A little bit of history

Jordi Cortadella
Department of Computer Science

Mechanization

The old dream of mechanical computing

Calculating-Table by Gregor Reisch: Margarita Philosophica, 1503.

Roman Abacus

18th-century calculating machine

Fig. 31A — Raisin press á masque à pouce sans le tuteur.
The first mechanical computer

- **Charles Babbage** (1791-1871), English mathematician, philosopher, inventor and mechanical engineer.
- Invented the *difference engine* (never finished).
- Useful to tabulate polynomial functions.

### ENIAC (1946)
- The first electronic general-purpose computer, designed to calculate artillery firing tables.
- 167 m², 27 tons, it could solve 5000 additions and 385 multiplications in 1 second.
- Programmable with switches and cables.
- Programming an algorithm could take several weeks.

Programmable digital computers

- Alan Turing (1912-1954), the father of modern digital computers.
- During 2nd World War, he broke German ciphers generated by the Enigma machine. The war in Europe was shortened by at least two years.
- The *bombe* was the electromechanical machine created to break Enigma.
- It could perform chains of logical deductions for each setting of the rotors.

"Sometimes it is the people no one imagines anything of who do the things no one can imagine."

- Alan Turing, played by Benedict Cumberbatch, *The Imitation Game*.
Turing machine

- A universal machine: theoretical model for computability.
- It can simulate any computer algorithm.
- Useful to understand the limits of mechanical computations.
- Turing machines were the inspiration for modern CPUs.

Modern digital computers

- Inspired by Turing’s work, he proposed the architecture of modern digital computers.
- Essential innovation: *programs stored in memory*.
- Most computers today are based on von Neumann’s architecture.
Why digital?

• Binary digit (bit): minimum unit of information  
  — yes/no, on/off, true/false, am/pm, open/close, ...

• Binary representations are very convenient for physical (electrical) implementations.

• Electronic computers use voltage levels to represent bits, e.g., 0 (0 volts) and 1 (3.3 volts)

Representing information with bits

1 bit → 2 states (0, 1)  
2 bits → 4 states (00, 01, 10, 11)  
3 bits → 8 states (000, 001, 010, 011, 100, 101, 110, 111)  
n bits → 2^n states

Example: RGB model for colors

Any color can be generated by adding the three components with different intensity.

Representing numbers with bits

\[ 154 = 1 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0 \]

\[ 154 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \]

\[ = 128 + 16 + 8 + 2 \]

\[ \rightarrow 10011010 \]
Arithmetic operations

\[ \begin{array}{c}
10011010 \quad (154) \\
+ 00110011 \quad (51) \\
\hline
11001101 \quad (205)
\end{array} \]

Conventional computers work with 32- or 64-bit representations.

Not all the results can be represented (not enough bits).

\[ \begin{array}{c}
01001 \quad (9) \\
x 10011 \quad (19) \\
\hline
01001 \\
00000 \\
00000 \\
01001 \\
010101011 \quad (171)
\end{array} \]

Logic circuits and chips

The evolution of technology

ENIAC (1946)

167 m², 270 tons
150 kW of power
5,000 additions / sec
17,468 vacuum tubes
7,200 crystal diodes, 1,500 relays
70,000 resistors, 10,000 capacitors

Intel Ivy Bridge-HE-4 (2013)

160 mm², 4 cores
130 W of power
10 billion additions / sec
1.4 billion transistors (22nm)

The fascinating world of technology
Back to von Neumann’s architecture

Introduction to Programming © Dept. CS, UPC

Machine code

MIPS architecture (32 bits):

<table>
<thead>
<tr>
<th>000000 00001 00010 00110 00000 100000</th>
</tr>
</thead>
<tbody>
<tr>
<td>arith R1 R2 R6</td>
</tr>
<tr>
<td>add</td>
</tr>
<tr>
<td>R6 ← R1 + R2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>100011 00011 01000 00000 00001 000100</th>
</tr>
</thead>
<tbody>
<tr>
<td>load R3 R8 68</td>
</tr>
<tr>
<td>R8 ← Memory [R3 + 68]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>000010 00000 00000 00000 10000 000000</th>
</tr>
</thead>
<tbody>
<tr>
<td>jump 1024</td>
</tr>
<tr>
<td>PC ← PC + 1024</td>
</tr>
</tbody>
</table>

Assembly language

MIPS assembly language
Every line corresponds to an instruction of machine code

Function that calculates the factorial of a number (n!)

High-level programming languages

C AREA OF A TRIANGLE - HERON'S FORMULA

C INPUT - CARD READER UNIT 5, INTEGER INPUT
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPLAY ERROR OUTPUT CODE 1 IN JOB CONTROL LISTING
INTEGER A,B,C
READ(5,501) A,B,C

501 FORMAT(3I5)

IF (A.EQ.0 .OR. B.EQ.0 .OR. C.EQ.0) STOP 1
S = (A + B + C) / 2.0
AREA = SQRT( S * (S - A) * (S - B) * (S - C))
WRITE(6,601) A,B,C,AREA

601 FORMAT(4H A= ,I5,5H B= ,I5,5H C= ,I5,8H AREA= ,F10.2,12HSQUARE UNITS)
STOP
END

C AREA OF A TRIANGLE - HERON'S FORMULA
C INPUT - CARD READER UNIT 5, INTEGER INPUT
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPLAY ERROR OUTPUT CODE 1 IN JOB CONTROL LISTING
INTEGER A,B,C
READ(5,501) A,B,C

501 FORMAT(3I5)

IF (A.EQ.0 .OR. B.EQ.0 .OR. C.EQ.0) STOP 1
S = (A + B + C) / 2.0
AREA = SQRT( S * (S - A) * (S - B) * (S - C))
WRITE(6,601) A,B,C,AREA

601 FORMAT(4H A= ,I5,5H B= ,I5,5H C= ,I5,8H AREA= ,F10.2,12HSQUARE UNITS)
STOP
END
High-level programming languages

• Programming in assembly language is a tedious and unpleasant task.

• High-level languages:
  – Independent from the computer
  – Higher levels of abstraction
  – Closer to natural language (if-then, while-do, ...)

• The first high-level language: **FORTRAN** (IBM, 1957).

• Today, there are thousands of programming languages:
  – Basic, Cobol, Java, C++, Go, Python, Ruby, Lisp, Haskell, ML, Prolog, Perl, Excel, Pascal, Javascript, PHP, C#, R, Matlab, SQL, Smalltalk, Eiffel, Scratch, …

---

**Function to calculate n!**

\[ n! = n \cdot (n - 1) \cdot (n - 2) \cdots 2 \cdot 1 \]

**Haskell**

```haskell
factorial n = 
  if n == 0 then 1 
  else n * factorial(n - 1)
```

**C++**

```cpp
int factorial(int n) {
  if (n == 0) return 1;
  return n * factorial(n - 1);
}
```

**Python**

```python
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
```

**MIPS assembly language**

```assembly
# Fact
factorial:
  bne $a0, $zero, gen
  ori $v0, $zero, 1
  jr $ra

gen:
  addiu $sp, $sp, -8
  sw $ra, 4($sp)
  sw $a0, 0($sp)
  addiu $a0, $a0, -1
  jal fact
  lw $a0, 0($sp)
  lw $ra, 4($sp)
  addiu $sp, $sp, 8
  mul $v0, $v0, $a0
  jr $ra
```

---

**The top 25 programming languages**

IEEE Spectrum
Sept. 10, 2015

---

**Compilers**

Compilers translate programs written in high-level languages into machine code.
Our challenge in this course

Design a program that solves a difficult Sudoku in one second!

The trip to computer programming starts now:
You define your limits.

Let's make it happen together!