Programming basics
(GKNB_INTA023)

Hatwagner F. Miklós, PhD.

Széchenyi István University, Győr, Hungary

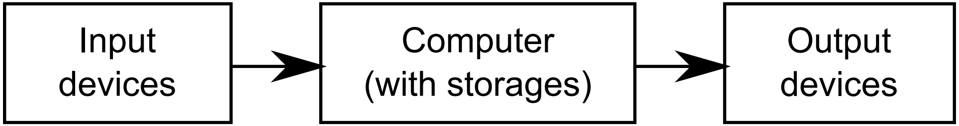
September 21, 2020

Capabilities of a computer

Questions:

- What sort of problems can be solved by a computer? (hardware capabilities, software libraries, programming languages, . . .)
- Which parts of the problem are appropriate to solve with a computer?
- Unique problem $\rightarrow$ general solution

Computer: information processing tool



von Neumann architecture

Essence:

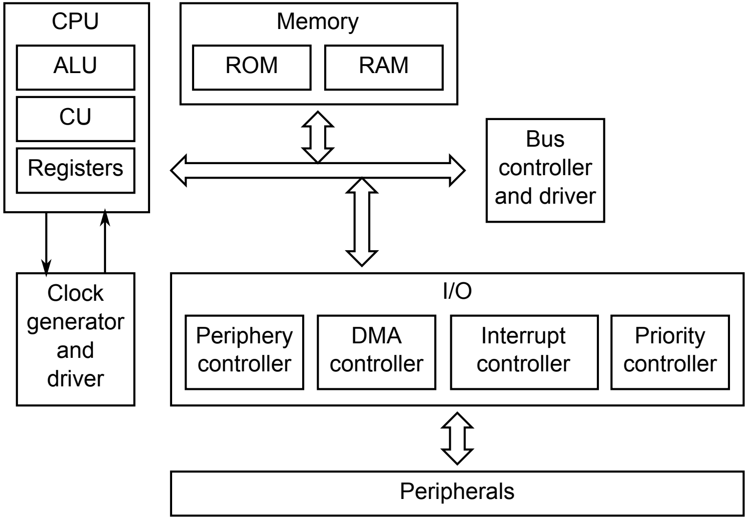
- Sequential instruction execution
- Binary number system
- Both user data *and* program code are stored in the same memory (see also Harvard architecture)
- Fully electronic
- General usage
- Central Processing Unit (automatic operation)

Parts of the computer:

- Central Processing Unit, CPU
  - Arithmetic/Logic Unit, ALU
  - Control Unit, CU
- Memory
- I/O devices

See von Neumann architecture

Functional model of a computer with bus system



Information

Categories:

- (User or program) Data (to process)
- Program (to execute)

Data

The *data* of a task is all the information from which we can get to the solution by performing *operations* and transforming them, and data is all the information, including the solution, that is generated from the initial data during operations and transformations.

Program

A *program* is information that describes how a computer have to work to get the solution it is looking for using the baseline data.

A program:

- contains instructions (communication, initiation of basic activities)
- defines the order of instruction execution

Data

Data handling:

- (Constant) literals (writing the value to the appropriate place)
- with variables

According to the amount of data the variable may be:

- Basic / primitive / primary (one unit of data)
- Compound / derived (data group)

Data

Properties of basic variables

- name (id) $\rightarrow$ usable characters, destination/function, expressive name, conventions
- type
    - How to store the data in memory? (data representation and required memory capacity)
    - What sort of instructions can be executed with it?
    - The nature of data (numeric, string $\rightarrow$ data representation)
- Memory area
    - stores the value according to the data type
    - in most cases it is not initialized automatically

Fixed-point arithmetic

Unsigned case

- $2018_{10} = 2 \cdot 10^3 + 0 \cdot 10^2 + 1 \cdot 10^1 + 8 \cdot 10^0$
- $2018_{10} = 0000\ 0111\ 1110\ 0010_2 = 1 \cdot 2^{10} + 1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^1$
- $2018_{10} = 3742_8 = 3 \cdot 8^3 + 7 \cdot 8^2 + 4 \cdot 8^1 + 2 \cdot 8^0$
- $2018_{10} = \mathrm{7E2}_{16} = 7 \cdot 16^2 + 14 \cdot 16^1 + 2 \cdot 16^0$

Integer part	Remainder of division by 10
2018	8
201	1
20	0
2	2
0	

Integer part	Remainder of division by 16
2018	2
126	E
7	7
0	

Fixed-point arithmetic

- Usual lengths: 8, 16, 32, 64 bits (1, 2, 4, 8 bytes; usually one byte is the smallest addressable unit $\rightarrow$ prefixes)
- $V_{\text{unsigned integer}} = \sum_{i=0}^{N-1} b_i \cdot 2^i$
- Interval: $[0; 2^N - 1]$

No. of bits	No. of values
8	256
16	65 536
32	$4,29 \cdot 10^9$
64	$1,84 \cdot 10^{19}$

Fixed-point arithmetic

Usage of signs

- two's complement
- one's complement, then $+1$
- value multiplied by -1: subtraction from 2^N
- Sign bit $\leftrightarrow$ sign flag bit
- $V_{\text{two's complement}} = -b_{N-1} \cdot 2^{N-1} + \sum_{i=0}^{N-2} b_i \cdot 2^i$
- Interval: $[-2^{N-1}; 2^{N-1} - 1]$

$$\begin{array}{rr} & 1\ 0000\ 0000 \\ - & 0100\ 1100 \\ \hline & 1011\ 0100 \end{array}$$

$$\begin{array}{rr} & 256 \\ - & 76 \\ \hline & 180 \end{array}$$

Bits	Value
0111 1111	127
0111 1110	126
⋮	⋮
0000 0001	1
0000 0000	0
1111 1111	−1
1111 1110	−2
⋮	⋮
1000 0000	−128

Floating-point arithmetic

Representing racional numbers

- Normal form of numbers (Scientific notation)
- $m \cdot 2^c$, where m means mantissa, c characteristic (exponent)
- $1/2 \leq m < 1$
- $0,1111110001 \cdot 2^{10} = 2018_{10}$
- Example of a value given by excess-128 representation:

$$01111110\ 00100000\ 00000000|10001010_2 = 2018_{10}$$

IEEE754

Character coding

Characters

- Letters, digits, punctuation marks, ...
- The world of PCs': ASCII (American Standard Code for Information Interchange)
- 7 bit code: the first 128 characters are always the same, the others depend on code pages (eg. 852)
- The first 32 values correspond to control signals/characters
- Letters: in alphabetical order, digits in increasing order
- new character encoding ways (see Unicode)

Texts

- string
- "C" language: terminating 0 character $\rightarrow$ size: number of characters $+$ 1 (needs time to calculate the length of the string)
- Pascal: the first byte encodes the length of the string (limits the maximum length)

'J'	'o'	'e'	'\0'
74	111	101	0
0100 1010	0110 1111	0110 0101	0000 0000
4A	6F	65	00

Compound (derived, user) variables

Describes a group of data. Types:

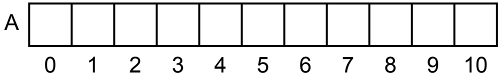
- array
- structure (Pascal: record)

4th property of an array: *dimension*, the layout of data:

- *one dimension* (vector)
- two dimensions (matrix, table)

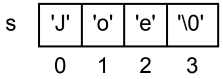
Indexing

- ordering the elements
- $0 \leq x < \mathrm{size}, x \in N$
- `A[0], A[1], ..., A[10]`



Compound (derived, user) variables

- Can be created from several basic types
- Array elements can be used everywhere, where the usage of the corresponding basic variables are allowed
- Strings are one dimensional arrays in "C" language



Notice that

- the number of letters (characters) is 3,
- and s[3] is the terminating '\0'.

Characters can be considered as

- characters
- small integer numbers

Programming languages

- Machine code
- Assembly

example02.asm (Source: Agárdi Gábor: Gyakorlati Assembly)

```
Pelda02 Segment                            ;Segment definition
        assume cs:Pelda02,ds:Pelda02       ;cs es ds registers are set
                                           ;to the start of the segment.
Start:  mov     ax,Pelda02                 ;Set ds register
        mov     ds,ax
        mov     ax,0b800h                  ;Loads the segment and offset addresses
        mov     es,ax                      ;of screen memory to the es register
        mov     di,1146                    ;Sets the offset address in
                                           ;di index register
        mov     al,"A"                     ;Loads the ASCII code of letter "A"
                                           ;in register al
        mov     ah,7                       ;The color of the character is black
                                           ;on white background
        mov     es:[di],ax                 ;Writes the content
                                           ;(letter "A" with the specified colors)
                                           ;to the address described by es:di
        mov     ax,4c00h                   ;Back to DOS
        int     21h
Pelda02 Ends                               ;Segment end
        End     Start                      ;Program end
```

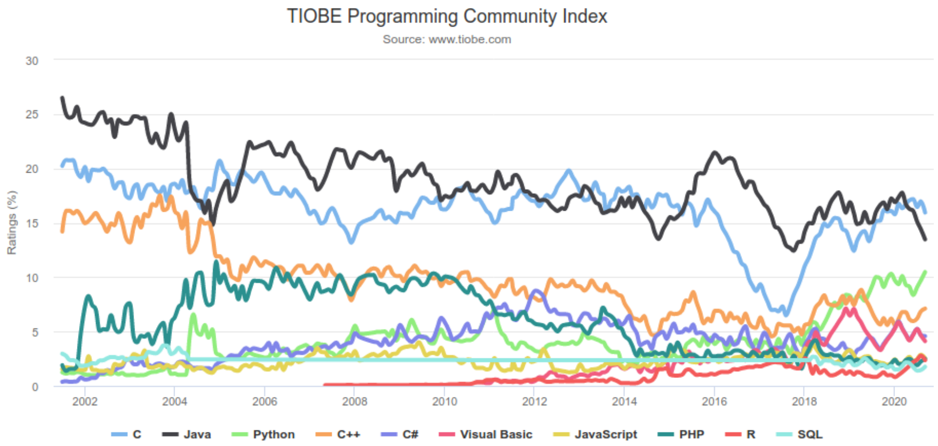
Programming languages

- C
  - Dennis Ritchie, Bell Laboratories (1969-1973): “C” programming language $\rightarrow$ UNIX operating system
  - “Standards”: K&R (1978), ANSI (or C89, 1989), C99, C11.
  - Properties: general purpose, imperative (tells *how* the program have to operate in order to achieve the required state changes), structured (source files, blocks, loops, etc.)
- C++
  - Bjarne Stroustroup (1979): “C with Classes”
  - “Standards”: C++ (1983), “The C++ Programming Language” (1985), . . . , ISO/IEC 14882:2017, C++20
  - Properties: general purpose, procedural, functional, object-oriented, mostly “C” compatible

Programming languages

- Literature
  - C Programming Language, 2nd Edition by Brian W. Kernighan, Dennis M. Rithcie
  - C Programming: A Modern Approach, 2nd Edition by K. N. King
  - Programming in C, 4th edition by Stephen G. Kochan
  - C Traps and Pitfalls by Andrew Koenig
  - The C++ Programming Language, 4th Edition by Bjarne Stroustrup
- Software
  - Microsoft Azure Dev Tools
  - QT Creator IDE
  - GNU Compiler Collection
  - Code::Blocks
  - Geany
  - repl.it – online editor

Programming languages



TIOBE Index for September 2020

Programming languages

numbers.c

```c
#include <stdio.h>

int main(void) {
  int i;
  for(i=1; i<=10; i++)
    printf("%d ", i);
  printf("\n");
  return 0;
}
```

Numbers.java

```java
class Numbers {
  public static void main(String[] args) {
    for(int i=1; i<=10; i++)
      System.out.print(i + " ");
    System.out.println();
  }
}
```

numbers.php

```php
<?php
  for($i=1; $i<=10; $i++)
    echo $i.' ';
?>
```

numbers.js

```javascript
let str = "";
for(let i=1; i<=10; i++)
  str += i + " ";
console.log(str);
```

From source code to running

1. Editing the source code (usually .c file extension, ASCII text file format)

first.c

```
/* This line is a comment */
#include<stdio.h>

int main(void) {
  printf("This is our first program written in C!\n");
  return 0;
}
```

From source code to running

2. Building

```
gcc -Wall -o first first.c
```

Command line arguments

-Wall
: It warns of easy-to-avoid, questionable solutions (potential errors)

-o
: Name of the executable file (here: `first`)

3. Running

in a Linux terminal window

```
wajzy@wajzy-notebook:~/Dokumentumok/gknb_inta023/lecture01$ ./first
This is our first program written in C!
wajzy@wajzy-notebook:~/Dokumentumok/gknb_inta023/lecture01$
```

From source code to running

Details of the build process

1. Compiling

$$\begin{tikzcd} \text{first.c} \arrow[r] & \boxed{\text{compiler}} \arrow[r] & \text{first.o} \end{tikzcd}$$

Compilation with GCC

```
gcc -Wall -c first.c
```

Meaning of the command line arguments

`-c`
: Compile only, executable file will not be created

Types of messages:

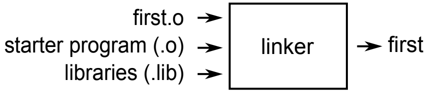
- errors → syntactic problems, object file cannot be created
- warnings → warns of suspicious solutions, proposes alternatives, object file is generated

From source code to running

Details of the build process

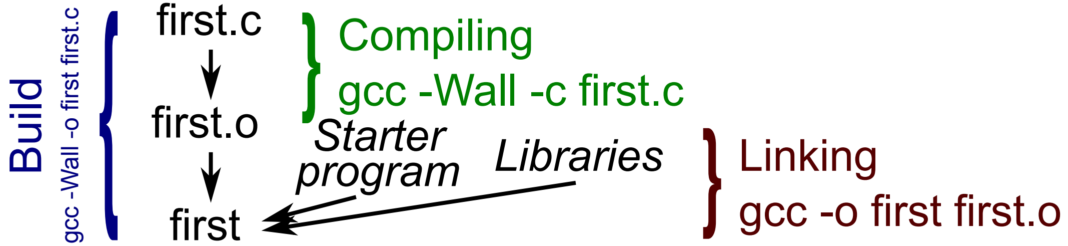
2. Linking
   - object codes of functions can be found in static libraries (.lib, run-time library or standard library)

```
gcc -o first first.o
```



Messages of the linker

From source code to running



From source code to running

first.c

```c
/* This line is a comment */
#include<stdio.h>

int main(void) {
  printf("This is our first program written in C!\n");
  return 0;
}
```

From source code to running

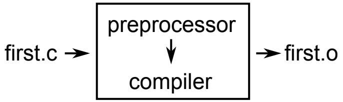
Comments for the developers:

- after // to the end of the line (can be used only with C99 compliant and newer compilers)
- between /* and */ even through several lines
- The preprocessor deletes them

Directives:

- lines beginning with #
- #include<. . . > includes the content of the header file → eg. to use constants, library functions (eg. `/usr/include/stdio.h`)

Directives and comments are processed the the preprocessor



From source code to running

The `main` function

- Function: group of data and executable instructions. Their operation can be influenced by arguments and they may return a value.
- Definition of a function: providing all information about the function
- *return_type function_name(argument_list) { function_body }*
- The function `main` has a special purpose: it is the entry point of the program
- Returns a status (exit) code to the OS (0: everything is fine)
- Return value: after `return`

; indicates the end of a statement

From source code to running

Standard streams

- Output (stdout, ≈ screen), used by eg. `printf`
- Input (stdin, ≈ keyboard), used by eg. `scanf`
- Error (stderr, ≈ screen), used by eg. `fprintf` (unbuffered)

Calling `printf`

- Goal: prints formatted strings to the standard output
- Prints the string between quotation marks to standard output
- `\n` is an escape sequence to execute terminal commands described by non printable characters

From source code to running

Esc. sequence.	Meaning
\a	alert signal (bell)
\b	backspace
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab, HTAB
\v	vertical tab, VTAB
\\	backslash
\?	question mark
\'	apostrophe
\"	quotation mark
\ooo	octal number
\xhh	hexadecimal number
\0	the character whoose ASCII code is zero