Problem 1

PART 1

In this section, given the dataword, we are expected to find the codeword. From the table, we have

C [0] = d [6] ^ d [4] ^ d [3] ^ d [1] ^ d [0]
C [1] = d [6] ^ d [5] ^ d [3] ^ d [2] ^ d [0]
C [2] = d [7] ^ d [3] ^ d [2] ^ d [1]
C [3] = d [7] ^ d [6] ^ d [5] ^ d [4]

a. Dataword: **0000 0010**
   From equations above
   C [0] = 0 ^ 0 ^ 0 ^ 1 ^ 0 = 1
   C [1] = 0 ^ 0 ^ 0 ^ 0 ^ 0 = 0
   C [2] = 0 ^ 0 ^ 0 ^ 1 = 1
   C [3] = 0 ^ 0 ^ 0 ^ 0 = 0

   Codeword: **0101 00000010**

b. Dataword: **1010 1001**
   From equations above
   C [0] = 0 ^ 0 ^ 1 ^ 0 ^ 1 = 0
   C [1] = 0 ^ 1 ^ 1 ^ 0 ^ 1 = 1
   C [2] = 1 ^ 1 ^ 0 ^ 0 = 0
   C [3] = 1 ^ 0 ^ 1 ^ 0 = 0

   Codeword: **0010 10101001**

c. Dataword: **0110 1110**
   From equations above
   C [0] = 1 ^ 0 ^ 1 ^ 1 ^ 0 = 1
   C [1] = 1 ^ 1 ^ 1 ^ 1 ^ 0 = 0
   C [2] = 0 ^ 1 ^ 1 ^ 1 = 1
   C [3] = 0 ^ 1 ^ 1 ^ 0 = 0

   Codeword: **0101 01101110**

d. Dataword: **1101 1011**
   From equations above
   C [0] = 1 ^ 1 ^ 1 ^ 1 ^ 1 = 1
   C [1] = 1 ^ 0 ^ 1 ^ 0 ^ 1 = 1
   C [2] = 1 ^ 1 ^ 0 ^ 1 = 1
   C [3] = 1 ^ 1 ^ 0 ^ 1 = 1

   Codeword: **1111 11011011**

PART 2

In this section, given the codeword, we find the dataword assuming that at most 1 bit can be an error

a. Codeword: **0000 00000111**
   - ➔ Given check bits = 0000
   - ➔ Given dataword = 0000 0111

   Using table given, the original checkbits for this dataword are
   C [0] = 0 ^ 0 ^ 0 ^ 1 ^ 1 = 0
   C [1] = 0 ^ 0 ^ 0 ^ 1 ^ 1 = 0
   C [2] = 0 ^ 0 ^ 1 ^ 1 = 0
   C [3] = 0 ^ 0 ^ 0 ^ 0 = 0
   Syndrome = original check bits ^ given checkbits = 0000 ^ 0000 = 0000 -> **No error**

b. Codeword: **1011 10000111**
   - ➔ Given check bits = 1011
   - ➔ Given dataword = 1000 0111

   Using table given, the original checkbits for this dataword are
   C [0] = 0 ^ 0 ^ 0 ^ 1 ^ 1 = 0
   C [1] = 0 ^ 0 ^ 0 ^ 1 ^ 1 = 0
   C [2] = 1 ^ 0 ^ 1 ^ 1 = 1
   C [3] = 1 ^ 0 ^ 0 ^ 0 = 1
   Syndrome = original check bits ^ given checkbits = 1100 ^ 1011 = 0111 -> **Error in *name3***

c. Codeword: **0010 10101001**
   - ➔ Given check bits = 0010
   - ➔ Given dataword = 1010 1001

   Using table given, the original checkbits for this dataword are
   C [0] = 0 ^ 0 ^ 1 ^ 0 ^ 1 = 0
   C [1] = 0 ^ 1 ^ 1 ^ 0 ^ 1 = 1
   C [2] = 1 ^ 1 ^ 0 ^ 0 = 0
   C [3] = 1 ^ 0 ^ 1 ^ 0 = 0
   Syndrome = original check bits ^ given checkbits = 0010 ^ 0010 = 0000 -> **No error**

d. Codeword: **0111 11001010**
   - ➔ Given check bits = 0111
   - ➔ Given dataword = 1100 1010

   Using table given, the original checkbits for this dataword are
   C [0] = 1 ^ 0 ^ 1 ^ 1 ^ 0 = 1
   C [1] = 1 ^ 0 ^ 1 ^ 0 ^ 0 = 0
   C [2] = 1 ^ 1 ^ 0 ^ 1 = 1
   C [3] = 1 ^ 1 ^ 0 ^ 0 = 0
   Syndrome = original check bits ^ given checkbits = 0101 ^ 0111 = 0010 -> ***error in checkbit***

Problem 2

Multiplicand = 1100 0101 = -59
Multiplier = 0101 0101 = 85

Expected result = -59 * 85 = -5015

- ➔ 2-bit Booth encoding of multiplier is **1 1 1 1** (Procedure illustrated in discussion session slides)
- ➔ Actual Multiplication (sign extend multiplicand)

Multiplicand		2-bit encoding		
1111 1111 1100 0101	*****	**1**	**=**	**1111 1111 1100 0101 [PP1]**
Shift left by 2				
1111 1111 0001 0100	*****	**1**	**=**	**1111 1111 0001 0100 [PP2]**
Shift left by 2				
1111 1100 0101 0000	*****	**1**	**=**	**1111 1100 0101 0000 [PP3]**
Shift left by 2				
1111 0001 0100 0000	*****	**1**	**=**	**1111 0001 0100 0000 [PP4]**

- ➔ Add all Partial products (PP1 to PP4)
  Result is **11 1110 1100 0110 1001 =** -5015 [Expected]

Problem 3

PART 1
17.384

(a) 17 -> **1 0001**

(b) 0.384 = [0 * 2 ^ (-1)] + [1 * 2 ^ (-2)] + [1 * 2 ^ (-3)] + [0 * 2 ^ (-4)] + [0 * 2 ^ (-5)] + [0 * 2 ^ (-6)] + [1 * 2 ^ (-7)] + etc …

(c) Hence, 0.384 in binary is **0.0110001001001101111**

(d) 17.384 would be **10001.0110001001001101111**

(e) Normalize to get **1.000101100010010011011111 * 2 ^ (4)**

(f) Excess 127 to exponent => E = 4 +127 = 131. E = **1000 0011**

(g) Since it is a positive number, *sign = 0*

(h) So ***Sign = 0***, ***exponent = 1000 0011*** and ***mantissa = 00010110001001001101111***

Hence representation is **0 10000011 00010110001001001101111**

PART 2

Given number is **1 10000001 10001001001000000000000**

(a) We can observe sign bit is **1**

(b) Exponent is **1000 0001 = 129**
Due to excess 127, we have e = 129 – 127 = 2 (i.e., 2+127 will give us E = 129)

(c) Mantissa is **10001001001000000000000**

Hence, number is
(-1) ^ (sign) * 1.(mantissa) * 2 ^ (exponent)
(-1) ^ (1) * 1.(10001001001000000000000) * 2 ^ (2)

Mantissa = 10001001001000000000000 = 2 ^ (-1) + 2 ^ (-5) + 2 ^ (-8) + 2 ^ (-11) = 0.53564453125

Final result is
-(1 + 0.53564453125) * 2 ^ (2) = -6.142578125

The representation above is for the number **-6.142578125**

Problem 4

(a) Initial State

P0		P1		P2		Memory
State	Data	State	Data	State	Data	Data
I	0	I	0	I	0	7

(b) **Operation:** Processor P0 executes **lw $1, 100 ($0)**
Action: P0 misses in the cache and reads the block from the memory
Memory State

P0		P1		P2		Memory
State	Data	State	Data	State	Data	Data
S	7	I	0	I	0	7

(c) **Operation:** Processor P1 executes **addi $1, $0, 13** and **sw $1, 100 ($0)**
Action: P1 misses, reads the block from memory and invalidates P0
Memory State

P0		P1		P2		Memory
State	Data	State	Data	State	Data	Data
I	7	M	13	I	0	7

(d) **Operation:** Processor P2 executes **lw $1, 100 ($0)**
Action: P2 misses, reads the block, P1 intervenes and supplies data, updating the memory
Memory State

P0		P1		P2		Memory
State	Data	State	Data	State	Data	Data
I	7	S	13	S	13	13

(e) **Operation:** Processor P1 executes **addi $1, $0, 19** and **sw $1, 100 ($0)**
Action: P1 hits, but does nto have permission to write. P1 invalidates P2's copy
Memory State

P0		P1		P2		Memory
State	Data	State	Data	State	Data	Data
I	7	M	19	I	13	13