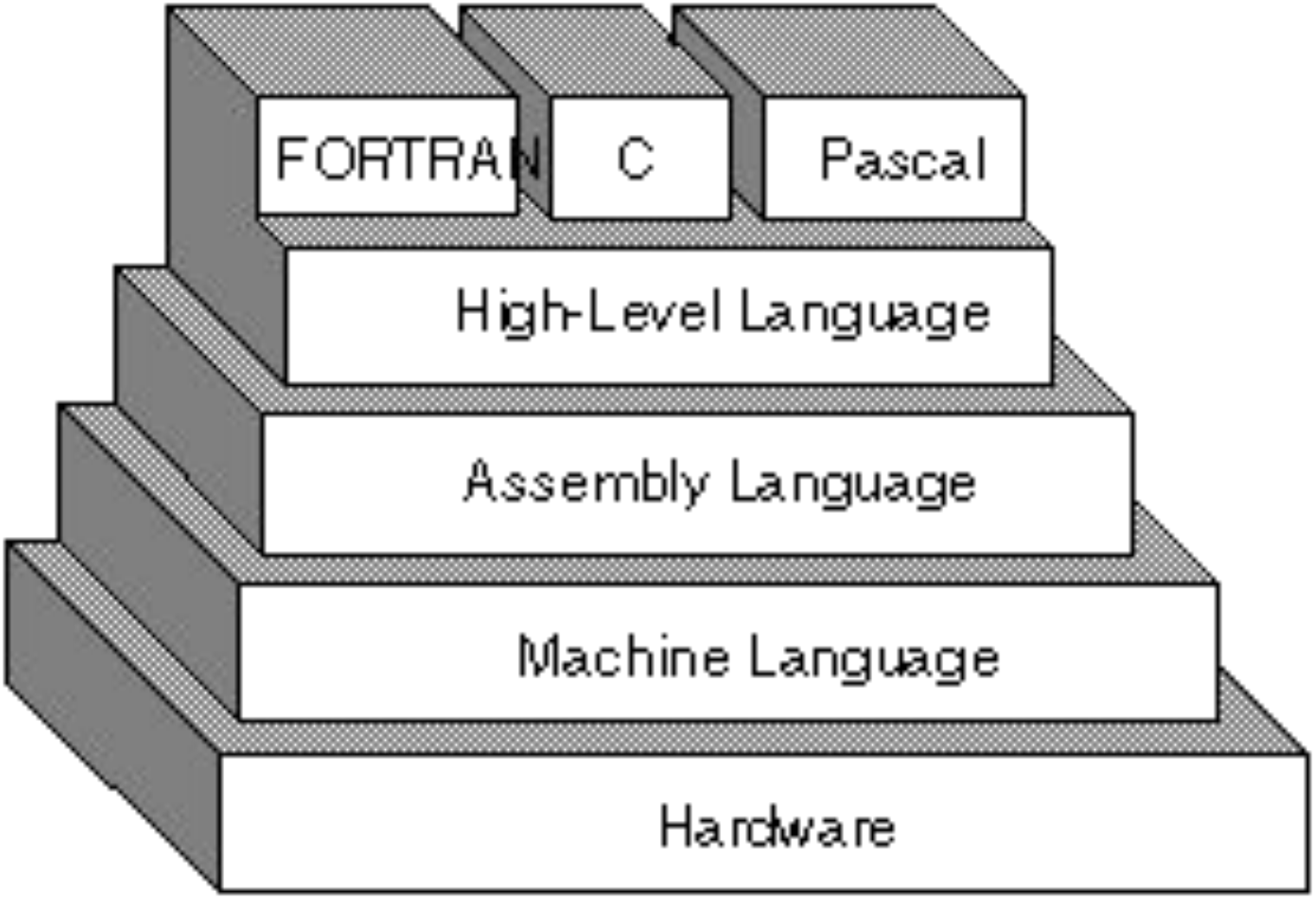
Family of Programming Languages


FORTRAN
C
Pascal
High-Level Language
Assembly Language
Machine Language
Hardware

Programming Languages

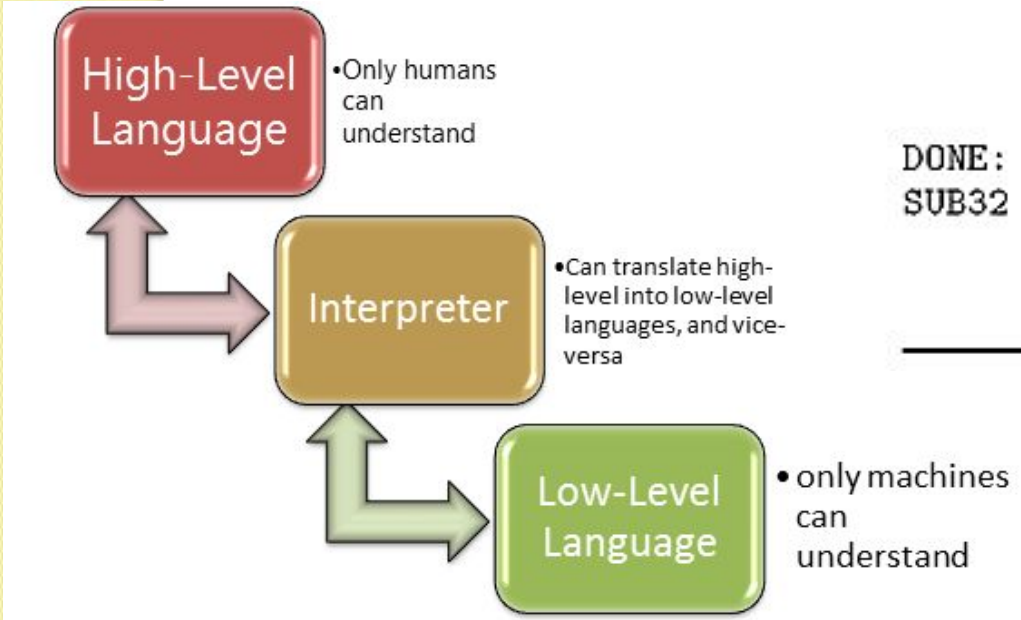
- When computers were first invented they had to be programmed in their own natural language, i.e. *binary* **machine code**.
- Later, programmers developed more sophisticated systems to make programming easier and more efficient**.**
- **Assembly languages** were developed as **the 2nd generation** of languages.
- **High-level languages** (3rd & 4th generation) later made programming even more productive.

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```

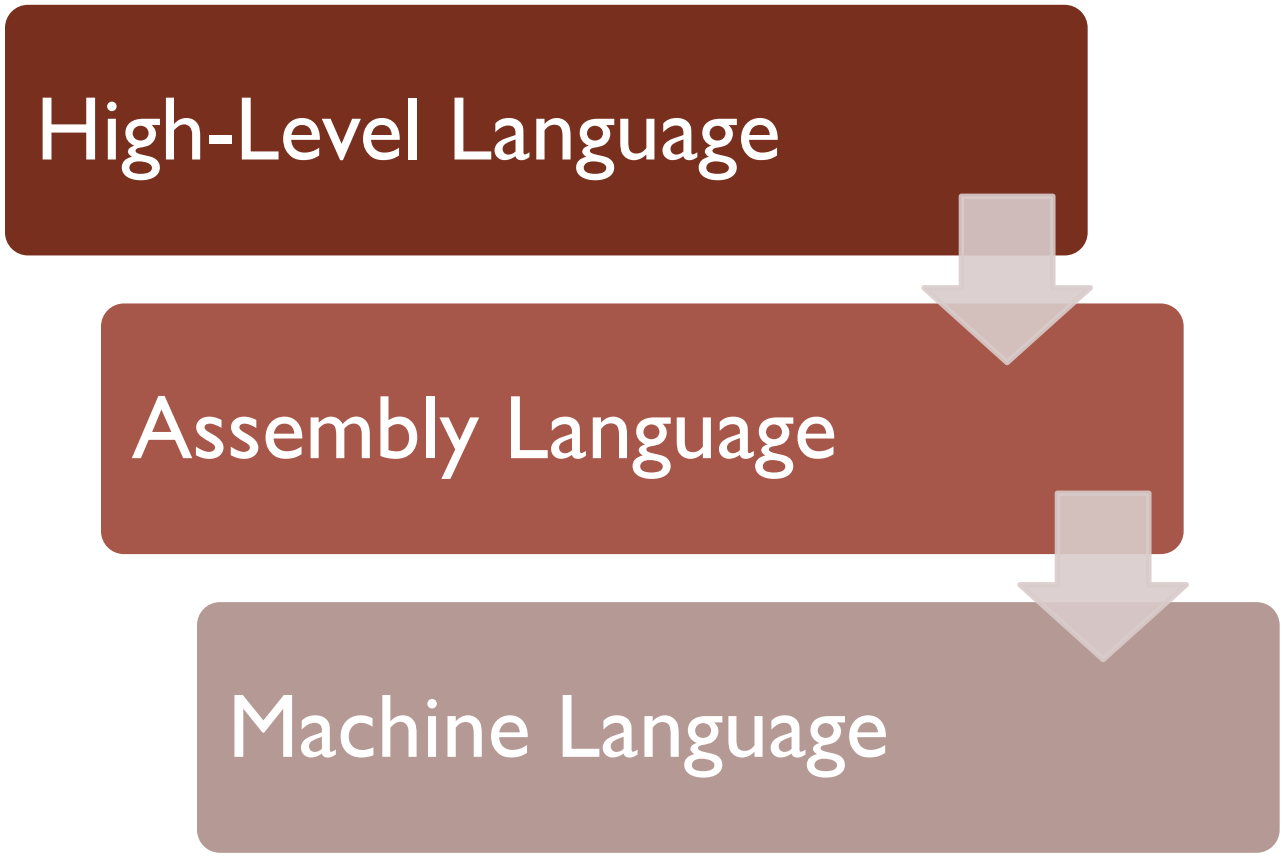
```
Example of IBM PC assembly language
Accepts a number in register AX;
subtracts 32 if it is in the range 97-122;
otherwise leaves it unchanged.

SUB32   PROC            ; procedure begins here
        CMP   AX,97     ; compare AX to 97
        JL    DONE      ; if less, jump to DONE
        CMP   AX,122    ; compare AX to 122
        JG    DONE      ; if greater, jump to DONE
        SUB   AX,32     ; subtract 32 from AX
DONE:   RET             ; return to main program
SUB32   ENDP            ; procedure ends here
```

FIGURE 17. Assembly language



The Generations



Machine Code

- These are the only kind of instructions the CPU can directly understand and execute.
- Stream of binary bit patterns that represent the instructions that are to be carried out.
- The binary bit patterns are decoded by the processor's logic circuits.
- They are then acted upon or executed, one after another.
- Machine code is a type of low-level code.

Machine Code

- Writing programs in machine code is difficult and time-consuming.
- It's difficult to remember all the bit patterns and what they do.
- Each operation of the processor has to be defined.
- Each machine instruction causes the processor to carry out just one operation.
- Nearly all machine code instructions consist of two parts:
  - An opcode, which tells the processor what to do
  - An operand, which tells the processor what to do it to.

Instruction Set

- This is the full set of machine code instructions, which the CPU "understands" and can execute directly without translation.

Assembly Languages

- The commands are still the basic CPU instruction set, but in an easier-to-read form.
- **One** assembly language instruction corresponds to **one** machine code instruction.
- Assembly language instructions have to be translated into machine code by an **Assembler** before the CPU can execute them.

Assembly Languages

- Low-Level Language
- As with machine language, each instruction causes just one processor operation
- These use **mnemonic instructions** (op-codes) instead of binary codes and hex or decimal in operands.
- These make programming less error prone and more productive than programming in pure binary code.

High-Level Languages

- High-Level languages do not have the same one-to-one correspondence between commands and machine code instructions as an assembler.
- A high-level command may represent several machine code instructions:
  - In a high-level language we can multiply two numbers together in one command.
  - At machine level this is not possible and it has to be done by repeated addition.

High Level Languages

- High level commands have to be turned into binary instructions the machine can understand; this is translation.
- There are two basic ways of translating high-level code
  - Complier: converts the whole code into machine code before running it
  - Interpreter: converts the code one instruction at a time, running each instruction before translating the next.

High-Level Languages

- Source code is the code written by the programmer.
- A compiler translates this source code into an object code in machine language. Object code runs independently of the source code and translator.
- An interpreter does not create object code so the source code has to be translated each time it is run.
- This means interpreted languages need the source code and translator present every time the code is run.

- https://habrahabr.ru/post/257331/