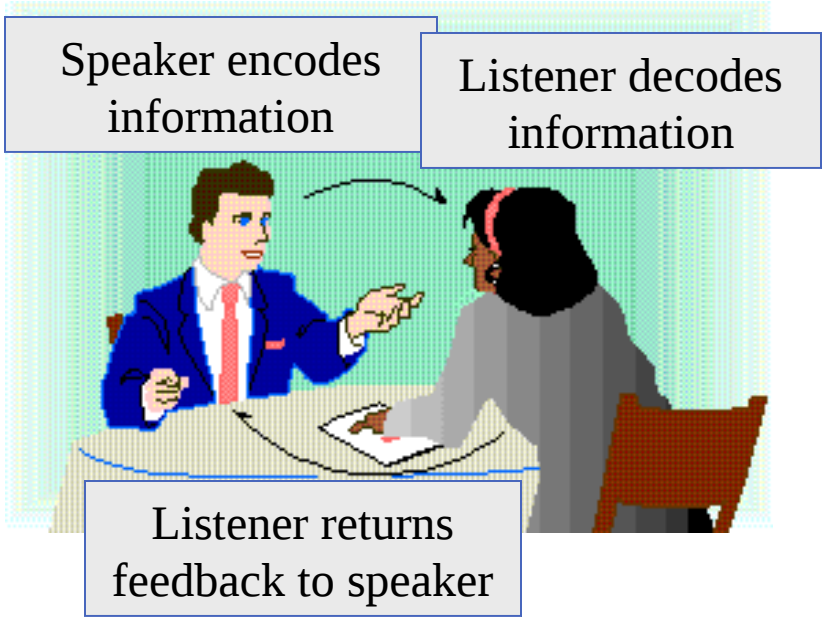
Computer Languages, Algorithms and Program Development

How do computers know what
we want them to do?

Communicating with a Computer

Communication cycle

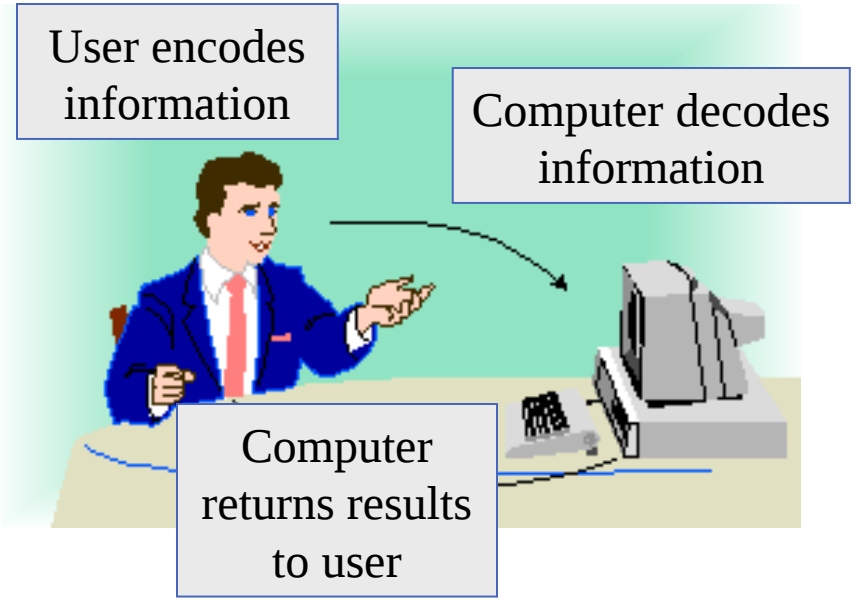
- One complete unit of communication.
  - An idea to be sent.
  - An encoder.
  - A sender.
  - A medium.
  - A receiver.
  - A decoder.
  - A response.



Communicating with a Computer

Substituting a computer for one of the people in the communication process.

- Process is basically the same.
  - Response may be symbols on the monitor.



Communicating with a Computer

Programming languages bridge the gap between human thought processes and computer binary circuit.

- **Programming language**: A series of specifically defined commands designed by human programmers to give directions to digital computers.
  - Commands are written as sets of instructions, called **programs**.
  - All programming language instructions must be expressed in binary code before the computer can perform them.

The Programming Language

In the beginning… To use a computer, you needed to know how to program it.

Today… People no longer need to know how to program in order to use the computer.

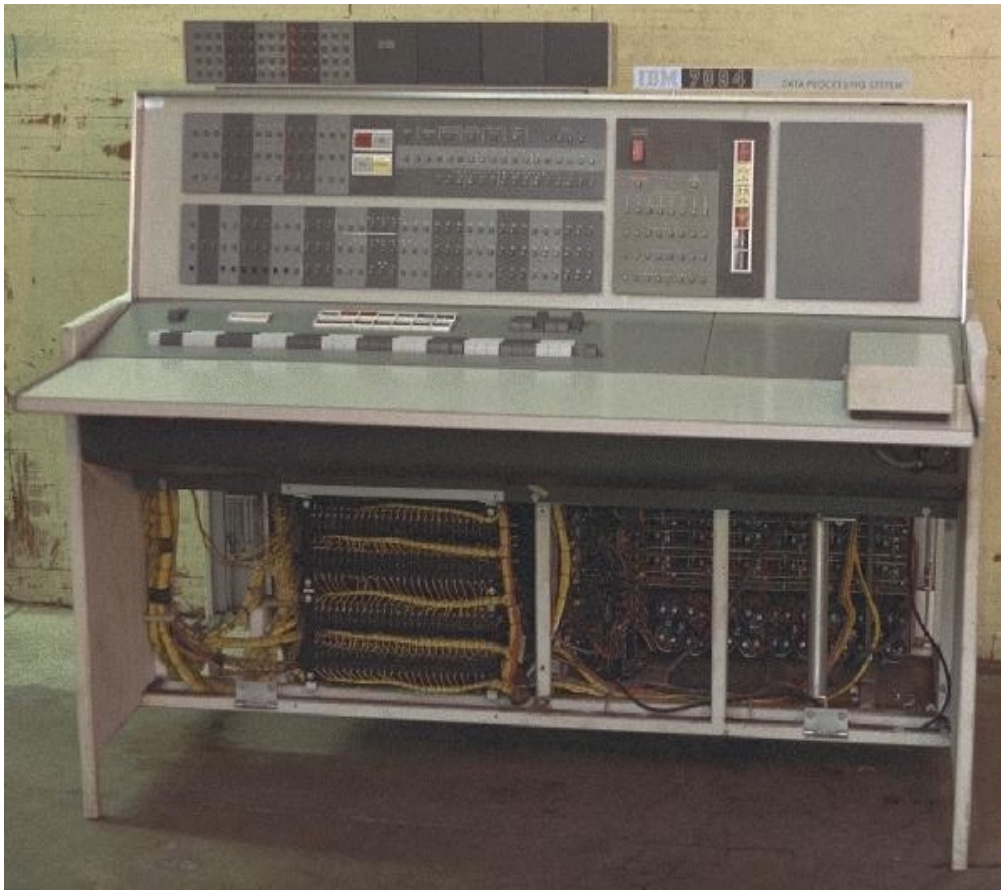
How this was accomplished:

- First Generation
  - – Machine Language (code)

The Programming Language

- Second Generation - Assembly Language

Second gen. computer



The Programming Language

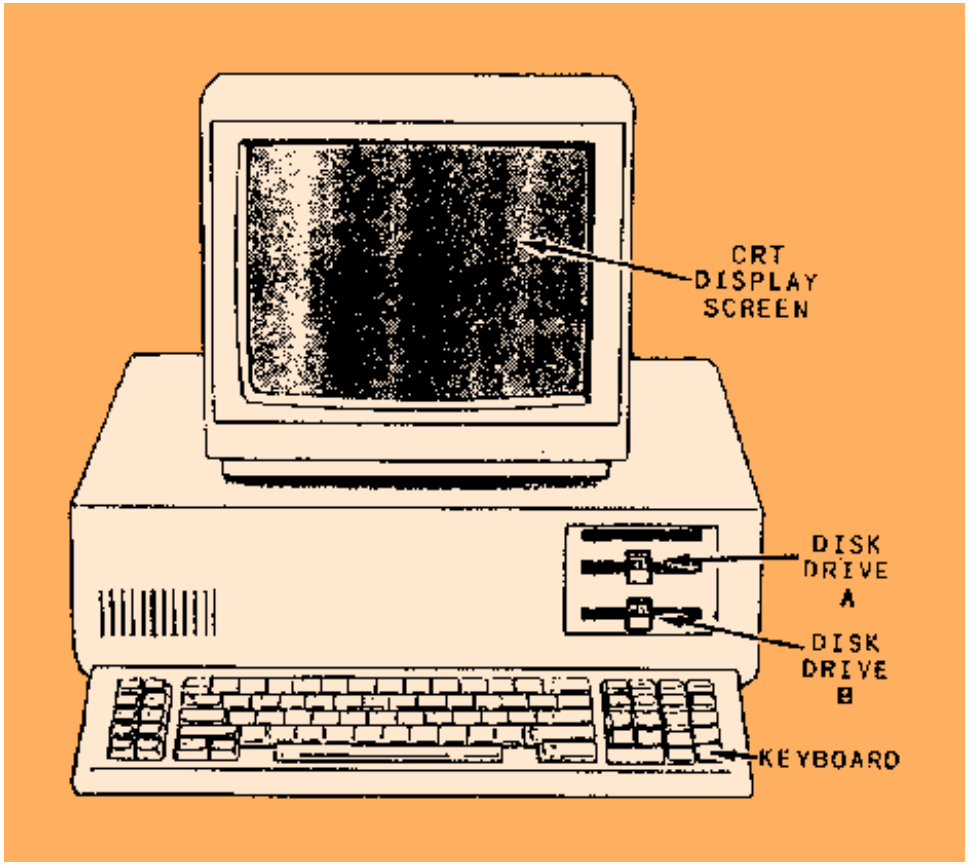
- Third Generation – People - Oriented Programming Languages

Third
Gen. Comp.

The Programming Language

- Fourth Generation - Non-Procedural Languages

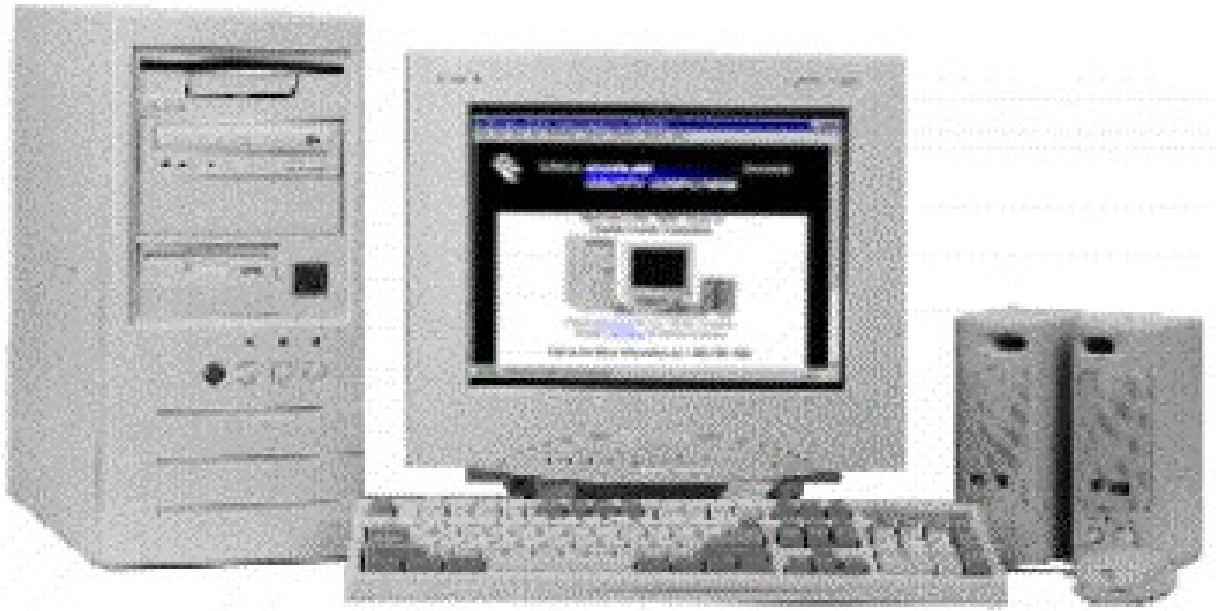
Forth gen. Comp.



The Programming Language

Fifth Generation - Natural Languages

Fifth gen. Comp.



The Programming Language

First Generation - Machine Language (code)

- **Machine language** programs were made up of instructions written in binary code.
  - This is the “native” language of the computer.
    - Each instruction had two parts: Operation code, Operand
    - **Operation code** (**Opcode**): The command part of a computer instruction.
    - **Operand**: The address of a specific location in the computer’s memory.
  - **Hardware dependent**: Could be performed by only one type of computer with a particular CPU.

The Programming Language

Second Generation - Assembly Language

- **Assembly language** programs are made up of instructions written in mnemonics.

```
READ    num1
READ    num2
LOAD    num1
ADD     num2
STORE   sum
PRINT   sum
STOP
```

- **Mnemonics**: Uses convenient alphabetic abbreviations to represent operation codes, and abstract symbols to represent operands.
- Each instruction had two parts: Operation code, Operand
- Hardware dependent.
- Because programs are not written in 1s and 0s, the computer must first translate the program before it can be executed.

The Programming Language

Third Generation - People-Oriented Programs

- Instructions in these languages are called statements.
  - **High-level languages**: Use statements that resemble English phrases combined with mathematical terms needed to express the problem or task being programmed.
  - Transportable: NOT-Hardware dependent.
  - Because programs are not written in 1s and 0s, the computer must first translate the program before it can be executed.

The Programming Language

Pascal Example: Read in two numbers, add them, and print them out.

```
Program sum2(input,output);
var
  num1,num2,sum : integer;

begin
  read(num1,num2);
  sum:=num1+num2;
  writeln(sum)
end.
```

The Programming Language

Fourth Generation - Non-Procedural Languages

- Programming-like systems aimed at simplifying the programmers task of imparting instructions to a computer.
- Many are associated with specific application packages.
  - Query Languages:
  - Report Writers:
  - Application Generators:

The Programming Language

- **Query Languages:**
  - Enables a person to specify exactly what information they require from the database.
  - Usually embedded within database management programs.
- **Report Writers:**
  - Takes information retrieved from databases and formats into attractive, usable output.
- **Application Generators:**
  - A person can specify a problem, and describe the desired results.
  - Included with many micro-computer programs (macros).

The Programming Language

Fourth Generation - Non-Procedural Languages (cont.)

- **Object-Oriented Languages**: A language that expresses a computer problem as a series of objects a system contains, the behaviors of those objects, and how the objects interact with each other.
  - **Object**: Any entity contained within a system.
    - Examples:
      - » A window on your screen.
      - » A list of names you wish to organize.
      - » An entity that is made up of individual parts.
  - Some popular examples: C++, Java, Smalltalk, Eiffel.

The Programming Language

Fifth Generation - Natural Languages

- **Natural-Language**: Languages that use ordinary conversation in one's own language.
  - Research and experimentation toward this goal is being done.
    - Intelligent compilers are now being developed to translate natural language (spoken) programs into structured machine-coded instructions that can be executed by computers.

Assembled, Compiled, or Interpreted Languages

All programs must be translated before their instructions can be executed.

Computer languages can be grouped according to which translation process is used to convert the instructions into binary code:

- Assemblers
- Interpreters
- Compilers

Assembled, Compiled, or Interpreted Languages

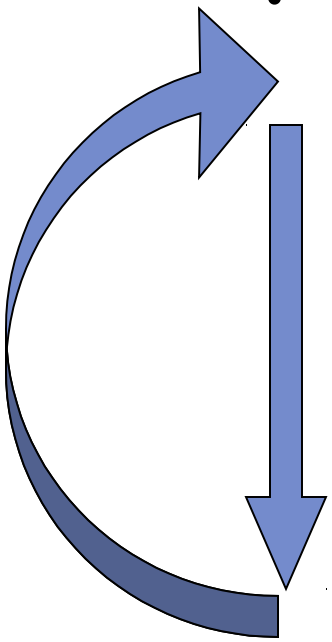
Assembled languages:

- **Assembler:** a program used to translate Assembly language programs.
- Produces one line of binary code per original program statement.
  - The entire program is assembled before the program is sent to the computer for execution.

Assembled, Compiled, or Interpreted Languages

Interpreted Languages:

- **Interpreter:** A program used to translate high-level programs.
- Translates one line of the program into binary code at a time:
  - An instruction is **fetched** from the original source code.
  - The Interpreter checks the single instruction for errors. (If an error is found, translation and execution ceases. Otherwise…)
  - The instruction is translated into binary code.
  - The binary coded instruction is **executed**.
  - The fetch and execute process repeats for the entire program.



Assembled, Compiled, or Interpreted Languages

Compiled languages:

- **Compiler:** a program used to translate high-level programs.
- Translates the entire program into binary code before anything is sent to the CPU for execution.
  - The translation process for a compiled program:
    - First, the Compiler checks the entire program for syntax errors in the original **source code**.
    - Next, it translates all of the instructions into binary code.
      - » Two versions of the same program exist: the original **source code** version, and the binary code version (**object code**).
    - Last, the CPU attempts execution only after the programmer requests that the program be executed.

Programming for Everyone

Several ways to control what your computer does or the way it accomplishes a particular task:

- Using Macros
- Using HTML to create Web Pages
- Scripting

Each allows customization of current applications.

Programming for Everyone

Using Macros

- **Macro**: Set of operations within the computer application that have been recorded for later execution.
  - The macro can be used repeatedly on any document within that application.
  - In word processors, macros are commonly used to speed up repetitive tasks.
    - Example: SIG can be stored as a macro that includes a signature message at the end of a document.

Programming for Everyone

Using HTML to create Web Pages

- HTML (HyperText Markup Language): A computer language consisting of special codes intended to design the layout (or markup) of a Web page.
  - Web browsers interpret the HTML code and display the resulting Web pages.
  - Web browser: A program that displays information from the WWW.
  - Each line of HTML is called a tag (formatting instruction).

Programming for Everyone

```
<HTML>
<HEAD>
<TITLE> Title of Web Page </TITLE>
</HEAD>
<BODY bgcolor=#ffffff text=#000000 >
<BODY>
<H1>
<CENTER> Sample Web Page
</CENTER> </H1>
<HR>
<A HREF="http://www.dogpile.com">
    dogpile search engine </A>
</BODY>
</HTML>
```

Designates an HTML document
Beginning of Header section
Contents of Title bar
End of Header section
Background=white, text=black
Top of the body of the document
H1=largest text size, H6 is smallest
CENTER turns on centering
Turns off centering and large text
Displays a horizontal rule: thin line
Links to the dogpile search engine
</BODY> and </HTML>designate the bottom of the document

Programming for Everyone

Scripting

- **Scripting**: A series of commands, written to accomplish some task.
  - Very similar to the concept of a program.
  - Extends the capabilities of the application where it is being used.
  - Examples of scripting languages:
    - Perl, C++, VBScript, JavaScript
    - **JavaScript**: A scripting language that allows the Web page designer to add functional features to a formatted web page created in HTML.

Building a Program

Whatever type of problem needs to be solved, a careful thought out plan of attack, called an algorithm, is needed before a computer solution can be determined.

1) Developing the algorithm.
2) Writing the program.
3) Documenting the program.
4) Testing and debugging the program.

Building a Program

1) Developing the algorithm.
   - **Algorithm**: A detailed description of the exact methods used for solving a particular problem.
   - To develop the algorithm, the programmer needs to ask:
     - What data has to be fed into the computer?
     - What information do I want to get out of the computer?
     - **Logic**: Planning the processing of the program. It contains the instructions that cause the input data to be turned into the desired output data.

Building a Program

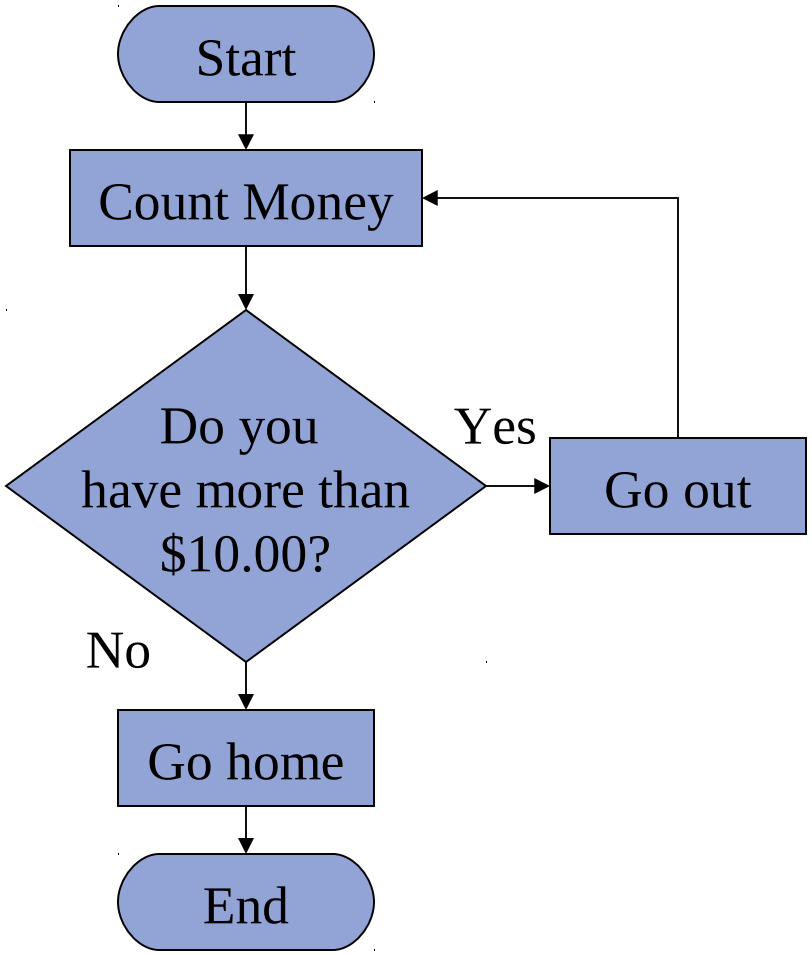
A step-by-step program plan is created during the planning stage.

The three major notations for planning detailed algorithms:

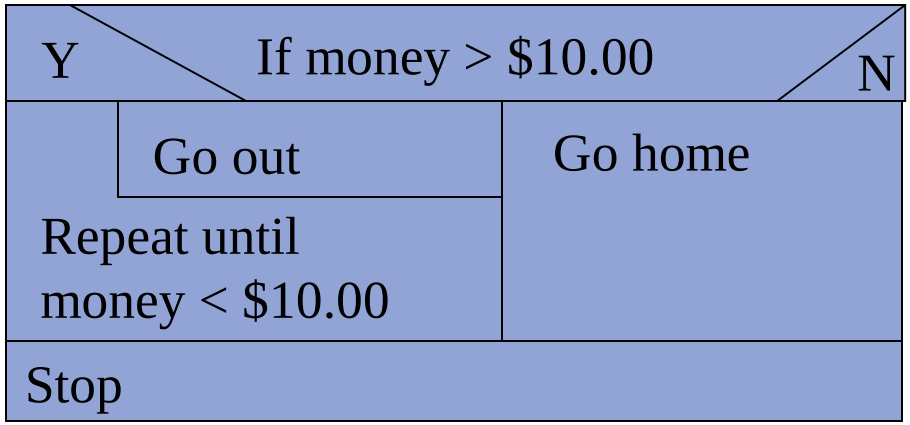
- **Flowchart**: Series of visual symbols representing the logical flow of a program.
- **Nassi-Schneidermann charts**: Uses specific shapes and symbols to represent different types of program statements.
- **Pseudocode**: A verbal shorthand method that closely resembles a programming language, but does not have to follow a rigid syntax structure.

Building a Program

Flow chart:



Nassi-Schneidermann chart:



Pseudocode:

1. If money < $10.00 then go home
   Else Go out
2. Count money
3. Go to number 1

Building a Program

2) Writing the Program

- If analysis and planning have been done, translating the plan into a programming language should be a quick and easy task.

3) Documenting the Program

- During both the algorithm development and program writing stages, explanations called documentation are added to the code.
  - Helps users as well as programmers understand the exact processes to be performed.

Building a Program

4) Testing and Debugging the Program.

- The program must be free of **syntax errors**.
- The program must be free of **logic errors**.
- The program must be **reliable**. (produces correct results)
- The program must be **robust**. (able to detect execution errors)

- **Alpha testing**: Testing within the company.
- **Beta testing**: Testing under a wider set of conditions using “sophisticated” users from outside the company.

Strokovne besede

Gap – vrzel

Consist - sestavljati

Hardware – strojna oprema

Computer languages – računalniški jeziki

Encoder – kodirnik

Sender – pošiljatel

Decoder – dekodirnik

Binary code – binarna koda

Simplifying – poenostavljanje

Strokovne besede

Programing – programiranje

Compilers – prevajalniki

Assemblers – monterji

Interpreters – tolmači

Charts – lestvice

Testing – testiranje

Debugging – odkrivanje napak

Algorithms – algoritem

Logic - logika

Strokovne besede

Macro – makro

HTML – programski jezik

Data base – zbirka podatkov

Fetched – prenesen

Errors – napake

Medium – medij

Response – odziv

Programmnig languages – programski jeziki

Computer - računalnik