WELCOME

Lectures: CM, every Tuesday from 9:30 to 11:00

Lecturer (CM): William JALBY William.Jalby@uvsq.fr

Lectures through slideware half in English half in French. Multiple views of the same topic.

An excellent generic reference: Wikipedia: www.wikipedia.org

A remarkable specialized reference (in French):

https://fr.wikibooks.org/wiki/Fonctionnement_d%27un_ordinateur

A good and simple reference : Y. N. Patt and S. J. Patel, *Introduction to Computing Systems: from bits & gates to C & beyond*, 2nd edition, McGraw-Hill, 2004, ISBN 0-07-121503-4

Five teaching assistants (but 4 groups/classes) for the exercise sessions:

Stéphane Bouhrour: stephane.bouhrour@uvsq.fr,

Kevin Camus: kevin.camus@uvsq.fr,

Thomas Dionisi: thomas.dionisi@uvsq.fr,

Salah Ibanamar: mohammed-salah.ibnamar@uvsq.fr,

Mathieu Tribalat : mathieu.tribalat@uvsq.fr,

ALL INTERACTIONS SHOULD BE DONE THROUGH MOODLE OR EMAIL

Material for lectures will be provided through moodle.

Exercise sessions (text and solutions) will also be managed through moodle.

There will be (likely) 12 exercise sessions 2 hours long (not 3 hours)

Only one time slot for the exercise sessions: Tuesday from 4:00 PM to 6:00/7:00 PM.

All of the lectures will be on line.

Exercise sessions will alternate between F2F sessions and on line sessions see CELCAT calendar for getting up-to-date info.

There will be quiz/exams/homeworks/QCM organized by teaching assistants for grading. Stay tuned.

INFORMATION REPRESENTATION: PART 1

- Data Representation issues
- Text representation
- Unsigned Integer representation
- Base β representation

How do we represent data in a computer?

At the lowest level, a computer is an electronic machine.

works by controlling the flow of electrons

Easy to recognize two conditions:

1. presence of a voltage – we'll call this state "1"
2. absence of a voltage – we'll call this state "0"

Could base state on *value* of voltage,
but control and detection circuits more complex.

compare turning on a light switch to
measuring or regulating voltage

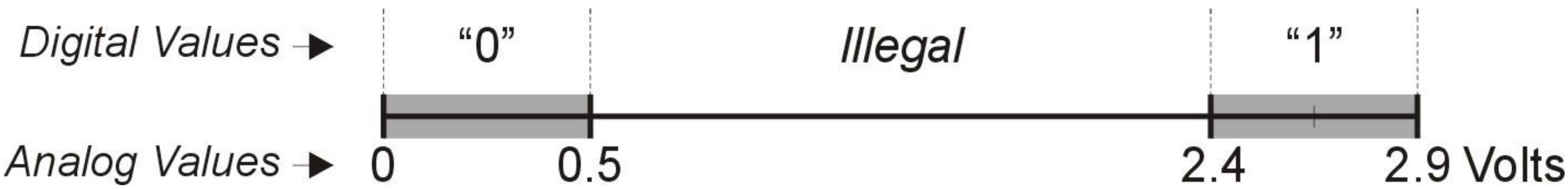
Computer is a binary digital system.

Digital system:

- finite number of symbols

Binary (base two) system:

- has two states: 0 and 1



Basic unit of information is the *binary digit*, or *bit*.

Values with more than two states require multiple bits.

A collection of two bits has four possible states:
00, 01, 10, 11

A collection of three bits has eight possible states:

000, 001, 010, 011, 100, 101, 110, 111

A collection of n bits has 2^n possible states.

What kinds of data do we need to represent?

Numbers – signed, unsigned, integers, floating point, complex, rational, irrational, …

Text – characters, strings, …

Images – pixels, colors, shapes, …

Sound

Logical – true, false

Instructions

…

Data type:

representation and *operations* within the computer

First (rough) classification

- Numerical data: data is operated upon mostly by means of the classical arithmetic operations: +, -, *, /
- Non numerical data: data is operated upon by means of standard set operators: union, intersection, idfentity, inclusion etc …
- In between (gray zone): image, movie, sound close to numeric data but complex operators used: filters, FFT, DFT,

Going from real world (RW) to machine world (MW)

With each object in the real world, we need to associate a representation in the machine world.

First, real world has to be structured. For example: text, paragraph, sentence, words, letters. Final element called atom of information.

Then two major steps:

- Encoding: associating with each atom a machine representation: Going from RW to MW.
- Decoding: reverse operation of encoding. Going from MW to RW.

Both operations should be easy to carry out.

Finite Real World

Let us assume that RW is finite: latin character set upper case + 4 punctuation symbols: 30 different symbols to be represented.

EASY SOLUTION: each symbol is represented by a “block” of 5 bits. The “5 bits” can be viewed as labels associated with boxes and encoding can be viewed as distributing symbols across the 32 different boxes.

KEY PROPERTY: make sure that each box contains at most one symbol. This will make decoding obvious.

This works because $\underline{32 = 2^5}$ is greater than the number of symbols to be represented: 30. The number of boxes is greater than the number of RW elements.

Finite Real World

We could use 4 bits to represent the latin character sets + 4 punctuation symbol.

Encoding is still fast and even saves space: 4 bits instead of 5 bits.

Decoding is much harder because à priori a same box contains multiple symbols. What is the right symbol to pick ??

KEY TRICK: associate within a same box, two letters with radically different occurrence frequency (depends upon language). For example, for French put A and Z in the same box.

TRICK FOR DECODING: for each box, there will a preferred symbol : the most frequent one. Then use a dictionary

Infinite Real World (N)

N the set of positive integers.

Since the number of bits in the machine is finite, the number of positive numbers “exactly” represented in the machine will be finite!!

In fact, only a segment of N is represented: [0, A]. All of the elements (which constitute a finite set) in the segment are exactly represented in the machine.

All of the numbers above A are exceeding machine capacity and they correspond to Overflow Zone.

Infinite Real World (Z)

Z the set of negative + positive integers.

Since the number of bits in the machine is finite, the number of positive numbers “exactly” represented in the machine will be finite!!

In fact, only a segment of N is represented: [-A, A]. All of the elements (which constitute a finite set) in the segment are exactly represented in the machine.

All of the numbers outside of [-A, A] are exceeding machine capacity and they correspond to Overflow Zone.

Infinite Real World (Q and R)

Q (resp. R) is the set of rational (resp. real) numbers.

MAJOR ISSUE: [0, 1] contains an infinite number of rational and real numbers.

The segment trick does not work any more.

Numbers exactly represented in the machine have to be spread in a clever manner over the interval of interest.

Two major techniques:

- Fixed point representation
- Floating point representation

MAJOR ISSUE: how to associate an arbitrary number with a machine representation ??

Accuracy/Approximation/Truncation.

Key characteristics of numerical representations

- SPACE: number of bits
- NUMERICAL OPERATION COMPLEXITY
- RANGE: largest number exactly represented in the machine, smallest positive (non zero) number exactly represented in the machine.
- ACCURACY: both for numbers and operations.

Hexadecimal Notation

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

Binary	Hexadecimal	Decimal
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

It is often convenient to write binary (base-2) number using hexadecimal notation: replace blocks of 4 bits by hexadecimal symbols

- Fewer digits -- four bits per hex digit
- Less error prone -- easy to corrupt long string of 1's and 0's
- More compact than decimal : only one symbol

Converting from Binary to Hexadecimal

Every four bits is replaced by an hexadecimal digit/symbol.

Start grouping from right-hand side

011	1010	1000	1111	0100	1101	0111
3	**A**	**8**	**F**	**4**	**D**	**7**

This is not a new machine representation, just a convenient way to write the number.

Text: ASCII Characters

ASCII: Maps 128 characters to 7-bit code.

both printable and non-printable (ESC, DEL, …) characters

For more details see
https://fr.wikibooks.org/wiki/Les_ASCII_de_0_%C3%A0_127/La_table_ASCII

00	nul	10	dle	20	sp	30	0	40	@	50	P	60	`	70	p
01	soh	11	dc1	21	!	31	1	41	A	51	Q	61	a	71	q
02	stx	12	dc2	22	"	32	2	42	B	52	R	62	b	72	r
03	etx	13	dc3	23	#	33	3	43	C	53	S	63	c	73	s
04	eot	14	dc4	24	$	34	4	44	D	54	T	64	d	74	t
05	enq	15	nak	25	%	35	5	45	E	55	U	65	e	75	u
06	ack	16	syn	26	&	36	6	46	F	56	V	66	f	76	v
07	bel	17	etb	27	'	37	7	47	G	57	W	67	g	77	w
08	bs	18	can	28	(	38	8	48	H	58	X	68	h	78	x
09	ht	19	em	29	)	39	9	49	I	59	Y	69	i	79	y
0a	nl	1a	sub	2a	*	3a	:	4a	J	5a	Z	6a	j	7a	z
0b	vt	1b	esc	2b	+	3b	;	4b	K	5b	[	6b	k	7b	{
0c	np	1c	fs	2c	,	3c	<	4c	L	5c	\	6c	l	7c	|
0d	cr	1d	gs	2d	-	3d	=	4d	M	5d	]	6d	m	7d	}
0e	so	1e	rs	2e	.	3e	>	4e	N	5e	^	6e	n	7e	~
0f	si	1f	us	2f	/	3f	?	4f	O	5f	_	6f	o	7f	del

Interesting Properties of ASCII Code

What is relationship between a decimal digit ('0', '1', …) and its ASCII code?

What is the difference between an upper-case letter ('A', 'B', …) and its lower-case equivalent ('a', 'b', …)?

Given two ASCII characters, how do we tell which comes first in alphabetical order?

Are 128 characters enough?

Beyond ASCII Code

- Unicode: 16 bit superset of ASCII providing representation of many different alphabets and specialized symbol sets.

http://www.unicode.org/

- EBCDIC: IBM's mainframe representation.

Extended Binary Coded Decimal Interchange Code

https://fr.wikipedia.org/wiki/Extended_Binary_Coded_Decimal_Interchange_Code

Other Data Types

- Text strings
  - sequence of characters, terminated with NULL (0)
  - typically, no hardware support
- Image
  - array of pixels
    - monochrome: one bit (1/0 = black/white)
    - color: red, green, blue (RGB) components (e.g., 8 bits each)
    - other properties: transparency
  - hardware support:
    - typically none, in general-purpose processors
    - MMX -- multiple 8-bit operations on 32-bit word
- Sound
  - sequence of fixed-point numbers

Unsigned Integers

Non-positional notation

A number ("5") could be represented with a string of ones ("11111")

Usual method for represented a number between 0 and 10 using fingers.

problems?

Weighted positional notation

like decimal numbers: "329"

"3" is worth 300, because of its position, while "9" is only worth 9

most significant 101 *least significant*

329 → 10^2 10^1 10^0

101 → 2^2 2^1 2^0

$\boxed{3\text{x}100 + 2\text{x}10 + 9\text{x}1 = 329}$

$\boxed{1\text{x}4 + 0\text{x}2 + 1\text{x}1 = 5}$

Convention

Most significant bit on the left, least significant bit on the right.

Arbitrary decision: follows usual decimal representation

In fact, in machine world, two conventions coexist: from left to right and right to left: LITTLE ENDIAN, BIG ENDIAN

https://fr.wikipedia.org/wiki/Boutisme

Unsigned Binary Integers: examples

$$A = a_3a_2a_1a_0 = a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$$

(where the digits $a_3a_2a_1a_0$ can each take on the values of 0 or 1 only)

N = number of bits

Range is:
$0 \leq i < 2^N - 1$

	3-bits	5-bits	8-bits
0	000	00000	00000000
1	001	00001	00000001
2	010	00010	00000010
3	011	00011	00000011
4	100	00100	00000100

Unsigned Binary Integers: general case

$$A = a_{(N-1)}\ldots\ldots a_2 a_1 a_0 = a_{(N-1)} 2^{(N-1)} + \ldots + a_2 2^2 + a_1 2^1 + a_0 2^0$$

where $a_{(N-1)}\ldots\ldots a_2 a_1 a_0$ are bits which can each take on the values of 0 or 1 only

N = number of bits

Range is:

$0 \leq i < 2^N - 1$

0 is represented

1 is the smallest non zero integer represented

$2^N - 1$ is the largest integer represented

Unsigned Binary Arithmetic: addition (1)

Base-2 addition – just like base-10!

1. First build addition table
2. Second add from right to left, propagating carry

Addition +	**0**	**1**
0	**0**	**1**
1	**1**	**10**

Unsigned Binary Arithmetic: addition (2)

When adding two binary numbers, carry value is necessarily 0 or 1.

No longer true when simultaneously adding more than 2 binary numbers

```
  10010        10010        1111
+  1001      +  1011      +    1
  -----        -----        -----
  11011        11101        10000
```

carry

```
  10111
+   111
  -----
```

Subtraction, multiplication, division,…

Unsigned Binary Arithmetic: multiplication

Multiplication:

1. Compute partial products: multiply by 0 or 1 (easy)
2. Then sum up the partial products properly shifted

			1	0	1	1
		X	1	0	0	1
			1	0	1	1
		0	0	0	0	
	0	0	0	0		
1	0	1	1			
1	1	0	0	0	1	1

Converting Binary to Decimal (unsigned)

1. Add powers of 2 that have "1" in the corresponding bit positions.

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

$$
\begin{aligned}
X &= 1101000_{two} \\
&= 2^6+2^5+2^3 = 64+32+8 \\
&= 104_{ten}
\end{aligned}
$$

More Examples

$$X = 0100111_{two}$$
$$= 2^5+2^2+2^1+2^0 = 32+4+2+1$$
$$= 39_{ten}$$

$$X = 00011010_{two}$$
$$= 2^4+2^3+2^1 = 16+8+2$$
$$= 26_{ten}$$

n	**2^n**
0	**1**
1	**2**
2	**4**
3	**8**
4	**16**
5	**32**
6	**64**
7	**128**
8	**256**
9	**512**
10	**1024**

Converting Decimal to Binary (unsigned numbers)

First Method: *Euclidian Division*

1. Find magnitude of decimal number. (Always positive.)
2. Divide by two – remainder is least significant bit.
3. Keep dividing by two until answer is zero,
   writing remainders from right to left.

$X = 104_{ten}$

104/2	=	52 r0	*bit 0*
52/2	=	26 r0	*bit 1*
26/2	=	13 r0	*bit 2*
13/2	=	6 r1	*bit 3*
6/2	=	3 r0	*bit 4*
3/2	=	1 r1	*bit 5*
1/2	=	0 r1	*bit 6*

$X = 01101000_{two}$

Converting Decimal to Binary

Second Method: *Subtract Powers of Two*

1. Find magnitude of decimal number.
2. Subtract largest power of two less than or equal to number.
3. Put a one in the corresponding bit position.
4. Keep subtracting until result is zero.

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

$X = 104_{ten}$

104 - 64	=	40		*bit 6*
40 - 32	=	8		*bit 5*
8 - 8	=	0		*bit 3*

$X = 1101000_{two}$

Base β representation

$$A = a_{(N-1)}\ldots..a_2a_1a_0 = a_{(N-1)}\beta^{(N-1)} + \ldots. + a_2\beta^2 + a_1\beta^1 + a_0\beta^0$$

where β is an integer greater or equal to 2

And where $a_{(N-1)}\ldots..a_2a_1a_0$ are such that:

$$0 \leq i < \beta$$

N = number of “digits/symbols”

Range is:

$0 \leq a_i < \beta^N - 1$

0 is represented

1 is the smallest non zero integer represented

$\beta^N - 1$ is the largest integer represented

Generating Base β representation

Repetitive use of Euclidian division:

$A = \beta\, q_1 + r_1$ with $0 \le r_1 < \beta$

$q_1 = \beta\, q_2 + r_2$ with $0 \le r_2 < \beta$

$q_2 = \beta\, q_3 + r_3$ with $0 \le r_3 < \beta$

……………

…….. Until

$q_p = \beta\, q_p + r_p$ with $0 \le r_p < \beta$ and with $0 \le q_p < \beta$

$A = a_{(N-1)} \ldots\ldots a_2 a_1 a_0 = a_{(N-1)}\beta^{(N-1)} + \ldots + a_2\beta^2 + a_1\beta^1 + a_0\beta^0$

Generating Base β representation

Then substitution

$A = \beta\, q_1 + r_1 = \beta\,(\beta\, q_2 + r_2) + r_1$ etc……………………

……..

$A = q_p \ldots r_2 r_1 r_0 = q_p \beta^p + \ldots + r_3 \beta^2 + r_2 \beta^1 + r_1 \beta^0$

Base β examples

- $\beta = 10$ standard decimal
- $\beta = 4$
- $\beta = 8$ octal

When $\beta > 10$, a small issue with symbols. Until 10, regular digits can be used but beyond it does not work any longer.

For example let us assume $\beta = 16$.

If reusing standard decimal what does 11 mean in base 16.

Amibiguous:

$11 = 1 \times 16 + 1 = 17$

$11 = 0 \times 16 + 11 = 11$

Either searators have to be used or special symbols. Hexadecimal encoding uses 6 extra letters.

Base β Arithmetic: addition (1)

Base β addition – just like base-10!

1. First, build addition table
2. Second, add from right to left, propagating carry

+ Base 5	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	10
2	2	3	4	10	11
3	3	4	10	11	12
4	4	10	11	12	13

Acknowledgements

Some of these slides were inspired by slides developed by:

- Y.N. Patt (Univ of Texas Austin)
- S. J. Patel (Univ of Illinois Urbana Champaign)
- Walid A. Najjar (Univ California Riverside)
- Brian Linard (Univ California Riverside)
- G.T. Byrd (Univ North Carolina)