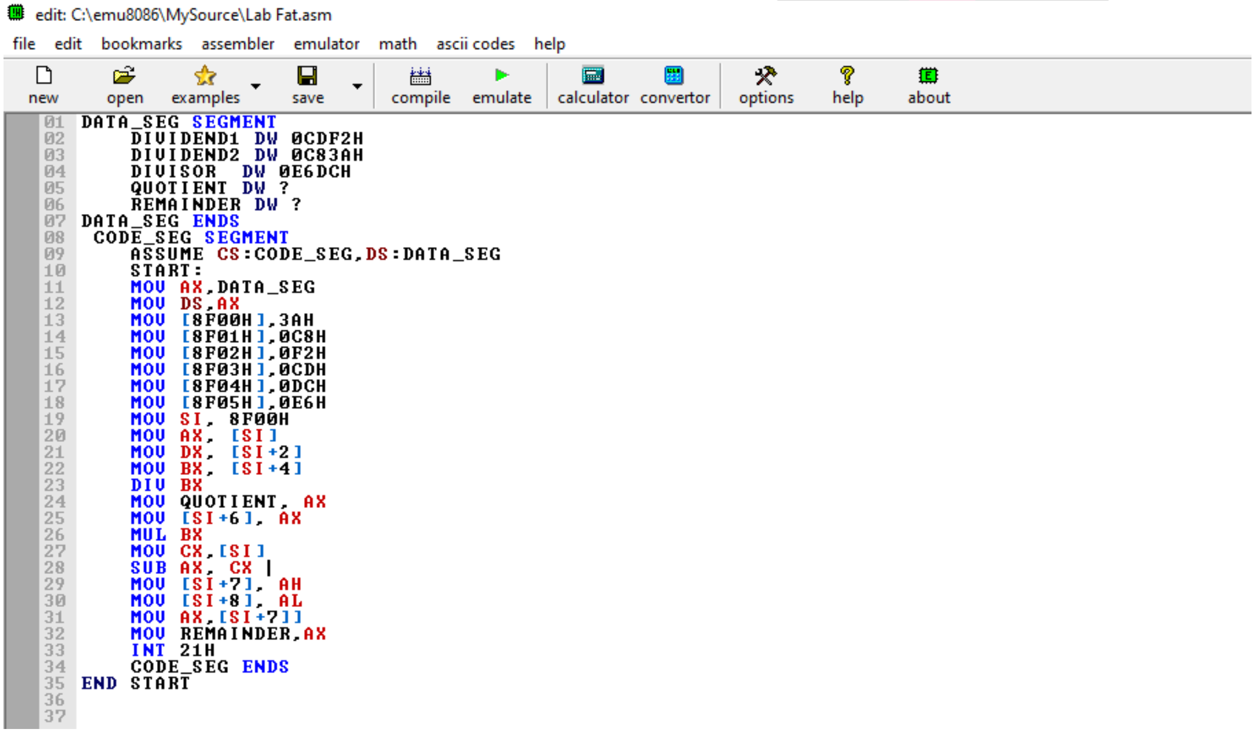
```
E6DC) CDF2C83A ( E460
      CA008
    ---------------
       3F248
       39B70
      ----------
         56D83
         56928
       ---------
         45BA
```

Quotient: E460

Remainder= 45BA

Program Code:

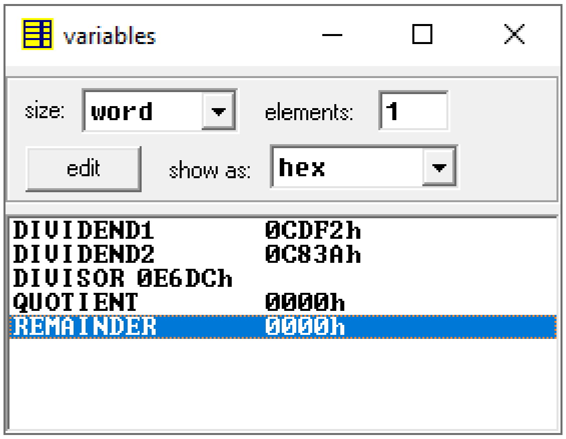
```
DATA_SEG SEGMENT
   DIVIDEND1 DW 0CDF2H
   DIVIDEND2 DW 0C83AH
   DIVISOR  DW 0E6DCH
   QUOTIENT DW ?
   REMAINDER DW ?
DATA_SEG ENDS
 CODE_SEG SEGMENT
   ASSUME CS:CODE_SEG,DS:DATA_SEG
   START:
   MOV AX,DATA_SEG
   MOV DS,AX
   MOV [8F00H],3AH
   MOV [8F01H],0C8H
   MOV [8F02H],0F2H
   MOV [8F03H],0CDH
   MOV [8F04H],0DCH
   MOV [8F05H],0E6H
   MOV SI, 8F00H
   MOV AX, [SI]
   MOV DX, [SI+2]
   MOV BX, [SI+4]
   DIV BX
   MOV QUOTIENT, AX
   MOV [SI+6], AX
   MUL BX
   MOV CX,[SI]
   SUB AX, CX
   MOV [SI+7], AH
   MOV [SI+8], AL
   MOV AX,[SI+7]]
   MOV REMAINDER,AX
   INT 21H
   CODE_SEG ENDS
END START
```


edit: C:\emu8086\MySource\Lab Fat.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

```
DATA_SEG SEGMENT
    DIVIDEND1 DW 0CDF2H
    DIVIDEND2 DW 0C83AH
    DIVISOR  DW 0E6DCH
    QUOTIENT  DW ?
    REMAINDER DW ?
DATA_SEG ENDS
 CODE_SEG SEGMENT
    ASSUME CS:CODE_SEG,DS:DATA_SEG
    START:
    MOV AX,DATA_SEG
    MOV DS,AX
    MOV [8F00H],3AH
    MOV [8F01H],0C8H
    MOV [8F02H],0F2H
    MOV [8F03H],0CDH
    MOV [8F04H],0DCH
    MOV [8F05H],0E6H
    MOV SI, 8F00H
    MOV AX, [SI]
    MOV DX, [SI+2]
    MOV BX, [SI+4]
    DIV BX
    MOV QUOTIENT, AX
    MOV [SI+6], AX
    MUL BX
    MOV CX,[SI]
    SUB AX, CX
    MOV [SI+7], AH
    MOV [SI+8], AL
    MOV AX,[SI+7]]
    MOV REMAINDER,AX
    INT 21H
    CODE_SEG ENDS
END START
```

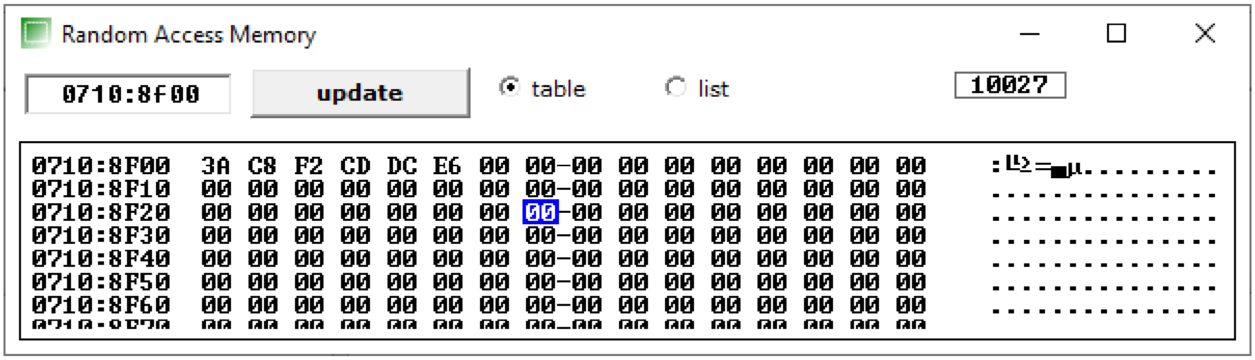
Before Emulation

Variables:



```
DIVIDEND1         0CDF2h
DIVIDEND2         0C83Ah
DIVISOR 0E6DCh
QUOTIENT          0000h
REMAINDER         0000h
```

Memory before Emulation

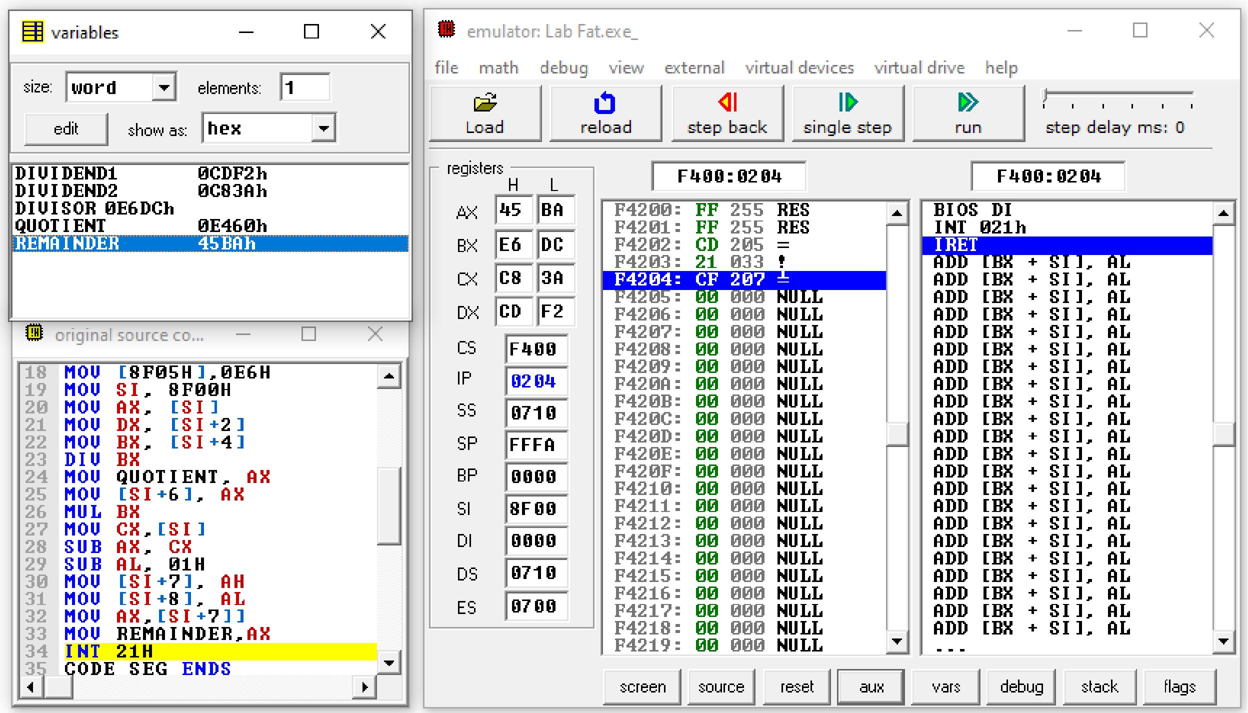


0710:8F00 — 10027

```
0710:8F00  3A C8 F2 CD DC E6 00 00-00 00 00 00 00 00 00 00   :╚≥═▄µ..........
0710:8F10  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:8F20  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:8F30  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:8F40  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:8F50  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:8F60  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0710:8F70  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

OUTPUT

After Emulation



size: word — elements: 1 — show as: hex

```
DIVIDEND1       0CDF2h
DIVIDEND2       0C83Ah
DIVISOR 0E6DCh
QUOTIENT        0E460h
REMAINDER       45BAh
```

```
MOV  [8F05H],0E6H
MOV  SI, 8F00H
MOV  AX, [SI]
MOV  DX, [SI+2]
MOV  BX, [SI+4]
DIV  BX
MOV  QUOTIENT, AX
MOV  [SI+6], AX
MUL  BX
MOV  CX,[SI]
SUB  AX, CX
SUB  AL, 01H
MOV  [SI+7], AH
MOV  [SI+8], AL
MOV  AX,[SI+7]]
MOV  REMAINDER,AX
INT  21H
CODE SEG ENDS
```

	H	L
AX	45	BA
BX	E6	DC
CX	C8	3A
DX	CD	F2
CS	F400	
IP	0204	
SS	0710	
SP	FFFA	
BP	0000	
SI	8F00	
DI	0000	
DS	0710	
ES	0700	

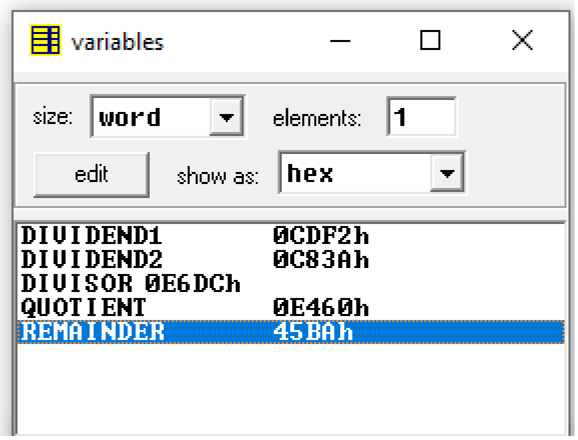
F400:0204

```
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
F4215:  00 000 NULL
F4216:  00 000 NULL
F4217:  00 000 NULL
F4218:  00 000 NULL
F4219:  00 000 NULL
```

F400:0204

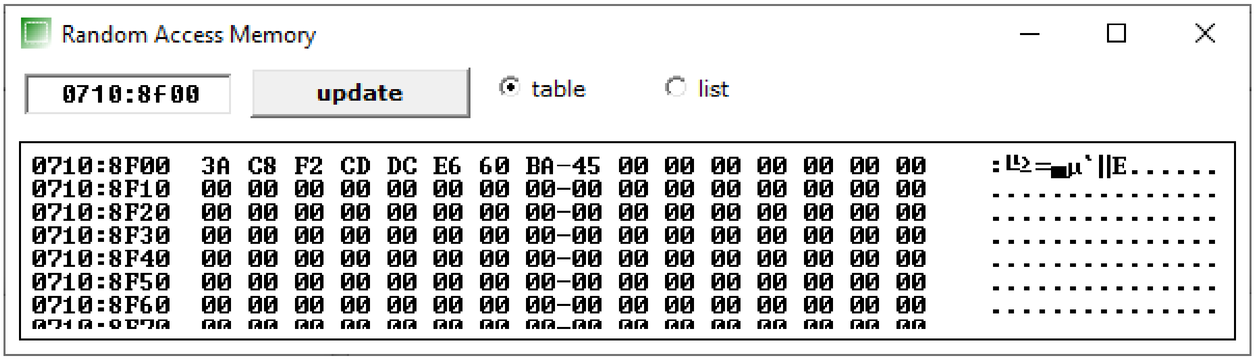
```
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
...
```

Variables after Emulation



```
DIVIDEND1        0CDF2h
DIVIDEND2        0C83Ah
DIVISOR 0E6DCh
QUOTIENT         0E460h
REMAINDER        45BAh
```

Memory after emulation



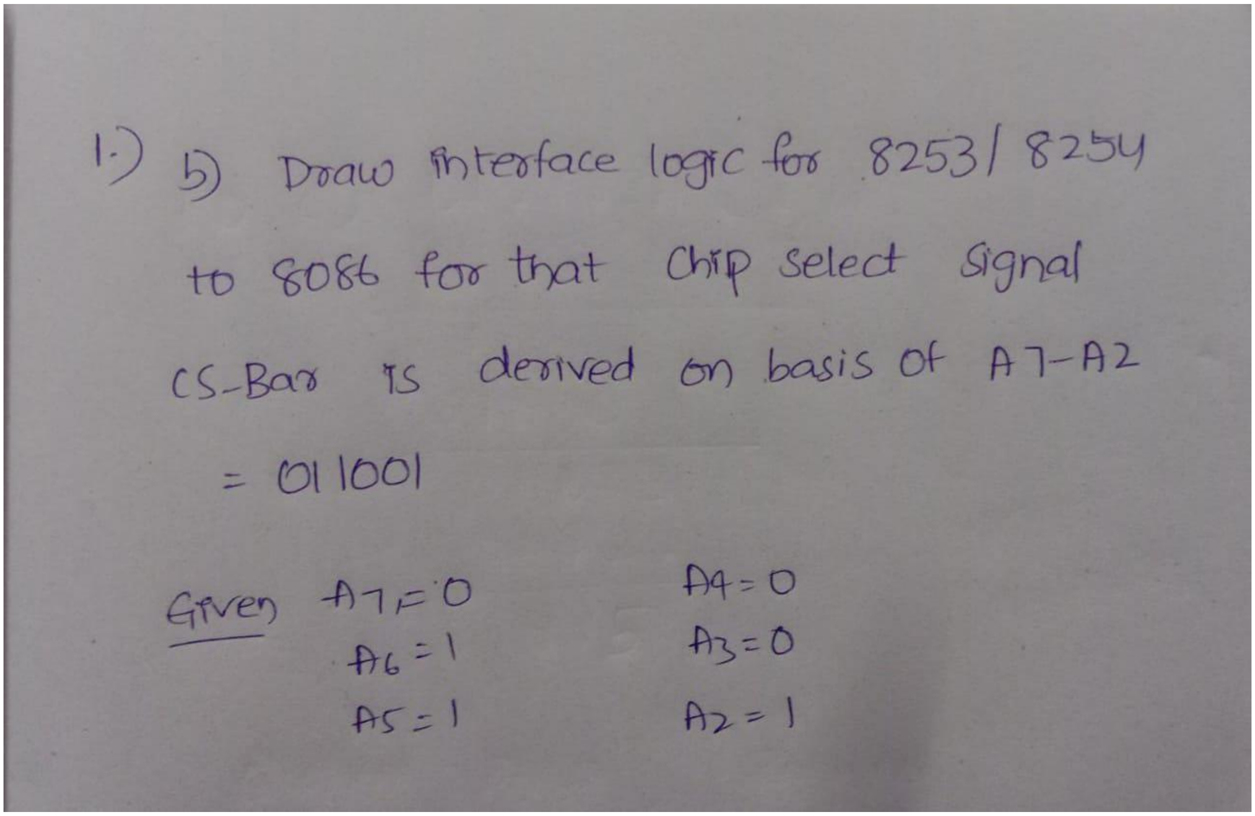
```
0710:8F00  3A C8 F2 CD DC E6 60 BA-45 00 00 00 00 00 00 00   :╚≥═▄µ`║E......
0710:8F10  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...............
0710:8F20  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...............
0710:8F30  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...............
0710:8F40  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...............
0710:8F50  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...............
0710:8F60  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...............
0710:8F70  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
```

Result and Inference:

- The accumulator initially had CDF2.
- The data register initially had C83A.
- The base register initially had E6DC
- The expected quotient of E460is stored in the memory location of 8F06H.
- The expected remainder of 45BAis stored in the memory location of 8F08H.
- Hence the quotient is E460 and remainder is 45BA as expected.

1 B)

1. b) . Draw the Interface logic for 8253/8254 to 8086 for that chip select signal (CS_Bar) is derived on the basis of A7-A2=011001

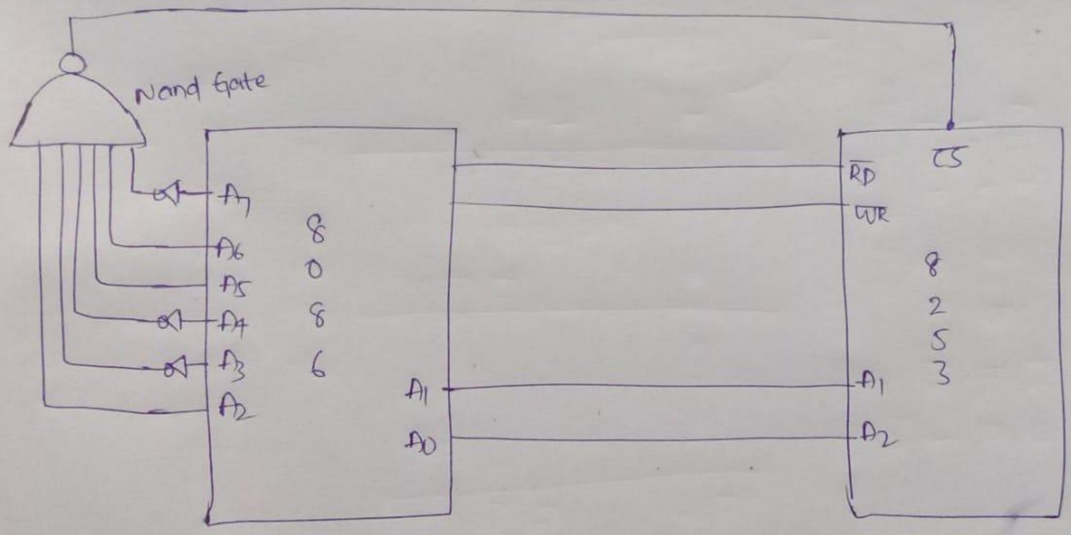


1.) b) Draw interface logic for 8253/ 8254 to 8086 for that Chip select Signal CS-Bar is derived on basis of A7-A2 = 011001

Given A7 = 0
A6 = 1
A5 = 1

A4 = 0
A3 = 0
A2 = 1

Interface logic



A7-A2 = 011001