CS47 - Lecture 03

Kaushik Patra
(kaushik.patra@sjsu.edu)

1

- Topics
  - Integer Representation
  - Format conversion
  - Negative Integer Representation

[Chapter 3 (3.2) of Computer Organization & Design by Patterson]

Integer Representation ...

2

Different Representations

- There are multiple number systems and its applications.
  - ***Decimal*** number system using 0 ... 9 digit [base 10]
  - ***Binary*** number system using 0 and 1 [base 2]
  - ***Octal*** number system using 0 … 7 digit [base 8]
  - ***Hexadecimal*** number system 0 … 9, a … f [base 16]

- In context of maths using multiple number systems, base value is written as subscript after the number.

$$V_K\{e.g.\ 10_{16}, 10_{10}, 10_8, 10_2\}$$

3

- Number systems are different literal representations of quantities.

- Base of a number system is the number of digits it has to express any integer number. For example decimal system has 10 digits (0 to 9), thus its base is 10. Similarly octal has 8 digits (0 to 7), thus its base is 8.

- Decimal number system is the most natural number system that has been adopted by human to perform mathematics.

- For need of electronic computer development and usage binary, octal and hexadecimal number systems are used. The reasons will be discussed in later section.

Conversion to / from Decimal

- The formula to translate a n-digit number in base K into corresponding decimal value is as following where d_i is the i^{th} digit in the original number, assuming 0 is the right most digit.

$$V_{10} = \sum_{i=0}^{n-1} d_i * K^i$$

- The algorithm to convert an unsigned decimal into a base K number as following.

```
// num stores a value in base 10
// solution will have digits in an array
index = 0 ;
while ( num != 0 )
 {
    remainder = num % K ;  // assume K > 1
    num       = num / K ;  // integer division
    digit[ index ] = remainder ;
    index ++ ;
 }
```

4

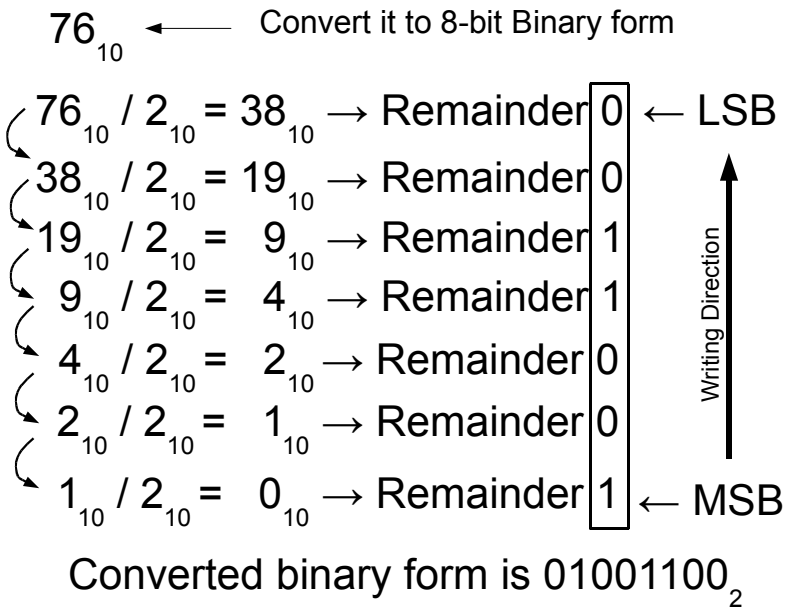
- Decimal value of 1010_2 is $(1*2^3 + 0*2^2 + 1*2^1 + 0*2^0)_{10} = 10_{10}$
- Hexadecimal value of 1482_{10} is determined as following
  - Num = 1482; H[0] = num % 16 = 10 or A_{16}; num = num / 16 = 92;
  - Num = 92; H[1] = num % 16 = 12 or C_{16}; num = num /16 = 5;
  - Num = 5; H[1] = num % 16 = 5 or 5_{16}; num = num /16 = 0;
  - Hence the hexadecimal is $5CA_{16}$.
- Decimal value of $5CA_{16}$ is $(5*16^2 + 12*16^1 + 10*16^0)_{10} = 1482_{10}$

•The algorithm snippet has been take from the site http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Data/toBaseK.html. More details can be found in that note.

Format Conversion ...

5

Decimal to Binary Conversion



6

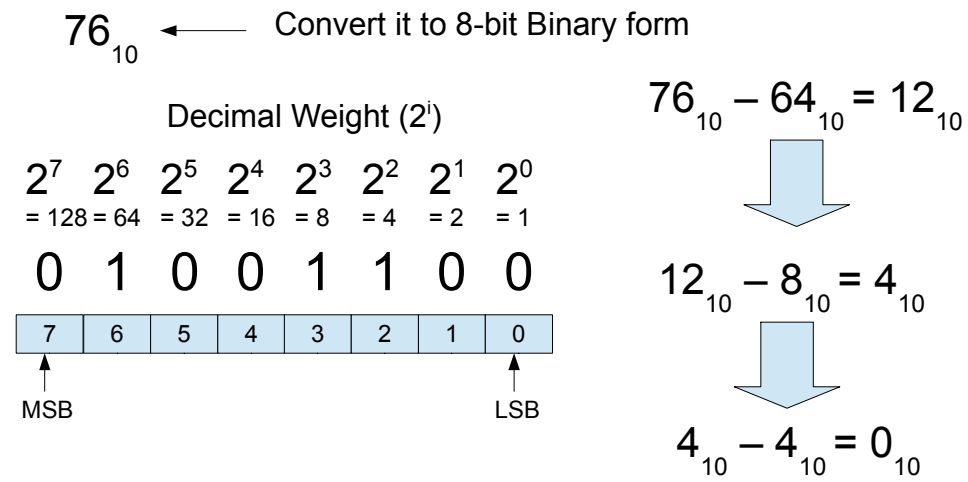
- To convert from decimal to binary format, we keep on dividing the given number by 2 and note down the quotient and remainder term at each step. Since we are dividing by 2, the remainder will be either 0 or 1 at each step. At each next step we take the quotient from the previous step as dividend, while divisor always remains to be 2. We stop this process as soon as we get a quotient of 0.

- The remainder at very first step is the 'Least Significant Bit' (LSB) and the remainder at the very last step is the 'Most Significant Bit' (MSB).

- We write down the result from MSB to LSB.

- If conversion is asked to meet certain number of bits (8 bit in this example), pad 0s at the beginning of result.

Decimal to Binary Conversion



Range for n-bit binary number [0 , $(2^n - 1)$]

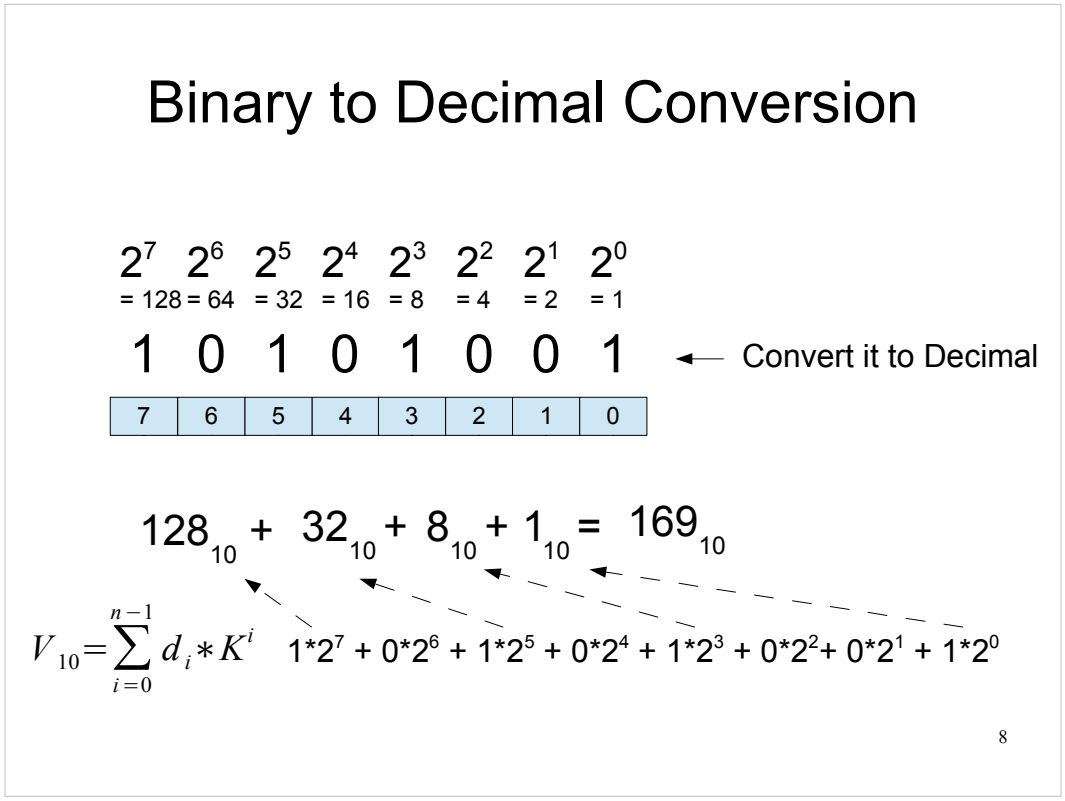
7

- In alternate process of converting decimal to binary, we first write down index at the bit position as (i-1) assuming lower bit position goes towards right hand side. For example, bit position 1 is the right most bit with index value (1-1) = 0. The highest bit position is the left most one. In this example it is the 8^{th} bit position at the left most side with index (8-1)=7.

- Then we determine decimal weight of each bit position with index I as 2^i . LSB has the lowest decimal weight and MSB has the highest decimal weight.

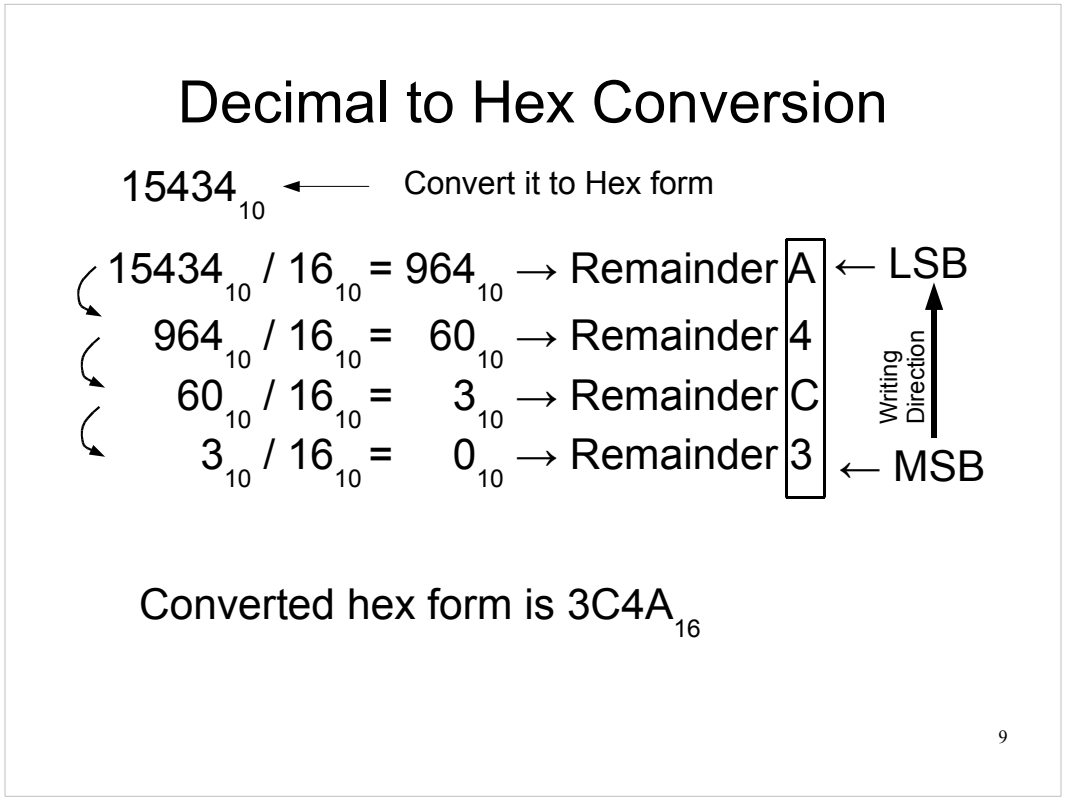
- Start from MSB position and put 0 as long as the decimal weight at that bit position is greater than the given number. If the decimal weight is less or equal to the number, put 1 in that bit position, subtract the decimal weight from the number and use the result as the number for next iteration. Stop the process as soon as subtraction results in 0 and fill up rest of the bits in the right hand side with 0.

- Range is very important to check if, for a given width of the binary number, it can represent given number. In this case of 8 bit width number from 0 – 255 can be represented.



- To convert binary number to decimal, like the previous decimal to binary conversion, we first compute the decimal weight at each position and then add up the weight of the positions where the bit value is 1.

- This is a shortcut method from the original summation based generic formula. In the summation based formula, we multiply digit value at the given position with the power of number system base. Since binary numbers are represented with digit 1 and 0, we can omit anything multiplied by 0 (because it will contribute 0 to final summation). Rest of the part is just the base to the power position index (here base is 2) since it is always multiplied by 1.



- It is the same process as decimal to binary conversion. The divisor is 16 since the target number system is a hexadecimal. Since dividing by 16, the remainder range is [0,15]. We need to note down the remainder in hexadecimal digit format (i.e. 0-9,A-F). Rest of the process is same as binary conversion.

Hex to Decimal Conversion

Hex Digit	Decimal Value	Hex Digit	Decimal Value
0	0	8	8
1	1	9	9
2	2	A	10
3	3	B	11
4	4	C	12
5	5	D	13
6	6	E	14
7	7	F	15

10

- We need to remember hexadecimal digit to its decimal value (as in the table) while converting from hexadecimal number to decimal number.

Hex to Decimal Conversion

$D5C23_{16}$ ← Convert it to Decimal Value

16^4	16^3	16^2	16^1	16^0
= 65536	= 4096	= 256	= 16	= 1
D	5	C	2	3
4	3	2	1	0

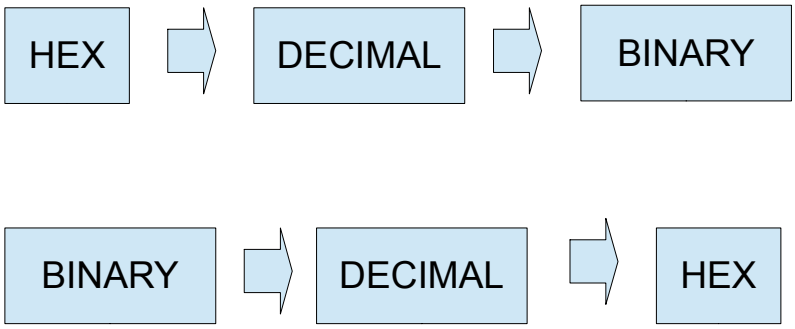
$$V_{10} = \sum_{i=0}^{n-1} d_i * K^i$$

$(13_{10} * 65536_{10}) + (5_{10} * 4096_{10}) +$
$(12_{10} * 256_{10}) + (2_{10} * 16_{10}) +$
$(3_{10} * 1_{10}) = 875555_{10}$

11

- Similar to binary to decimal conversion, we first compute the decimal weight of each index position. In this case since the number system has base 16, the decimal weight is 16^i . Now to convert the number from hexadecimal format, we multiply decimal equivalent of the hex digit and the decimal weight of its position and then sum the multiplication result.

Hex(Binary) to Binary(Hex) Conversion



12

- For a given hex number, we can always convert it decimal and then binary. Similarly we can also convert binary to decimal and then hex. Let's see it there is any shortcut ways to bypass the decimal conversion.

Hex to Decimal Conversion

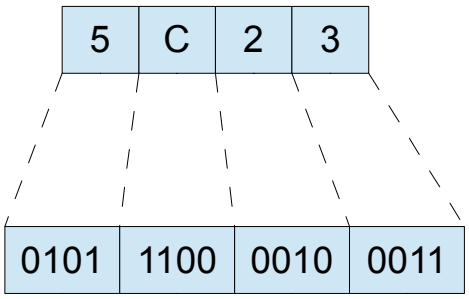
Hex Digit	Decimal Value	4-bit Binary	Hex Digit	Decimal Value	4-bit Binary
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	A	10	1010
3	3	0011	B	11	1011
4	4	0100	C	12	1100
5	5	0101	D	13	1101
6	6	0110	E	14	1110
7	7	0111	F	15	1111

13

- Since max value represented by hex digit is 15, we can represent value of each hex digit as a 4 bit binary number. We need to remember these pattern mapping.

Hex to Binary Conversion

$5C23_{16}$ ← Convert it to Binary Value

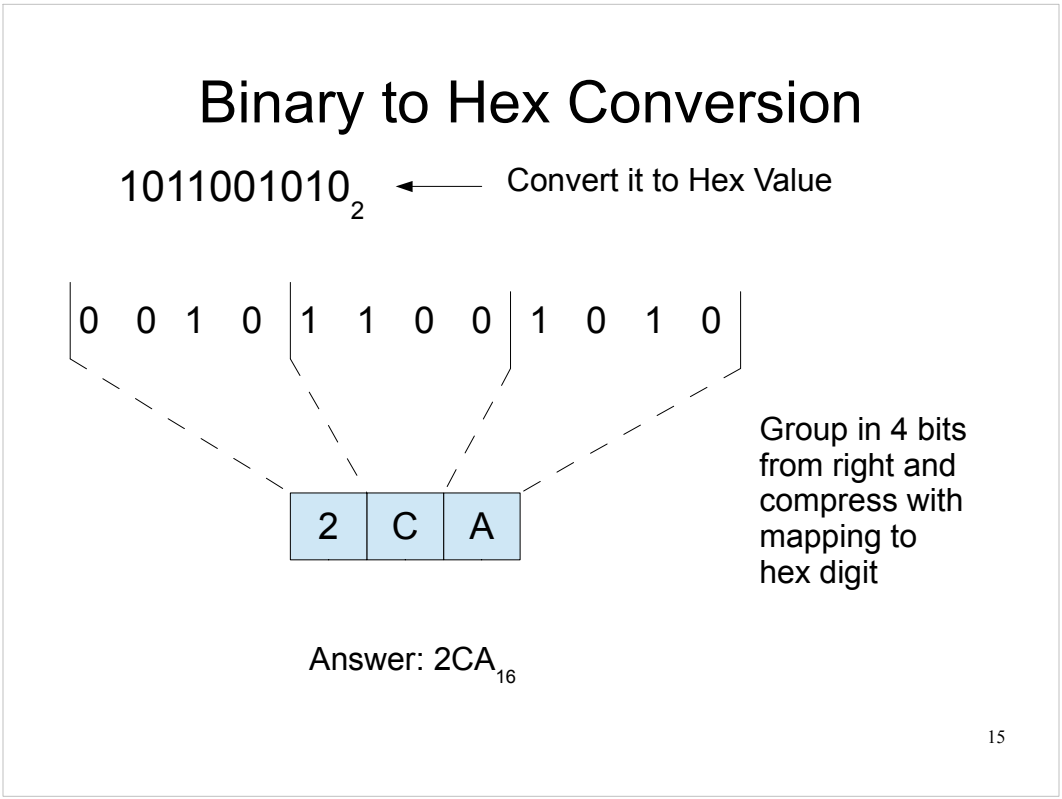


Blowup each
hex digit into
4 digit binary pattern

Answer: 0101110000100011_2

14

- For a given hex number, to convert it to corresponding binary, we need map each hex digit to its 4 bit binary pattern in sequence. The resultant mapping is the binary representation of the given hex number.

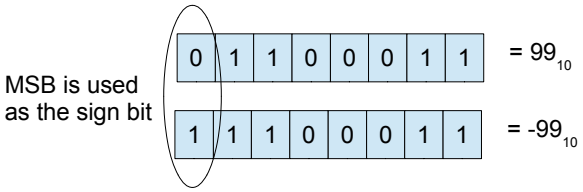


- To convert number in binary format to equivalent hex format, we group the whole binary representation in group of 4 bits. The grouping starts from LSB position. At the MSB position if needed 0s are added to make the last group as group of 4 bits. Once the grouping is done, each group (the bit pattern) is mapped to its equivalent hex digit.

Negative Integer Representation ...

-ve Number Representation

- How to represent negative numbers in binary? There is no '-' sign in electronic computer world.
- One way would be to use sign and magnitude representation of binary numbers using one of the bit as sign bit. However it has problems.
  - Where to put the sign bit? MSB or LSB?
  - May need an extra step for a sign-magnitude ALU to determine the sign of the result.
  - The system has two zeros !!! +ve 0 and -ve 0 ???



17

- MSB = **M**ost **S**ignificant **B**it, which is the left most bit of a binary number.
- LSB = **L**east **S**ignificant **B**it, which is the right most bit of a binary number.

- Early computer used this sign-magnitude approach. However, it presented problem for both hardware engineers (adding extra steps to determine result's sign bit) and software engineers (what to do with +ve 0 and -ve 0). Extra circuitry was needed of subtraction operation.

- In search of a better representation binary subtraction operation was carefully observed. If there is a need of subtracting a bigger number from a smaller one, what do we do? We start borrow from leading zeros which would result in string of 1s. This will be clear once we learn how to do binary subtraction.

- It was then made that any number with leading zeros will be +ve and with ones will be -ve. This is a 2's complement form which is now used in every computer.

-ve Number Representation

0	0	0	0	$= 0_{10}$	1	0	0	0	$= -8_{10}$
0	0	0	1	$= 1_{10}$	1	0	0	1	$= -7_{10}$
0	0	1	0	$= 2_{10}$	1	0	1	0	$= -6_{10}$
0	0	1	1	$= 3_{10}$	1	0	1	1	$= -5_{10}$
0	1	0	0	$= 4_{10}$	1	1	0	0	$= -4_{10}$
0	1	0	1	$= 5_{10}$	1	1	0	1	$= -3_{10}$
0	1	1	0	$= 6_{10}$	1	1	1	0	$= -2_{10}$
0	1	1	1	$= 7_{10}$	1	1	1	1	$= -1_{10}$

- 2's complement form represent any +ve binary number with leading 0 and any -ve binary number with leading 1.
- For a 4 bit binary number it can represent a range of +7 to -8.
- There is no problem of two zeros unlike sign-magnitude representation.
- There is no need for subtraction circuit – addition circuit is enough for a subtraction operation. We'll ignore any overflow. [(7 - 1) == (7 + (-1))]
- Implementing 2's complement form is easier for hardware engineers, but may pose some challenge with software side due to the lowest number represented with the system.

18

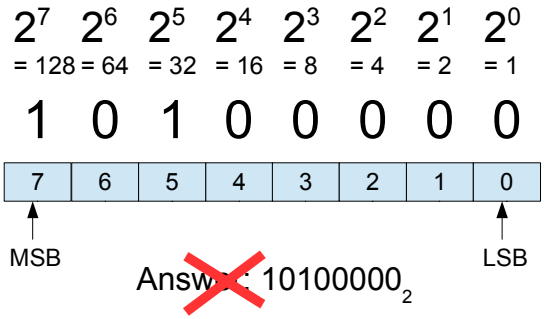
- For a four bit binary number when MSB is 1 for the first time (i.e the number is 1000_2) its unsigned decimal value is 8. However, since any leading 1 means a -ve number it is assumed to be -8 in decimal. From that point on wards, we can add the unsigned value represented by rest of the 3 lower bits to -8 and get the corresponding -ve number. For example 1011_2 is ($-8_{10} + 011_2 = -8_{10} + 3_{10} = -5_{10}$).

- 2's complement of a -ve number will have it's +ve counter part, except for the lowest -ve number represented in the sytem. For example 2's complement of 1101_2 (-3_{10}) is ($0010_2 + 1_2 = 0011_2 = 3_{10}$). However, 2's complement of 1000_2 (-8_{10}) is ($0111_2 + 1_2 = 1000_2 = -8_{10}$). This boundary condition may pose some challenge to software engineering side.

- Since all the leading bits will be 1 in case of binary -ve number in 2's complement representation, it is sufficient to test the MSB for to determine the sign of the number. This also gives the ease of extending a number in 2's complement form. If it is required to sign extend a number from shorter size to longer size, it is sufficient to repeat the MSB of the shorter representation into rest of the higher bit positions in the longer representation. For example if 4-bit number 1101_2 has to be extended to 8-bit, the MSB (which is 1) needs to be repeated through rest of the higher order bits in the 8-bit representation. The result is 11111101_2

-ve Number Representation

160_{10} ← Convert it to 8-bit 2's complement binary form



Answer: 10100000_2

$160_{10} - 128_{10} = 32_{10}$

$32_{10} - 32_{10} = 0_{10}$

Range for n-bit 2's complement binary number [$-2^{(n-1)}$, $(2^{(n-1)} - 1)$]

19

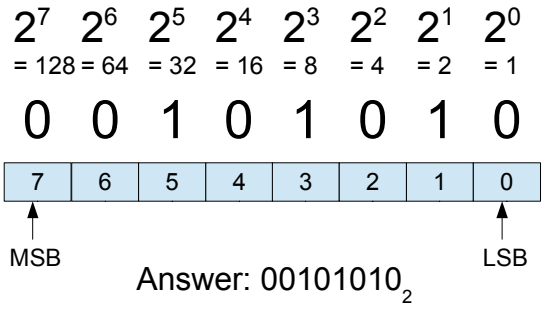
- For a 8-bit 2's complimentary representation, this result to represent 160 is wrong, since in this representation, any leading 1 means a -ve number.

- The range, compared to unsigned 8 bit representation (where range is [0,255]), of 2's complement representation is very different. I can represent half of the integer set in absolute term, but include both -ve and position representation (8-bit has range of [-128,127]).

- Always check the range before conversion It may happen that the given number can not be represented with the given width of the binary representation.

-ve Number Representation

42_{10} ← Convert it to 8-bit 2's complement binary form



Answer: 00101010_2

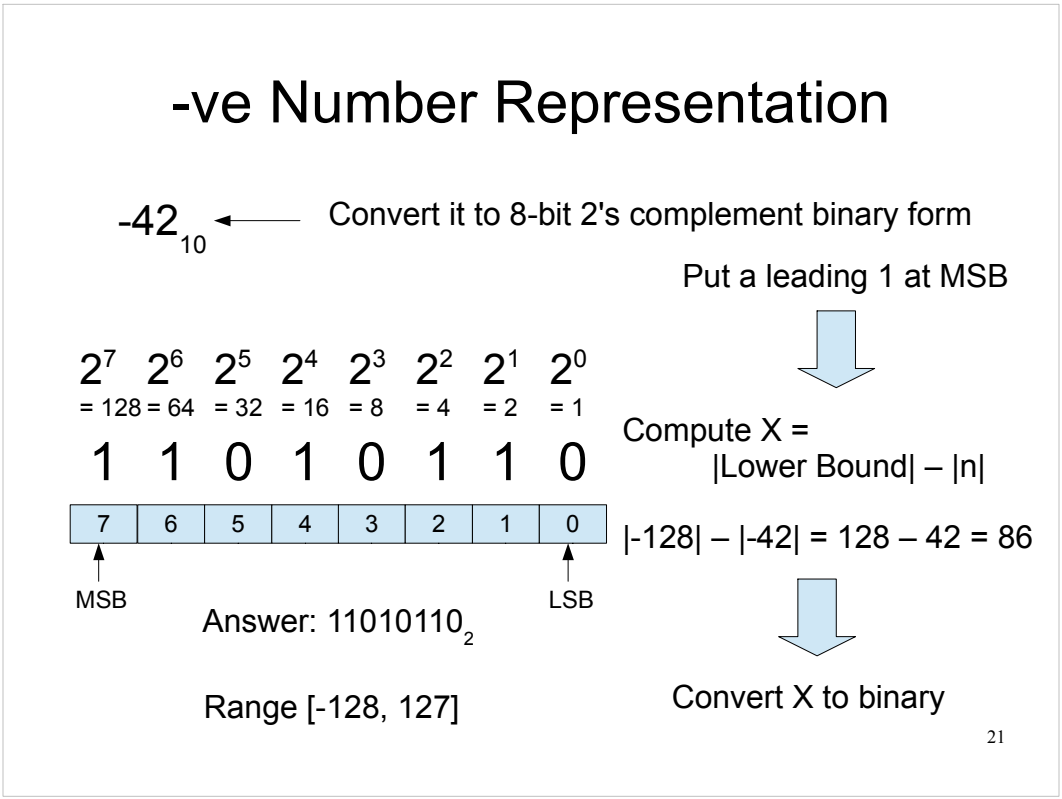
Range [-128, 127]

Put a leading 0 at MSB

Convert as usual

20

- To convert a positive number put 0 to MSB and then convert the number to binary as usual. Remember to check the range before starting the conversion.



- To convert a negative number, put 1 to MSB position. Then compute difference between absolute lower bound and absolute of the number given. Now convert the difference to binary as usual.

CS47 - Lecture 03

Kaushik Patra
(kaushik.patra@sjsu.edu)

22

- Topics
  - Integer Representation
  - Format conversion
  - Negative Integer Representation

[Chapter 3 (3.2) of Computer Organization & Design by Patterson]