Chapter 1

BINARY SYSTEMS

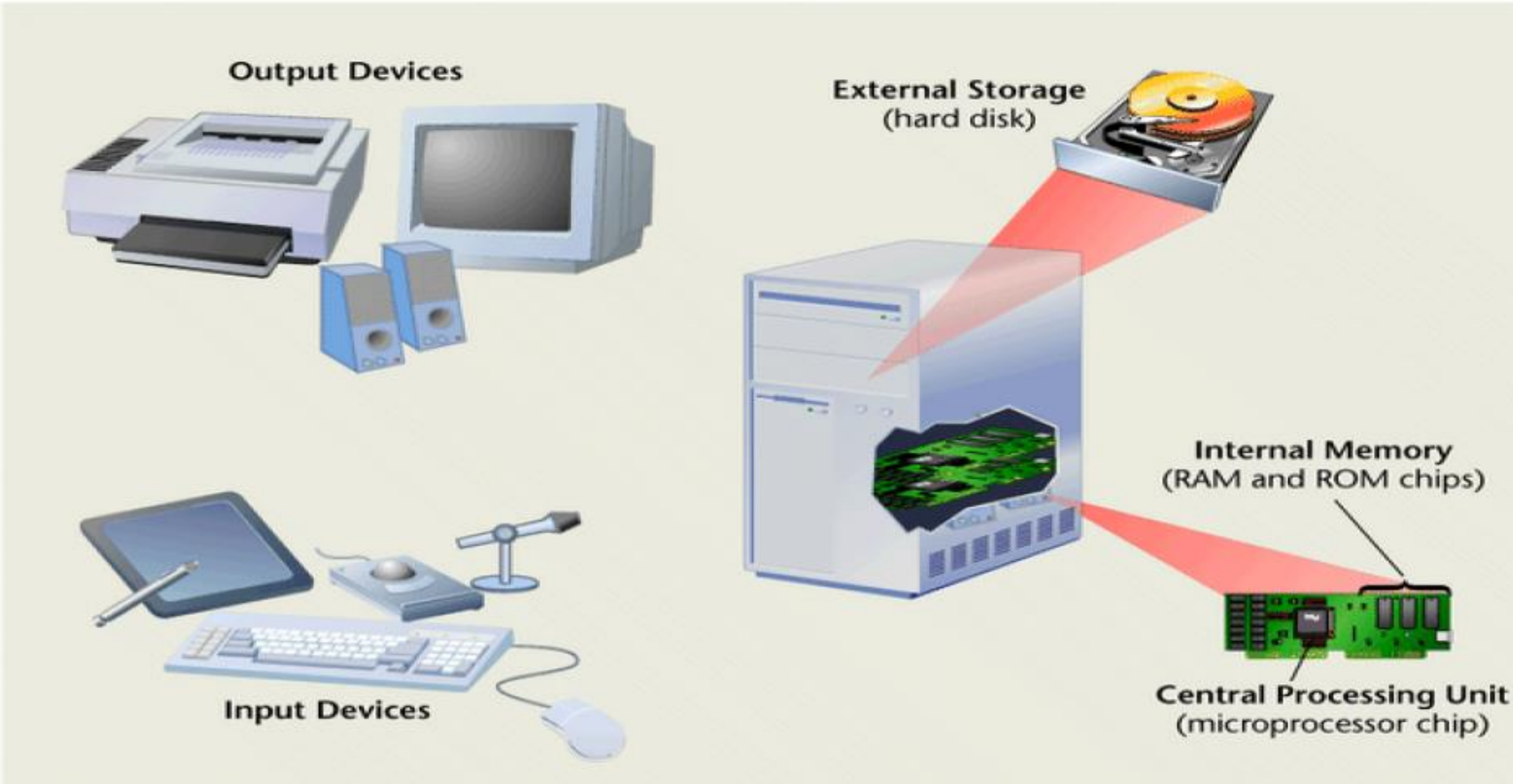
Topics

- Digital Systems
- Numbering systems
  - Base, representation, converting between numbering systems
- Arithmetic Operations in the Binary Numbering System
  - Addition, subtraction, multiplication, division, modulo
  - Defining 1's complement, 2's complement
  - Using the complement for subtraction
  - Signed numbers
- Binary Codes

“Digital Age”

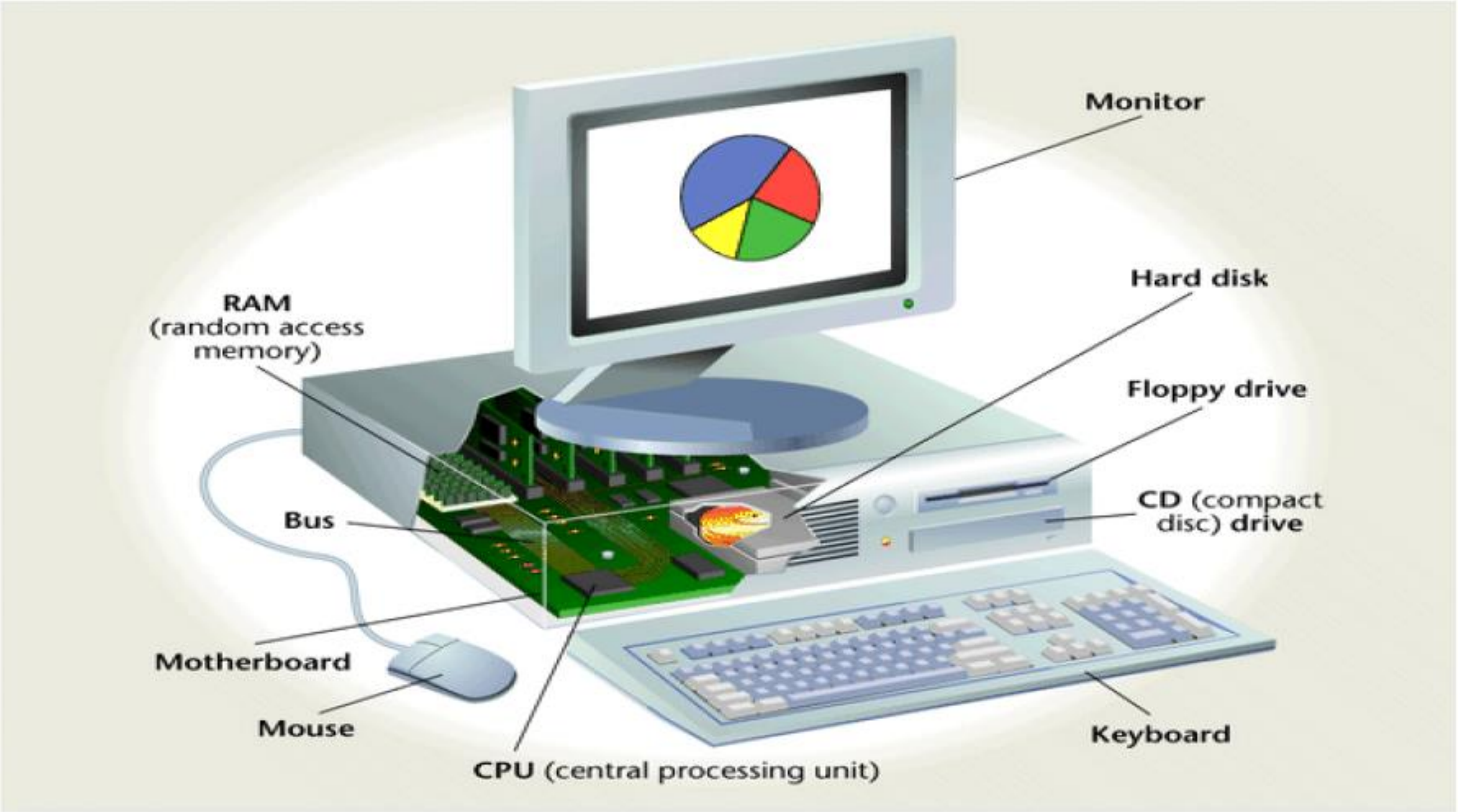

Gaming
Console
PDA
Home
Computer
Business
Mobile
Home Office
Phone
Office
Phone
Office
Computer
TV / PVR

The Central Tool of Modern Information Systems



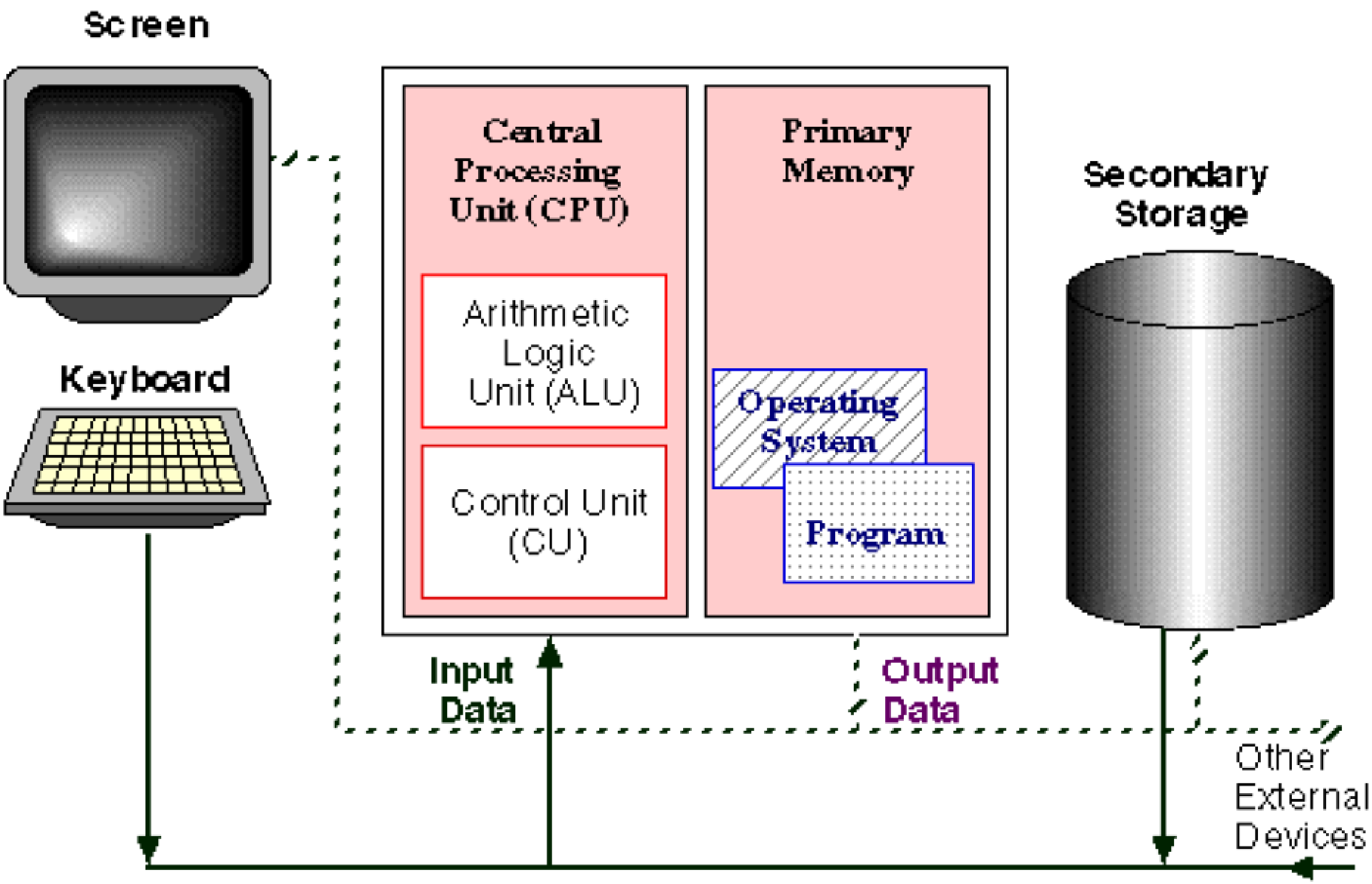
All computers have the same basic components.

Inside the computer



A look inside a computer

Block Diagram of a Digital Computer



Digital Systems

→ Early computers were designed to perform numeric computations

→ They used ***discrete*** elements of information named digits (finite sets)

→DIGITAL SYSTEMS: manipulate ***discrete*** elements of information

such as the 10 decimal digits or the 26 letters of the alphabet

we live in the "Digital Age"!

Binary System and Logic Circuits

- What kind of data do computers work with?
  - Deep down inside, it’s all 1s and 0s
- What can you do with 1s and 0s?
  - Boolean algebra operations
  - These operations map directly to hardware circuits (logic circuits)

Different Numbering Systems

- Decimal (Arabic): (0,1,2,3,4,5,6,7,8,9):
  Example: **$(452968)_{10}$**
- Octal: (0,1,2,3,4,5,6,7):
  Example **$(4073)_{8}$**
- Hexadecimal(0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)
  Example: $(2BF3)_{16}$
- Binary: (0,1):
  Example: $(1001110001011)_{2}$

Base in Numbering systems

- The decimal numbering system uses **base 10**. The values of the positions are calculated by taking 10 to some power.

1	6	2	.	3	7	5	Digits
100	10	1		1/10	1/100	1/1000	Weights

1	6	2	.	3	7	5	Digits
10^2	10^1	10^0		10^{-1}	10^{-2}	10^{-3}	Weights

- Base 10 for decimal numbers?

It uses 10 digits: The digits 0 through 9.

Base in Numbering systems [2]

- The binary numbering system is called binary because it uses **base 2**. The values of the positions are calculated by taking 2 to some power.
- Base 2 for binary numbers :

It uses 2 digits. The digits 0 and 1.

Representation of Numbers

→ ***There are two possible ways of writing a number in a given system:***

1- Positional Notation

2- Polynomial Representation

Positional Notation

$$N = (a_{n-1}a_{n-2} \dots a_1a_0 \, . \, a_{-1}a_{-2} \dots a_{-m})_r$$

Where

$.$ = radix point

r = radix or base

n = number of integer digits to the left of the radix point

m = number of fractional digits to the right of the radix point

a_{n-1} = most significant digit (MSD)

a_{-m} = least significant digit (LSD)

Positional Notation

The Decimal Numbering System

- The decimal numbering system is a positional number system.
- Example:

$(4\ 6\ 2\ 1)_{10}$ $\quad 1 \times 10^0$

(1000 100 10 1) $\quad 2 \times 10^1$

6×10^2

4×10^3

Positional Notation

Binary Numbering System

- The Binary Numbering System is also a positional numbering system.
- Instead of using ten digits, 0 - 9, the binary system uses only two digits, the 0 and the 1.

→Example of a binary number & the values of the positions.

1	0	1	0	1	0	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0

1	1	0	1	.	0	1	Binary digits, or bits
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	Weights (in base 10)

Polynomial Notation

$N = a_{n-1} \text{ x } r^{n-1} + a_{n-2} \text{ x } r^{n-2} + \ldots + a_0 \text{ x } r^0 + a_{-1} \text{ x } r^{-1} \ldots + a_{-m} \text{ x } r^{-m}$

$$= \sum_{i=-m}^{n-1} a_i r^i$$

Example:

Positional (N) ***Polynomial (N)***

$N = (651.45)_{10} = 6 \text{ x } 10^2 + 5 \text{ x } 10^1 + 1 \text{ x } 10^0$

$+ 4 \text{ x } 10^{-1} + 5 \text{ x } 10^{-2}$

Important number systems

→ There are three important number systems

- Binary Number System
- Octal Number System
- Hexadecimal Number System

Binary numbers

Digits = {0, 1}

Positional | Polynomial

$(11010.11)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$

– **1 K (kilo) = 2^{10} = 1,024**

– **1M (mega) = 2^{20} = 1,048,576**

– **1G (giga) = 2^{30} = 1,073,741,824**

Converting Decimal to Binary

$$N = (a_{n-1}a_{n-2} \ldots a_1a_0 \,.\, a_{-1}a_{-2} \ldots a_{-m})_r$$

← *Integer* → ← *Fractional* →

Radix point

→ **Integer** part and **Fractional** part are **converted differently**

Converting the Integer Part

- keep dividing by 2 until the quotient is 0. Collect the remainders *in reverse order*.
- Example: $(162)_{10}$.

162 / 2	= 81	rem 0
81 / 2	= 40	rem 1
40 / 2	= 20	rem 0
20 / 2	= 10	rem 0
10 / 2	= 5	rem 0
5 / 2	= 2	rem 1
2 / 2	= 1	rem 0
1 / 2	= 0	rem 1

- Then $(162)_{10} = (10100010)_2$

Converting the Fraction Part

→ keep multiplying the *fractional part* by 2 until it becomes 0. Collect the integer parts (in forward order).

– However this may not terminate!

–Example: $(0.375)_{10}$

$0.375 \times 2 = 0.750$
$0.750 \times 2 = 1.500$
$0.500 \times 2 = 1.000$

→ So, $(.375)_{10} = (.011)_2$

And $(162.375)_{10} = (10100010.011)_2$

Why does this work?

- This method can be applied to convert from decimal to *any* base
- Try converting 162.375 from decimal to decimal.

162 / 10 = 16 rem 2
16 / 10 = 1 rem 6
1 / 10 = 0 rem 1

- Each division "strips off" the rightmost digit (the remainder). The quotient represents the remaining digits in the number.

Why does this work? [2]

$$0.375 \times 10 = 3.750$$
$$0.750 \times 10 = 7.500$$
$$0.500 \times 10 = 5.000$$

- Each multiplication "strips off" the leftmost digit (the integer part). The fraction represents the remaining digits.

Converting binary to decimal

- To convert binary, or base 2, numbers to decimal we first obtain the polynomial representation of the number, then sum the products.
  - Example: $(1101.01)_2$

1	1	0	1	.	0	1	Binary digits, or bits
2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	Weights (in base 10)

$$(1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) =$$

$$8 + 4 + 0 + 1 + 0 + 0.25$$

→ The decimal value is: $(13.25)_{10}$

Octal number system

–Digits = {0, 1, 2, 3, 4, 5, 6, 7}

Positional = Polynomial

$(127.4)_8 = 1 \text{ x } 8^2 + 2 \text{ x } 8^1 + 7 \text{ x } 8^0 + 4 \text{ x } 8^{-1}$

- **Octal (base 8) digits range from 0 to 7.**

Since $8 = 2^3$, one octal digit is equivalent to 3 binary digits.

Converting Decimal to Octal

→ Integer part: keep dividing by 8 until the quotient is 0. Collect the remainders *in reverse order*.

→ Fractional Part: keep multiplying the *fractional part* by 8 until it becomes 0. Collect the integer parts (in forward order).

Same method as for the decimal to binary

Converting Octal to Decimal

• To convert Octal, or base 8, numbers to decimal we first obtain the polynomial representation of the number, then sum the products.

Example

$$(127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (87.5)_{10}$$

Converting Binary to Octal

- To convert from binary to octal, make groups of 3 bits, starting from the binary point. Add 0s to the ends of the number if needed. Then convert each group of bits to its corresponding octal digit.

Example

$$(10110100.001011)_2 = (010\ 110\ 100\ .\ 001\ 011)_2$$

$$= (2\ 6\ 4\ .\ 1\ 3)_8$$

Converting Octal to Binary

- To convert from octal to binary, replace each Octal digit with its equivalent 3-bit binary sequence.

- Example

$$
\begin{aligned}
(261.35)_8 &= (2 \quad 6 \quad 1 \quad . \quad 3 \quad 5\,)_8 \\
&= (010 \quad 110 \quad 001 \quad . \quad 011 \quad 101)_2
\end{aligned}
$$

Hexadecimal numbers

-Digits = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Positional Polynomial

- $(B65F)_{16} = 11 \times 16^{3} + 6 \times 16^{2} + 5 \times 16^{1} + 15 \times 16^{0}$

• *Hexadecimal* (base 16) digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. Since $16 = 2^4$, one hexa digit is equivalent to 4 binary digits.

– It's often easier to work with a number like B5 instead of 10110101.

Converting Decimal to Hexadecimal

→ **Integer part**: keep dividing by 16 until the quotient is 0. Collect the remainders *in reverse order*.

→ **Fractional Part**: keep multiplying the *fractional part* by 16 until it becomes 0. Collect the integer parts (in forward order).

Same method as for the decimal to binary conversion

Converting Hexadecimal to Decimal

- To convert Hexadecimal, or base 16, numbers to decimal, first obtain the polynomial representation of the number, then sum the products.

Example

$$\mathbf{(B65F)_{16} = 11 \times 16^{3} + 6 \times 16^{2} + 5 \times 16^{1} + 15 \times 16^{0}}$$

$$\mathbf{= (46{,}687)_{10}}$$

Converting Hexadecimal to Binary

- To convert from hexadecimal to binary, replace each hex digit with its equivalent 4-bit binary sequence.
- Example

$$261.35_{16} = (2 \quad 6 \quad 1 \quad . \quad 3 \quad 5)_{16}$$
$$= (0010 \quad 0110 \quad 0001 \quad . \quad 0011 \quad 0101)_2$$

Converting Binary to Hexadecimal

- To convert from binary to hex, make groups of 4 bits, starting from the binary point. Add 0s to the ends of the number if needed. Then convert each group of bits to its corresponding hex digit.
- Example

$$(10110100.001011)_2 = (1011\ 0100.\ 0010\ 1100)_2$$
$$= (B\ 4\ .\ 2\ C)_{16}$$

Decimal	Binary	Octal	Hex
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	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

Summary

- Converting from numerical system with base ***r*** to decimal

$$N = a^{n-1} \times r^{n-1} + a^{n-2} \times r^{n-2} + \ldots + a^{0} \times r^{0} + a^{-1} \times r^{-1} + \ldots + a^{-m} \times r^{-m}$$

- Converting from decimal to number of base ***r*:**

$N_{10} = I_1.F_1$

Integer Part	Fractional Part
$I_1 / r = Q_1$ rem r_1	$F_1 * r = D_1 . f_1$
$Q_1 / r = Q_2$ rem r_2	$f_1 * r = D_2 . f_2$
…	…
$Q_{n-2} / r = Q_{n-1}$ rem r_{n-1}	$f_{n-2} * r = D_{n-1} . f_{n-1}$
$Q_{n-1} / r = 0$ rem r_n	$f_{n-1} * r = D_n . 0$
$I_r = (r_n r_{n-1} \ldots r_2 r_1)_r$	$F_r = (D_1 D_2 \ldots D_{n-1} D_n)_r$

- Number base r is $N_r = I_r.F_r$

Summary (continued)

- Converting from Binary to Octal
  - Divide the bits into groups of 3 – translate each 3-bits into an octal digit (see slide 31)
- Converting from Octal to binary
  - Convert each octal digit to its 3-bit value (see slide 31)
- Converting from Binary to Hexadecimal
  - Divide the bits into groups of 4 – translate each 4-bits into a hexadecimal digit (see slide 31)
- Converting from Hexadecimal to binary
  - Convert each hexadecimal digit to its 4-bit value (see slide 31)

ARITHMETIC OPERATIONS IN A BINARY SYSTEM

Binary Addition

```
         0        0              1            1
        +0      +  1            +0           +1
         --        --             --           --
sum      0        1               1           10
```

(sum of 0 and carryover of 1)

Apply carry-over when adding multiple bit numbers

Examples:

```
  1 0 0 1      0 0 0 1        1 1 0 0
+ 0 1 1 0    + 1 0 0 1      + 0 1 0 1
  -------      -------      ---------
  1 1 1 1      1 0 1 0      1 0 0 0 1
```

carryover

Binary Addition-Examples

		Check your work
Carry	1 1 1 1 0 1	
	1 0 1 1 0 1	$(45)_{10}$
	+ 0 1 1 1 0 1	+ $(29)_{10}$
Sum	1 0 0 1 0 1 0	= 74

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 32 + 0 + 8 + 4 + 1 = 45$$

Binary Addition- Examples

Addition of three Binary Digits

x	y	CarryIn	Sum	CarryOut
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Addition of Large Binary Numbers

- Example showing larger numbers:

```
     1010  0011  1011  0001
  +  0111  0100  0001  1001
  -------------------------
  1  0001  0111  1100  1010
```

Binary Subtraction

	0	^{1}0	1	1
	- 0	- 1	- 0	- 1
	--	--	--	--
difference	0	1	1	0

(Requires a borrow, note that the value 10 has a decimal value of 2)

$$
\begin{array}{r}
{\scriptstyle 0\;\;1\;0\;1\;0\;0} \\
\cancel{1}\,0\,\cancel{1}\,0\,1\,1 \\
{\scriptstyle 0\;\;\;\;\;0\;\;\;\;\;\;\;\;\;\;} \\
-\;0\,1\,0\,1\,0\,1 \\
\hline
0\,1\,0\,1\,1\,0
\end{array}
$$

Decimal Examples

$$
\begin{array}{r}
{\scriptstyle 2} \\
\cancel{3}86 \\
-\;94 \\
\hline
2
\end{array}
\;\Rightarrow\;
\begin{array}{r}
2\;18\;6 \\
-\;\;9\;4 \\
\hline
2\;\;9\;2
\end{array}
$$

$$
\begin{array}{r}
1{,}002 \\
-\;398 \\
\hline
\end{array}
\;\Rightarrow\;
\begin{array}{r}
{\scriptstyle 0\;\;\;\;9\;\;\;\;9\;\;\;\;\;} \\
\cancel{1}\;\cancel{10}\;\cancel{10}\;12 \\
-\;3\;\;\;9\;\;\;8 \\
\hline
6\;\;\;0\;\;\;4
\end{array}
$$

Binary Multiplication

	0	0	1	1
	x 0	x 1	x 0	x 1
	--	--	--	--
Product	0	0	0	1

```
  A      0 010000.010
x B      0 001000.010
-------------------------
              0000000000
             0010000010
            0000000000
           0000000000
          0000000000
         0000000000
        0010000010
```

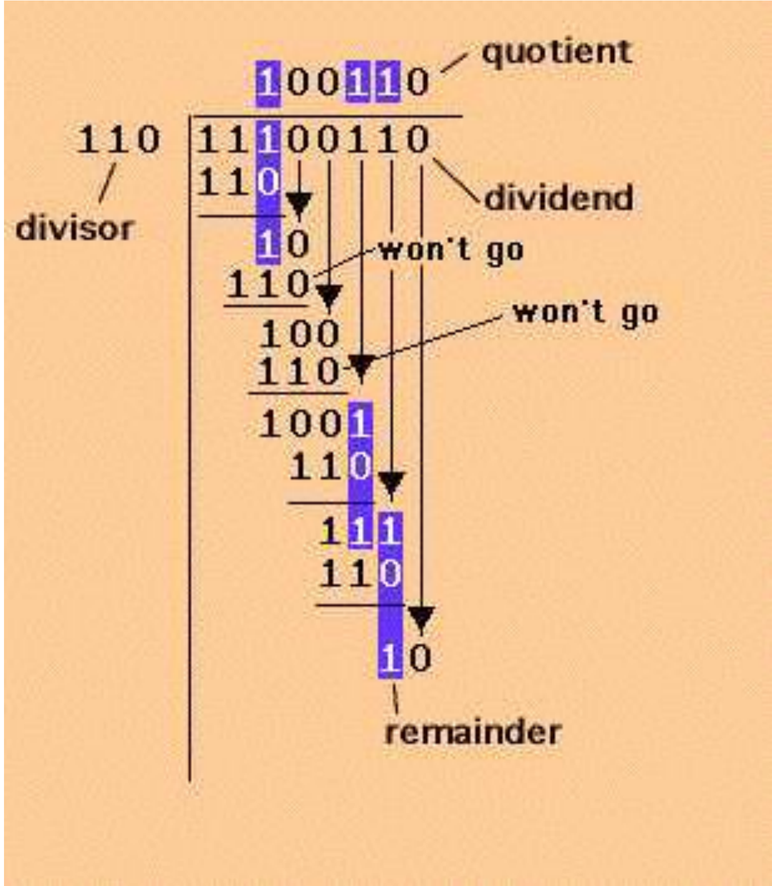
Decimal Example

423 x 211

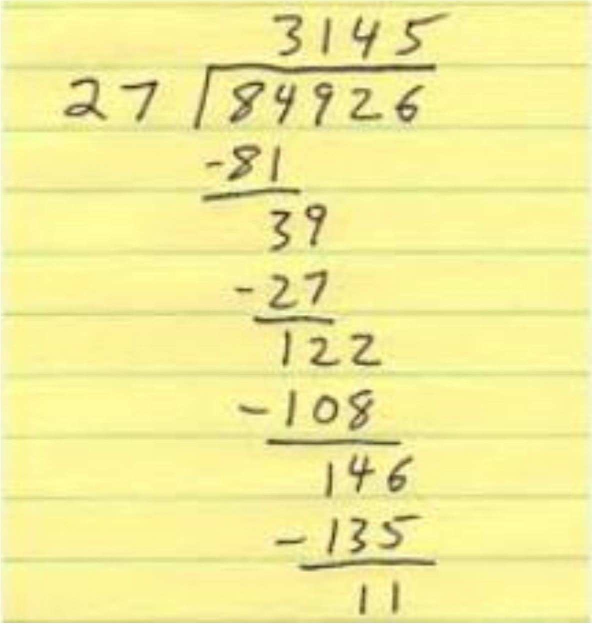
```
    423
  x 211
  -----
    423
   423
+ 846
  -----
  89253
```


1000110.000100

Binary Division



Decimal Example



Complements in Numbering Systems

- Complements are used in digital systems (computers) for simplifying the Subtraction operation and for logical manipulation

- There are two types of complements for each ***base* ‘*r*’ *system*:**

1- *Radix complement → r’s complement*

Ex. ***base 10-→ 10’s complement***

base 2 → 2’s complement

2- *Diminished radix complement → (r-1) complement*

Ex. ***base 10 → 9’s complement***

Base 2 → 1’s complement

Radix complement (r's complement)

$$[N]_r = r^n - (N)_r$$

where n is the number of digits in $(N)_r$.

Example

- **What 2's complement of $(N)_2 = (101001)_2$**

$$[N]_2 = 2^6 - (101001)_2 = (1000000)_2 - (101001)_2 = (010111)_2$$

- **10's complement of $(N)_{10} = (72092)_{10}$**

$$[N]_{10} = (100000)_{10} - (72092)_{10} = (27908)_{10}.$$

Obtaining 2's complement

- **Can be obtained directly from the given number by**
  1-copying each bit of the number starting at the least significant bit and proceeding forward the most significant bit until the first 1 has been copied.

  2- After the first 1 has been copied replace each of the remaining 0s and 1s by 1s and 0s respectively

(a) $[1010100]_2 = 2^7 - (1010100)_2 = (10000000)_2 - (1010100)_2 = (0101100)_2$

(b) $[101001]_2 = 2^6 - (101001)_2 = (1000000)_2 - (101001)_2 = (010111)_2$

	(a)	(b)
	1 0 1 0 1 0 0	1 0 1 0 0 1
2 's →	**0 1 0 1 1 0 0**	**0 1 0 1 1 1**

Diminished radix complement (*r-1*'s complement)

$$[N]_{r-1} = (\boldsymbol{r^n} - 1) - (N)_r$$

-9's complement of $[546700]_9$

= 999999 – 546700 = 453299

-1's complement of [1011000]

= $(10000000 - 1)_2 - (1011000)_2 = (0100111)_2$

Obtaining 1's complement

- ***1 's complement can be obtained directly from the given number by replacing each of 0s and 1s by 1s and 0s of the number (i.e. complement each bit)***

$$\mathbf{[1011000]} = (10000000 - 1)_2 - (1011000)_2$$

$$= \mathbf{(0100111)_2}$$

1 0 1 1 0 0 0

1 's complement → **0 1 0 0 1 1 1**

Summary of Complements

$[N]_r = r^n - (N)_r$

- 10's Complement
  - Subtract from 10^n
  - Take 9's complement and add 1
- 2's Complement
  - Subtract from 2^n
  - Start from LSB, copy bit until 1 is copied, then change bit up to MSB
  - Take 1's complement and add 1

$[N]_{r-1} = (r^n - 1)_r - (N)_r$

- 9's Complement
  - Subtract from 10^n-1
  - Subtract each digit from 9
- 1's Complement
  - Subtract from 2^n-1
  - Flip each bit

Subtraction with 2's Complement

- **2's complement are used to convert subtraction to addition, which reduces hardware requirements (only adders are needed).**

$$A - B = A + (-B)$$

(add *2*'s complement of *B* to *A*)

- 2's Complement has the properties of the minus sign

A + (-A) = 0 **A + 2's{A} = 0**

- (- A) = A **2's{ 2's{A}} = A**

A - B = A + (-B) **A - B = A + 2's{B}**

Subtraction with 2's Complement [2]

Examples:

A= 1010100 B= 1000011

- 2's complement

A - B = A + (-B)= A+ [B]

= (1010100) + (0111101) = (0010001)

End carry = 1, means that B is less than A, and result is a positive number

$$
\begin{array}{rr}
 & 1010100 \\
+ & 0111101 \\
\hline
\cancel{1} & 0010001
\end{array}
$$

End carry = 0, means that B is larger than A, and result is 2's complement number (a negative value), try B-A in our example.

Subtraction with 1's Complement

Examples: note: same proprieties minus sign as 2's complement.

A= 1010100 B= 1000011

- 1's complement

A - B = A+ [B]= (1010100) + (0111100) = (0010001)

$$
\begin{array}{rr}
 & 1010100 \\
+ & 0111100 \\
\hline
1 & 0010000 \\
 & +\ \ 1 \\
\hline
 & 0010001
\end{array}
$$

Add end carry

Subtraction with 10s/9'sComplements

$(72)_{10} - (32)_{10} = (40)_{10}$

10's Complement

$[32] = 10^2 - (32)_{10} = (68)_{10}$

$(72)_{10} + (68)_{10} = \cancel{1}\ (40)_{10}$

9's Complement

$[32] = (10^2 - 1) - (32)_{10} = (67)_{10}$

$(72)_{10} + (67)_{10} = (1 + 39)_{10} = (40)_{10}$

Signed Binary numbers

• Recall that digital Systems are made with devices that take on exactly two states : 0 and 1.

•**The only states are "1" and "0". There is no "-" state!**

→because of hardware limitations computers represent negative numbers by using the leftmost bit for the sign bit.

-- "0" indicates a positive number,

-- while a "1" indicates a negative number

Signed Magnitude

•The leftmost bit indicates the sign of the number. The remaining bits give the magnitude of the number

→Using 8 bits to represent binary number the value in the example is:

sign ← **magnitude** →

1	0	0	0	0	0	1	1

-3 = 10000011 = 1/ (sign bit) 0000011

→ Sign Magnitude representation is good for having the ability for a human to read and understand what number is represented

Signed Complement

(a) Signed Magnitude representation

1	0	0	0	0	0	1	1

-3 = 10000011 = 1/ (sign bit) 0000011

(b) Signed 1's representation

1	1	1	1	1	1	0	0

-3 = 10000011 = 1/ (sign bit) 1111100

(c) Signed 2's representation

1	1	1	1	1	1	0	1

-3 = 10000011 = 1/ (sign bit) 1111101

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	—	—

Fixed-Length Registers

- All practical digital devices have fixed-length registers
- This means that numbers in a computer are represented by a fixed number of bits
  - The earliest microprocessors were 4-bit devices
  - Intel 8080 and the 6502 (Apple II) chips were 8-bit
  - Intel 8088 (IBM PC) and Motorola 68000 (Mac) are 16-bit devices
  - Pentium chips and PowerPC chips are 32-bit

Range of a number Overflow during addition

- A fixed-length register can only hold a ***Range*** of numbers

→For a 4-bit device, the *range* of positive integers is 0 - 15

→For an 8-bit device the *range* of positive integers is 0 – 255

→ When adding positive integers, ***Overflow*** occurs when the sum falls outside the range of the register

Carry bit out of MSB = 1

Overflow in Signed Complements

•when numbers are treated as signed complement, a "carry" of 1 from the addition of the most significant bits **DOES NOT** indicate an overflow,

3 00011

+ (-3)+11101

= 00000, with a carry of "1" (2'complement) : We know that addion operation in 2's complement the end-carry is discarded !

•For signed complement, overflow occurs when:

→***The addition of two positive numbers results in a negative number***

OR → ***The addition of two negative numbers results in a positive number***

Overflow Examples

•In a 6-bit register with signed 2's complement

+ 17 = 010001

+ 16 = +010000

=100001

100001 = - (11111) = -(31)$_{10}$ instead of **+ (33)$_{10}$**

•Same with a 7-bit register

+ 17 = 0 010001

+ 16 = +0 010000

= 0100001

0100001 = + 33 No Overflow

Binary codes: *BCD* (1)

- To represent information as strings of alpha-numeric characters.
- ***Binary Coded Decimal (BCD)***
  - Used to represent the decimal digits 0 - 9.
  - 4 bits are used.
  - Each bit position has a weight associated with it (***weighted code***).
  - Weights are: 8, 4, 2, and 1 from MSB to LSB (called 8-4-2-1 code).

Binary codes: *BCD* (2)

– BCD Codes:

0→ 0000	1→ 0001	2→ 0010
3→ 0011	4→0100	5→ 0101
6→ 0110	7→0111	8→ 1000
9→ 1001		

– Used to encode numbers for output to numerical displays

–***Example***: $(9750)_{10} = (1001011101010000)_{BCD}$

Binary codes: *ASCII* [2]

- ***ASCII*** (American Standard Code for Information Interchange) (see table 1.7 of textbook)
  - Most widely used character code.
  - ***Example***: ASCII code representation of the word ***'Digital'***

Character	Binary Code	Hexadecimal Code
D	1000100	44
i	1101001	69
g	1100111	67
i	1101001	69
t	1110100	74
a	1100001	61
l	1101100	6C

Practice Problems Solved in the Class

Examples: Signed Complements 2's

	Sign-bit	
$(9)_{10}$	0	1001
$+(6)_{10}$	0	0110
	0	1111
$(9)_{10}$	0	1001
$-(6)_{10}$	1	1010
	0	0011
$(6)_{10}$	0	0110
$-(9)_{10}$	1	0111
	1	1101

Examples: Signed Complements 1's

	Sign-bit	
$(9)_{10}$	0	1001
$+(6)_{10}$	0	0110
	0	1111

$(9)_{10}$	0	1001
$-(6)_{10}$	1	1001
1	0	0010 = (0010) + (0001) = (0011)

$(6)_{10}$	0	0110
$-(9)_{10}$	1	0110
	1	1 1 0 0

Problems

Question

(a) Convert le following binary number into (i) Octal, (ii) Decimal, (iii) hexadecimal

10101101.10110

(b) Convert A= 16.25 and B = 8.25 into binary, use 7 bits to represent the integer part and 3 bits to represent the fractional part, then perform the following operations

I) C= A + B

ii) D = A – B

Note: Compute C and D

(a) using non-signed binary numbers and without complements

(b) using signed 2's complement

(c) Convert le following number into (i)Decimal, (ii) Octal, (iii) binary

$$(\mathbf{FD8.C2B})_{16}$$

Problems

10101101.10110

Answers:

i) Octal→ (010 101 101.101 100)

(2 5 5 . 5 4)$_8$

iii) Hexa → (1010 1101.1011 0000)2

(A D . B 0)16

ii) Decimal

$$(10101101.10110)_2 = 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2}\ 1 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5}$$

$$= (173.6875)_{10}$$

Problems

Integer part

- Conversion de $(16)_{10}$.

16 / 2	= 8	remainder 0
8 / 2	= 4	remainder 0
4 / 2	= 2	remainder 0
2 / 2	= 1	remainder 0
1 / 2	= 0	remainder 1

- Then $(16)_{10} = (10000)_2$

Fractional part

– Converting: $(0.25)_{10}$

$0.25 \times 2 = 0.50$
$0.50 \times 2 = 1.00$

• then, $(.25)_{10} = (.01)_2$

and $(16.25)_{10} = (10000.01)_2$

Same as for A

→ B = 8.25: $(8.25)_{10} = (1000.01)_2$

→ Representation using 7 bits and 3 bits

$$A = (16.25)_{10} = (0010000.010)_2$$

$$B = (8.25)_{10} = (0001000.010)_2$$

Problems

Non Signed Binary

A	0010000.010
+ B	0001000.010
	0 011000.100

A	0010000.010
- B	0001000.010
	0001000.000

Problems

Signed 2' complement

A	0 010000.010
+ B	0 001000.010
	0 011000.100

A	0 010000.010
+ (-B)	1 110111.110
	0 001000.000

Problems

A	0 010000.010
x B	0 001000.010

```
            0000000000
           0010000010
          0000000000
         0000000000
        0000000000
       0000000000
      0010000010
```

1000110.000100

Problems

A÷ B

Dividend	Divider
10000.010	1000 .010
1000010	1.1.. Quotient
01000 0000	
1000010	

Question 1:

Convert $A = (00010010.0101)_{BCD}$ and $B= (2.25)_{10}$ into pure binary format employing 8 bits for the integer part and 3 for the fractional part, including the sign bit. Perform the following operations in specific signed complement as indicated for each operation.

(i) $C = -A - B$ using signed 2's complement
(ii) $D = -A + B$ using signed 1's complement
(iii) $E = A - B$ using 9's complement (show all intermediate steps)

Problems

$A = (12.50)_{10}$ $B = (2.25)_{10}$ $(12.50)_{10} - (2.25)_{10}$ using 9's complement

9's complement of (2.25)
First step use the same number of position to represent the two numbers (02.25) then obtain the 9's complement

$[02.25]_9 = (10^4 - 1)_{10} - (02.25)_{10} = (99.99 - 02.25)_{10} =$
$(97.74)_{10}$

$E = A - B = A + [B]_9 =$

$$
\begin{array}{lr}
 & 12.50 \\
+ & 97.74 \\
\hline
\text{(end carry) } 1 & 10.24 \\
 & +1 \\
\hline
 & 10.25
\end{array}
$$