## King Fahd University of Petroleum & Minerals
## Computer Engineering Dept

COE 200 – Fundamentals of Computer Engineering

Term 043

Dr. Ashraf S. Hasan Mahmoud

Rm 22-144

Ext. 1724

Email: ashraf@ccse.kfupm.edu.sa



# Number Systems – Base r

## Number Systems – Base r

- General number in base r is written as:

$$\underbrace{A_{n-1}\,A_{n-2}\,\ldots\,A_2\,A_1\,A_0}_{\text{Integer Part (n digits)}}\;\blacksquare\;\underbrace{A_{-1}\,A_{-2}\,\ldots\,A_{-(m-1)}\,A_{-m}}_{\text{Fraction Part (m digits)}}$$

Radix Point

- Note that All $A_i$ (digits) are less than r:
  - i.e. Allowed digits are 0, 1, 2, …, r – 1 ONLY
- $A_{n-1}$ is the MOST SIGNIFACT Digit (MSD) of the number
- $A_{-m}$ is the LEAST SIGNIFICANT Digit (LSD) of the number

$A_{n-1}$ is the MSD of the integer part
$A_0$ is the LSD of the integer part
$A_{-1}$ is the MSD of the fraction part
$A_{-m}$ is the LSD of the fraction part



## Number Systems – Base r

- The (base r) number

$$A_{n-1}\,A_{n-2}\,\ldots\,A_2\,A_1\,A_0\,\blacksquare\,A_{-1}\,A_{-2}\,\ldots\,A_{-(m-1)}\,A_{-m}$$

FORM or SHAPE OF NUMBER

is equal to

$$A_{n-1}\,X\,r^{n-1} + A_{n-2}\,X\,r^{n-2} + \ldots A_2\,X\,r^2 + A_1\,X\,r^1 + A_0\,X\,r^0 + A_{-1}\,X\,r^{-1} + A_{-2}\,X\,r^{-2} + \ldots A_{-(m-1)}\,X\,r^{-(m-1)} + A_{-m}\,X\,r^{-m}$$

VALUE OF NUMBER

## Example – Decimal or Base 10

- For decimal system (base 10), the number

$$(724.5)_{10}$$

is equal to

$$7X10^{2} + 2X10^{1} + 4X10^{0} + 5X10^{-1}$$
$$= 7 X 100 + 2 X 10 + 4 X 1 + 5 X 0.1$$
$$= 700 + 20 + 4 + 0.5$$
$$= 724.5$$

**It is all powers of 10**:
…
$10^3 = 1000$,
$10^2 = 100$,
$10^1 = 10$,
$10^0 = 1$,
$10^{-1} = 0.1$,
$10^{-2} = 0.01$,
…



## Example –Base 5

- Base 5 → r = 5
- Allowed digits are: 0, 1, 2, 3, and 4 ONLY
- The number

$$(312.4)_5$$

is equal to

$$3X5^{2} + 1X5^{1} + 2X5^{0} + 4X5^{-1}$$
$$= 3 X 25 + 1 X 5 + 2 X 1 + 4 X 0.2$$
$$= 75 + 5 + 2 + 0.8$$
$$= (82.8)_{10}$$

Therefore $(312.4)_5 = (82.8)_{10}$

**It is all powers of 5**:
…
$5^3 = 125$,
$5^2 = 25$,
$5^1 = 5$,
$5^0 = 1$
$5^{-1} = 0.2$
$5^{-2} = 0.04$,
…

## A Third Example –Base 2

- Base 2 ➔ r = 2
  - This is referred to as the **BINARY SYSTEM**
- Allowed digits are: 0 and 1 ONLY
- The number

$(110101.11)_2$

is equal to

$1X2^5 + 1X2^4 + 0X2^3 + 1X2^2 + 0X2^1 + 1X2^0 + 1X2^{-1} + 1X2^{-2}$

$= 1 X 32 + 1 X 16 + 1 X 4 + 1 X 2 + 1 X 0.5 + 1 X 0.25$

$= 32 + 16 + 4 + 1 + 0.5 + 0.25$

$= (53.75)_{10}$

Therefore $(110101.11)_2 = (53.75)_{10}$

**It is all powers of 5**:
…
$2^4 = 16$
$2^3 = 8$,
$2^2 = 4$,
$2^1 = 2$,
$2^0 = 1$
$2^{-1} = 0.5$
$2^{-2} = 0.25$,
…



## Decimal to Binary Conversion of Integer Numbers

- Conversion from base 2 to base 10 (for real numbers) – See previous slide
- To convert a decimal *integer* to binary ➔ decompose into powers of 2
  - Example: $(37)_{10} = (?)_2$

37 has ONE 32 ➔ remainder is 5
5 has ZERO 16 ➔ remainder is 5
5 has ZERO 8 ➔ remainder is 5
5 has ONE 4 ➔ remainder is 1
1 has ZERO 2 ➔ remainder is 1
1 has ONE 1 ➔ remainder is 0

Therefore $(37)_{10} = (100101)_2$

## Decimal to Binary Conversion of Integer Numbers– cont'd

- Or we can use the following (see table):
- You stop when the division result is ZERO
- Note the order of the resulting digits

| No | No/2 | Remainder | |
|---|---|---|---|
| 37 | 18 | 1 | LSD |
| 18 | 9 | 0 | |
| 9 | 4 | 1 | |
| 4 | 2 | 0 | |
| 2 | 1 | 0 | |
| 1 | 0 | 1 | MSD |

- Therefore $(37)_{10} = (100101)_2$
- To check:

$1X2^5+1X2+1 = 32+4+1 = 37$

In general: to convert a decimal integer to its equivalent in base r we use the above procedure but dividing by r



## A Very Useful Table

- To represent decimal numbers from 0 till 15 (16 numbers) we need FOUR binary digits $B_3B_2B_1B_0$
- In general to represent N numbers, we need $\lceil \log_2 N \rceil$ bits
- Note how
  - $B_0$ flipped or COMPLEMENTED at every increment
  - $B_1$ flipped or COMPLEMENTED every 2 steps
  - $B_2$ flipped or COMPLEMENTED every 4 steps
  - $B_3$ flipped or COMPLEMENTED every 8 steps

| Decimal | Binary | Decimal | Binary |
|---|---|---|---|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | 10 | 1010 |
| 3 | 0011 | 11 | 1011 |
| 4 | 0100 | 12 | 1100 |
| 5 | 0101 | 13 | 1101 |
| 6 | 0110 | 14 | 1110 |
| 7 | 0111 | 15 | 1111 |

# A Very Useful Table – cont'd

- Note that zeros to the left of the number do not add to its value
- When we need DIGITS beyond 9, we will use the alphabets as shown in Table
  - Example: base 16 system has 16 digits; these are: 0, , 1, 2, 3, …, 8, 9, A, B, C, D, E, F
  - This is referred to as HEXADECIMAL or HEX number system

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



# Decimal to Binary Conversion of Fractions

- Example: $(0.234375)_{10} = (?)_2$
- Solution: We use the following procedure
- **Note:**
  - The binary digits are the integer part of the multiplication process
  - The process stops when the number is 0
- There are situations where the process DOES NOT end – See next slide
- Therefore $(0.234375)_{10} = (0.001111)_2$
- To check: $(0.001111)_2 = 1X2^{-2} +1X2^{-3}+1X2^{-4}+ 1X2^{-5}+1X2^{-5}$ $1X2^{-6} = (0.234375)_{10}$

| No | NoX2 | Integer Part | |
|---|---|---|---|
| 0.234375 | 0.46875 | 0 | MSD |
| 0.46875 | 0.9375 | 0 | |
| 0.9375 | 1.875 | 1 | |
| 0.875 | 1.75 | 1 | |
| 0.75 | 1.5 | 1 | |
| 0.5 | 1.0 | 1 | LSD |
| 0 | | | |

In general: to convert a decimal fraction to its equivalent in base r we use the above procedure but multiplying by r

## Decimal to Binary Conversion of Fractions – cont'd

- Example: $(0.513)_{10} = (?)_2$
- Solution: As in previous slide

| No | NoX2 | Integer Part |
|---|---|---|
| 0.513 | 1.026 | 1 |
| 0.026 | 0.052 | 0 |
| 0.052 | 0.104 | 0 |
| 0.104 | 0.208 | 0 |
| 0.208 | 0.416 | 0 |
| 0.416 | 0.832 | 0 |
| 0.832 | 1.664 | 1 |
| 0.664 | 1.328 | 1 |
| 0.328 | 0.656 | 0 |
| … | | |

Therefore $(0.513)_{10} = (0.100000110 \ldots)_2$

If we chose to round to 1 significant figure ➔ $(0.1)_2$

Or to 7 significant figures ➔ $(0.1000001)_2$

Etc.



## Octal Number System

- Base r = 8
- Allowed digits are = 0, 1, 2, …, 6, 7
- Example: the number $(127.4)_8$ has the decimal value $1X8^2 + 2X8^1 + 7X8^0 + 4X8^{-1}$

= 1 X 64 + 2 X 8 + 7 + 0.5

$= (87.5)_{10}$

**<u>It is all powers of 8</u>**:
…
$8^4 = 4096$
$8^3 = 512$,
$8^2 = 64$,
$8^1 = 8$,
$8^0 = 1$
$8^{-1} = 0.125$
$8^{-2} = 0.015625$,
…

## Conversion between Octal and Binary

- **Example:** $(127)_8 = (?)_2$
- **Solution:** we can find the decimal equivalent (see previous slide) and then convert from decimal to binary

$(127)_8 = (87)_{10} \rightarrow (?)_2$

From long division

$(127)_8 = (87)_{10} = (1010111)_2$

To check:

$1X2^6+1X2^4+1X2^2+1X2^1+1X2^0$

$= 64 + 16 + 4 + 2 + 1$

$= 87$

| No | No/2 | Remainder |
|---|---|---|
| 87 | 43 | 1 |
| 43 | 21 | 1 |
| 21 | 10 | 1 |
| 10 | 5 | 0 |
| 5 | 2 | 1 |
| 2 | 1 | 0 |
| 1 | 0 | 1 |



## Conversion between Octal and Binary– cont'd

- **NOTE:** $(127)_8 = (1010111)_2$
- Lets group the binary digits in groups of 3 starting from the LSD

$$(1010111)_2 \rightarrow (001 \quad 010 \quad 111)_2$$

$$\updownarrow \quad \updownarrow \quad \updownarrow$$

$$1 \quad 2 \quad 7$$

- That is the decimal equivalent of the first group 111 → 7
  of the second group 010 → 2
  of the third group 001 → 1
- Hence, to convert from Octal to Binary one can perform direct translation of the Octal digits into binary digits:
  ONE Octal digit ←→ THREE Binary digits

## Conversion between Octal and Binary – cont'd

- To convert from Binary to Octal, Binary digits are grouped into groups of three digits and then translated to Octal digits

- Example: $(1011101.10)_2 = (?)_8$
- Solution:

$$(1011101.10)_2 = (001\ 011\ 101\ .\ 100)_2$$
$$= (1 \quad 3 \quad 5 \ . \ 4)_8$$
$$= (135.4)_8$$

**Note:**
We can add zeros to the left of the number or to the right of the number after the radix point to form the groups



## Conversion From Decimal to Octal

- **Problem**: What is the octal equivalent of $(\mathbf{32.57})_{10}$?
- **Solution**:

a) We can covert $(32.57)_{10}$ to binary and then to Octal or
b) We can do:

$32_{10} \Rightarrow$ 32/8 = 4 and remainder is 0 → 0
4/8 = 0 and remainder is 4 → 4

hence, $32_{10} = 40_8$

$(0.57)_{10} \Rightarrow$ 0.57 X 8 = 4.56 → 4
0.56 X 8 = 4.48 → 4
0.48 X 8 = 3.84 → 3
0.84 X 8 = 6.72 → 6
…

hence, $(0.57)_{10} = (0.4436)_8$

Therefore, $(32.57)_{10} = (40.4436)_8$

What is $(0.4436)_8$ rounded for
-Two fraction digits?
-One fraction digit?

## Hexadecimal Number Systems

- Base r = 16
- Allowed digits: 0, 1, 2, …, 8, 9, A, B, C, D, E, F
- The values for the alphabetic digits are as show in Table

| Hex | Value |
|---|---|
| A | 10 |
| B | 11 |
| C | 12 |
| D | 13 |
| E | 14 |
| F | 15 |

- **Example 1:**

$(B65F)_{16} = BX16^3+6X16^2+5X16^1+FX16^0$
$= 11X4096 + 6X256 + 5X16 + 15$
$= (46687)_{10}$

- **Example 2:**

$(1B.3C)_{16} = 1X16^1+BX16^0+3X16^{-1}+CX16^{-2}$
$= 16+11+3X0.0625+ 12X0.00390625$
$= (27.234375)_{10}$



## Conversion Between Hex and Binary

- **Example:** $(1B.3C)_{16} = (?)_2$
- **Solution:** we can find the decimal equivalent (see previous slide) and then convert from decimal to binary

$(1B)_{16} = (27)_{10}$ ➔ $(?)_2$
From long division
$(1B)_{16} = (27)_{10} = (11011)_2$
(0.3C)16 = $(0.234375)_{10}$=$(0.001111)_2$

➔Therefore $(1B.3C)_{16} = (11011.\ 001111)_2$

**Verify This Result**

## Conversion Between Hex and Binary – cont'd

- **Note:**

$(1B.3C)_{16} = (11011.\ 001111)_2$ from previous example

Lets group the binary bits in groups of 4 starting from the radix point, adding zeros to the left of the number or to the right as needed

➔ (0001 1011 . 0011 1100)

$$\begin{array}{cccc} 0001 & 1011 & . & 0011 & 1100 \\ \updownarrow & \updownarrow & & \updownarrow & \updownarrow \\ \mathbf{1} & \mathbf{B} & \mathbf{.} & \mathbf{3} & \mathbf{C} \end{array}$$

- Hence, to convert from Hex to Binary one can perform direct translation of the Hex digits into binary digits: ONE Hex digit ←→ FOUR Binary digits



## Conversion between Hex and Binary – cont'd

- To convert from Binary to Hex, Binary digits are grouped into groups of four digits and then translated to Hex digits

- Example: $(1011101.10)_2 = (?)_{16}$
- Solution:

$$\begin{aligned}(1011101.10)_2 &= (0101\ 1101\ .\ 1000)_2 \\ &= (\ 5 \quad D \ .\ 8\ )_{16} \\ &= (5D.8)_{16}\end{aligned}$$

**Note:**
We can add zeros to the left of the number or to the right of the number after the radix point to form the groups

## Sample Exam Problem

- **Problem**: What is the radix r if
  $((33)_r + (24)_r) \text{ X } (10)_r = (1120)_r$
- **Solution**:

$(33)_r = 3r + 3,$
$(24)_r = 2r + 4,$
$(10)_r = r,$
$(1120)_r = r^3 + r^2 + 2r$
therefore:

$$[(3r+3)+(2r+4)] \text{ X } r$$
$$= r^3 + r^2 + 2r \rightarrow r^3 - 4r^2 - 5r = 0, \text{ or}$$
$$r(r-5)(r+1) = 0$$

This means, the radix r is equal to 5



## Number Ranges - Decimal

- Consider a decimal integer number of n digits:
  $A_{n-1}A_{n-2}\ldots A_1A_0$ where $A_i \in \{0,1,2, \ldots, 9\}$

Smallest integer is $0_{n-1}0_{n-2}\ldots 0_1 0_0 = 0$
Largest integer is $9_{n-1}9_{n-2}\ldots 9_1 9_0 = 10^n - 1$

**Example**: for n equal to 3 ➔ 3 digits integer decimals;
the maximum integer is 999 or $10^3 - 1$

## Number Ranges – Decimal – cont'd

- Consider a decimal fraction of m digits:

  $0.A_{-1}A_{-2}\ldots A_{-(m-1)}A_{-m}$ where $A_i \in \{0,1,2, \ldots, 9\}$

Smallest non-zeros fraction is $0.0_{-1}0_{-2}\ldots 0_{-(m-1)}1_{-m} = 10^{-m}$
Largest fraction is $0.9_{-1}9_{-2}\ldots 9_{-(m-1)}9_{-m} = 1 - 10^{-m}$

**Example**: for m equal to 3 ➔ 3 digits decimal fraction;

The minimum fraction is $10^{-3}$ or 0.001
The maximum number is $1 - 10^{-3}$ or 0.999



## Number Ranges – Base-r Numbers

- Consider a base-r integer of n digits:

  $A_{n-1}A_{n-2}\ldots A_1A_0$ where $A_i \in \{0,1,2, \ldots, r\text{-}1\}$

Smallest integer is $0_{n-1}0_{n-2}\ldots 0_1 0_0 = 0$
Largest integer is $(r-1)_{n-1}(r-1)_{n-2}\ldots(r-1)_1(r-1)_0 = r^n - 1$

**Example**: for r = 5, n equal to 3 ➔ 3 digits base-5 integer;

The maximum integer is $(444)_5$ or $(5^3 - 1)_{10}$

To check:

the decimal equivalent of $(444)_5$ is $4X5^2+4X5^1+4 = (124)_{10}$ or simply $5^3 - 1 = (124)_{10}$

## Number Ranges - Base-r Numbers

- Consider a base-r fraction of m digits:

  $0.A_{-1}A_{-2}\ldots A_{-(m-1)}A_{-m}$ where $A_i \in \{0,1,2, \ldots, r\text{-}1\}$

Smallest non-zero fraction is

$$(0.0_{-1}0_{-2}\ldots 0_{-(m-1)}1_{-m})_r = (r^{-m})_{10}$$

Largest fraction is

$$(0.(r\text{-}1)_{-1}(r\text{-}1)_{-2}\ldots(r\text{-}1)_{-(m-1)}(r\text{-}1)_{-m})_r = (1 - r^{-m})_{10}$$

**Example**: for r = 5 and m equal to 3 ➔ 3 digits base-5 fraction;

The maximum number is $(0.444)_5$ or $1 - 5^{-3} = 0.992$



## Number Ranges - Base-r Numbers – cont'd

| | | Decimal (r=10) | Binary (r = 2) | Octal (r = 8) | Hex (r = 16) |
|---|---|---|---|---|---|
| Integer | Min | $0_{n-1}0_{n-2}\ldots 0_1 0_0$ = 0 | $0_{n-1}0_{n-2}\ldots 0_1 0_0$ = 0 | $0_{n-1}0_{n-2}\ldots 0_1 0_0$ = 0 | $0_{n-1}0_{n-2}\ldots 0_1 0_0$ = 0 |
| | Max | $9_{n-1}9_{n-2}\ldots 9_1 9_0$ $= 10^n - 1$ | $(1_{n-1}1_{n-2}\ldots 1_1 1_0)_2$ $= (2^n - 1)_{10}$ | $(8_{n-1}8_{n-2}\ldots 8_1 8_0)_8$ $= (8^n - 1)_{10}$ | $(F_{n-1}F_{n-2}\ldots F_1 F_0)_{16}$ $= (16^n - 1)_{10}$ |
| fraction | Min | $0.0_{-1}0_{-2}\ldots 0_{-(m-1)}1_{-m}$ $= 10^{-m}$ | $(0.0_{-1}0_{-2}\ldots 0_{-(m-1)}1_{-m})_2$ $= (2^{-m})_{10}$ | $(0.0_{-1}0_{-2}\ldots 0_{-(m-1)}1_{-m})_8$ $= (8^{-m})_{10}$ | $(0.0_{-1}0_{-2}\ldots 0_{-(m-1)}1_{-m})_{16}$ $= (16^{-m})_{10}$ |
| | Max | $0.9_{-1}9_{-2}\ldots 9_{-(m-1)}9_{-m}$ $= 1 - 10^{-m}$ | $(0.1_{-1}1_{-2}\ldots 1_{-(m-1)}1_{-m})_2$ $= (1 - 2^{-m})_{10}$ | $(0.7_{-1}7_{-2}\ldots 7_{-(m-1)}7_{-m})_8$ $= (1 - 8^{-m})_{10}$ | $(0.F_{-1}F_{-2}\ldots F_{-(m-1)}F_{-m})_{16}$ $= (1 - 16^{-m})_{10}$ |

## Exercises

- What is $8^4$ equal to in octal?
  $(8^4)_{10} = (10000)_8$
- What is $2^5$ equal to in binary?
  $(2^5) = (100000)_2$
- What is $16^4 - 1$ equal to in Hex?
- What is $2^3 - 2^{-2}$ equal to in Binary?
- What is $16^5 - 16^4$ equal to in Hex?
- What is $3^4 - 3^{-2}$ equal to in base-3?
- What is $2^4 - 2^{-2}$ equal to in base-3?



# Addition and Subtraction of (Unsigned) Numbers

## Binary Addition of UNSIGEND Numbers

- Consider the following example:
  Find the summation of $(1100)_2$ and $(11001)_2$

**Solution:**

```
                110000  ← Carry
Augend           01100
Addend          +11001
------------    ---------
sum             100101
```

- Note that
  - 0+0 = 0, 0+1 = 1+0 = 1, and 1 + 1 = 0 and the carry is 1
  - If the maximum no of digits for the augend or the addend is n, then the summation has either n or n+1 digits
  - This procedure works even for non-integer binary numbers



## Binary Subtraction of UNSIGEND Numbers

- Consider the following example:
  Subtract $(10010)_2$ from $(10110)_2$

**Solution:**

```
Minuend          10110
Subtrahend      -10010
------------    ---------
Difference       00100
```

- Note that
  - $(10110)_2$ is greater than $(10010)_2$ → The result is POSITIVE
  - 0-0 = 0, 1-0 = 1, and 1 - 1 = 0
  - The difference size is always less or equal to the size of the minued or the subtrahend
  - This procedure works even for non-integer binary numbers

## Binary Subtraction – cont'd

- Consider the following example:
  Subtract $(10011)_2$ from $(10110)_2$

**Solution:**

| | | |
|---|---|---|
| | 00110 | ← Borrow |
| Minuen | **10110** | |
| Subtrahend | **-10011** | |
| ------------ | --------- | |
| Difference | **00011** | |

- Note that
  - $(10110)_2$ is greater than $(10011)_2$ → result is positive
  - 0-1= 1, and the borrow from next significant digit is 1
  - This procedure works even for non-integer binary numbers



## Binary Subtraction – cont'd

- Consider the following example:
  Subtract $(11110)_2$ from $(10011)_2$

**Solution:**

| | | | | |
|---|---|---|---|---|
| | | | 00110 | ← Borrow |
| Minuen | **10011** | | **11110** | |
| Subtrahend | **-11110** | | **-10011** | |
| ------------ | --------- | 2 | ---------- | 1 |
| Difference | **-01011** | ← | **01011** | |

- Note that
  - $(10011)_2$ is smaller than $(11110)_2$ → result is negative
  - This procedure works even for non-integer binary numbers

## Binary Multiplication of UNSIGEND Numbers

- Consider the following example:
  Multiply $(1011)_2$ by $(101)_2$

**Solution:**

```
        Multiplicand          1011
        Multiplier          X 101
        ------------        ---------
                              1011
                             0000
                            1011
                           -------
        Product            110111
```



## Sums and Products in Base r (Unsigned Numbers)

- For sums and Products in base-r (r > 2) systems
  - Memorize tables for sums and products
  - Convert to Dec ➔ perform operation ➔ convert back to base-r
- **Example**: Find the summation of $(59F)_{16}$ and $(E46)_{16}$?

010 ← Carry
59F
+E46
------
13E5

1

F → 15
6 → 6
----
$21_{10} \rightarrow 16_{10} + 5_{10}$
$\rightarrow 15_H$

2

1
9 → A → 10
4 → 4 → 4
----
$14_{10} \rightarrow E_H$

3

5 → 5
E → 14
----
$19_{10} \rightarrow 16_{10} + 3_{10}$
$\rightarrow 13_H$

- This procedure is used for any base-r

## Sums and Products in Base-r – cont'd

- **Example**: Find the multiplication of $(762)_8$ and $(45)_8$?
- **Solution**:

```
 3310  ← Carry (for 4)
 4310  ← Carry (for 5)
  762
 X 45
 ------
 4672
3710
-------
43772
```

| Octal | Decimal | Octal |
|---|---|---|
| 5X2 | = 10 → 8 + 2 | = 12 |
| 5X6+1 | = 31 →24 + 7 | = 37 |
| 5X7+3 | = 38 →32 + 6 | = 46 |
| 4X2 | = 8 → 8 + 0 | = 10 |
| 4X6+1 | = 25 → 24 + 1 | = 31 |
| 4X7+3 | = 24+7 | = 37 |

Therefore, product = $(43772)_8$



# Decimal Codes

## Decimal Codes

- There are $2^n$ **DISTINCT** n-bit binary codes (group of n bits)
  - n bits can count $2^n$ numbers
- For us, humans, it is more natural to deal with decimal digits rather than binary digits
- 10 different digits ➔ we can use 4 bits to represent any digit
  - 3 bits count 8 numbers
  - 4 bits count 16 numbers ➔ to represent 10 digits we need 4 bits at least



## Binary Coded Decimal (BCD)

- Let the decimal digits be coded as show in table
- Then we can write numbers as

| Decimal Digit | Binary Code | Decimal Digit | Binary Code |
|---|---|---|---|
| 0 | 0000 | 5 | 0101 |
| 1 | 0001 | 6 | 0110 |
| 2 | 0010 | 7 | 0111 |
| 3 | 0011 | 8 | 1000 |
| 4 | 0100 | 9 | 1001 |

$(396)_{10} = (0011\ 1001\ 0110)_{BCD}$

Since 3 ➔ 0011, 9 = 1001, 6 = 0110

Although we are using the equal sign – but they are not equal in the mathematical sense; this is **JUST a code**

Note that $(396)_{10} = (110001100)_2 \neq (0011\ 1001\ 0110)_{BCD}$

## BCD Addition – Example 1

- Consider:

```
 000 ← Carry
  241
 +105
-------
  346
```

Addition in the Decimal Domain

```
        0010 → Carry
BCD for 1 = 0001
BCD for 5 = 0101
        --------
        0110 → BCD for 6

        0000 → Carry
BCD for 4 = 0100
BCD for 0 = 0000
        --------
        0100 → BCD for 4

        0000 → Carry
BCD for 2 = 0010
BCD for 1 = 0001
        --------
        0011 → BCD for 3
```

Addition in the Decimal Domain

**Hence, we can add BCD codes to obtain the correct decimal result. Is true always?**



## BCD Addition – Example 2

- Consider:

```
 110 ← Carry
  448
 +489
-------
  937
```

Addition in the Decimal Domain

```
        0010 → Carry
BCD for 8 = 1000
BCD for 9 = 1001
        --------
      1 0001 → > 9 → Need a correction step
       +0110    (add 6)
       --------
      1 0111 → (BCD for 7)

        0001 → Carry
BCD for 4 = 0100
BCD for 8 = 1000
        --------
        1101 → > 9 → Need a correction step
       +0110    (add 6)
       --------
      1 0011 → (BCD for 3)

        0001 → Carry
BCD for 4 = 0100
BCD for 4 = 0100
        --------
        1001 → BCD for 9
```

Addition in the Decimal Domain

## BCD Subtraction – Example 3

- Consider:

**110 ← Borrow**
**234**
**+135**
**-------**
**099**

Subtraction in the Decimal Domain

```
        1 1111 → borrow
BCD for 4 =  0100
BCD for 5 =  0101
             --------
              1111 → a borrow occurred – need correction
            - 0110    (subtract 6)
             --------
              1001 → (BCD for 9 – Also Let A1
        1 1111 → borrow
BCD for 2 =  0010
BCD for 8 =  0011
             --------
              1111 → a borrow occurred – need correction
           -  0110 (subtract 6)
             --------
              1011 → (BCD for 9 – Also Let A2 = A2 -1 = 1)

         0  0000 → Carry
BCD for 1 =   0001
BCD for 1 =   0101
              --------
               0000 → BCD for 0 – no borrow occurred – no
                      need for correction
```

Subtraction in the Decimal Domain



## BCD Addition – Summary

- BCD codes: decimal digits are assigned 4 bit codes
- We can perform additions using the BCD digits
  - If the result of adding two BCD digits is greater than 9, a correction step is required in order produce the correct BCD digit
  - To correct: add 6
  - If a carry is produced → move it to next BCD digits addition

## Alphanumeric Codes

- We have
  - 10 decimal digits
  - 26 X 2 (English) letters: capital and small case
  - Some special characters {; , . : + - etc}
- If we assign each character of these a binary code, then computers can exchange alphanumeric information (letters, numbers, etc) by exchanging binary digits
- One binary code is the American Standard Code for Information Interchange (ASCII)



## ASCII

- A 7-bits code → 128 distinct codes
  - 96 printable characters (26 upper case letter, 26 lower case letters, 10 decimal digits, 34 non-alphanumeric characters)
  - 32 non-printable character
    - Formatting effectors (CR, BS, …)
    - Info separators (RS, FS, …)
    - Communication control (STX, ETX, …)
- Computers typically use words sizes that are multiples of 2
  - Usually 8 bits are used for the ASCII code with the $8^{th}$ (left most bit) bit set to zero, OR
  - The ASCII code is extended → Extended ASCII (platform dependant)
- A good reference about ASCII and Extended ASCII is found at http://www.cplusplus.com/doc/papers/ascii.html

# ASCII – cont'd

- A 7-bits code ➔ 128 distinct codes
- The American Standard Code for Information Interchange (ASCII) uses seven binary digits to represent 128 characters as shown in the table.

```
00 NUL| 01 SOH| 02 STX| 03 ETX| 04 EOT| 05 ENQ| 06 ACK| 07 BEL
08 BS | 09 HT | 0A NL | 0B VT | 0C NP | 0D CR | 0E SO | 0F SI
10 DLE| 11 DC1| 12 DC2| 13 DC3| 14 DC4| 15 NAK| 16 SYN| 17 ETB
18 CAN| 19 EM | 1A SUB| 1B ESC| 1C FS | 1D GS | 1E RS | 1F US
20 SP | 21  ! | 22  " | 23  # | 24  $ | 25  % | 26  & | 27  '
28  ( | 29  ) | 2A  * | 2B  + | 2C  , | 2D  - | 2E  . | 2F  /
30  0 | 31  1 | 32  2 | 33  3 | 34  4 | 35  5 | 36  6 | 37  7
38  8 | 39  9 | 3A  : | 3B  ; | 3C  < | 3D  = | 3E  > | 3F  ?
40  @ | 41  A | 42  B | 43  C | 44  D | 45  E | 46  F | 47  G
48  H | 49  I | 4A  J | 4B  K | 4C  L | 4D  M | 4E  N | 4F  O
50  P | 51  Q | 52  R | 53  S | 54  T | 55  U | 56  V | 57  W
58  X | 59  Y | 5A  Z | 5B  [ | 5C  \ | 5D  ] | 5E  ^ | 5F  _
60  ` | 61  a | 62  b | 63  c | 64  d | 65  e | 66  f | 67  g
68  h | 69  i | 6A  j | 6B  k | 6C  l | 6D  m | 6E  n | 6F  o
70  p | 71  q | 72  r | 73  s | 74  t | 75  u | 76  v | 77  w
78  x | 79  y | 7A  z | 7B  { | 7C  | | 7D  } | 7E  ~ | 7F DEL
```



# Unicode

- Unicode describes a 16-bit standard code for representing symbols and ideographs for the world's languages.

**First 256 Codes for Unicode**[a]

| | Control | | ASCII | | | | | | Control | | Latin 1 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 000 | 001 | 002 | 003 | 004 | 005 | 006 | 007 | 008 | 009 | 00A | 00B | 00C | 00D | 00E | 00F |
| 0 | CTRL | CTRL | SPACE | 0 | @ | P | ` | p | CTRL | CTRL | NB SP | ° | À | Ð | à | D |
| 1 | CTRL | CTRL | ! | 1 | A | Q | a | q | CTRL | CTRL | ¡ | ± | Á | Ñ | á | ñ |
| 2 | CTRL | CTRL | " | 2 | B | R | b | r | CTRL | CTRL | ¢ | ² | Â | Ò | â | ò |
| 3 | CTRL | CTRL | # | 3 | C | S | c | s | CTRL | CTRL | £ | ³ | Ã | Ó | ã | ó |
| 4 | CTRL | CTRL | $ | 4 | D | T | d | t | CTRL | CTRL | ¤ | ´ | Ä | Ô | ä | ô |
| 5 | CTRL | CTRL | % | 5 | E | U | e | u | CTRL | CTRL | ¥ | µ | Å | Õ | å | õ |
| 6 | CTRL | CTRL | & | 6 | F | V | f | v | CTRL | CTRL | ¦ | ¶ | Æ | Ö | æ | ö |
| 7 | CTRL | CTRL | ' | 7 | G | W | g | w | CTRL | CTRL | § | · | Ç | × | ç | ÷ |
| 8 | CTRL | CTRL | ( | 8 | H | X | h | x | CTRL | CTRL | ¨ | ¸ | È | Ø | è | ø |
| 9 | CTRL | CTRL | ) | 9 | I | Y | i | y | CTRL | CTRL | © | ¹ | É | Ù | é | ù |
| A | CTRL | CTRL | * | : | J | Z | j | z | CTRL | CTRL | ª | º | Ê | Ú | ê | ú |
| B | CTRL | CTRL | + | ; | K | [ | k | { | CTRL | CTRL | « | » | Ë | Û | ë | û |
| C | CTRL | CTRL | , | < | L | \ | l | \| | CTRL | CTRL | ¬ | ¼ | Ì | Ü | ì | ü |
| D | CTRL | CTRL | - | = | M | ] | m | } | CTRL | CTRL | - | ½ | Í | Ý | í | ý |
| E | CTRL | CTRL | . | > | N | ^ | n | ~ | CTRL | CTRL | ® | ¾ | Î | þ | î | þ |
| F | CTRL | CTRL | / | ? | O | _ | o | CTRL | CTRL | CTRL | ¯ | ¿ | Ï | ß | ï | ÿ |

[a] Unicode, Inc., The Unicode Standard: Worldwide Character Encoding, Version 1.0, Volume 1, © 1990, 1991 by Unicode, Inc. Reprinted by permission of Addison-Wesley Publishing Company, Inc.

## Problems of Interest

- Problem List:

- Homework: Chapter 1, pages 24-26**:** 2, 3, 8, 12, 16, 19, 24, 26
  Due date: Saturday July 16, 2005 (in class)



# Signed Numbers Representations

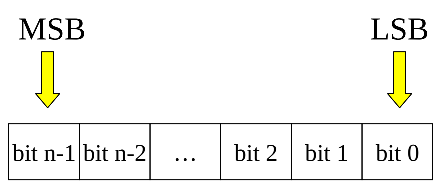
# Machine Representation of Numbers

- Computers store numbers in special digital electronic devices called REGISTERS
- REGISTERS consist of a fixed number of storage elements
- Each storage element can store one BIT of data (either 1 or 0)
- A register has a FINITE number of bits
  - Register size (n) is the number of bits in this register
  - N is typically a power of 2 (e.g. 8, 16, 32, 64, etc.)
  - A register of size n can represent $2^n$ distinct values
  - Numbers stored in a register can be either signed or unsigned



# N-bit Register

- N-storage elements




- Each storage element capable of holding ONE bit (either 1 or –0
- n-bits can represent $2^n$ distinct values
  - For example if unsigned integer numbers are to be represented, we can represent all numbers from 0 to $2^n$-1 (recall the number ranges for n-bits)
  - If we use it to represent signed numbers, still it can hold 2n different numbers – we will learn about the ranges of these numbers in the coming slides

## N-bit Register – cont'd

- Using a 4-bit register, $(13)_{10}$ or $(D)_H$ is represented as follows:

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | 1 |

- Using an 8-bit register, $(13)_{10}$ or $(D)_H$ is represented as follows:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |

- Note that ZEROS are used to pad the binary representation of 13 in the 8-bit register
- We are still using UNSIGNED NUMBERS

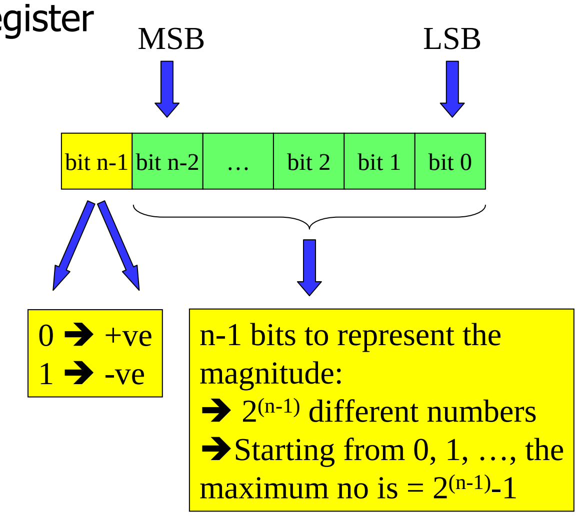


## Signed Number Representation

- To report a "signed" number, you need to specify its:
  - Magnitude (or absolute value), and
  - Sign (positive or negative)
- There are to major techniques to represent signed numbers
  1. Signed Magnitude Representation
  2. Complement Method

## Signed Magnitude Representation

- N-bit register






## Signed Magnitude Representation – Example 1:

- Show how +6, -6, +13, and –13 are represented using a 4-bit register

- Solution: Using a 4-bit register, the leftmost bit is reserved for the sign, which leaves 3 bits only to represent the magnitude

➔ The largest magnitude that can be represented = $2^{(4-1)} - 1 = 7 < 13$

Hence, the numbers +13 and –13 can NOT be represented using the 4-bit register

## Signed Magnitude Representation – Example 1: cont'd

- Solution (cont'd):

However both –6 and +6 can be represented as follows:

Singed-Magnitude representation of +6 →

| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 0 |

notice the difference

Singed-Magnitude representation of -6 →

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 0 |



## Signed Magnitude Representation – Example 2:

- Show how +6, -6, +13, and –13 are represented using an 8-bit register

- Solution: Using an 8-bit register, the leftmost bit is reserved for the sign, which leaves 7 bits only to represent the magnitude

➔ The largest magnitude that can be represented = $2^{(8-1)} - 1 = 127$

Hence, the numbers can be represented using the 8-bit register

## Signed Magnitude Representation – Example 2: cont'd

- Solution (cont'd):

  Since 6 and 13 are equal to : 110 and 1101 respectively, the required representations are

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Singed-Magnitude representation of +6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Singed-Magnitude representation of -6 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Singed-Magnitude representation of +13 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| Singed-Magnitude representation of -13 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |



## Things We Learned About Signed-Magnitude Representation

- For an n-bit register
  - Leftmost bit is reserved for the sign (0 for +ve and 1 for –ve)
  - Remaining n-1 bits represent the magnitude
  - $2^{(n-1)}$ different numbers: – minimum is zero and maximum is $2^{(n-1)}-1$
- Two representations for zero: +0 and –0
- Range of numbers: from $-\{2^{(n-1)}-1\}$ to $+\{2^{(n-1)}-1\}$ ➔ symmetric range

## Complement Representation

- +ve numbers (+N) are represented exactly the same way as in signed-magnitude representation
- -ve numbers (-N) are represented by the *complement* of N or N′

How is the complement of N or N′ defined?

N′ = M – N     where M is some constant



## Properties of the Complement Representation

- The complement of the complement of N is equal to N:

Proof: (N′)′ = M – (M – N) = -(-N) = N

Same as with –ve numbers definition!

- The complement method representation of signed numbers simplifies implementation of arithmetic operations like subtraction:

e.g.: A – B can be replaced by A + (-B) or A + B′ using the complement method

Therefore to perform subtraction using computers we complement and add the subtrahend

## How to Choose M?

- Consider the following number:

$$X = X_{n-1} \ldots X_2 X_1 X_0 \,.\, X_{-1} X_{-2} \ldots X_{-(m-1)} X_{-m}$$

(n integral digits – m fractional digits)

- Using the base-r number system, there can be two types of the complement representation
  - Radix Complement (R's Complement)
  
  ➔ $M = r^n$
  - Diminished Radix Complement (R-1's Complement):
  
  ➔ $M = r^n - r^{-m}$
  
  $= r^n - \text{ulp}$

Recall that $r^n = 1_n 0_{n-1} \ldots 0_1 0_0$
= 1 followed by n zeros
Recall that $r^{-m} = 0\ldots00.00..01$
= unit in the least position



## How to Choose M? – cont'd

- Note that:
  - $M = r^n - r^{-m}$ is the LARGEST unsigned number that can be represented
  - From the definitions of M, Rs complement of N is equal to R-1's complement of N plus ulp

## Summary of Complement Method

- R's Complement:

| Number System | R's Complement | Complement of X |
|---|---|---|
| Decimal | 10's Complement | $X'_{10} = 10^n - X$ |
| Binary | 2's Complement | $X'_2 = 2^n - X$ |
| Octal | 8's Complement | $X'_8 = 8^n - X$ |
| Hexadecimal | 16's Complement | $X'_{16} = 16^n - X$ |



## Summary of Complement Method – cont'd

- R-1's Complement:

| Number System | R-1's Complement | Complement of X |
|---|---|---|
| Decimal | 9's Complement | $X'_9 = (10^n - 10^{-m}) - X$<br>$= 99...99.99...99 - X$ |
| Binary | 1's Complement | $X'_1 = (2^n - 2^{-m}) - X$<br>$= 11...11.11...11 - X$ |
| Octal | 7's Complement | $X'_7 = (8^n - 8^{-m}) - X$<br>$= 77...77.77...77 - X$ |
| Hexadecimal | 15's Complement | $X'_{15} = (16^n - 16^{-m}) - X$<br>$= FF...FF.FF...FF - X$ |

## Example 1a:

- Find the 9's and 10's complement of 2357?
- <u>Solution</u>:

$X = 2357 \Rightarrow n = 4$

$X'_9 = (10^4 - ulp) - X$

$= (10000 - 1) - 2357$

$= 9999 - 2357$

$= 7642$

$X'_{10} = 10^4 - X$

$= 10000 - 2357$

$= 7643$

Or alternatively,

$X'_{10} = X'_9 + ulp = 7642 + 1 = 7643$

Note that: $X + X'_9 = 2357 + 7642 = 9999 = M$
While $X + X'_{10} = 2357 + 7643 = 1\ 0000 = M$



## Example 1b:

- Find the 9's and 10's complement of 2895.786?
- <u>Solution</u>:

$X = 2895.786 \Rightarrow n = 4, m = 3$

$X'_9 = (10^4 - ulp) - X$

$= (10000 - 0.001) - 2895.786$

$= 9999.999 - 2895.786$

$= 7104.213$

$X'_{10} = 10^4 - X$

$= 10000 - 2895.786$

$= 7104.214$

Or alternatively,

$X'_{10} = X'_9 + ulp = 7642 + 1 = 7104.214$

Note that: $X + X'_9 = 2895.786 + 7104.213 = 9999.999 = M$
While $X + X'_{10} = 2895.786 + 7104.214 = 1\ 0000.000 = M$

## Example 2a:

- Find the 1's and 2's complement of 110101010?
- Solution:

$X = 110101010 \Rightarrow n = 9$

$X'_1 = (2^9 - ulp) - X$

$= (1000000000 - 1) - 110101010$

$= 111111111 - 110101010$

$= 001010101$

$X'_2 = 2^9 - X$

$= 1000000000 - 110101010$

$= 001010110$

Or alternatively,

$X'_2 = X'_1 + ulp = 001010101 + 1 = 001010110$

Note that: $X + X'_1 = 110101010 + 001010101$
$= 111111111 = M$
While $X + X'_2 = 110101010 + 001010110$
$= 1\ 000000000 = M$



## Example 2b:

- Find the 1's and 2's complement of 1010.001?
- Solution:

$X = 1010.001 \Rightarrow n = 4, m = 3$

$X'_1 = (2^4 - ulp) - X$

$= (10000 - 0.001) - 1010.001$

$= 1111.111 - 1010.001$

$= 0101.110$

$X'_2 = 2^4 - X$

$= 10000 - 1010.001$

$= 0101.111$

Or alternatively,

$X'_2 = X'_1 + ulp = 0101.110 + 0.001 = 0101.111$

Note that: $X + X'_1 = 1010.001 + 0101.110$
$= 1111.111 = M$
While $X + X'_2 = 1010.001 + 0101.110$
$= 1\ 0000.000 = M$

## Notes On 1's and 2's Complements Computation:

- 1's complement can be obtained by bitwise complementing the bits of X

Examples (from previous slide)

$$X = 1010.001 \Rightarrow X'_1 = 0101.110$$

- 2's complement of X can be obtained by:
  1. Adding ulp to its 1's complement, or

upl is added

$$X = 1010.001 \Rightarrow X'_1 = 0101.110 \Rightarrow X'_2 = 0101.111$$

  2. Scanning X from right to left, copy all digits including first 1, complement all remaining digits

$$X = \underbrace{1010.00}_{\text{You complement subsequent bits}}1 \Rightarrow X'_2 = 0101.11\underbrace{1}_{\text{1st one you keep}}$$



## Example 3a:

- Find the 7's and the 8's complement of the following octal number 6770?
- <u>Solution</u>:

$X = 6770 \Rightarrow n = 4$

$X'_7 = (8^4 - \text{ulp}) - X$

$= (10000 - 1) - 6770$

$= 7777 - 6770$

$= 1007$

$X'_8 = 8^4 - X$

$= 10000 - 6770$

$= 1010$

Or alternatively,

$X'_8 = X'_7 + \text{ulp} = 1007 + 1 = 1010$

# Example 3b:

- Find the 7's and the 8's complement of the following octal number 541.736?
- Solution:

$X = 541.736 \rightarrow n = 3, m = 3$

$X'_7 = (8^3 - ulp) - X$

$= (10000 - 0.001) - 541.736$

$= 777.777 - 541.736$

$= 236.041$

$X'_8 = 8^3 - X$

$= 10000 - 541.736$

$= 236.042$

Or alternatively,

$X'_8 = X'_7 + ulp = 236.041 + 0.001 = 236.042$



# Example 4a:

- Find the 15's and the 16's complement of the following Hex number 3FA9?
- Solution:

$X = 3FA9 \rightarrow n = 4$

$X'_{15} = (16^4 - ulp) - X$

$= (10000 - 1) - 3FA9$

$= FFFF - 3FA9$

$= C056$

$X'_{16} = 16^4 - X$

$= 10000 - 3FA9$

$= C057$

Or alternatively,

$X'_{16} = X'_{15} + ulp = C056 + 1 = C057$

## Example 4b:

- Find the 15's and the 16's complement of the following Hex number 9B1.C70?
- Solution:

$X = 9B1.C70 \rightarrow n = 3, m = 3$

$X'_{15} = (16^3 - ulp) - X$

$= (1000 - 0.001) - 9B1.C70$

$= FFF.FFF - 9B1.C70$

$= 64E.38F$

$X'_{16} = 16^3 - X$

$= 1000 - 9B1.C70$

$= 64E.390$

Or alternatively,

$X'_{16} = X'_{15} + ulp = 64E.38F + 0.001 = 64E.390$



## Complement Representation – Example 5:

- Show how +53 and -53 are represented in 8-bit registers using signed-magnitude, 1's complement and 2's complement?
- Solution:

Note that 53 = 32 + 16 + 4 + 1,

Therefore using 8-bit signed-magnitude:

  - +53 → **0**0110101    -53 → **1**0110101
- To find the representation in complement method:

## Complement Representation – Example 5: cont'd

- Solution: cont'd

To find the representation in complement method.
(53)10 = $(00110101)_2$ when written in 8-bit binary

1's complement ➔ 11001010 (inverting every bit)

2's complement ➔ 11001011 (adding ulp to $X'_1$)



## Complement Representation – Example 5: cont'd

- Solution: cont'd

Putting all the results together in a table

| | +53 | -53 |
|---|---|---|
| Signed-Magnitude | 00110101 | 10110101 |
| 1's Complement | 00110101 | 11001010 |
| 2's Complement | 00110101 | 11001011 |

Note:
- +53 representation is the same for all methods
- For +53, the leftmost bit is 0 (+ve number)
- For -53, the leftmost bit is 1 (-ve number)

## Example 6:

- For the shown 4-bit representations, indicate the corresponding decimal value in the shown representation



## Example 6: cont'd

- Signed-Magnitude and 1's complement are symmetrical representations with TWO representations for ZERO
- Range from signed-magnitude and 1's complement is from -7 to +7
- 2's complement representation is not symmetrical
- Range for 2's complement is from -8 to +7 – with one representation for ZERO

| | Unsigned | Signed-Magnitude | 1's Complement | 2's Complement |
|---|---|---|---|---|
| 0000 | 0 | 0 | 0 | 0 |
| 0001 | 1 | 1 | 1 | 1 |
| 0010 | 2 | 2 | 2 | 2 |
| 0011 | 3 | 3 | 3 | 3 |
| 0100 | 4 | 4 | 4 | 4 |
| 0101 | 5 | 5 | 5 | 5 |
| 0110 | 6 | 6 | 6 | 6 |
| 0111 | 7 | 7 | 7 | 7 |
| 1000 | 8 | -0 | -7 | -8 |
| 1001 | 9 | -1 | -6 | -7 |
| 1010 | 10 | -2 | -5 | -6 |
| 1011 | 11 | -3 | -4 | -5 |
| 1100 | 12 | -4 | -3 | -4 |
| 1101 | 13 | -5 | -2 | -3 |
| 1110 | 14 | -6 | -1 | -2 |
| 1111 | 15 | -7 | -0 | -1 |

## Summary

- The following table summarizes the properties and ranges for the studied signed number representations

| | Signed-Magnitude | 1's Complement | 2's Complement |
|---|---|---|---|
| Symmetric | Y | Y | N |
| No of Zeros | 2 | 2 | 1 |
| Largest | $2^{(n-1)}-1$ | $2^{(n-1)}-1$ | $2^{(n-1)}-1$ |
| Smallest | $-\{2^{(n-1)}-1\}$ | $-\{2^{(n-1)}-1\}$ | $-2^{(n-1)}$ |



## Exercise

- Find the binary representation in signed magnitude, 1's complement, and 2's complement for the following decimal numbers: +13, -13, +39, +1, -1, +73, and -73. For all numbers, show the required representation for 6-bit and 8-bit registers

## 10's Complement

- For n = 1 and 2

| $X'_{10}$ (n=1) | $X'_{10}$ using+/- in decimal |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | -5 |
| 6 | -4 |
| 7 | -3 |
| 8 | -2 |
| 9 | -1 |

| $X'_{10}$ (n=2) | $X'_{10}$ using+/- in decimal |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 02 | 2 |
| .. | .. |
| 09 | 9 |
| 10 | 10 |
| 11 | 11 |
| 12 | 12 |
| ... | .. |
| 49 | 49 |
| 50 | -50 |
| 51 | -49 |
| 52 | -48 |
| ... | ... |
| 98 | -2 |
| 99 | -1 |



## 8's Complement

- For n = 1 and 2

| $X'_8$ (n=1) | $X'_8$ using+/- in decimal |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | -4 |
| 5 | -3 |
| 6 | -2 |
| 7 | -1 |

| $X'_8$ (n=2) | $X'_8$ using+/- in decimal |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 02 | 2 |
| .. | .. |
| 07 | 7 |
| 10 | 8 |
| 11 | 9 |
| 12 | 10 |
| ... | .. |
| 36 | 30 |
| 37 | 31 |
| 40 | -32 |
| 41 | -31 |
| ... | ... |
| 70 | -8 |
| 71 | -7 |
| ... | ... |
| 76 | -2 |
| 77 | -1 |

## 16's Complement

- For n = 1 and 2

| $X'_{16}$ (n=1) | $X'_{16}$ using+/- in decimal |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | -8 |
| 9 | -7 |
| A | -6 |
| B | -5 |
| C | -4 |
| D | -3 |
| E | -2 |
| F | -1 |

| $X'_{16}$ (n=2) | $X'_{16}$ using+/- in decimal |
|---|---|
| 00 | 0 |
| 01 | 1 |
| … | … |
| 0E | 14 |
| 0F | 15 |
| 10 | 16 |
| 11 | 17 |
| … | … |
| 1F | 31 |
| 20 | 32 |
| 21 | 33 |
| … | … |
| 7E | 126 |
| 7F | 127 |
| 80 | -128 |
| 81 | -127 |
| … | … |
| F0 | -16 |
| F1 | -15 |
| … | … |
| FD | -3 |
| FE | -2 |
| FF | -1 |



# Operations On Binary Numbers

## Operation On Binary Numbers

- Assuming we are dealing with n-bit binary numbers
  - UNSIGNED, or
  - SIGNED (2's complement)
- A subtraction can always be made into an addition operation A – B = A + (-B) or A + (B')
  - Compute the 2's complement of the subtrahend and added to the minuend



## Operations on Binary Numbers

- The GENERAL OPERATION looks like:

$$
\begin{array}{lllllll}
C_n & C_{n-1} & C_{n-2} & \ldots & C_2 & C_1 & C_0 \quad \leftarrow \text{Carry generated} \\
 & A_{n-1} & A_{n-2} & \ldots & A_2 & A_1 & A_0 \quad \rightarrow \text{Number A (signed or otherwise)} \\
+ & B_{n-1} & B_{n-2} & \ldots & B_2 & B_1 & B_0 \quad \rightarrow \text{Number B (signed or otherwise)} \\
\hline
C_n & S_{n-1} & S_{n-2} & \ldots & S_2 & S_1 & S_0
\end{array}
$$

- Note that although we start with n-bit numbers, we can end up with a result consisting of n+1 bits
  - Remember we are using n-bit registers!!
  - What to do with this extra bit ($C_n$)?

## Addition of Unsigned Numbers - Review

- For n-bit words, the n-bit UNSIGNED binary numbers range from $(0_{n-1}0_{n-2}\ldots0_1 0_0)_2$ to $(1_{n-1}1_{n-2}\ldots1_1 1_0)_2$

  i.e. they range from 0 to $2^{n-1}$
- When adding A to B as in:

$$
\begin{array}{llllllll}
C_n & C_{n-1} & C_{n-2} & \ldots & C_2 & C_1 & C_0 & \leftarrow \text{Carry generated} \\
 & A_{n-1} & A_{n-2} & \ldots & A_2 & A_1 & A_0 & \rightarrow \text{Number A (unsigned)} \\
+ & B_{n-1} & B_{n-2} & \ldots & B_2 & B_1 & B_0 & \rightarrow \text{Number B (unsigned)} \\
\hline
C_n & S_{n-1} & S_{n-2} & \ldots & S_2 & S_1 & S_0 &
\end{array}
$$

- If $C_n$ is equal to ZERO, then the result **DOES** fit into n-bit word ($S_{n-1}$ $S_{n-2}$ ... $S_2$ $S_1$ $S_0$)
- If $C_n$ is equal to ONE, then the result **DOES NOT** fit into n-bit word → An "OVERFLOW" indicator!



## Subtraction of Unsigned Numbers

- How to perform A – B (both defined as n-bit unsigned)?
- Procedure:
  1. Add the the 2's complement of B to A; this forms A + ($2^n$ – B)
  2. If (A >= B), the sum produces end carry signal ($C_n$); discard this carry
  3. If A < B, the sum does not produce end carry signal ($C_n$); result is equal to $2^n$ – (B-A), the 2's complement of B-A – Perform correction:
     - Take 2's complement of sum
     - Place –ve sign in front of result
     - Final result is –(A-B)

## Subtraction of Unsigned Numbers - NOTES

- Although we are dealing with unsigned numbers, we use the 2's complement to convert the subtraction into addition
- Since this is for UNSIGEND numbers, we have to use the –ve sign when the result of the operation is negative



## Subtraction of Unsigned Numbers – Example

- Example: X = 1010100 or $(84)_{10}$, Y = 1000011 or $(67)_{10}$ – Find X-Y and Y-X
- Solution:

n = 7

A) X – Y: X = 1010100
2's complement of Y = 0111101
Sum = 10010001
Discard $C_n$ (last bit) = 0010001 or $(17)_{10}$ ← X – Y

B) Y – X: Y = 1000011
2's complement of X = 0101100
Sum = 1101111
$C_n$ (last bit) is zero → need to perform correction
Y - X = -(2's complement of 1101111) = - 001001

## n-bit Unsigned Number Operations - Summary

N-bit operation

- Subtraction (A-B)
  - Take 2's complement of B and perform A+(B') → result
    - A < B
      - $C_n$ = 0;
      - - need correction
      - -find 2's complement of result
      - - Put –ve sign in front of the 2's complement
    - A ≥ B
      - $C_n$ = 1;
      - - no need for correction
      - - discard $C_n$
      - - Result valid
- Addition (A+B)
  - Perform regular binary addition A + B → result
    - $C_n$ = 0
      - Result valid
    - $C_n$ = 1
      - Result invalid (overflow)



## 2's Complement Review

- **For n-bit words, the 2's complement SIGNED binary numbers range from $-(2^{n-1})$ to $+(2^{n-1}-1)$**

  **e.g. for 4-bit words, range = – 8 to +7**
- **Note that MSB is always 1 for –ve numbers, and 0 for +ve numbers**

# Addition/Subtraction of n-bit Signed Numbers by Example (1)

- Consider

```
       01 1000         111 0000
  +6   00 0110     -6   11 1010
+ 13   00 1101    +13   00 1101
------ ------------ ----- -----------
 +19   01 0011     +7   00 0111      Cn = 1 → discarded

       00 1100         110 0100
  +6   00 0110     -6   11 1010
-  13  11 0011    - 13  11 0011
------ ------------ ----- -----------
  - 7  11 1001    -19   101101       Cn = 1 → discarded
```

n = 6 → range $-2^{6-1}$ = - 32 to $(2^{6-1}-1)$ = 31
**Hence:**
-All used numbers are valid (within the range)
-All results are also valid (within the range)

- **Any carry out of sign bit position is DISCARDED**
- **-ve results are automatically in 2's complement form (no need for an explicit –ve sign)!**

Are there cases when the results do not fit the n-bit register?



# Addition/Subtraction of n-bit Signed Numbers by Example (2)

- **Consider**

$C_n$ $C_{n-1}$

**0 10 0000**
16 01 0000
+ 23 01 0111
------ -------------
~~+39 10 0011~~

$C_n$ $C_{n-1}$

**1 10 0000** ← carry
-16 11 0000
+23 01 0111
----- ------------
+7 1 00 0111 ← Result is valid Discard $C_n$

$C_n$ $C_{n-1}$

**0 00 0000**
+16 01 0000
- 23 10 1001
------ -------------
- 7 11 1001

Result is valid

$C_n$ $C_{n-1}$

**1 00 0000** ← carry
-16 11 0000
- 23 10 1001
----- ------------
~~-39 1 01 1001~~

n = 6 → range $-2^{6-1}$ = - 32 to $(2^{6-1}-1)$ = 31
**Hence:**
-All used numbers are valid (within the range)
-All results are also valid (within the range)

though we started with valid 6-bit signed numbers the results is in valid for a 6-bit signed representation

## Addition/Subtraction of n-bit Signed Numbers by Example (2) – cont'd

- <u>NOTE:</u>
- The result is invalid (not within range) only if $C_{n-1}$ and $C_n$ are different! ➔ An OVERFLOW has occurred
- The result is valid (within range) if $C_{n-1}$ and $C_n$ are the same
  - If $C_n$ = 1; it needs to be discarded
- If result is valid and –ve, it will be in the correct 2's complement form



## Addition/Subtraction of n-bit Signed Numbers - Summary

- <u>Summary</u>