0.1 BINARY NUMERATION SYSTEM

Jean-Pierre Deschamps

University Rovira i Virgili, Tarragona, Spain

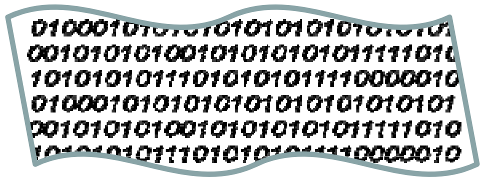
Universitat Autònoma de Barcelona

1. Computer information representation

A computer receives, stores, processes, transmits **data**.

Data types: numbers, characters, sounds (audio), pictures (video), etc.

Data encoding: strings of **zeroes and ones**.



Computer technology is based on electronic circuits able to process vectors of 0's and 1's (the so-called **digital electronic circuits**). For that reason all data are encoded by strings of 0's and 1's.

This type of information encoding is called **binary encoding system.**

2. Numeration systems

Most used systems:

- **Decimal system**
- **Binary system**
- **Hexadecimal system**

Conversion methods will be presented..

2.1 Decimal system

UAB
Universitat Autònoma de Barcelona

- Uses ten **digits**:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

- **Positional** system **:** a weight is associated to every digit position so that position is relevant.

Example: 653

	6	**5**	**3**
(weights)	10^2	10^1	10^0

$$653 = 6 \cdot 10^2 + 5 \cdot 10^1 + 3 \cdot 10^0$$

6 hundreds, 5 tens, 3 units

2.2 Binary system

- Uses two digits (binary digits, **bits**): **0, 1**
- **Positional** system.

Example:

	1	**1**	**0**	**1**
(weights)	2^3	2^2	2^1	2^0

- To compute the decimal representation, add up the weights corresponding to the 1's of the binary representation:

$$(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 0 + 1 = 13$$

UAB
Universitat Autònoma
de Barcelona

Exercise

Compute the decimal representation of the following binary number: $(101001)_2$

Exercise (solution)

Compute the decimal representation of the following binary number: $(101001)_2$

	1	0	1	0	0	1
weights	2^5	2^4	2^3	2^2	2^1	2^0

$$(101001)_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 32 + 8 + 1 = 41$$

2.2 Binary system: representation range

- Pure binary system: **non-negative number** representation.
- With *n* bits: **2^n** distinct values.
- Representation range: **0 to 2^n - 1**.

EXAMPLE:

n = 4 bits

16 different combinations

from 0 to 15 = 2^4-1

Binary	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binary	Decimal
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

2.2 Binary system: representation range

- ✓ If n = 3 : 2^3 = 8 representable numbers, from 0 to 7.
- ✓ If n = 4 : 2^4 = 16 representable , from 0 to 15.
- ✓ If n = 5 : 2^5 = 32 representable , from 0 to 31.
- ✓ If n = 6 : 2^6 = 64 representable , from 0 to 63 …

Example: how many bits do we need to represent 48?

$31 \leq 48 \leq 63$

~~n=5~~ n=6

=> to represent decimal number 48_{10} we need 6 bits: 110000

0.1

UAB Universitat Autònoma

2.3 Hexadecimal system

- Uses sixteen dígits:

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

- **Positional** system

  3 A 9 F

  (weights) 16^3 16^2 16^1 16^0

- To compute the decimal representation, add up the digits multiplied by the corresponding weights:

$$(3A9F)_{16} = 3 \cdot 16^3 + 10 \cdot 16^2 + 9 \cdot 16^1 + 15 \cdot 16^0 = 15007_{10}$$

BINARY				HEXA
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

UAB Universitat Autònoma de Barcelona

3. Base conversion (HEXADECIMAL to BINARY) (BINARY to HEXADECIMAL)

- HEXADECIMAL to BINARY: 1 hexadecimal digit $\rightarrow$ 4 bits.

hexadecimal	**3**	**A**	**9**
binary	**0011**	**1010**	**1001**

- BINARY to HEXADECIMAL: 4 bits $\rightarrow$ 1 hexadecimal digit (starting from the four rightmost bits)

binary	**100**	**1011**	**1010**	**0101**
hexadecimal	**4**	**B**	**A**	**5**

Exercise

BINARY				HEXA
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Compute the hexadecimal representation:

$110110101001100_2 = ????_{16}$

Compute the binary representation:

$5F2C_{16} = ????_2$

UAB
Universitat Autònoma de Barcelona

Exercise (solution)

BINARY				HEXA
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Compute the hexadecimal representation:

$$110110101001100_2 = 6D4C_{16}$$

Compute the binary representation:

$$5F2C_{16} = 0101111100101100_2$$

0101 1111 0010 1100

4. Base conversion (DECIMAL to BINARY)

- Divide the decimal number by 2. Divide the obtained quotient by 2. Keep dividing the obtained quotients by 2 until the obtained quotient is equal to 1.
- The base 2 number consists of the last quotient 1 and the set of previously obtained remainders.

Example: $18_{(10} = ?$

$$\boxed{}_{(10} = \boxed{} \cdot 2 + \boxed{}$$

$$\begin{aligned} 18 &= 9 \cdot 2 + 0 \\ 9 &= 4 \cdot 2 + 1 \\ 4 &= 2 \cdot 2 + 0 \\ 2 &= 1 \cdot 2 + 0 \end{aligned}$$

$18_{(10} = 10010_{(2}$

UAB Universitat Autònoma de Barcelona

Exercise

$43_{(10}$ = binary number?

UAB Universitat Autònoma de Barcelona

Exercise (solution)

$43_{(10}$ = binary number?

$$
\begin{aligned}
43 &= 21 \cdot 2 + 1 \\
21 &= 10 \cdot 2 + 1 \\
10 &= 5 \cdot 2 + 0 \\
5 &= 2 \cdot 2 + 1 \\
2 &= 1 \cdot 2 + 0
\end{aligned}
$$

$43_{(10}$ = 101011

UAB
Universitat Autònoma de Barcelona

6. Sum and difference of binary numbers

Sum of 2 bits:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = **10** (current step bit: 0,
carry to the next step: 1)

Difference of 2 bits:

0 - 0 = 0

1 - 0 = 1

1 - 1 = 0

0 - 1 = **11** (current step bit: 1
borrow to the next step: 1)

Sum example:

```
      1 1 1        → carry
  1 0 1 0 0 1 0 1  = A
+   1 0 1 0 1 1 1  = B
-------------------
  1 1 1 1 1 1 0 0
```

Difference example:

```
  1 0 1 0 0 1 0 1  = A
-   1 0 1 0 1 1 1  = B
  1 0 1 1 1 1          → borrow
-------------------
  0 1 0 0 1 1 1 0
```

Exercise

	1 0 0 1 1 0 1 1 = A
+	1 0 1 0 0 1 1 = B

	1 0 0 1 1 0 0 1 = A
-	1 0 1 0 0 1 1 = B

UAB Universitat Autònoma de Barcelona

Exercise (solution)

	$\boxed{0\ 0\ 1\ 0\ 0\ 1\ 1}$	→ carry
	1 0 0 1 1 0 1 1	= A
+	1 0 1 0 0 1 1	= B
	1 1 1 0 1 1 1 0	

	1 0 0 1 1 0 0 1	= A
-	1 0 1 0 0 1 1	= B
	$\boxed{1\ \ \ \ \ 1\ \ 1}$	→ borrow
	0 1 0 0 0 1 1 0	

SUMMARY

- Computer information representation.
- Numeration systems (decimal, binary, hexadecimal).
- Pure binary system and representation range.
- Base conversion.
- Sum and difference of binary numbers.

0.2 ALGORITHM REPRESENTATION IN PSEUDOCODE

Jean-Pierre Deschamps

University Rovira i Virgili, Tarragona, Spain

Universitat Autònoma de Barcelona

1. Algorithm representation: pseudocode

Algorithm

- Sequence of operations whose objective is the solution of some problem such as:

complex computation, control of some process, etc.

- The result of the algorithm execution must be independent of the chosen type of representation.
- Some common representation methods are: natural language, **pseudocode**, flow diagrams and programming languages.

Pseudocode

Similar to programming language but more informal. It uses a mix of

- natural language sentences,
- programming language instructions,
- key words that define basic structures.

0 .2

2. Operations and control structures

- ASSIGNMENTS
- OPERATORS
  - Comparison
  - Logic operations
  - Arithmetic operations
- SELECTION STRUCTURES (DECISIONS)
  - Simple, Double, Multiple, Case
- ITERATION STRUCTURES (CYCLES)
  - While, For
- PROCEDURES

2. 1 Assignments and types of operators

ASSIGNMENTS

Assignment instruction **<=** : allows to store a value within a variable. Examples:

x <= 15, x <= 2x + y + z, etc.

variable **<=** expression;

where x, y and z are variables.

COMPARISON OPERATORS

(1) Sometimes we can use <=

(2) Sometimes we can use >=

(3) Sometimes we can use /=

Operator	Meaning
<	Smaller than
>	Greater than
=	Equal to
≤ (1)	Smaller than or equal to
≥ (2)	Greater than or equal to
≠ (3)	Different from

2. 1 Assignments and types of operators

LOGIC OPERATORS

Operator	Meaning
and	Logic product
or	Logic sum
not	Negation

ARITHMETIC OPERATORS

Operator	Meaning
+	Sum
-	Difference
*	Product
/	Division

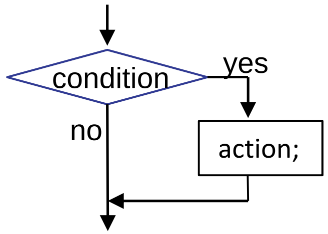
3. Control structures

UAB
Universitat Autònoma de Barcelona

SELECTION STRUCTURES

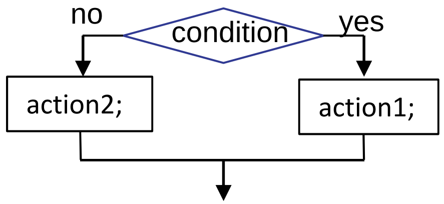
Simple

If *condition* **then** *action(s)*;
end if;



Double

If *condition* **then** *action1/s*;
else *action2/s*;
end if;

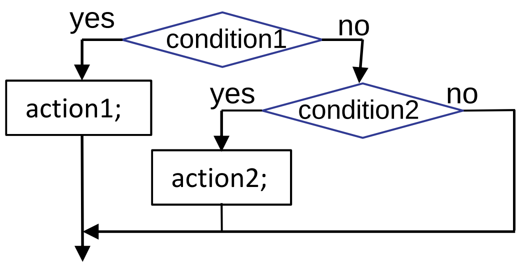


3. Control structures

SELECTION STRUCTURES

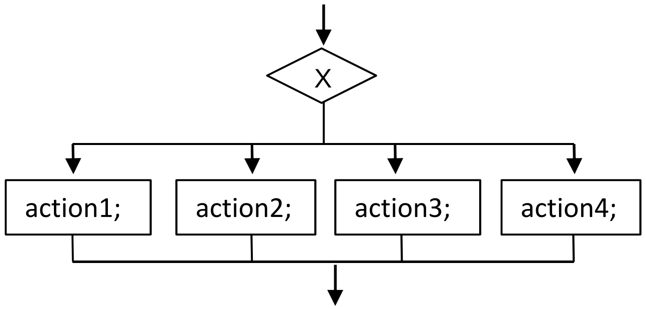
Multiple

If *condition1* **then** *action1/s*;
  elsif *condition2* **then** *action2/s*;
  end if;
end if;



Case

```
case x is
    when "value1" => action1;
    when "value2" => action2;
    when "value3" => action3;
    when "value4" => action4;
end case;
```



UAB Universitat Autònoma de Barcelona

3. Control structures (example 1)

y = X/2 , rounded down

Different cases:

- When X is even, the result of the division is exact (whether X is positive or negative)
- When X is odd, rounding must be calculated differently :
  - if X is negative: (X+1)/2
  - if X is positive (X-1)/2

```
If  (X is even) then  y<= X/2;
          elsif  (X<0)  then  y <= (X+1)/2;
            else y <= (X-1)/2;
          end if;
 end if;
```

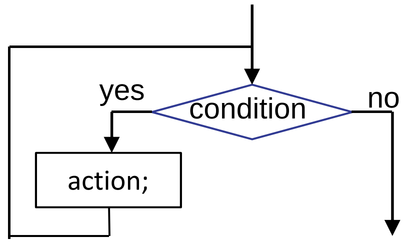
3. Control structures (example 2)

X is a decimal digit: we calculate the binary representation

```
case x is
    when "0" => y <= 0000;
    when "1" => y <= 0001;
    ........
    when "9" => y <= 1001;
end case;
```

3. Control structures

ITERATION STRUCTURES



While

While *condition* **loop**

action/s;

end loop;

For

for variable **in** min **to** max **loop**

action/s;

end loop;

3. Control structures (example 3)

Given two vectors of 8 positions:

a(0), a(1), ……a(7)
x(0), x(1), ……x(7)

We want to do the following calculation:

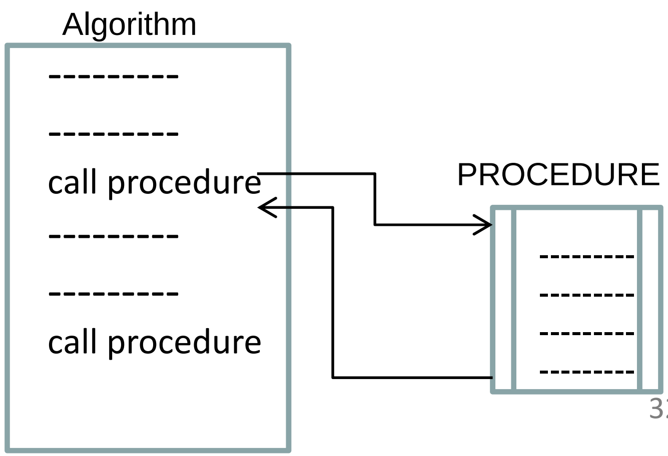
$y = a(0) \cdot x(0) + a(1) \cdot x(1) + a(2) \cdot x(2) + \ldots\ldots\ldots\ldots + a(7) \cdot x(7)$

```
acc <= 0;
for i in 0 to 7 loop
    acc <= acc+ a(i) · x(i);
end loop;
```

4. Procedures

Procedure call *name(parameters)*;

- **Procedure** (subroutine): sequence of instructions (operations and control structures) that execute some task (algorithm).
- A **name** and a set of **parameters** are associated to every procedure.
- A procedure can be **called** one or several times within a program. When called, values are given to its parameters.



4. Procedures (example 1)

$$z = a \cdot \sqrt{x} + b \cdot \sqrt{y}$$

Procedure *square_root(x, y, n) is*

----

end procedure

Algorithm

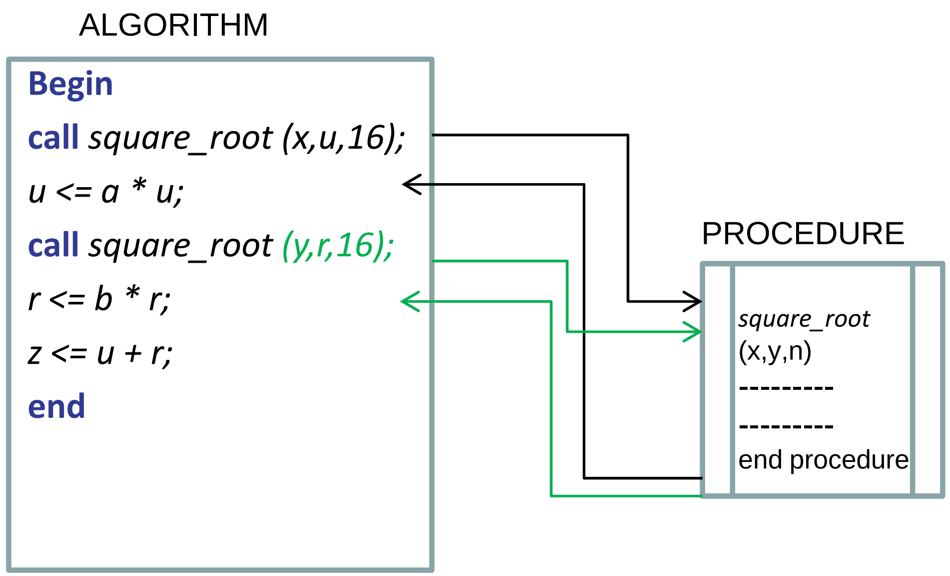
call *square_root (x, u, 16);*

*u <= a * u;*

call *square_root (y, r, 16);*

*r <= b * r;*

z <= u + r;



UAB
Universitat Autònoma de Barcelona

SUMMARY

- Algorithm representation
- Pseudocode
- Operations and control structures
- Procedures