| **UNIT-II** |
| --- |

**(MULTIPLE CHOICE QUESTIONS)**

| **S.NO** | **QUESTION** | **BLOOMS LEVEL** | **CLO** |
| --- | --- | --- | --- |
|  | Which method/s of representation of numbers occupies large amount of memory than others ?  **a) Sign-magnitude**  b) 1′s compliment  c) 2′s compliment  d) Both a and b |  |  |
|  | Which representation is most efficient to perform arithmetic operations on the numbers ?  a) Sign-magnitude  b) 1′s compliment  **c) 2′S compliment**  d) None of the above |  |  |
|  | Which method of representation has two representations for ’0′ ?  a) Sign-magnitude **b) 1′s compliment**  c) 2′s compliment  d) None of the above |  |  |
|  | When we perform subtraction on -7 and 1 the answer in 2′s compliment form is \_\_\_\_\_ .  a) 1010  b) 1110  **c) 0110**  d) 1000 |  |  |
|  | When we perform subtraction on -7 and -5 the answer in 2′s compliment form is \_\_\_\_\_ .  a) 11110  b) 1110  c) 1010  **d) 0010** |  |  |
|  | When we subtract -3 from 2, the answer in 2′s compliment form is \_\_\_\_\_\_\_ .  a) 0001  b) 1101  **c) 0101**  d) 1001 |  |  |
|  | The processor keeps track of the results of its operations using a flags called \_\_\_\_\_ .  **a) Conditional code flags**  b) Test output flags  c) Type flags  d) Status flags |  |  |
|  | The register used to store the flags is called as \_\_\_\_\_\_ .  a) Flag register  **b) Status registers**  c) Test register  d) Log register |  |  |
|  | The Flag ‘V’ is set to 1 indicates that,  **a) The operation is valid**  b) The operation is validated  c) The operation as resulted in an overflow  d) Both a and c |  |  |
|  | In some pipelined systems, a different instruction is used to add to numbers which can affect the flags upon execution. That instruction is \_\_\_\_\_\_\_ .  a) AddSetCC  b) AddCC  **c) Add++**  d) SumSetCC |  |  |
|  | The most efficient method followed by computers to multiply two unsigned numbers is \_\_\_\_\_\_\_ .  **a) Booth algorithm**  b) Bit pair recording of multipliers  c) Restoring algorithm  d) Non restoring algorithm |  |  |
|  | For the addition of large integers most of the systems make use of \_\_\_\_\_\_ .  a) Fast adders  b) Full adders  **c) Carry look-ahead adders**  d) Ripple adder |  |  |
|  | In a normal n-bit adder , to find out if an overflow as occurred we make use of \_\_\_\_\_ .  **a) And gate**  b) Nand gate  c) Nor gate  d) Xor gate |  |  |
|  | In the implementation of a Multiplier circuit in the system we make use of \_\_\_\_\_\_\_ .  a) Counter  b) Flip flop  **c) Shift register**  d) Push down stack |  |  |
|  | When 1101 is used to divide 100010010 the remainder is \_\_\_\_\_\_ . a) 101  **b) 11**  c) 0  d) 1 |  |  |
|  | The logic operations are implemented using \_\_\_\_\_\_\_ circuits. a) Bridge  b) Logical  c) Combinatorial  **d) Gate** |  |  |
|  | The carry generation function: ci + 1 = yici + xici + xiyi, is implemented in \_\_\_\_\_\_\_\_\_\_\_\_.  a) Half adders  **b) Full adders**  c) Ripple adders  d) Fast adders |  |  |
|  | The carry in the ripple adders,(which is true) a) Are generated at the beginning only.  b) Must travel through the configuration.  **c) Is generated at the end of each operation.**  d) None of the above |  |  |
|  | In full adders the sum circuit is implemented using \_\_\_\_\_\_\_\_. a) And & or gates  **b) NAND gate**  c) XOR  d) XNOR |  |  |
|  | The usual implementation of the carry circuit involves \_\_\_\_\_\_\_\_\_.  **a) And and or gates**  b) XOR  c) NAND  d) XNOR |  |  |
|  | **Problems in Multiplication** The product of 1101 & 1011 is a) 10001111  b) 10101010  **c) 11110000**  d) 11001100 |  |  |
|  | The product of -13 & 11 is a) 1100110011  **b) 1101110001**  c) 1010101010  d) 1111111000 |  |  |
|  | We make use of \_\_\_\_\_\_ circuits to implement multiplication. a) Flip flops  b) Combinatorial  **c) Fast adders**  d) Carry look ahead |  |  |
|  | The multiplier is stored in \_\_\_\_\_\_. a) PC Register  b) Shift register  **c) Cache**  d) IR |  |  |
|  | The \_\_\_\_\_\_ is used to co-ordinate the operation of the multiplier. **a) Controller**  b) Coordinator  c) Control sequencer  d) Program Counter |  |  |
|  | The method used to reduce the maximum number of summands by half is \_\_\_\_\_\_\_. a) Fast multiplication  **b) Bit-pair recording**  c) Quick multiplication  d) Carry Save Summand |  |  |
|  | The bits 1 & 1 are recorded as \_\_\_\_\_\_\_ in bit-pair recording. a) -1  **b) 0**  c) +1  d) both a and b |  |  |
|  | The multiplier -6(11010) is recorded as, a) 0-1-2  b) 0-1+1-10  **c) -2-10**  d) None of the above |  |  |
|  | The numbers written to the power of 10 in the representation of decimal numbers are called as \_\_\_\_\_. a) Height factors  b) Size factors  **c) Scale factors**  d) Space Factors |  |  |
|  | If the decimal point is placed to the right of the first significant digit, then the number is called as \_\_\_\_\_\_\_\_. a) Orthogonal  **b) Normalized**  c) Determinate  d) Diagonal |  |  |
|  | \_\_\_\_\_\_\_\_ constitute the representation of the floating number. a) Sign  **b) Significant digits**  c) Scale factor  d) All of the above |  |  |
|  | The sign followed by the string of digits is called as \_\_\_\_\_\_. a) Significant  b) Determinant  **c) Mantissa**  d) Exponent |  |  |
|  | ) In Booth’s algorithm, for Multiplier=1000 and Multiplicand=1100. How many number of cycles are required to get the correct multiplication result?  a. 4  **b. 5**  c. 3  d. 6 |  |  |
|  | In Booth’s algorithm, for Multiplier=100 and Multiplicand=1100. How many number of cycles are required to get the correct multiplication result?  a. 4  b. 5  **c. 3**  d. 6 |  |  |
|  | In IEEE 32-bit representations, the mantissa of the fraction is said to occupy \_\_\_\_\_\_ bits. **a) 24**  b) 23  c) 20  d) 16 |  |  |
|  | The normalized representation of 0.0010110 \* 2 ^ 9 is a) 0 10001000 0010110  b) 0 10000101 0110  **c) 0 10101010 1110**  d) 0 11110100 11100 |  |  |
|  | The 32 bit representation of the decimal number is called as \_\_\_\_. **a) Double-precision**  b) Single-precision  c) Extended format  d) None of the above |  |  |
|  | In 32 bit representation the scale factor as a range of \_\_\_\_\_\_\_\_. **a) -128 to 127**  b) -256 to 255  c) 0 to 255  d) -16 to 15 |  |  |
|  | In double precision format the size of the mantissa is \_\_\_\_\_\_. a) 32 bit  b) 52 bit  **c) 64 bit**  d) 72 bit |  |  |
|  | Which of the following is ordinary (average) multiplier in booth recoding multiplication?  a. 01010101  b. 00001111  **c. 11001100**  d. None of these |  |  |
|  | In booth recoding, M is multiplicand and -1 is booth recoded multiplier, then what will be the result of multiplication?  a. 1’s complement of M  **b. 2’s complement of M**  c. M  d. Right shift of M |  |  |
|  | In Booth’s algorithm, if Q0=0 and Q-1=0 then it will perform which operation,  **a. A=A-M**  b. A=A+M  c. Arithmetic right shift of A, Q and Q-1  d. A=M-A |  |  |
|  | In Booth’s algorithm, if Q0=1 and Q-1=1 then it will perform which operation,  a. A=A-M  **b. A=A+M**  c. Arithmetic right shift of A, Q and Q-1  d. A=M-A |  |  |
|  | In Booth’s algorithm, if Q0=1 and Q-1=0 then it will perform which operation,  a. A=A-M  b. A=A+M  **c. Arithmetic right shift of A, Q and Q-1**  d. A=M-A |  |  |
|  | In Booth’s algorithm, if Q0=0 and Q-1=1 then it will perform which operation,  a. A=A-M  b. A=A+M  c. Arithmetic right shift of A, Q and Q-1  **d. A=M-A** |  |  |
|  | What version of multiplicand will be selected if consecutive multiplier bits are 00?  **a. 0\*M**  b. +1\*M  c. -1\*M  d. 2\*M |  |  |
|  | What version of multiplicand will be selected if consecutive multiplier bits are 01?  a. 0\*M  **b. +1\*M**  c. -1\*M  d. -2\*M |  |  |
|  | )What version of multiplicand will be selected if consecutive multiplier bits are 10?  a. 0\*M  b. +1\*M  c. -1\*M  **d. 0\*M** |  |  |
|  | Which of the following is good multiplier in booth recoding multiplication?  a. 01010101  **b. 00001111**  c. 11001100  d. None of these |  |  |
|  | Which of the following is worst case multiplier in booth recoding multiplication?  **a. 01010101**  b. 00001111  c. 11001100  d. None of these |  |  |

**PART B**

**2 Marks with answers**

| **S.NO** | **QUESTION** | **BLOOMS LEVEL** | **CLO** |
| --- | --- | --- | --- |
| **1** | 1. Differentiate between restoring and non-restoring division |  |  |
| **2** | **2** Explain the design of a four bits carry look ahead adder circuit |  |  |
| **3** | **3** Add +5 and -9 using 2's compliment method |  |  |
| **4** | **4** Given Booth's algorithm to multiply two binary numbers, explain the working of the algorithm with an example. |  |  |
| **5** | **5** Explain with figure the design of a 4-bit carry look ahead adder |  |  |
| **6** | **6** With figure explain circuit arrangements for binary division. |  |  |
| **7** | **7** IEEE standard for floating point numbers, explain. |  |  |
| **8** | **8** Design 4 bit carry look ahead logic and explain how it is faster them 4 bit ripple adder |  |  |
| **9** | **9** Multiply 14 x - 8 using Booth's algorithm |  |  |
| **10** | **10** Explain normalization, excess - exponent and special values with respect to IEEE floating point representation |  |  |

**PART C**

**12 Marks (Only Question)**

| **S.NO** | **QUESTION** | **BLOOMS LEVEL** | **CLO** |
| --- | --- | --- | --- |
| **1** | **1** Discuss in detail Multiplication of positive numbers with Problem Solving |  |  |
| **2** | **2** Explain in detail Signed operand multiplication with Problem solving |  |  |
| **3** | **3** Explain in detail about Fast multiplication- Bit pair recoding of  Multipliers ,  Problem Solving |  |  |
| **4** | **4** Explain in detail about Carry Save Addition of summands,  Problem Solving |  |  |
| **5** | **5** Discuss in detail about Integer division – Restoring Division with Solving Problems |  |  |
| **6** | **6** Explain in detail Non Restoring Division with Solving Problems |  |  |
| **7** | **7** Discuss in detail about Floating point numbers and operations with  **Solving Problems** |  |  |
| **8** | **8** Explain in detail Addition and subtraction of Signed numbers with   Problem solving |  |  |
| **9** | **9** Discuss in detail about Design of fast adders, Ripple carry adder and Carry look ahead adder |  |  |