Logical Design

Lecture 3: Arithmetic Operations

Produced By: Dr. Haleema Essa

2023-2024

1- Binary Arithmetic operations

<u>Addition</u>

The 4 basic rules for adding binary digits (bits) are as follows:

0+0=0

0+1=1

1+0=1

1+1 = 0 with carry 1

Example1: 10001 + 11101= 1 0 1 1 1 0

Example 2: 10111 + 110001

Solution:

```
 1     1 1 1
      1 0 1 1 1
(+) 1 1 0 0 0 1
---------------
   1 0 0 1 0 0 0
```

H.W: 111000111011011 + 111111000110011

<u>Subtraction</u>

The four basic rules for subtracting bits as follows:

0−0=0

1−0 =1

1−1 =0

0−1= 10−1=1 with borrow of 1

Example 1: 0011010 – 001100

1 1 Borrow

0 0 1 1 0 1 0

(-) 0 0 1 1 0 0

---

0 0 0 1 1 1 0

Decimal Equivalent:

0 0 1 1 0 1 0 = 26

0 0 1 1 0 0 = 12

26 - 12 = 14

H.W: 111000111011011 - 111111000110011

Multiplication

The four basic rules for multiplying bits are as follows:

$0*0=0$

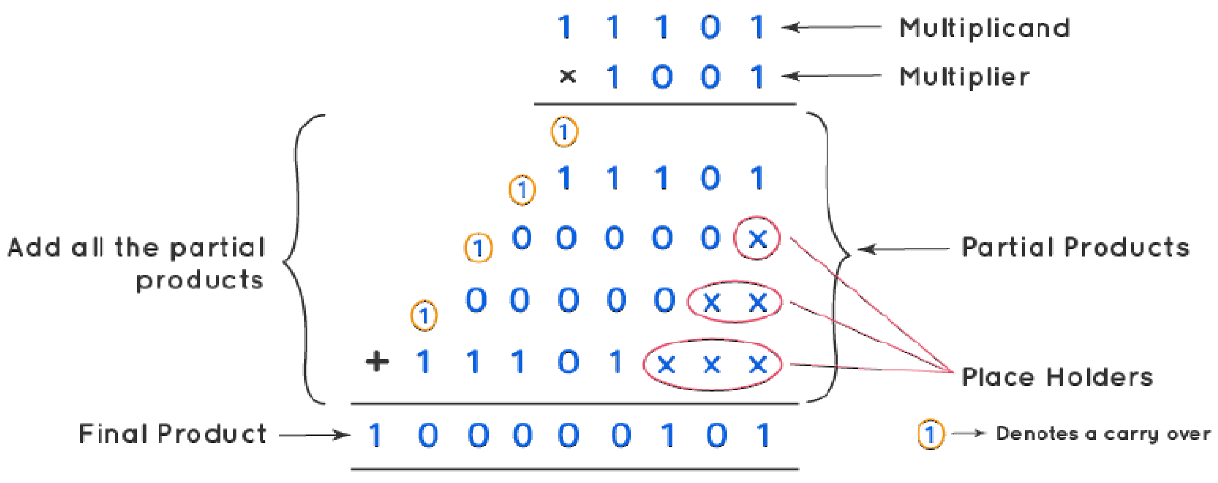
$0*1=0$

$1*0=0$

$1*0=0$

$1*1=1$

Example1:



Example2:

```
        100110     (38)
          1110     (14)
   ---------------------
        000000
       100110
      100110
  +  100110
   ---------------------
     1000010100    (532)
```

H.W: 11001110111 * 1111110011

2- Octal Arithmetic operations

Addition

Just as with decimal addition, octal uses "carry" when the sum of the values of a position exceeds 7_8

Example:

```
Carry     1 1 1
          1 2 3 4
        +   5 6 7
        ---------
          2 0 2 3
```

Example:

```
   1 1
     6 4 3 7₈
 +   2 5 1 0₈
 ------------
   1 1 1 4 7
```

H.W 1: $(13675)_8 + (34627)_8$

H.W 2: $(24636)_8 + (74527)_8$

Subtraction

Just as with decimal subtraction, octal uses "borrow" when the difference between the values of a position requires it

Example:

```
Borrow     1
          256
        - 137
        -----
          117
```

Example:

```
Borrow   1 1
          1 1 4 7₈
        - 6 4 3 7₈
        ---------
          2 5 1 0₈
```

H.W 1: $(563675)_8 - (34627)_8$

H.W 2: $(35575)_8 - (34627)_8$

3- Hexadecimal Arithmetic operations

Addition

Just as with decimal addition, hexadecimal uses "carry" when the sum of the values of a position exceeds $\mathbf{15_{10}}$ [or $\mathbf{F_{16}}$]

Example:

```
Carry     1         1 1
         5 6 7 8   1 2 3 4
        +8 A 1 1   D E F 0
        -------------------
         E 0 8 9   F 1 2 4
```

Example:

```
          1 1
          7 C 3 9₁₆
      +   3 7 F 2₁₆
          ---------
          B 4 2 B
```

H.W: $(A3F75)_{16} + (3E6C7)_{16}$

Subtraction

Just as with decimal subtraction, hexadecimal uses "borrow" when the difference between the values of a position requires it

Example:

```
Borrow      1  1          1
            8 A 1 1   D E F 0
         -  5 6 7 8   1 2 3 4
         --------------------
            3 3 9 9   C C B C
```

Example:

```
Borrow         1
            7 C 3 9
          - 3 7 F 2
          ---------
            4 4 4 7
```

H.W 1: $(A3F75)_{16}$ - $(3E6C7)_{16}$

H.W 2: $(D5FE2A)_{16}$ - $(A378F6)_{16}$

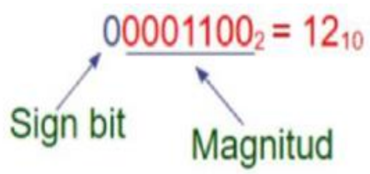
Signed Binary Numbers Representation

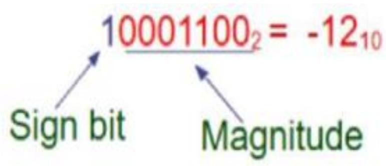
- Plus and Minus signs used for decimal numbers: 25 (or +25), -16, etc.
- For computers, it's desirable to represent everything as bits.
- Three types of signed binary number representation: Signed Magnitude, 1's Complement, 2's Complement.

Signed Magnitude

In the signed magnitude form, (−39) can be produced by changing the sign bit only to "1" and leave the magnitude bits as they are. So (−39) will be 10100111.

Example:





1's Complement

1's complement: the 1's complement form of any binary number is obtained simply by changing each 0 in the number to a 1 and each 1 to a 0. In other word, change each bit to its complement. For example:

1 0 1 1 0 1	Binary No.	0 1 1 0 1 0	Binary No.
↓↓↓↓↓↓		↓↓↓↓↓↓	
0 1 0 0 1 0	1's complement	1 0 0 1 0 1	1's complement

H.W: Find the 1'S Complement of $(11110000101010110011)_2$

2's Complement

2's complement: the 2's complement form of a binary number is formed simply by taking the 1's complement of the number and adding 1 to the least significant bit position.

2's complement = (1's complement) +1

Example: find 2's complement of 10110010.

1 0 1 1 0 0 1 0	binary number.
↓↓↓↓↓↓↓↓	
0 1 0 0 1 1 0 1	1's complement
+ 1	adding 1
0 1 0 0 1 1 1 0	2's complement

H.W: Find the 2'S Complement of $(11110000101010110011)_2$

1's Complement Subtraction

Example: find $11010_{(2)} - 10000_{(2)}$ using 1's complement method (Case 1).

As long as the carry appear, the number is positive and a carry must be added to the result.

```
  11010
+ 01111    1's complement of 10000
 101001
+     1
  01010
```

$11010_{(2)} - 10000_{(2)} = 01010_{(2)}$

Example: find $10000_{(2)} - 11010_{(2)}$ using 1's complement method (Case 2).

As long as no carry appear, the number is negative, then 1's complementing of the final result in needed.

```
  10000
+ 00101    1's complement of 11010
  10101
    ↓ 1's complement
  01010
```

$10000_{(2)} - 11010_{(2)} = -01010_{(2)}$

HW: Use 1's Complement to subtract the following:

1. $(12)_{10} - (1)_{10} =$ **?**
2. $(12)_{10} - (13)_{10} =$ **?**