King Fahd University of Petroleum & Minerals
Computer Engineering Dept

EE 200 – Logic Design

Term 151

Dr. Ashraf S. Hasan Mahmoud
Rm 22-420
Ext. 1724
Email: ashraf@kfupm.edu.sa

1.1 Digital Systems

- Analog signals versus discrete signals
- Digitization – analog-to-digital conversion (ADC)
  - Sampling along the time access and Quantization along the y-axis ➔ Sampling Thereon
- General purpose computer – best example of a digital system
- Why digital is good?
  - Programmability – cost reduction
  - Advances in integrated circuits technology
  - Digital systems can be made to operate with extreme reliability – error correcting codes

1.2 Binary Numbers

- General number in base r is written as:

$$\underbrace{a_{n-1}\, a_{n-2} \,\dots\, a_2\, a_1\, a_0}_{\text{Integer Part (n digits)}} \,\blacksquare\, \underbrace{a_{-1}\, a_{-2} \,\dots\, a_{-(m-1)}\, a_{-m}}_{\text{Fraction Part (m digits)}}$$

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

$$a_{n-1}\, a_{n-2} \,\dots\, a_2\, a_1\, a_0 \,.\, a_{-1}\, a_{-2} \,\dots\, a_{-(m-1)}\, a_{-m}$$

FORM or SHAPE
OF NUMBER

is equal to

$$a_{n-1} X\, r^{n-1} + a_{n-2} X\, r^{n-2} + \dots a_2 X\, r^2 + a_1 X\, r^1 + a_0 X\, r^0 + a_{-1} X\, r^{-1} + a_{-2} X\, r^{-2} + \dots a_{-(m-1)} X\, r^{-(m-1)} + a_{-m} X\, r^{-m}$$

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 (Binary)

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

```
     5    4     3   2     1   0         -1   -2
    (1    1     0    1    0    1    .    1    1)2
```

←Positions

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 2:

…

$2^4 = 16$

$2^3 = 8$,

$2^2 = 4$,

$2^1 = 2$,

$2^0 = 1$

$2^{-1} = 0.5$

$2^{-2} = 0.25$,

…

Arithmetic Operations in Base r

- Mainly follow the same rules for decimal numbers
  - Base r → ONLY the r allowed digits are used
- Example operations for r = 2 (BINARY)

```
             1111 ← carry                        1100  ← borrow
Augend:     101101           Minuend:            101101
Addend:    +100111           Subtrahend:        -100111
           -------                              -------
Sum:       1010100           Difference:         000110


Multiplicand:     1011
Multiplier:       x101
                  -----
                  1011
                 0000
                1011
                -------
Product:        110111
```

partial product

Powers Of 2

- Textbook page 21

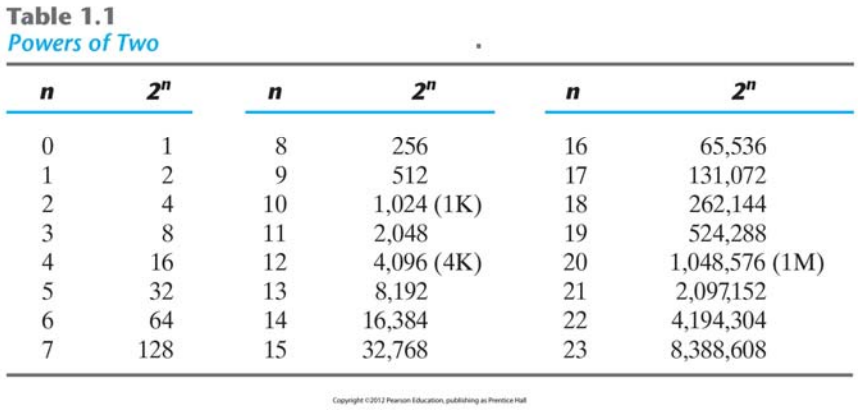


Table 1.1
Powers of Two

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024 (1K)	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096 (4K)	20	1,048,576 (1M)
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

- What is 1G? How many bytes does 1GB RAM has?

1.3 Base Number Conversions

- Representations of a number in a different radix are EQUIVALENT if they have the same decimal representation
  - E.g. $(0011)_8$ is equivalent to $(1001)_2$ – both are equal to decimal value 9
- Converting a base r number to decimal is done by expanding the number in a power series and adding all the terms
  - Refer to slides 5, 6, and 7.
- How to convert from decimal to base r?

Example 1.1: Decimal to Binary Conversion of Integer Numbers

- Convert 41 to binary.
- To convert a decimal *integer* to binary → decompose into powers of 2
  - Example: $(41)_{10} = (?)_2$

41 has ONE 32 → remainder is 9
9 has ZERO 16s → remainder is 9
9 has ONE 8 → remainder is 1
1 has ZERO 4s → remainder is 1
1 has ZERO 2s → remainder is 1
1 has ONE 1 → remainder is 0

Therefore $(41)_{10} = (101001)_2$

Example 1.1: 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	
41	20	1	← LSD
20	10	0	
10	5	0	
5	2	1	
2	1	0	
1	0	1	← MSD

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

- Therefore $(41)_{10} = (101001)_2$
- To check:

$1X2^5+1X2^3+1 = 32+8+1 = 41$

Example 1.2: Decimal to Octal Conversion of Integer Numbers– cont'd

- Convert 153 to octal.
- Using the table method of previous example:

No	No/8	Remainder	
153	19	1	LSD
19	2	3	
2	8	2	MSD

- Therefore $(153)_{10}$ = $(231)_8$
- To check:

$2X8^2+3X8^1+1 = 128+24+1 = 153$

Example 1.3: 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: $(0.001111)_2 = 1X2^{-3} +1X2^{-4}+1X2^{-5}+ 1X2^{-6} = (0.234375)_{10}$

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

Example 1.4: Decimal to Octal Conversion of Fractions

- Convert $(0.513)_{10}$ to to octal.
- Using the same procedure of Example 1.3, we get:

```
0.513 x 8 = 4.104
0.104 x 8 = 0.832
0.832 x 8 = 6.656
0.248 x 8 = 1.984
0.984 x 8 = 7.872
…
```

- Therefore $(0.513)_{10}$ = $(0.406517…)_8$
- Check the answer?
- Note the fraction representation in Octal may require infinite number of digits – here 1st 7 significant digits are used

How to Convert Decimal Numbers with BOTH Integer and Fraction Part to Base r?

- Answer:
  - Integer part is converted alone – as in Examples 1.1 and 1.2
  - Fraction part is converted alone – as in Examples 1.2 and 1.3
  - Combine the two answers
- Example: convert (153.513)10 to Octal

Using results of example 1.2 and 1.3

$(153)_{10}$ = $(231)_8$ and $(0.513)_{10}$ = $(0.406517)_8$

➔ $(153.513)_{10}$ = $(231.406517)_8$

Example: 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} ➔ 32/8 = 4 and remainder is 0 → 0

4/8 = 0 and remainder is 4 → 4

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

$(0.57)_{10}$ ➔ 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?

1.4 Octal and Hexadecimal Numbers

- The conversion from and to binary, octal, and hexadecimal plays an important role in digital computers
- Binary, octal, and hexadecimal systems are RELATED to one another since 2^3 = 8 and 2^4 = 16
  - ➔ Each octal digit corresponds to 3 binary digits, and
  - ➔ Each hexadecimal digit corresponds to 4 binary digits
- How

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 than:
  - 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

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

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

↕ ↕ ↕

1 2 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:

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

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

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

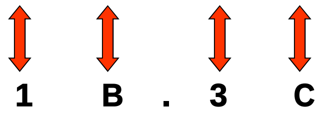
Conversion Between Hex and Binary – cont'd

- **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)



1 B . 3 C

- 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

Decimal, Binary, Octal, and Hexadecimal Systems Again:

- Textbook page 25

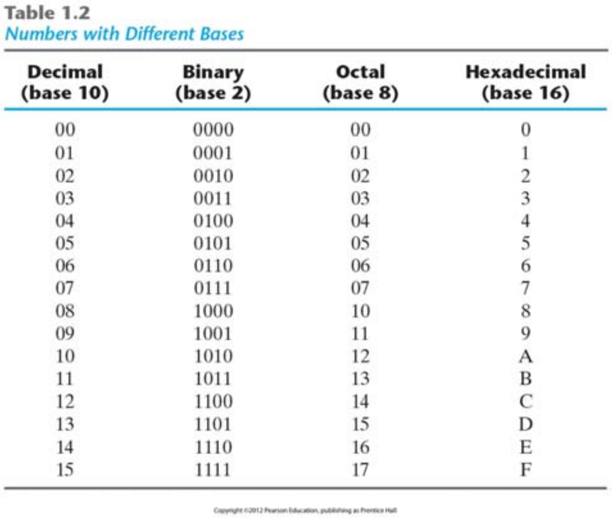


Table 1.2
Numbers with Different Bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Sample 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

1.5 Complements Of Numbers

- Used in digital computers to **simplify the subtraction operation**

- Two types of complements:
  - Diminished Radix Complement or (r-1)'s complement, and
  - Radix Complement or r's complement
- Examples:
  - r = 2 (BINARY) → 2's complement and 1's complement
  - r = 10 (DECIMAL) → 10's complement and 9's complement

Diminished Radix Complement

- Given: Base r, number of digits n, and number N
- Diminished radix complement is defined as

$$N' = (r^n - 1) - N$$

- What is the diminished radix complement of N'?

Diminished Radix Complement – cont’d

- Example 1: r = 10 for any N of n decimal digits ➔ N’ = (10^n – 1) - N
  - 10^n is 1 followed by n zeros ➔ (10^n – 1) is n 9’s!
- Example 2: r = 10, n = 4 ➔ N’ = $(9999)_{10}$ - N
- Example 3: r = 10, n = 6 – compute 9’s complement of
  - N = 546700 ➔ N’ = 999999 – 546700 = 453299
  - N = 012398 ➔ N’ = 999999 – 012398 = 987601

Diminished Radix Complement – cont’d

- Example 4: r = 2, for any N of n binary digits
  - 2^n is 1 followed by n zeros ➔ (2^n – 1) is n 1’s!
- Example 5: r = 2, n = 4 ➔ N’ = $(1111)_2$ - N
- Example 3: r = 2, n = 7 – compute 1’s complement of
  - N = 1011000 ➔ N’ = 1111111 – 1011000 = 0100111
  - N = 0101101 ➔ N’ = 1111111 – 0101101 = 1010010

9's Complement

- For n = 1 and 2

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

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	-49
51	-48
52	-47
…	…
98	-1
99	-0

1's Complement

- For n = 2 and 3

N'_2 (n=2)	N'_2 using+/- in decimal
00	0
01	1
10	-1
11	-0

N'_2 (n=3)	N'_2 using+/- in decimal
000	0
001	1
010	2
011	3
100	-3
101	-2
110	-1
111	-0

7's Complement

- For n = 1 and 2

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

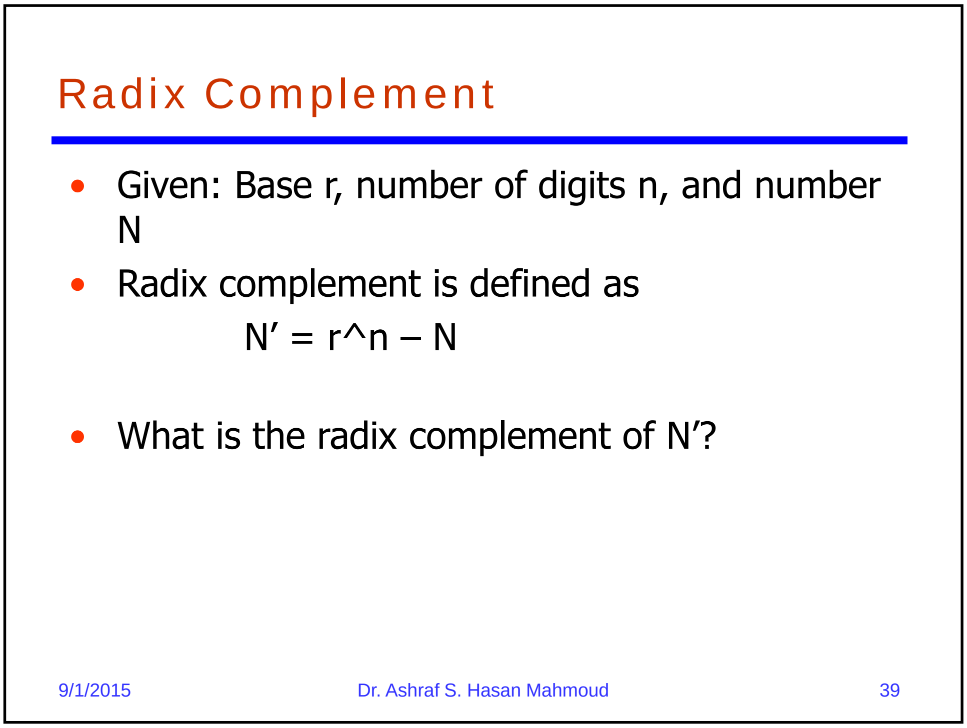
N'_8 (n=2)	N'_8 using+/- in decimal
00	0
01	1
02	2
..	..
07	7
10	8
11	9
12	10
…	..
36	30
37	31
40	-31
41	-36
…	…
70	-?
71	-?
…	…
76	-1
77	-0

15's Complement

- For n = 1 and 2

N'_{16} (n=1)	N'_{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

N'_{16} (n=2)	N'_{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

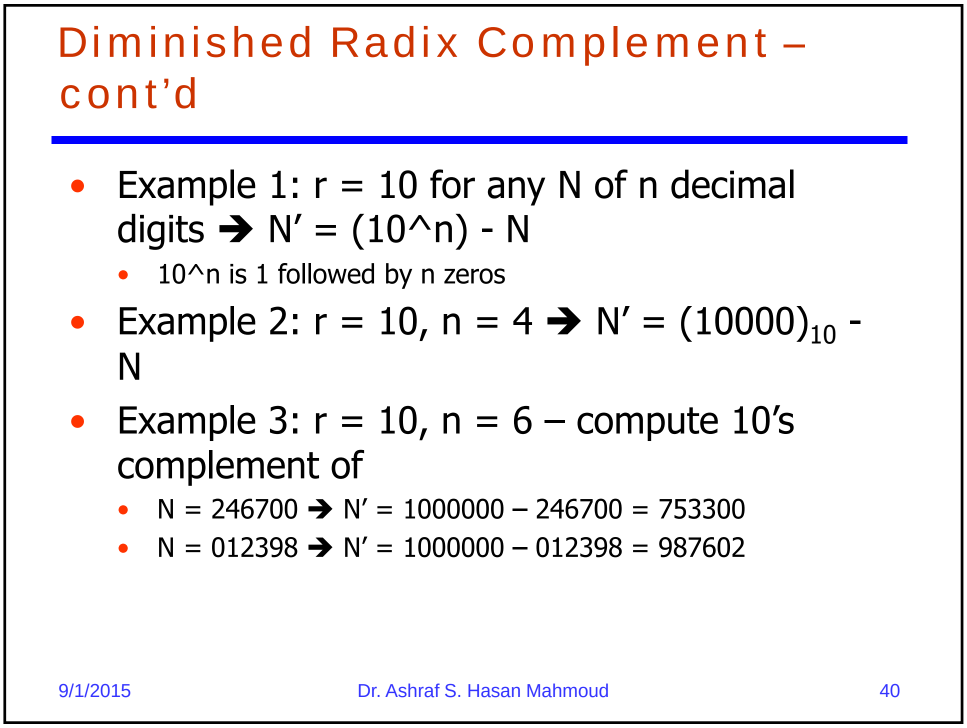


Radix Complement

- Given: Base r, number of digits n, and number N
- Radix complement is defined as

  N′ = r^n – N

- What is the radix complement of N′?



Diminished Radix Complement – cont'd

- Example 1: r = 10 for any N of n decimal digits ➔ N′ = (10^n) - N
  - 10^n is 1 followed by n zeros
- Example 2: r = 10, n = 4 ➔ N′ = $(10000)_{10}$ - N
- Example 3: r = 10, n = 6 – compute 10's complement of
  - N = 246700 ➔ N′ = 1000000 – 246700 = 753300
  - N = 012398 ➔ N′ = 1000000 – 012398 = 987602

Diminished Radix Complement – cont'd

- Example 4: r = 2, for any N of n binary digits
  - 2^n is 1 followed by n zeros
- Example 5: r = 2, n = 4 ➔ N' = $(10000)_2$ - N
- Example 3: r = 2, n = 7 – compute 2's complement of
  - N = 1101100 ➔ N' = 10000000 – 1101100 = 0010100
  - N = 0110111 ➔ N' = 10000000 – 0110111 = 1001001

10's Complement

- For n = 1 and 2

N'_{10} (n=1)	N'_{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

2's Complement

- For n = 2 and 3

N'_2 (n=2)	N'_2 using+/- in decimal
00	0
01	1
10	-2
11	-1

N'_2 (n=3)	N'_2 using+/- in decimal
000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

8's Complement

- For n = 1 and 2

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

N'_8 (n=2)	N'_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

N'$_{16}$ (n=1)	N'$_{16}$ 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

N'$_{16}$ (n=2)	N'$_{16}$ 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

Example: Signed Number Representation – r = 2, n = 4

- 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

Example: Signed Number Representation – r = 2, n = 4 - 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	$-\{2^{(n-1)}-1\}$	$-\{2^{(n-1)}-1\}$	$-2^{(n-1)}$

Summary – cont'd

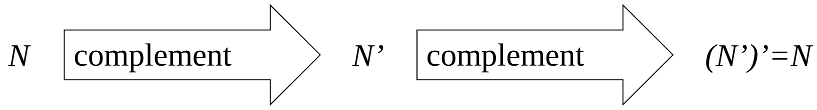
- The complement of the complement restores the number to its original value
- Proof:

  Given N, then N' is r^n – N

  Then (N')' should be r^n – (N') = r^n – (r^n) + N = N!

  Therefore, (N')' = N.

The above proof is the same for diminished radix complement.



1.6 Representation of Singed Binary Numbers

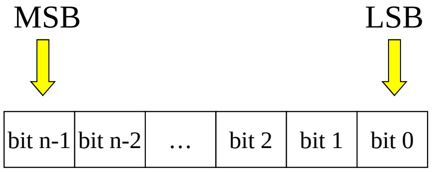
- There are two main techniques to represent signed numbers
  1. Signed Magnitude Representation
  2. Complement Method
     - Diminished-Radix complement
     - Radix complement

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



MSB

LSB

bit n-1	bit n-2	…	bit 2	bit 1	bit 0

- 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 2^n 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

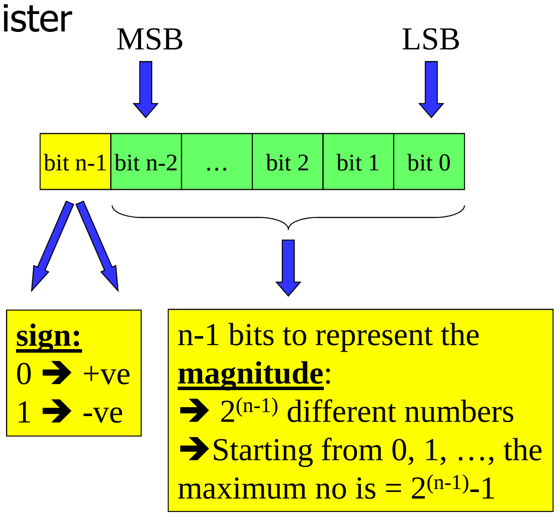
- Note that ZEROS are used to pad the binary representation of 13 in the 8-bit register
- NOTE: we are still using UNSIGNED NUMBERS

Signed Number Representation

- To report a “signed” number, you need to specify its:
  - Magnitude (or absolute value), and
  - Sign (positive or negative)

Signed Magnitude Representation

- N-bit register



bit n-1	bit n-2	…	bit 2	bit 1	bit 0

sign:
0 ➔ +ve
1 ➔ -ve

n-1 bits to represent the **magnitude**:
➔ $2^{(n-1)}$ different numbers
➔Starting from 0, 1, …, the maximum no is = $2^{(n-1)}$-1

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 = $2^{(4-1)} - 1 = 7 < 13$

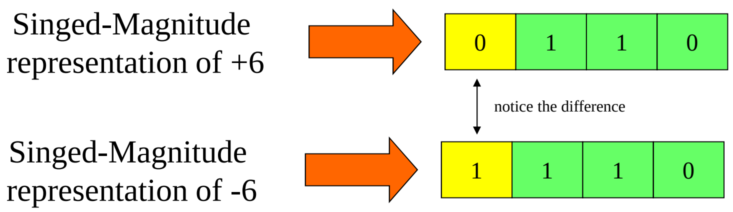
Hence, the numbers +13 and –13 can NOT be represented using the 4-bit register

---

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

- <u>Solution</u>: 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

- <u>Solution (cont'd)</u>:

  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:

<u>Proof</u>: (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

Signed Binary Numbers (r = 2, n = 4)

- Given a binary representation of a number, how can you tell whether the number is +ve or –ve?
- **<u>Sign extension rule?</u>** How would you write the number shown in table using r = 2 and n = 8?
  - i.e. what is -4 in 2'complement using n = 8?

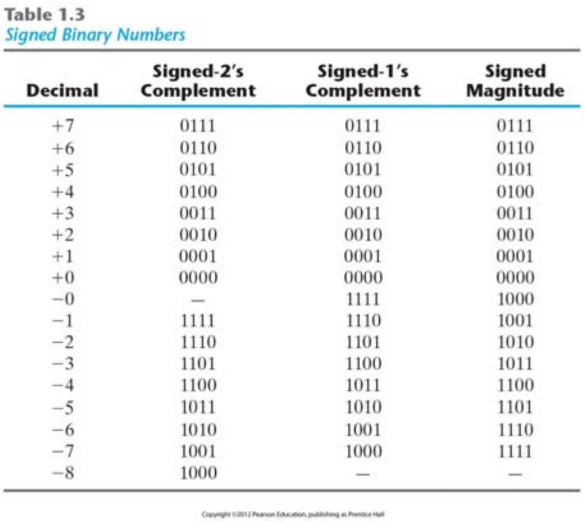


Table 1.3
Signed Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

Operations On Binary Numbers – REVIEW

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:

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

- 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 BINARY Numbers - Review

- **For n-bit words, the n-bit UNSIGNED binary numbers range from $(0_{n-1}0_{n-2}\ldots0_10_0)_2$ to $(1_{n-1}1_{n-2}\ldots1_11_0)_2$**

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

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

- **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 BINARY 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 BINARY 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 BINARY Numbers – Example (1)

- **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

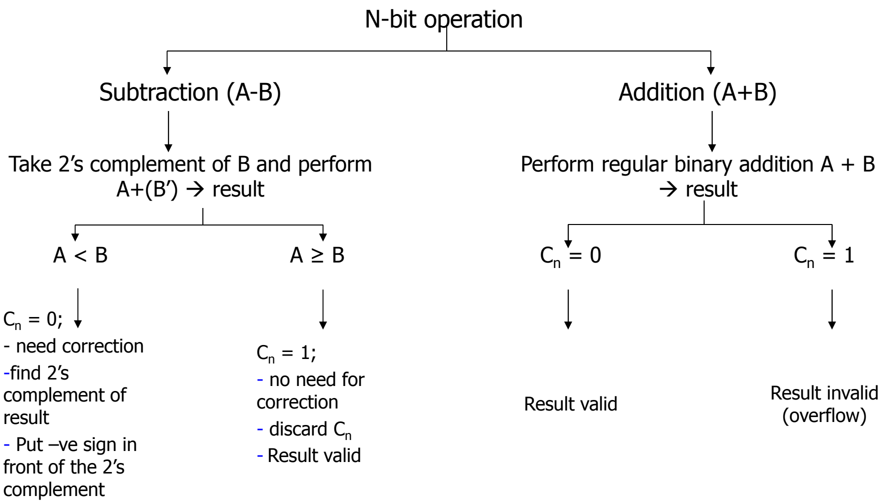
Sum = 1101111

C_n (last bit) is zero → need to perform correction

Y - X = -(2's complement of 1101111) = - 0010001

---

n-bit Unsigned BINARY 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 BINARY Numbers by Example (2)

- Consider

```
        01 1000          111 0000
   +6   00 0110      -6   11 1010
 +  13  00 1101     +13   00 1101
------  -----------  -----  ------------
  +19   01 0011      +7   00 0111

        00 1100          110  0100
   +6   00 0110      -6   11 1010
 -  13  11 0011     - 13  11 0011
------  -----------  -----  ------------
  - 7  11 1001      -19   101101
```

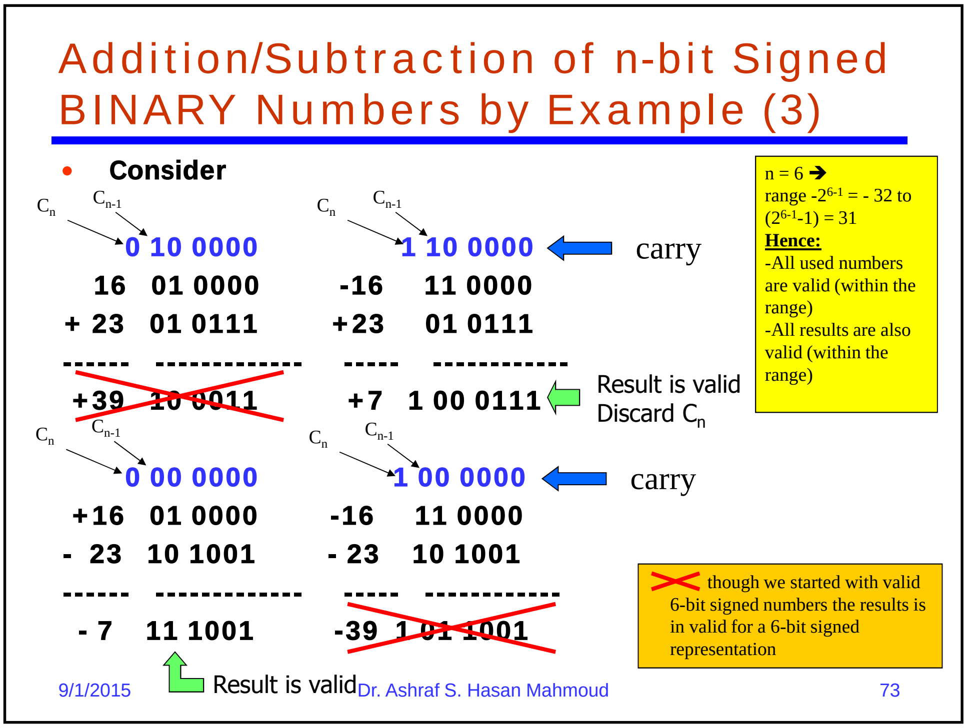
$C_n = 1$ ➔ discarded

$C_n = 1$ ➔ 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 BINARY Numbers by Example (3)

- **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~~

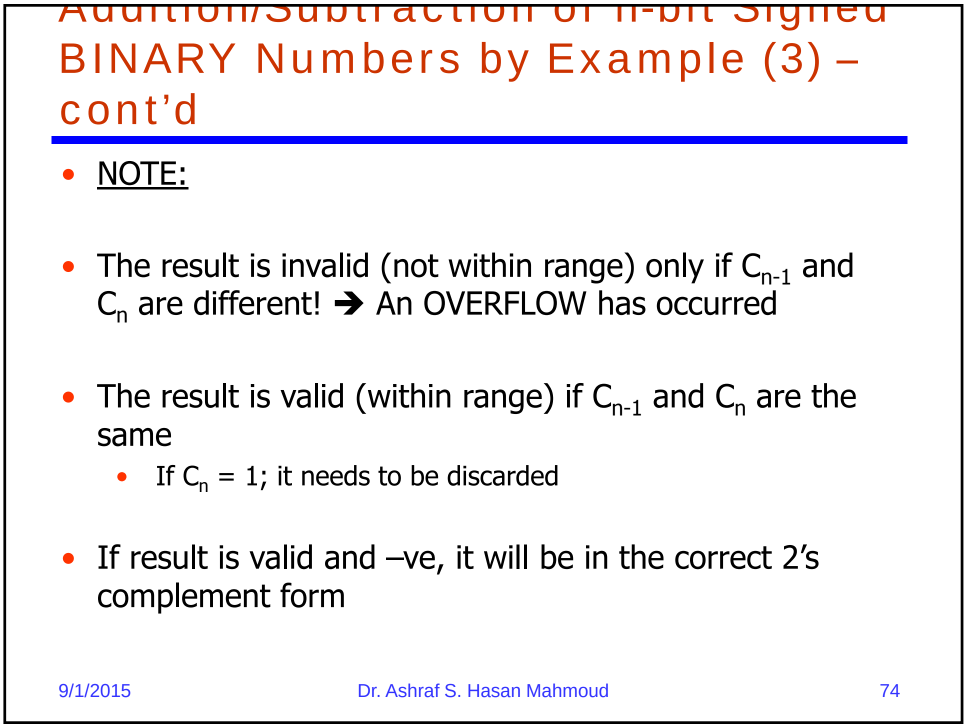
n = 6 ➔ range -2^{6-1} = - 32 to $(2^{6-1}-1)$ = 31

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

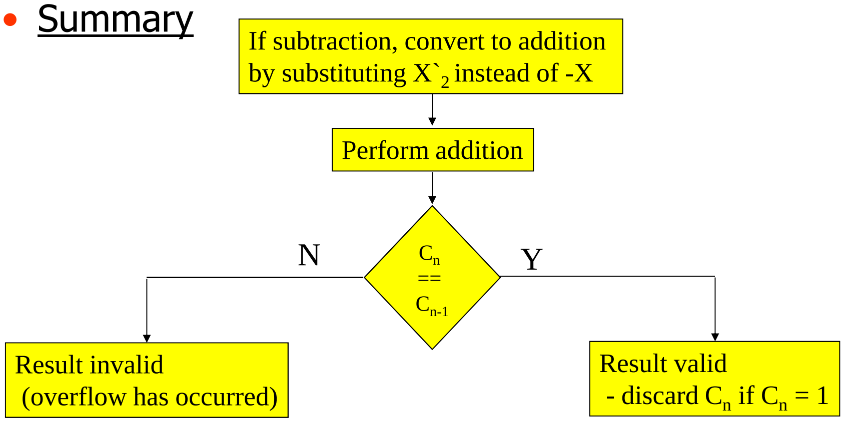


Addition/Subtraction of n-bit Signed BINARY Numbers by Example (3) – cont'd

- NOTE:
- 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 BINARY Numbers - Summary

- Summary



If subtraction, convert to addition by substituting $X`_2$ instead of -X

Perform addition

$C_n == C_{n-1}$

N

Y

Result invalid (overflow has occurred)

Result valid - discard C_n if $C_n = 1$

1.7 Binary Codes

- N-bit code → group of n bits → can give 2^n distinct combinations
  - Make every combination represent one element in the set of interest
  - Example – n = 2 ➔ 2^n = 4 distinct combinations: 00, 01, 10, 11
  - Example – n = 3 ➔ 2^3 = 8 distinct combinations: 000, 001, 010, 011, 100, 101, 110, 111
- Question: how many distinct combinations we can have from n decimal digits?

1.7 Binary Codes – cont'd

- Question 1: how many distinct combinations we can have from n decimal digits?
- Question 2: If I have m elements and I want to use m distinct codes, what is the minimum number of bits required?
  - Ans: we want 2^n >= m ➔ n = ceil{log2(m)}

Decimal Codes

- For us, humans, it is more natural to deal with decimal digits rather than binary digits
- m = 10 different decimal digits

  ➔ n = ceil (log2(m))
  = ceil (log2(10))
  = ceil(log2(3.3219))
  = 4

- Hence, we can use 4 bits to represent any digit ➔ BCD system
- Question: what is the maximum number of distinct codes given a 4-bit code?

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

This slide is borrowed from ?

BCD Arithmetic

1. BCD Unsigned Addition using 4-bit binary adders

What happens at a decimal digit?

→ If sum ≤ 1001: binary and BCD results are identical- No correction needed

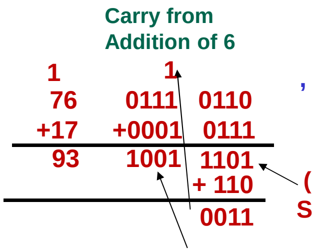
→ If sum > 1001 : (can be between 10 (1010) & 19 (10011)

should generate a BCD carry

and subtract $10)_d$ from result

Instead of subtracting $10)_d$, we add its 2's complement

2's complement of 1010 = 0110 = $6)_d$



Carry from Addition of 6

1	1	
76	0111	0110
+17	+0001	0111
93	1001	1101
		+ 110
		0011

(>9) Correction Needed: Subtract 10 by adding 6 and send a carry

(<= 9: No correction needed

in decimal in BCD

This slide is borrowed from ?

2. BCD Signed Subtraction through 10's Complement (Adding the 10s complement of subtrahend)

Signed BCD Numbers: We use One BCD Digit to represent the sign

Positive sign: 0 (=0000)   Negative Sign: 9 (=r-1) = 1001

Subtraction is done by addition of the 10's complement of the subtrahend

4-digit Signed-10's comp Subtraction

0395 - 0230 (using 10's complement in BCD)

The 10's complement of 0230 is (9)770.

10^4-N (here n = 4)

Subtraction Is converted To Addition

		Carries 1	1		
0	395	0000	0011	1001	0101
+9	770	+1001	0111	0111	0000
~~1~~0	165	1010	1011	1 0000	0101
		0110	0110	0110	
		~~1~~0000	0001	0110	0101

In Decimal

In BCD

= 5, No correction needed

= 16. Correction needed: Subtract 1010 by adding 0110

Logic and Computer Design Fundamentals
PowerPoint® Slides
© 2004 Pearson Education, Inc.

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 BCD 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 BCD 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 A_1 = A_1 -1 = 2)

              1 1111 → borrow
BCD for 2 =   0010
BCD for 3 =   0011
              --------
                1111 → a borrow occurred – need correction
            -   0110 (subtract 6)
              --------
               1001 → (BCD for 9 – Also Let A_2 = A_2 -1 = 1)

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

Subtraction in the BCD 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

---

Other Decimal Codes

- Weighted code – each bit position is given a weighting factor
- BCD and 2421 codes are examples of weighted codes
- Excess-3 is an unweighted code
- 8,4,-2,-1 code is an example of assigning both +ve and –ve weights

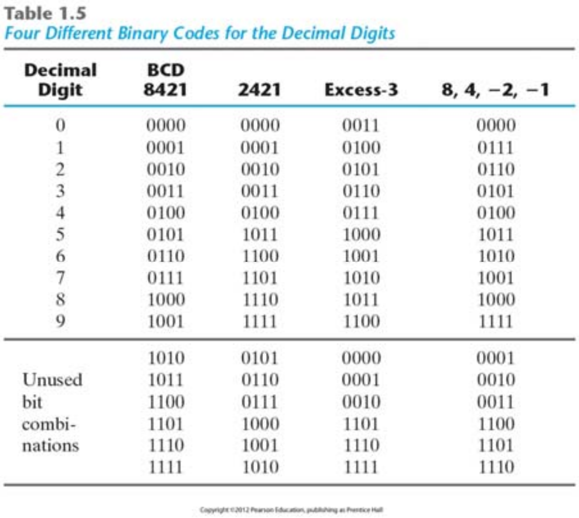


Table 1.5

Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combi- nations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Grey Code

- Note only one-bit change between NEIGHBORING code words
- Application: digital communication, representation of analog data by continuous change in angular position, etc.

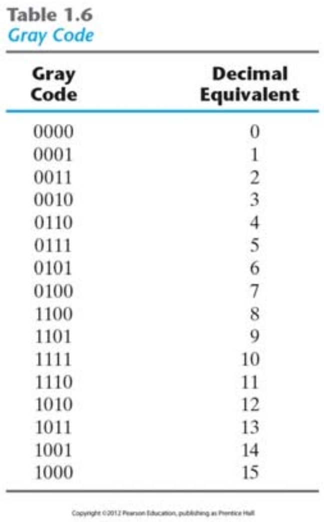


Table 1.6
Gray Code

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

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 Character Code

- 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 8th (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 Character Code

- 7-bits code ➔ 128 distinct combinations!
- Capital and small alphabetical characters differ by ONE bit (b6)
- Non printable characters
- One character is typically stored in ONE bytes – what is the use of the 8th bit?

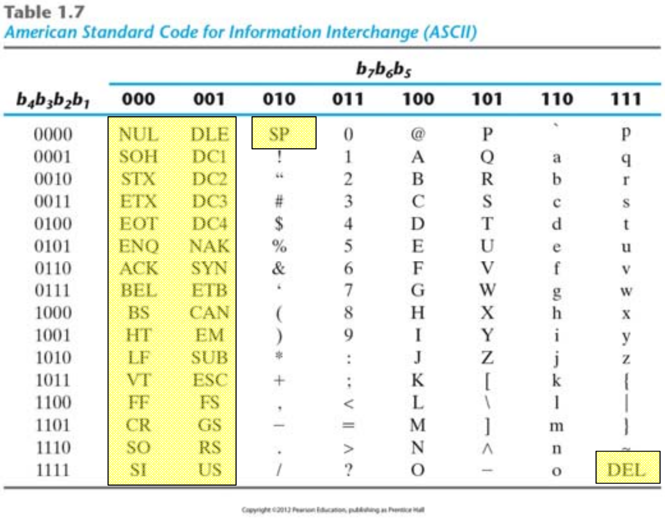
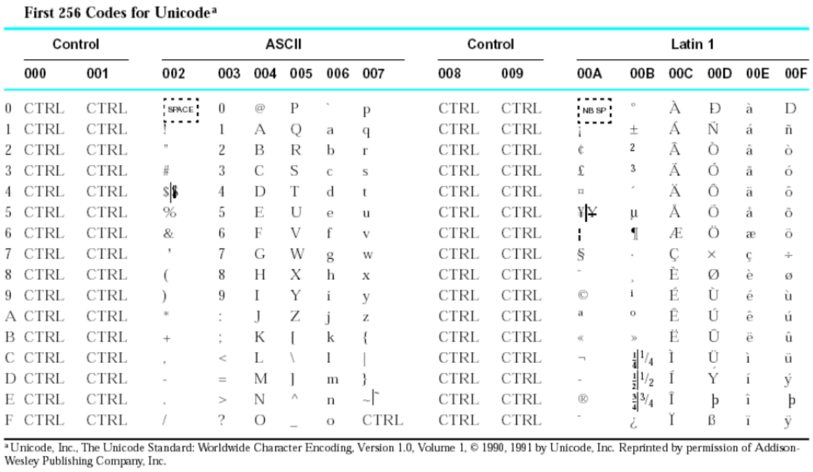


Table 1.7
American Standard Code for Information Interchange (ASCII)

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	–	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	–	o	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|1/4	Ì	Ü	ì	ü
D	CTRL	CTRL	-	=	M	]	m	}	CTRL	CTRL	-	½|1/2	Í	Ý	í	ý
E	CTRL	CTRL	.	>	N	^	n	~|~	CTRL	CTRL	®	¾|3/4	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.

Error-Detecting Code

- To detect errors in data communication and processing → add parity bit
  - 7-bit ASCII characters stored in 8-bit BYTES
- Even parity – the parity bit is added such that number of 1's is EVEN
- Odd parity – the parity bit is added such that number of 1's is ODD

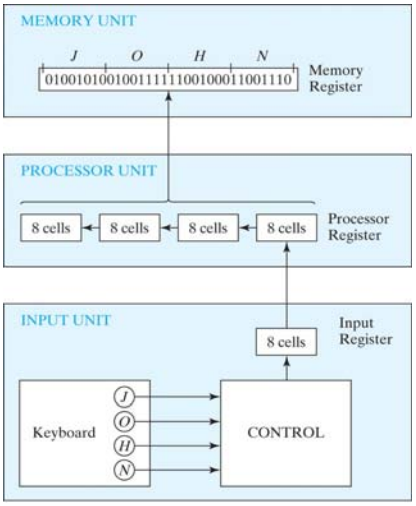
- Example:

	even parity	**odd parity**
ASCII A = 100 0001	**0**100 0001	**1**100 0001
ASCII B = 101 0100	**1**101 0100	**0**101 0100

- What types of errors can be detected with a single parity bit?
- What fraction of error can be detected with this system? Why?

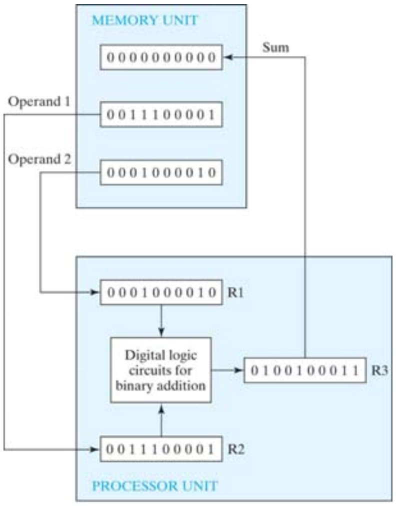
1.8 Binary Storage And Registers

- Register – group of binary cells
- n-bit register holds n bits – has 2^n possible states
- Register transfer – consists of a transfer of binary info from one set of registers to another
  - Refer to figure – ASCII characters produced by keyboard moved into processing unit register and copied to the memory register



1.8 Binary Storage And Registers – cont'd

- To process inputs in a digital systems – typically they are held in registers!
- See Figure – Digital logic circuit operates on R1 and R2 to produce R3
- Contents of R3 are transferred to the memory unit
- Where are the contents of R1 and R2 obtained from originally?



1.9 Binary Logic

- Deals with *binary* variables that take one of two discrete values
- Values of variables are called by a variety of very different names
  - *high* or *low* based on voltage representations in electronic circuits
  - *true* or *false* based on their usage to represent logic states
  - *one* (1) or *zero* (0) based on their values in Boolean algebra
  - *open* or *closed* based on its operation in *gate* logic
  - *on* or *off* based on its operation in *switching* logic
  - *asserted* or *de-asserted* based on its effect in digital systems

Basic Operations - AND

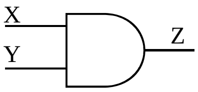
- Another Symbol is ".", e.g.

  Z = X AND Y or

  Z = X.Y or even

  Z = XY
- X and Y are inputs, Z is an output
- Z is equal to 1 if and only if X = 1 and Y = 1; Z = 0 otherwise (similar to the multiplication operation)
- Truth Table:
- Graphical symbol:

X	Y	Z=XY
0	0	0
0	1	0
1	0	0
1	1	1



Basic Operations - OR

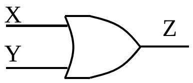
- Another Symbol is "+", e.g.

Z = X OR Y or

Z = X + Y

- X and Y are inputs, Z is an output
- Z is equal to 0 if and only if X = 0 and Y = 0; Z = 1 otherwise (similar to the addition operation)
- Truth Table:
- Graphical symbol:

X	Y	Z=X+Y
0	0	0
0	1	1
1	0	1
1	1	1



Basic Operations - NOT

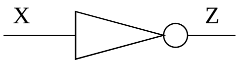
- Another Symbol is "¯", e.g.

Z = $\overline{X}$ or Z = X'

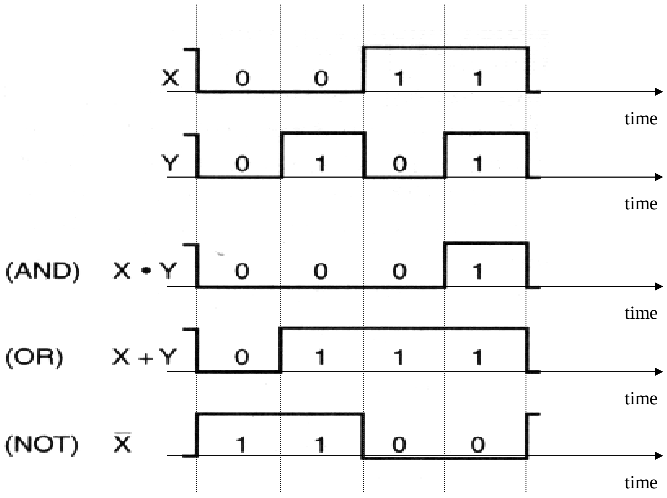
- X is the input, Z is an output
- Z is equal to 0 if X = 1; Z = 1 otherwise
- Sometimes referred to as the complement or invert operation
- Truth Table:

X	Z=X'
0	1
1	0

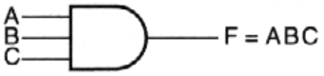
- Graphical symbol:



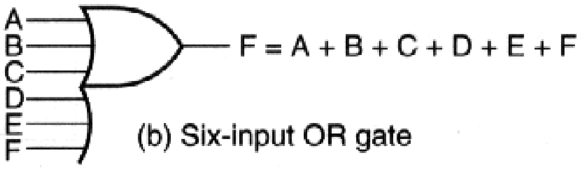
Time Diagrams



Multiple Input Gates



(a) Three-input AND gate



(b) Six-input OR gate

Textbook Examples: Pages 28 to 30

**Addition/Subtraction of UNSIGNED numbers using the COMPLEMENT system

Subtracting with Complements

- We want to perform: M – N
  - M and N are UNSIGNED NUMBERS
- **We WANT TO USE THE COMPLEMENT system to perform the subtraction**
- We can write: M – N = M + (-N) = M + N’
  - Change the subtraction to addition!
- Steps:
  - Add M to the r’s complement of N (i.e. N’)
  - If M >= N, the sum WILL produce a carry – ignore it.
  - If M < N, the sum does not produce a carry – the sum is the –ve of N – M (i.e. r^n – (N - M))

Example 1.5: Case for M >= N

- Using 10's complement subtract 72532 - 3250
- Solution: Note that r = 10, n = 5, M = 72532, N = 03250

M = 72532

10's complement of N = 96750 ← (100000 – 72532)

sum = **1**69282

discard the end carry 10^5 = -100000

ANSWER = 69282

Example 1.6: Case for M < N

- Using 10's complement subtract 3250 - 72532
- Solution: Note that r = 10, n = 5, M = 03250, N = 72532

M = 03259

10's complement of N = 27468 ←(100000 – 72532)

sum = 30718

There is NO end carry

➔ ANSWER = 30718

= – (10's complement of 30718)

= – 69282

Example 1.7: Using Binary Numbers

- Using the two binary numbers X = 1010100 and Y = 1000011, perform (a) X – Y and (b) Y – X using 2’s complement
- Solution: (a) X – Y:

X = 1010100

2’s complement of Y = 0111101 ← (10000000 – 1000011)

sum = 10010001

discard end carry 2^7 = -10000000

ANSWER = 0010001

r = 2, n = 7
- 2’s complement →
X = (1010100)2 = - 44
Y = (1000011)2 = - 61
→ X – Y = -44 – (-61) = 17
→ Y – X = -61 – (-44) = - 17

Example 1.7: Using Binary Numbers – cont’d

- Using the two binary numbers X = 1010100 and Y = 1000011, perform (a) X – Y and (b) Y – X using 2’s complement
- Solution: (a) Y – X:

Y = 0111101

2’s complement of X = 0101100 ← (10000000 – 1010100)

sum = 1101111

There is no end carry →

ANSWER is Y – X = - (2’s complement of 1101111)

= - 0010001

Example 1.8: Using Binary Numbers

- Using the two binary numbers X = 1010100 and Y = 1000011, perform (a) X – Y and (b) Y – X using 1's complement
- Solution: (a) X – Y:

X = 1010100

1's complement of Y = 0111100 ←
(1111111 - 1000011)

sum = 10010000

End-around carry = + 1

ANSWER = 0010001

r = 2, n = 7
- 1's complement →
X = (1010100)2 = - 43
Y = (1000011)2 = - 60
→ X – Y = -43 – (-60)
= 17
→ Y – X = -60 – (-43)
= - 17

Example 1.8: Using Binary Numbers –cont'd

- Using the two binary numbers X = 1010100 and Y = 1000011, perform (a) X – Y and (b) Y – X using 1's complement
- Solution: (b) Y – X:

Y = 1000011

1's complement of X = 0101011 ←
(1111111 - 1000011)

sum = 1101110

There is no end carry →

ANSWER is Y - X = - (1's complement of 1101110)

= - 0010001

More Examples

Subtraction of Unsigned Numbers – Example – Base 10

- **Example: X = $(72532)_{10}$, Y = $(3250)_{10}$ – Find X-Y and Y-X**
- **Solution:**

A) X – Y: X = 72532
10's complement of Y = 96750
Sum = 169282
Discard C_n (last bit) = $(69282)_{10}$ 🡐 X – Y

B) Y – X: Y = 3250
10's complement of X = 27468
Sum = 30718
C_n (last bit) is zero ➔ need to perform correction
Y - X = -(10's complement of 30718) = - 69282

The same procedure can be used for any base R system.

Example: Textbook page 37

- Example: Perform +375 + (-240) – take r = 10, n = 4
- Solution:

```
                                1 1 100  ← carry
                                  0 375
  10's complement of 240 =        9 760  ← (10000 – 240)
                     sum =      1 0 135
                  ANSWER =        0 135  ← Result is valid
```

Note end carry is discarded.