**QUESTIONS & ANSWERS**

***An example of a self complementing code is :***

(A) 8421 code (B) Gray code (C) Excess-3 code(yes) (D) 7421 code

***The Excess-3 decimal code is a self-complementing code because***

(A) The binary sum of a code and its 9’s complement is equal to 9.(yes)

(B) It is a weighted code.

(C) Complement can be generated by inverting each bit pattern.(yes)

(D) The binary sum of a code and its 10’s complement is equal to 9.

***The size of the ROM required to build an 8-bit adder/subtractor with mode control, carry input, carry output and two’s complement overflow output is given as***

(A) 216 × 8 (B) 218 × 10(YES) (C) 216 × 10 (D) 218 × 8

**Explanation:-**total input to the rom decoder will be (8+8 ( two 8 bit number ) +1( mode ) +1( carry in))

so total number of words out of decoder will be 2^18 . result will be 8 bit so 8 vertical lines +( 1 for carry ) +1 ( for saying underflow).

***8-bit 1’s complement form of –77.25 is***

(A) 01001101.0100 (B) 01001101.0010 (C) 10110010.1011(yes) (D) 10110010.1101

Explanation: convert 77 into binary . i.e.. 01001101

convert 0.25 into binary. i.e.. 0100

binary equivalent of 77.25 = 01001101.0100

make 1's into 0's and 0's into 1's = 10110010.1011

***A latch is constructed using two cross-coupled***

(A) AND and OR gates (B) AND gates

(C) NAND and NOR gates (D) NAND gates(yes)

**Explanation:-**Latch can be designed either Nand or Nor gates. But not both nand and nor gates.

***multiplexer is a logic circuit that***

(A) accepts one input and gives several output

(B) accepts many inputs and gives many output

(C) accepts many inputs and gives one output(yes)

(D) accepts one input and gives one output

**Explanation:-** Multiplexer means many to one. Encoder means one to many

***The hexadecimal number equivalent to (1762.46)8 is***

(A) 3F2.89 (B) 3F2.98 (C) 2F3.89 (D) 2F3.98

**Explanation:-** convert octal into binary form. convert each digit into triplet form (1762.46) = 001111110010.100110

0011 1111 0010.1001 1000 = 3F2.98

***Negative numbers cannot be represented in***

(A) signed magnitude form (B) 1’s complement form

(C) 2’s complement form (D) none of the above(yes)

**Explanation:-** All of the options has both positive and negative representations

***How many 64 x 8 RAM chips are needed to provide a memory capacity of 2048 bytes?***

Assuming that 64 x 8 RAM chips means 64 x 8 bit RAM chips,Since 8 bits = 1 byte,

Each RAM chip has 64 x 1 byte = 64 bytes.

Thus the number of chips to address a memory capacity of 2048 bytes will be,

2048/64 = 32 chips.

***How many 128 x 8 RAM chips are needed to provide a memory capacity of 2048 bytes?***

A chip size is 128 x 8bit = 128 byte

For 2048 byte = 2014 / 128 = 16 chip

**How many 128×8 bit RAMs are required to design 32K×32 bit RAM?**

a) 512 b) 1024(Ans) c) 128 d) 32

Number of RAM = (32 \* K \* 32) / 128 \* 8

= 2^5 \* 2^10 \* 2^5 / 2^7 \* 2^3

= 2^20 / 2^10 = 1024

***A combinational logic circuit which is used to send data coming from a single source to two or more separate destinations is called a***

(A) decoder (B) encoder (C) multiplexer (D) demultiplexer(yes)

***If each address space represents one byte of storage space, how many address lines are needed to access RAM chips arranged in a 4 x 6 array, where each chip is 8K x 4 bits***

A. 13 B. 14 C. 16 D. 17(yes)

**Explanation1 :-**

As there are 4\*6 = 24 chips so (as 25>= 24 ) 5 bits are required to address them.

in each RAM number of bytes=(8k\*4)/8 = 4k

So to represent 4k we need(as 212=4096) 12 bits

Therefore to represent total structure we need 12+5=17 bits.

**Explanation2 :-**

Number of chips = 4 \* 6 = 24 =>To address chips need 5 bits

Chip is byte addressable and number of bytes in a chip = (8K x 4) / 8 = 210+2 = 212

=> To address bytes of chips needs 12 bits

=> So total 17 address lines need in corresponding to 17 bits.

Reference: <http://gateoverflow.in/15027/how-many-address-lines-are-needed-to-access-ram-chips>

***The opcode itself specifies all the required addresses is***

(A) indirect addressing (B) implied addressing

(C) inclined addressing (D) immediate addressing(yes)

***If F and G are Boolean functions of degree n. Then, which of the following is true ?***

(A) F ≤ F + G and F G ≤ F

(B) G ≤ F + G and F G ≥ G

(C) F ≥ F + G and F G ≤ F

(D) G ≥ F + G and F G ≤ F

**Explanation:-**

let the F & G be two boolean function of degree 1:

with degree n =2^2^n total boolean function

with degree n= 2^2^1=4 boolean function

F^1-->F & G^1-->G

F=( 4 boolean function)

G= (4 boolean function)

F+G=( 8 boolean function)

F\*G= (16 boolean function)

so answer is B

***Which one of the following is decimal value of a signed binary number 1101010, if it is in 2’s complement form ?***

(A) – 42 (B) – 22(yes) (C) – 21 (D) – 106

***If X is a binary number which is the power of 2, then the value of X&(X-1) is:***

A)11….11 B)00…..00 C)100…..0 D)000…..1

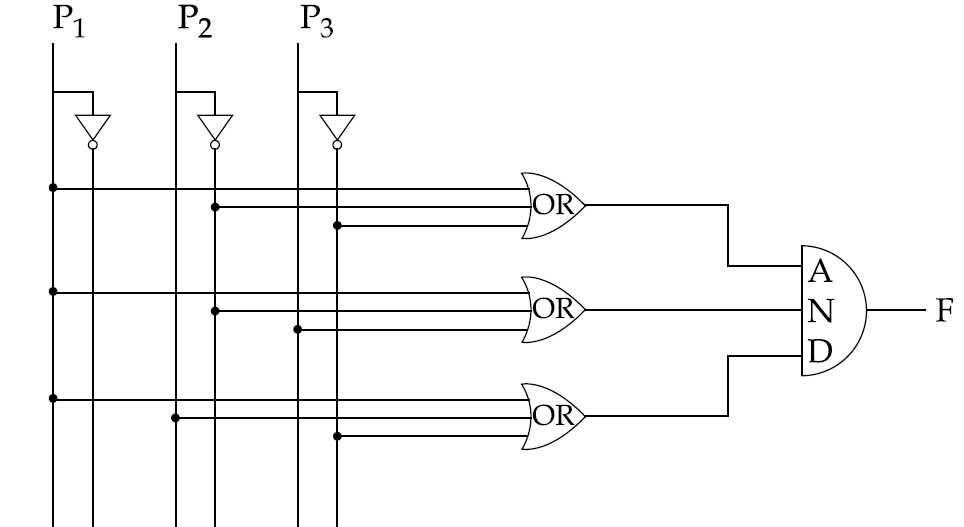
Explanation:-

let X=2^3=8=1000 then X-1=7=0111 now X&(X-1)=0000

(here & is bitwise AND= If both bits in the compared position of the bit patterns are 1, the bit in the resulting bit pattern is 1, otherwise 0)

so ans is B

***The output of the following combinational circuit is F.***



The value of F is :

(A) P +P P

(B) P +P P

(C) P +P P

(D) P +P P

Answer: (B)

Explanation: We know that (A+B)(A+C) = A + BC

ON the basis of Given question OR gates will produce output:

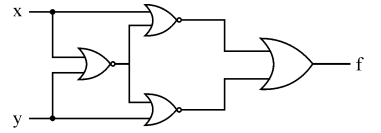
(P +P’ +P’ )

(P +P’ +P )

(P +P +P )

AND gate will take all three input produced by OR gate and generate (P + P’ P’ ) as output.

***Which logic operations is performed by the following given combinational of the following circuit ?***



A) EXCLUSIVE-OR B) EXCLUSIVE-NOR C) NAND D) NOR

**Explanation:-**

(X+(X+Y)')' + (Y+(X+Y)')'  
  
NOW USING DEMORGAN  LAW HERE   
  
X'(X+Y)+Y'(X+Y)  
  
XX'+X'Y+Y'X+Y'Y  
  
0+X'Y+Y'X+0  
  
X ex-or Y

***The simplified form of a boolean equation (AB'+AB'C+AC)(A'C'+B') is :***  
  
(1) AB'                      (2) AB'C (3) A'B                     (4) ABC  
  
**Explanation:**  
  
 =( AB' + AB'C + AC ) ( A'C' + B' )  
  
 =( AB'( 1 + C ) + AC ) ( A'C' + B' )                       [ x + 1 = 1 ]  
  
 =( AB' + AC ) ( A'C' + B' )  
  
 =( AA'B'C' + AB'B' + AA'CC' + AB'C )  
  
 =( 0 + AB' + 0 + AB'C )                                         [ x . x' = 0 ]  
  
 = AB'( 1 + C )  
  
 = AB'