

QUESTION 1 convert the hexadecimal number ABCDEF to binary and octal .

Answer : binary representation of the ABCDEF

10100 A

10110 B

11000 C

11001 D

11100 E

11110 F

To get the octal value makes the pair of three digit

101 001 011 011 000 110 011 110 011 110

2 0 1 1 0 3 1 3 1 3

So the octal representation of the no is (2011031313)

NOW THE REPRESENTATION OF THE HEXADECIMAL TO BINARY

Same rule will be followed

Now we collect all the binary value of ABCDEF

101001011011000110011110011110

Make the pairs for the binary representation of 4 bits

(0010 1001 0110 1100 0110 0111 1001 1110)

The binary representation of the hexadecimal no

(b): perform the arithmetic no operation (+28)+(-15) in binary using signed 2's complement representation for negative no .

Answer :

Signed value of(28) is 00111000

Signed value of (-15) is 10011110

Sum of the no are 10010110

The sum of the (28)+(-15) is 10010110

(c)compare static ram and dynamic ram.

Answer : STATIC RAM :it does not required refreshing .it is easier to use .it has less storage density .

DYNAMIC RAM : it needs periodic refreshing .it required extra control so it is difficult to use .it has higher storage density .

(d) explain the dma controller with the help of a block diagram . what is meant by a block transfer.

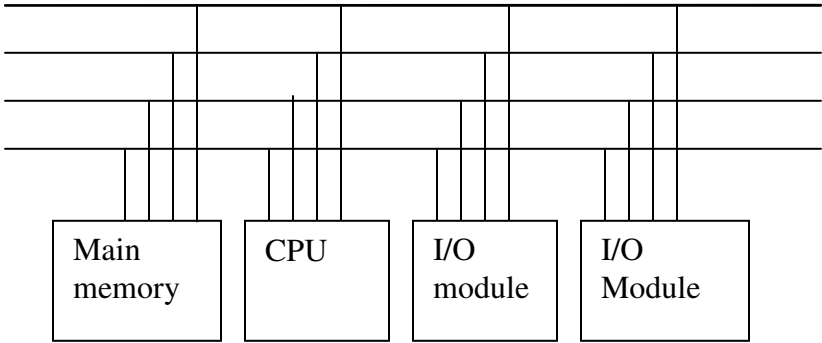
Answer: in dma as the name suggest the memory can be accessed directly by i/o module.thus overcome the drawback of programmed i\o and interrupt driven i\o and interrupt driven i/o where the cpu is responsible for extracting data from the memory for output & storing data in memory for input dma providing different information

(1) which operation (read/write) to be performed

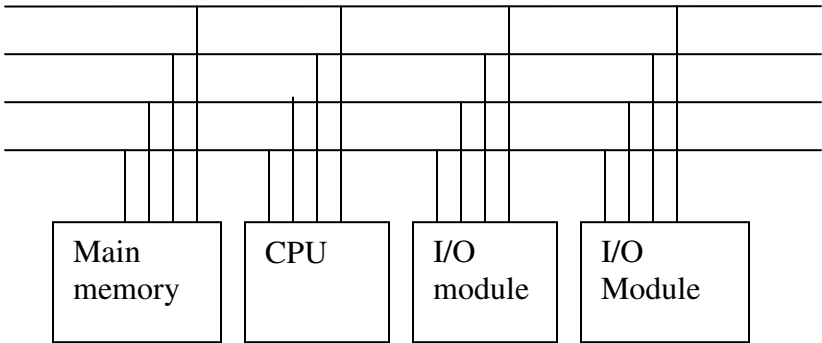
(2) the address of i/o device which is to be used .

(E) what is difference between memory mapped input /output and peripheral mapped input /output .

answer: MEMORY MAPPED : in memory mapped i/o the cpu treats the status and data register to i/o module. As memory location as a result memory & i/o device ca be accessed using the same instructions thenfigure show the memory mapped i/o system



PHERIPHIRAL MAPPED I/O: in pheriphiral mapped i/o the i/o device & memory are addressd and sepatly as in figure below

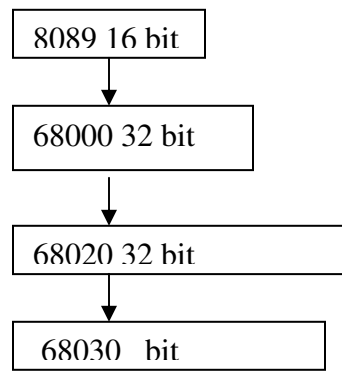


The figure shows that there are separate line for the memory & i/o device read or write .operation thus a memory refrence instruction does not affect on i/o device .

Question:2(b) what are the advantages of using segment registers in the 8086 microprocessor ?

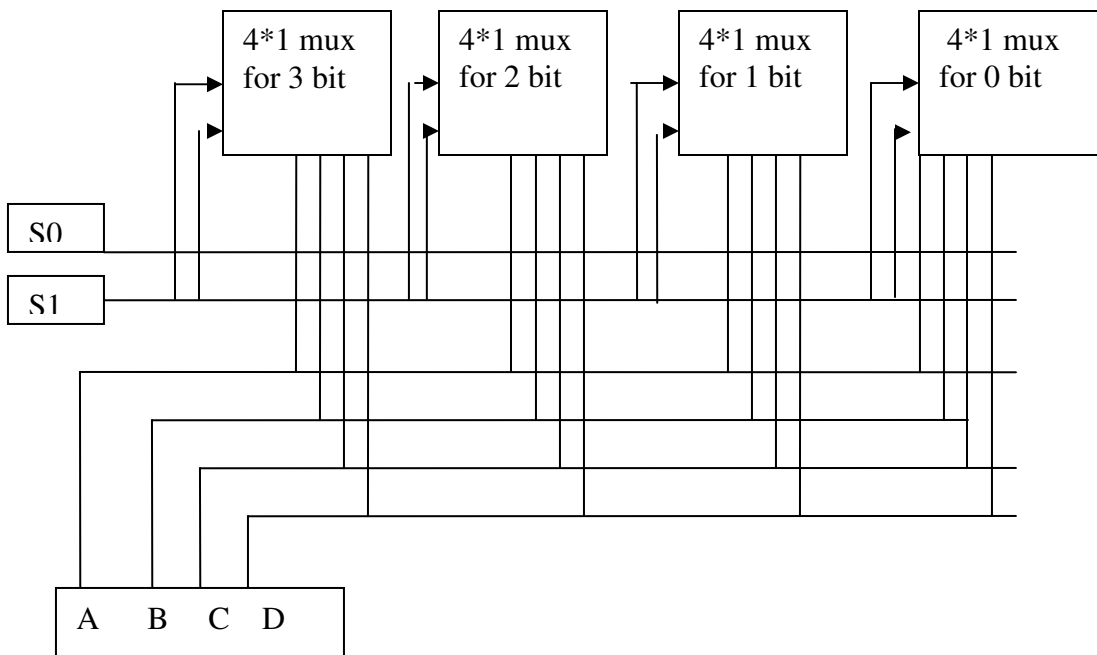
Answer: microprocessor :firstly developed in 1971 of intel . in 1993 intel announced Pentium with 64 bits microprocessor

Enhance version of microprocessor



Question 3 : design a 16 bit bus using registers and multiplexers and explain operation ?

Answer:



NOTE: Continue it to the 16 bit

Question 3(b) represent the number $(+64.6)_{10}$ as a floating point binary binary no with 24 bits . the normalized fraction mantissa has 16 and exponent has 8 bits .

Answer:

Here mantisa is 16 bit

Base is 10

And

Binary representation is 24 bits

Equation: $(+64.6)_{10}$

Now $(64)(0.6)_{10}$

$(64)(0.6) * 2^{127}$

NOTE: Now solve it

Question 4: 1) write a program in 8086 assembly to add 16 digit packed bcd numbers .

Answer:

```
Data segment source_str db 51h,21h,20h,33h,40h,45h,32h,35h.
```

```
dest_str db //dub 35h"$"
```

```
Data ends
```

```
Code segment
```

```
Assume cs:code ,data:data
```

```
Packed nbcd 51h,23h,25h.
```

```
ASCII values 35h,31h,32h,33h,35h.
```

```
Start :
```

```
Mov ax ,data
```

```
Mov ds ,dx
```

```
Mov si offset source_str
```

```
Mov si offset dest _str
```

```
 Mov ex_oah
```

```
Loop start :
```

```
Mov al [sj]
```

```
Mov bl ,al
```

```
Str al ,4
```

```
Add al,30h
```

```
Mov dest,of h
```

```
Add b1 ,30h
```

```
Int d1
```

```
Movdest _str[d1],bl
```

```
Loop loop start
```

```
Code ends
```

```
End start
```

Question 4)write a program in 8086 assembly language to do conversion 16 digit unpacked bcd number to a packed bcd number

Answer:

```
Data segment
```

```
Value 1 db 15h
```

```
Value 2 db 10h
```

```
Data ends
```

```
Code segment
```

```
Assume cs: code,ds: data
```

```
Start
```

```
Mov ax data
```

```
Mov ds ax
```

```
Mov al,value 1
```

```
Xchg value 2 ,al
```

```
Movvalue1,al
```

```
Mov ax 4c ool
```

```
Int 21h
```

Code ends ,
end start