De peste un milion de ani, rasa umană este într-o continuă dezvoltare. Omul a parcurs intelectual, cognitiv și fizic un drum imens de la adăposturile din peșteră la civilizația modernă pe care o cunoaștem. Chiar dacă se poate discuta despre sensul și justificarea utilizării tuturor beneficiilor ce-i stau la dispoziţie omului modern, este evident progresul imens al speciei umane, ce se accelerează într-o proporție exponențială.

Primul milion de ani de existență, omul l-a petrecut proiectând și creând diferite instrumente și mașini, ce i-au permis să se miște mai repede și să realizeze mai mult. Astfel, omul a reușit să răstoarne munții, să îmblânzească râurile și să călătorească de la un capăt al planetei la celălalt în mai puțin de o zi. Se poate spune că omul a reușit, în felul acesta, să-și sporească puterea fizică de câteva mii de ori.

Ceea ce deosebește, totuşi, omul de alte specii sunt instrumentele pe care le-a dezvoltat pentru a-și extinde și spori puterea intelectului. Un prim astfel de „instrument” este limbajul (vorbirea și scrisul). Vorbirea a făcut posibil ca omul să transmită cunoașterea și articularea intelectului, iar scrisul a oferit o perspectivă asupra cunoștințelor care au fost formate și dezvoltate de zeci de generații anterioare. Rândurile pe care tocmai le citiți sunt un exemplu ideal al puterii cuvântului scris, care vă permite să aflați ceva ce nu ați ştiut până acum și, în felul acesta, să încercaţi să folosiți puterea intelectului în domenii complet noi.

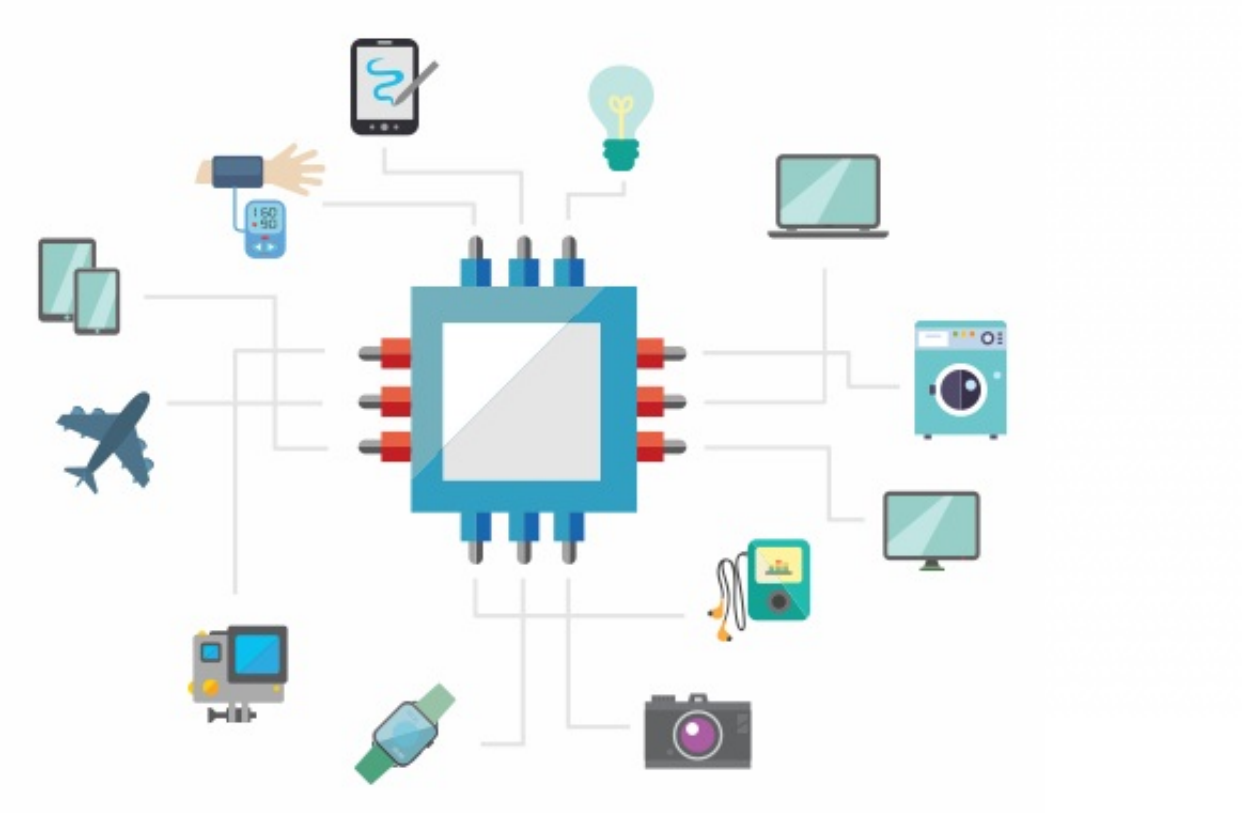
Articularea cuvintelor și apariția scrisului au permis omenirii să poată merge mai departe cu o viteză şi mai mare. Ultima dintr-o serie de realizări mari a speciei umane sunt calculatoarele – mașini care au permis omului să-și extindă puterea minţii până la limite inimaginabile.

Lumea calculatoarelor

Apariția calculatoarelor a schimbat complet lumea, cât și modul de viaţă al oamenilor. Calculatoarele sunt astăzi peste tot în jurul nostru, în desktop și laptop, telefoane inteligente, tablete, ceasuri inteligente, televizoare, mașini, avioane, mașini de spălat, aragaze, aparate de aer

condiționat, bancomate etc.

Majoritatea activităților din zilele de azi nu se pot realiza fără un calculator. Deşi poate nu sunteţi conștienți de acest lucru, un număr mare de operații zilnice obişnuite sunt în totalitate dependente de calculator. Avertismentul alarmei pe un ceas digital, încălzirea micului dejun într-un cuptor cu microunde, călătoria cu o mașină modernă la serviciu sau la școală, ridicarea banilor din ATM-uri, vizionarea filmului preferat la televizor – sunt doar câteva din exemplele celor mai comune activități care nu ar putea fi posibile fără existența calculatoarelor. Deci se poate spune, pe bună dreptate, că lumea de astăzi este lumea calculatoarelor.



Imaginea 1.1. Lumea calculatoarelor

Pe lângă activitățile zilnice, care sunt realizate în totalitate de calculatoare, nici celelalte domenii ale lumii moderne de astăzi nu pot

fi imaginate fără existența unui calculator. Astfel, practic, nu există niciun domeniu al tehnologiei, artei, științei, care să nu utilizeze calculatoare într-o măsură mai mare sau mai mică. Operația ochiului cu laser, proiectarea drumurilor, analiza probelor de sânge, tipărirea ziarelor, asamblarea autoturismelor pe o linie de producție – sunt doar câteva dintre exemplele de utilizare intensivă a calculatoarelor în toate domeniile de activitate.

Ce este un calculator?

O definiție comună a unui calculator ar fi – *calculatorul este un dispozitiv capabil să execute un număr arbitrar de* operații aritmetice și logice. Dacă această propoziţie se descompune în mai multe părți, se poate spune că un calculator este un dispozitiv care poate:

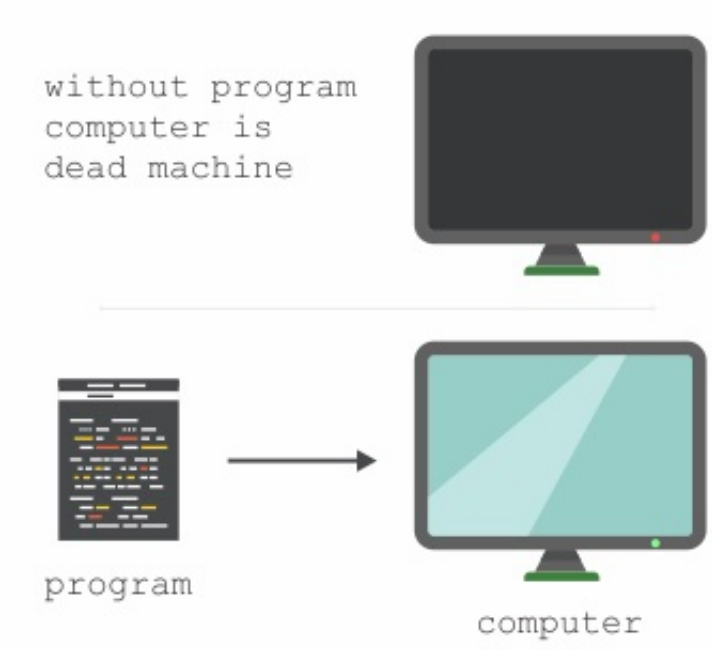
- accepta unele date de intrare;
- efectua prelucrarea datelor;
- produce o ieșire care poate fi înțeleasă de om sau de o altă mașină.

Deși noțiunea de calculator se identifică cu calculatoarele desktop și laptop, este important să înțelegem că, de fapt, calculatoarele există într-o gamă foarte largă de forme. Mii de calculatoare sunt ascunse în dispozitivele din jurul nostru, chiar dacă, în unele situații, nici nu suntem conștienți de acest lucru. Aceste calculatoare pot fi numite specializate, având în vedere că sunt destinate să efectueze operații stabilite în avans. De exemplu, calculatoarele din mașinile moderne sunt concepute pentru a controla funcționarea motoarelor și a celorlalte dispozitive. Astfel, calculatorul principal din mașină are rolul de a controla injecția de carburant, funcționarea supapei etc. Un astfel de calculator, desigur, nu poate fi folosit pentru a căuta pe internet. De aceea, nouă ne sunt mult mai familiare calculatoarele care pot fi numite universale. Acestea sunt concepute pentru a realiza o gamă largă de operații universale, care depind în exclusivitate de nevoile

utilizatorilor. Calculatoarele universale de astăzi există, în primul rând, sub formă de calculatoare desktop, laptop și tablete, precum și sub forma telefoanelor inteligente/smartphone-urilor. Astfel de dispozitive pot fi utilizate pentru a efectua o varietate de operații, de la simpla navigare pe internet, scrierea textului, vizualizarea audio și video, până la gestionarea și controlul altor tipuri de dispozitive digitale.

Ce sunt programele?

Pe lângă toate misterele ce înconjoară calculatoarele în general, este important să înțelegem că ele nu sunt un fel de mașini magice, adică nu pot face nimic singure. Fiecare calculator controlează un program, ce este de fapt un set de instrucțiuni care îi spun calculatorului ce trebuie să facă. Fără aceste instrucțiuni, calculatorul ar fi o mașină care nu ar avea nicio utilizare practică (imaginea 1.2.).



Imaginea 1.2. Programul inspiră viaţă calculatorului

Cu un set adecvat de instrucțiuni, integrate în cadrul programului, calculatorul poate deveni un instrument care poate fi utilizat pentru

comunicarea intercontinentală, lansarea unei rachete nucleare, cercetarea unor molecule noi care pot vindeca cele mai grave boli etc. Având în vedere faptul că calculatoarele se nasc ca dispozitive fără conștiință, este clar că în procesul de scriere a instrucțiunilor, combinate în interiorul programului, în calculatoare se inspiră viaţă. Acest proces se numeşte programare, ceea ce reprezintă și tema principală a acestui program complet.

Ce este programarea?

Ceea ce separă calculatoarele de celelalte mașini pe care omul le-a creat în timp este posibilitatea programării. În practică, acest lucru înseamnă că calculatorul, privit în general, este un instrument universal care, în funcție de instrucțiunile care îi sunt oferite, poate executa diferite operaţiuni. Tocmai în acest fapt se ascunde cea mai mare putere a acestor mașini – posibilitatea completă de programare.

Procesul de creare a programelor de calculator se numește **programare**. Astfel, programarea este un act de scriere a instrucțiunilor care permit calculatorului să efectueze o operație. Deși adesea se pot auzi și alte opinii, programarea este un proces foarte creativ, care cuprinde în mare măsură diferite aspecte ale artei, ingineriei și științei.

Cine sunt programatorii?

Programele de calculator sunt create de către programatori, oameni care dețin cunoștințele necesare pentru a putea scrie instrucțiuni ce dirijează calculatorul. Programarea modernă a calculatorului poate fi considerată o profesie relativ nouă, având o vechime de aproximativ 50 de ani. Totuşi, primele forme de programare au fost înregistrate mult mai devreme, în timpul existenței calculatoarelor mecanice, cu o construcţie complet diferită de cele pe care le cunoaștem astăzi.

Primul programator din lume se consideră a fi **Ada Lovelace** (imaginea 1.3.), fiica poetului Lord Byron.

Imaginea 1.3. Ada Lovelace, primul programator din lume[1]

În 1840, Ada Lovelace a primit să traducă un text de specialitate de la Charles Babbage, un designer pionier în domeniul calculatoarelor. Textul conținea o prezentare a designului și funcționării noului său calculator mecanic, care se numea Analytical Engine. În timp ce traducea acel text din limba franceză în engleză, Ada Lovelace a scris primul set de instrucțiuni, al căror scop era să permită compilarea numerelor lui Bernouli pe calculatorul mecanic. În cinstea acestei femei inovatoare, în 1980, Departamentul Apărării al Statelor Unite a numit noul limbaj creat pentru uz intern, Ada.

Limbajele de programare

Calculator, *program*, *programator* și *programare* sunt noţiuni interconectate, dependente. Printr-un proces numit programare, un programator creează un program care îi spune calculatorului ce trebuie să facă. Totuși, acestui proces, pentru a fi complet, îi lipseşte un

element. Este vorba de limbajul cu care programatorul se adresează mașinii în timpul procesului de creare a programului.

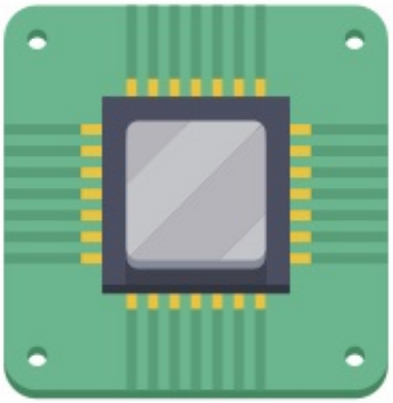
În viața de zi cu zi, oamenii comunică între ei prin folosirea unor limbi naturale (engl. *natural languages*), cum ar fi limba engleză, rusă, sârbă, română etc. Limba nu este nimic altceva decât un set prestabilit de reguli, care se aplică pentru a se articula gândurile și a le prezenta unei alte persoane. În același mod se realizează şi procesul de comunicare cu calculatoarele. Totuşi, calculatoarele nu înțeleg niciun fel de limbă naturală folosită de oameni, de aceea este necesar să ne adresăm calculatorului pe limba lui.

Componenta esențială a fiecărui calculator este unitatea de procesare centrală (*central processing unit, CPU*) sau, simplu, procesorul. În timpul funcționării lui, procesorul folosește și diverse alte componente care sunt partea componentă a calculatorului, dar, totuşi, procesorul este componenta cu care se comunică direct atunci când sunt transmise instrucțiunile pe care calculatorul trebuie să le execute. Astfel, comunicarea cu un calculator este de fapt o *discuţie* cu o unitate centrală de procesare.

Notă

CPU

Unitatea centrală de procesare este componenta esențială a fiecărui calculator. Ea efectuează operațiile de bază aritmetice și logice care sunt definite prin instrucțiuni în cadrul programului.



Imaginea 1.4. Unitatea centrală de procesare (engl. CPU)

De-a lungul timpului, designul procesoarelor de calculator s-a schimbat foarte mult, însă modul de bază al funcționării acestora a rămas neschimbat. Fiecare procesor efectuează ciclic operații aritmetice și logice cu operanzi. Astfel, procesoarele de calculator, pe lângă complexitatea structurală enormă, sunt proiectate să efectueze operații foarte simple, cum ar fi adunarea sau compararea a două numere. Totuşi, procesoarele moderne pot face milioane de astfel de operațiuni în doar o fracțiune de secundă.

Un procesor de calculator nu este în stare să înțeleagă limba pe care o folosesc zilnic oamenii. Mai mult decât atât, procesorul nu poate înțelege niciun fel de comandă mai complexă. Tot ce înțelege procesorul este o serie de biți, adică 0 și 1, care este compilată în anumite instrucțiuni. O astfel de serie de biți, care este interpretată în diferite comenzi, se numește limbaj mașină.

Limbajul mașină

Limbajul mașină este limba maternă a calculatorului, adică singura limbă pe care o înțelege calculatorul. Codul de programare al acestui limbaj este format dintr-o serie de 0 și 1, iar

fiecare dintre aceste cifre se numeşte **bit.**

Notă

Sistemul de numerație binar

În viața de zi cu zi, pentru a reprezenta numerele, se foloseşte sistemul zecimal, ce este format din zece cifre (1, 2, 3, 4, 5, 6, 7, 8, 9 și 0). Prin combinarea acestor zece cifre, se formează toate celelalte numere. Totuşi, pe lângă sistemul de numerație zecimal cu care suntem cu toții obișnuiți, în matematică mai există încă câteva sisteme de numerație, respectiv modalităţi în care pot fi prezentate numerele. Unul dintre aceste sisteme este deosebit de important pentru circuitele electronice digitale și calculatoare. Este vorba de sistemul de numerație binar.

În sistemul de numerație binar există doar două cifre: 0 și 1. Din acest motiv, adesea se spune că baza acestui sistem este 2, în timp ce baza, de exemplu, a sistemului zecimal este 10.

Sistemul de numerație binar este sistemul de bază pentru prezentarea datelor pe un calculator. Procesorul calculatorului gestionează doar datele binare, în timp ce același format se utilizează şi pentru stocarea datelor în cadrul oricărui tip de memorie a calculatorului. Utilizarea unui astfel de sistem de numerație în electronica digitală este condiționată de proprietățile elementelor în sine, din care sunt compuse componentele calculatorului, în special procesoarele. Procesoarele moderne sunt compuse din sute de milioane de elemente semiconductoare (tranzistoare), care se comportă ca întrerupătoarele şi pot avea două stări (on-off, true-false, zero-one). Tocmai datorită acestor proprietăți structurale ale elementelor calculatoarelor, electronica digitală se bazează pe un sistem de numerație binar.

Deși la prima vedere poate părea că sistemul de numerație binar este destul de limitat la existența a doar două cifre, acest lucru de fapt nu este aşa. Mai mult chiar, orice număr al sistemului de numerație zecimal poate fi prezentat ca binar, și invers.

Un număr prezentat în sistemul de numerație binar arată așa:

1101

Dacă aţi vedea numărul prezentat în afara contextului acestei lecții, probabil că l-ați citi ca o mie o sută unu. Datorită acestui fapt, adesea, atunci când se scriu numerele unor sisteme de numerație specifice, se foloseşte indicarea bazei de date pe care o utilizează un astfel de sistem de numerație. Pentru a specifica că numărul menţionat mai sus aparține sistemului binar, este necesar să se scrie:

$1101_{(2)}$

Așa cum puteți vedea, în interiorul subscript-ului se specifică baza sistemului de numerație care se foloseşte. Desigur, așa cum s-a menționat deja, pentru un sistem de numerație binar, baza este 2, deoarece există doar două cifre.

Numărul binar prezentat se poate compila foarte uşor într-unul zecimal, adică într-o formă care este mult mai ușor de înțeles pentru noi toți:

$1101_{(2)} = 1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 = 13$

Din calculul afișat, se poate înțelege cu ușurință logica folosită pentru conversia numerelor binare în numere zecimale. Fiecare cifră a numărului binar, începând din partea dreaptă, se înmulțește cu numărul 2, gradată în funcție de poziția pe care o astfel de cifră o deține în cadrul numărului binar complet. Astfel, prima cifră din exemplul prezentat se înmulțeşte cu 2^0, a doua cu 2^1, a treia cu 2^2 și așa mai departe. După efectuarea calculelor în modul ilustrat, se

determină că numărul binar 1101 din sistemul de numerație zecimal are valoarea 13.

Numerele sistemului de numerație zecimal pot fi, de asemenea, ușor convertite în binare. Pentru a realiza o astfel de conversie, se utilizează o logică ceva mai diferită, care implică împărțirea numărului zecimal cu doi și înregistrarea restului. Procesul complet de conversie a numărului zecimal 13 într-unul binar ar arăta așa:

13:2 = 6 şi restul **1**

6:2 = 3 şi restul **0**

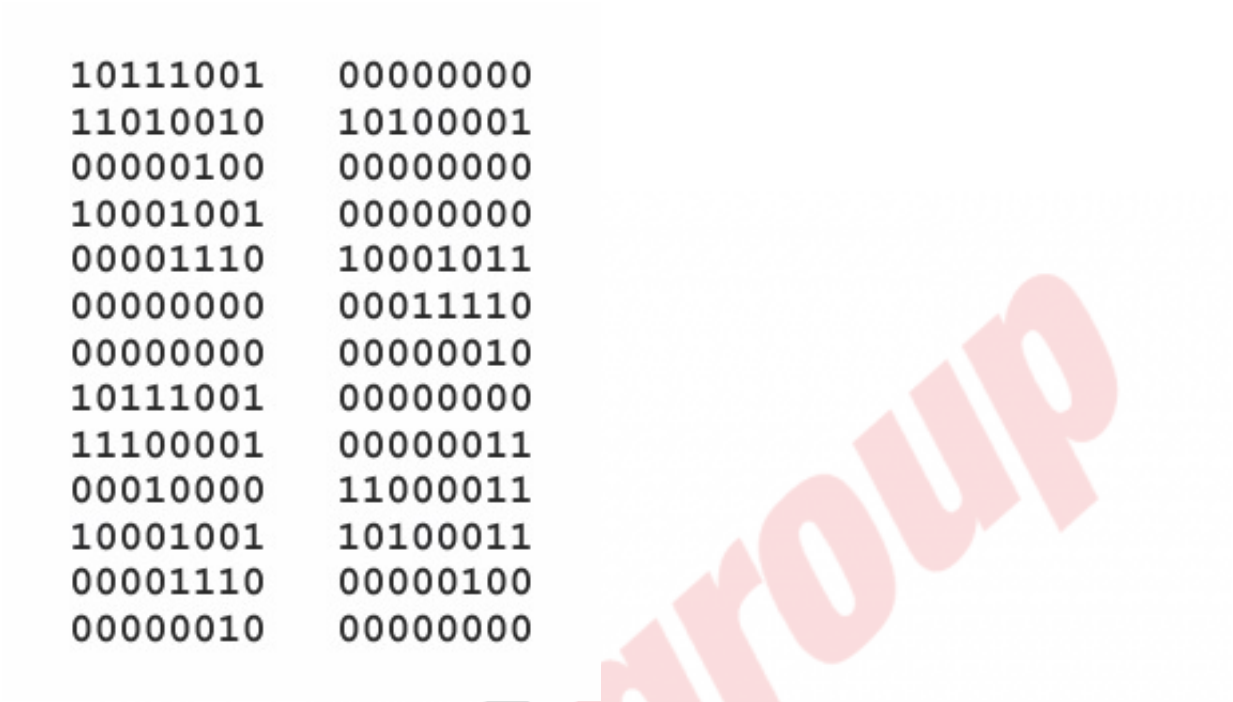
3:2 = 1 şi restul **1**

1:2 = 0 şi restul **1**

După cum puteți vedea din exemplu, numărul zecimal care trebuie convertit în număr binar se împarte cu numărul doi. Apoi, fiecare coeficient obținut se împarte cu numărul doi. Restul, atunci când se împarte orice număr cu numărul doi, poate fi 0 sau 1. Astfel, resturile rezultate din împărțire formează un număr binar. Valoarea numărului binar se pliază de la dreapta la stânga. În final, se poate scrie:

$1101_{(2)} = 13_{(10)}$

Folosirea limbajului mașină pentru a trimite instrucţiuni calculatorului, după cum puteți presupune şi singuri, poate fi o muncă foarte solicitantă și de lungă durată. De exemplu, pentru a se face o adunare simplă a numerelor 1234 și 4321 folosind limbajul mașină, este necesar să scrieți codul ilustrat în imaginea 1.5.



Imaginea 1.5. Program de adunare a numerelor 1234 şi 4321 scris în limbajul maşină

Setul de instrucțiuni prezentat, scris în limbajul mașină, reprezintă programarea la nivelul cel mai de jos posibil, adică adresarea directă către procesorul calculatorului. Seturile de biți prezentate, respectiv 0 și 1, reprezintă toate instrucțiunile necesare ce sunt trimise calculatorului pentru a face operațiile pe care le-a gândit programatorul. În acest moment, se poate pune următoarea întrebare: *cum a știut programatorul cum vor arăta comenzile pe care trebuie să le trimită calculatorului pentru ca el să facă adunarea celor două numere din exemplul prezentat?*

Răspunsul la această întrebare se află în **setul de instrucțiuni** (engl. *instruction set*). Fiecare procesor cunoaște un set predefinit de instrucțiuni, care este scris în hardware printr-o combinație specifică a unui tranzistor din interiorul unei părți a procesorului care este destinat instrucțiunilor de decodare. Dacă se trimite procesorului o instrucțiune pe care nu o știe, el, desigur, nu o va

executa. De aici se poate concluziona că limbajul mașină este complet dependent de procesorul pe care este executat. Acelaşi program scris cu un cod mașină nu se poate executa pe două procesoare cu proprietăți diferite.

Pentru ca programatorul să nu fie nevoit să scrie programul de mai multe ori, pentru fiecare procesor care există pe piață, cu timpul, au fost dezvoltate diferite arhitecturi de seturi de instrucțiuni (engl. *instruction set architecture, ISA*). **Arhitectura setului de instrucțiuni** este doar o abstracție, adică un contract sau un standard care definește ce instrucțiuni trebuie să cunoască calculatorul. Este vorba de unul dintre cele mai importante contracte din lumea calculatoarelor, care permite execuția unor programe identice pe o gamă largă de tipuri de calculatoare diferite. Existența arhitecturii seturilor de instrucțiuni, de exemplu, permite execuția unui program de calculator identic pe un computer cu un procesor 486, dar și pe un calculator modern cu un procesor Intel i7 de ultimă generaţie.

Notă

Arhitectura setului de instrucțiuni și microarhitectura

Având în vedere faptul că arhitectura setului de instrucțiuni este doar o abstracție, adică un standard, ea nu determină în niciun fel microarhitectura însăși, ce este modalitatea în care procesorul este proiectat intern, atunci când este vorba de componentele lui fizice. Această afirmaţie este susținută şi de faptul că același program de calculator se poate executa fără probleme şi pe un calculator AMD Ryzen, dar și pe un calculator cu un procesor Intel Kaby Lake. Chiar dacă există diferențe mari în microarhitectura dintre cele două familii de procesoare enumerate, ambele au aceeași arhitectură a setului de instrucțiuni – x86.

În rândurile anterioare, ne-am implicat profund în esența limbajului mașină și în modul în care procesorului i se trimit instrucţiuni în forma originală. Tot ce vă voi spune vă va ajuta foarte mult să înţelegeți

noţiunile care urmează. Ceva mai devreme, în imaginea 1.5., a fost prezentat un exemplu care ilustrează codul mașină pentru efectuarea adunării a două numere. Acum, imaginați-vă situația în care va trebui să scrieți un program ceva mai complicat sau chiar un program care să ruleze pe calculatoarele moderne, utilizând limbajul mașină. Pe bună dreptate se poate spune că o astfel de muncă ar epuiza şi pe cel mai tenace și cel mai calm programator, iar rezultatul ar fi, în mare parte, renunţarea. De aceea, primul pas în simplificarea lucrărilor de programare a venit în 1949, când au fost atribuite diferitelor comenzi binare, unele nume mai ușor de înțeles, ce reflectau scopul lor. În felul acesta, a fost creat limbajul de asamblare sau, prescurtat, assembler (engl. *assembly language* or *assembler*).

Asembler

Assembler-ul sau limbajul de asamblare, adesea abreviat ca asm, reprezintă limbajul de programare de nivel scăzut, în care comenzile programelor de bază sunt scrise folosind cuvinte predefinite care simbolizează comanda pe care o reprezintă. Astfel de construcții ale unui limbaj de asamblare sunt adesea numite şi simboluri, iar assembler-ul însuși, un limbaj simbolic. Folosind numele pentru scrierea comenzilor de program, limbajul de asamblare a permis o scriere a programelor mult mai ușoară, deoarece, utilizând assembler-ul, nu a mai trebuit să se memoreze seriile de 0 și 1 din care erau făcute comenzile. Totuşi, deși este mult mai ușor de înțeles, limbajul de asamblare funcționează la fel ca limbajul mașină, deoarece, între comenzile assembler-ului şi cele de maşină, există aproape întotdeauna un raport de unu la unu.

Operaţia de adunare a numerelor 1234 și 4321, a cărei execuţie a fost ilustrată anterior utilizând limbajul mașină, de data aceasta, folosind limbajul de asamblare, poate fi realizată ca în imaginea 1.6.

```
MOV CX,1234
MOV DS:[0],CX

MOV CX,4321
MOV DS:[2],CX

MOV AX,DS:[0]
MOV BX,DS:[2]

ADD AX,BX

MOV DS:[4],AX
```

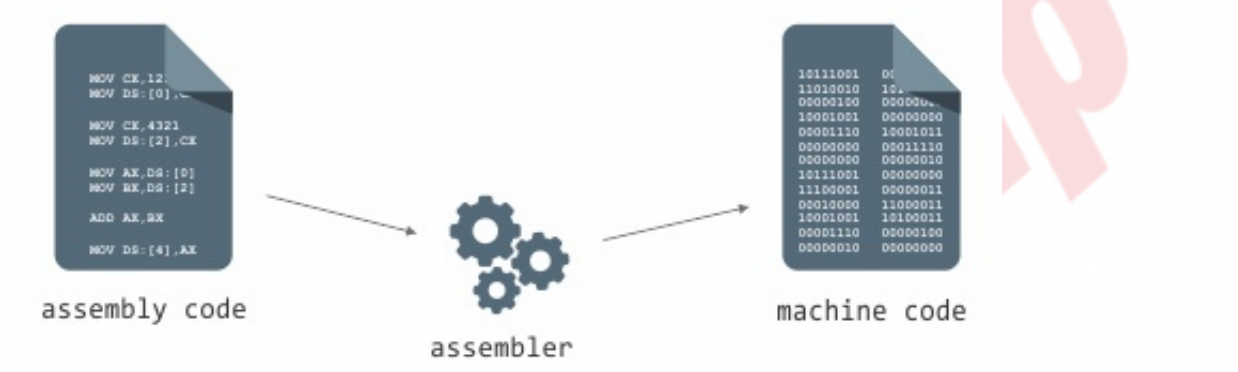
Imaginea 1.6. Program de adunare a numerelor 1234 şi 4321 scris în limbaj de asamblare

După cum puteți vedea, limbajul de asamblare este mult mai ușor de înțeles decât cel mașină. În loc de seturi de 0 și 1, acum sunt folosite pentru a fi trimise procesorului comenzi scrise cu cuvinte în limba engleză. De exemplu, `MOV` înseamnă a muta, respectiv scrieți, iar `ADD` a aduna. `AX`, `BX`, `CX` și altele, de asemenea se referă la locațiile de memorie, respectiv la registrele de procesoare utilizate pentru citirea și scrierea datelor. După cum puteţi concluziona şi singuri, apariția limbajului de asamblare a reprezentat un pas imens înainte pentru profesia de programator.

Mai devreme am spus că calculatoarele, respectiv unitățile de procesare, pot fi înțelese doar de limbajul mașină. În rândurile anterioare, am vorbit despre assembler, aşa încât apare următoarea întrebare - *cum a învățat calculatorul peste noapte un limbaj nou și a devenit capabil să înțeleagă, pe lângă cel mașină, și codul de asamblare?*

Desigur, calculatorul nu a învățat comenzile de asamblare, ci ele,

pentru ca procesorul să fie în stare să le înțeleagă, înainte de executare, le compilează într-un format de cod mașină. Fiecare comandă de asamblare este tradusă într-un set corespunzător de 0 și 1. Această muncă este efectuată de un program numit **assembler**. El este responsabil cu compilarea comenzilor de asamblare în cod mașină (imaginea 1.7.).



Imaginea 1.7. Compilarea unui cod assembler într-unul mașină

În imaginea 1.6., puteți vedea în ce mod programul de asamblare efectuează conversia instrucțiunilor scrise în limbajul de asamblare în codul mașină al procesorului. Este clar că magia completă a limbajului de asamblare este plasată într-o bucată de software numită assembler.

La fel ca limbajul mașină, assembler-ul este dependent în totalitate de setul de instrucțiuni pe care procesorul le suportă. Aceasta înseamnă că procesoarele arhitecturilor de instrucțiuni necesită existența diferitelor programe de asamblare.

Apariţia assembler-ului a ușurat foarte mult munca programatorilor. Totuşi, nu a fost suficient pentru ca munca de scriere a programelor de calculator să fie simplificată şi mai mult, mai rapid și mai standardizat. De aceea, abstractizarea a fost ridicată cu încă o scală, creând limbajele de programare de nivel înalt.

Limbaje de nivel înalt

Ambele tipuri de limbaje de programare se numără printre limbajele de nivel scăzut. Limbajul mașină și limbajul de asamblare sunt complet dependente de proprietățile fizice ale procesoarelor pe care se execută și necesită în mare măsură o cunoaștere a arhitecturii fizice a calculatorului de la programatori. Din acest motiv, limbajele de nivel scăzut sunt foarte pretenţioase de învățat și necesită mult timp pentru scrierea programelor. Tocmai de aceea, cu timpul, a fost creat un număr mare de limbaje de programare de nivel înalt, care prin abstractizarea detaliilor fizice ale calculatoarelor, au simplificat și accelerat semnificativ munca de scriere a programului.

Spre deosebire de limbajele de nivel scăzut, limbajele de programare de nivel înalt folosesc într-o măsură mult mai mare elementele limbii naturale, în general cuvinte și abrevieri preluate din limba engleză și, astfel, îmbunătățesc lizibilitatea și facilitează crearea codului de program. Pentru a înțelege în cea mai bună măsură avantajul limbajelor de nivel înalt, să vedem cum ar arăta un program simplu de adunare a două numere scris într-unul din limbajele de nivel înalt (imaginea 1.8.).

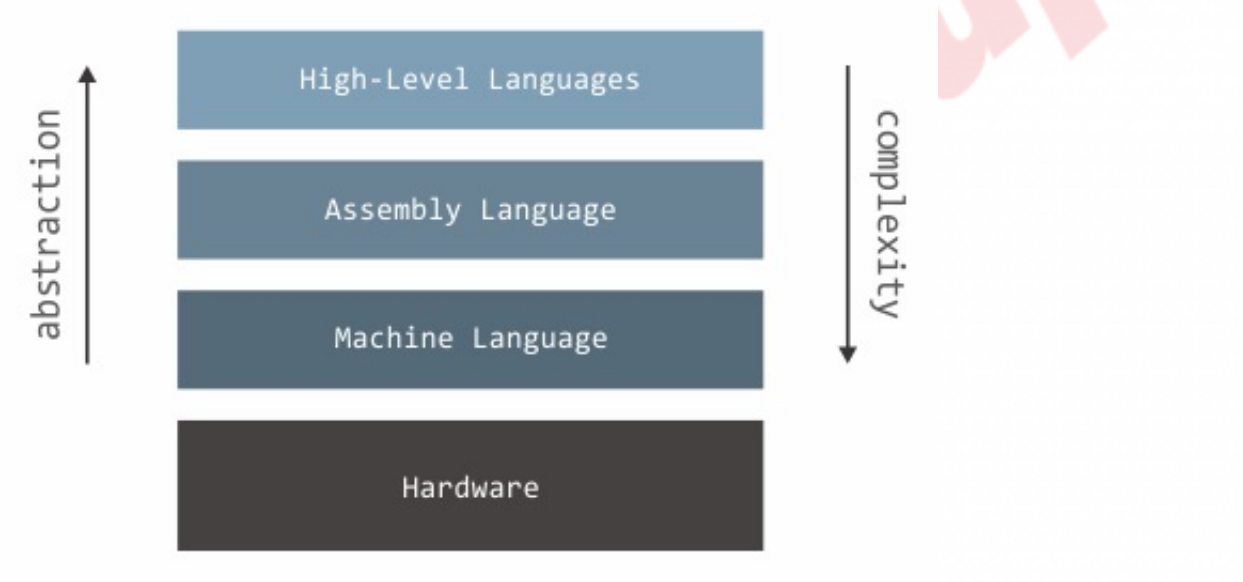
```
int a,b,c;

a = 1234;
b = 4321;

c = a + b;
```

Imaginea 1.8. Program de adunare a numerelor 1234 şi 4321 scris în C

Realizarea unei probleme identice, ce a fost ilustrată deja utilizând limbajul mașină și de asamblare, în exemplul prezentat s-a făcut folosind limbajul de programare C, unul dintre limbajele de nivel înalt. Ceea ce puteți observa imediat este codul de program semnificativ mai scurt, mai simplu și mai inteligibil.

Pe lângă reducerea complexității, limbajele de nivel înalt au şi sisteme diferite care eliberează complet programatorii de necesitatea de a gestiona anumite aspecte de execuție a programului. De exemplu, majoritatea limbajelor de programare de nivel înalt gestionează automat alocarea și eliberarea memoriei pe care programul o foloseşte în timpul execuției.

Raportul între calculatorul şi limbajul de nivel scăzut şi înalt poate fi ilustrat ca în imaginea 1.9.



Imaginea 1.9. Ierarhia diferitelor limbaje de programare

După cum puteți vedea în imaginea 1.9., odată cu trecerea de la hardware la limbajele de nivel înalt, se măreşte abstractizarea și se reduce complexitatea. Acest lucru înseamnă, practic, că limbajele de nivel înalt sunt caracterizate printr-un grad mai ridicat de abstractizare, iar limbajul mașină are o complexitate semnificativ mai mare.

În decursul timpului, a fost creat un număr mare de limbaje de programare de nivel înalt. Dacă începeți să învățați programarea de astăzi, aproape mai mult ca sigur că veți învăța unul dintre limbajele

de nivel înalt. De asemenea, diferitele limbaje de nivel înalt diferă între ele în mare măsură, în ceea ce privește posibilitățile, performanțele, sintaxa și nivelul de abstractizare. Unele limbaje de nivel înalt au şi posibilitatea programării low-level, permițând ca părțile unui program de calculator să fie scrise în limbajul de asamblare a unui anumit tip de procesor.

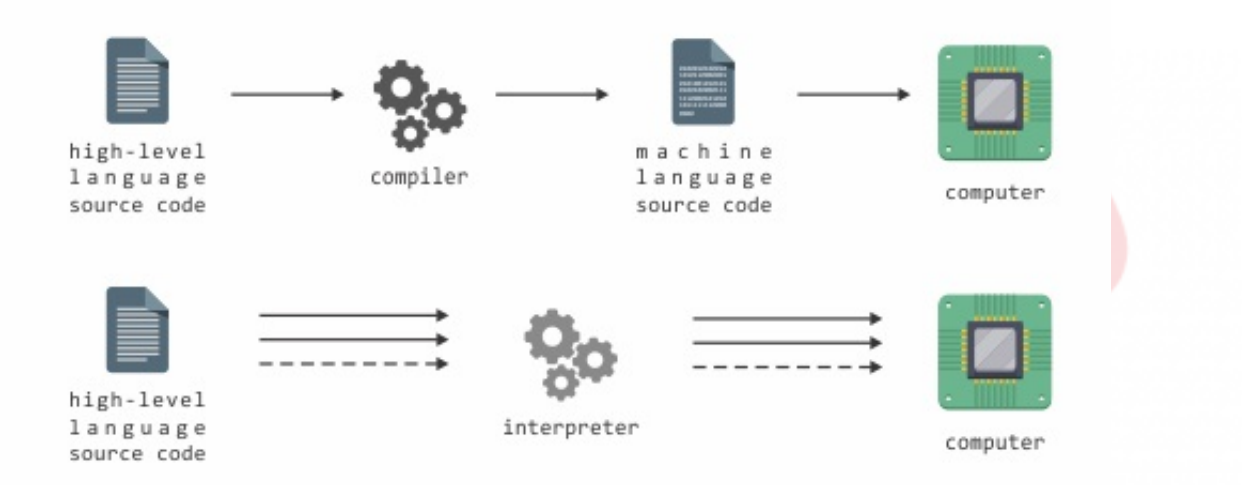
Toate avantajele pe care limbajele de nivel înalt le aduc trebuie să fie realizate într-un fel. Deoarece singurul limbaj pe care îl înțelege calculatorul este limbajul maşină, este clar că programele scrise în limbaje de nivel înalt trebuie să fie compilate într-un fel în limbajul mașină înainte să fie executate de calculator, aşa cum a fost în cazul codului de asamblare. Limbajele de programare de nivel înalt, pentru a efectua o astfel de operaţie, utilizează două componente foarte importante ale programării moderne - compilatoare și interpretoare.

Compilatoarele și interpretoarele sunt componente independente ale limbajelor de programare de nivel înalt care efectuează aceeași operaţie de compilare a codului limbajelor mai mari în limbaj maşină, în moduri diferite.

Compilatorul (engl. *compiler*) este un program de calculator al cărui scop principal este de a genera alte programe de calculator. Compilatorul compilează codul sursă al programelor scrise în limbaje de nivel înalt în limbaj mașină, pentru ca acesta să poată fi executat de calculator. Operaţia pe care compilatorul o face se numește compilare, iar limbajele pe care le foloseşte compilatorul, limbaje compilate. Operaţia de compilare se face dintr-odată, înainte ca un anumit program scris într-un limbaj mai mare să fie executat pe calculator. Rezultatul compilației este un fișier executabil cu conținutul scris în codul maşinii.

Interpretorul efectuează aceeași operaţie ca şi compilatorul, dar într-un mod ceva mai diferit. Spre deosebire de compilator, interpretorul efectuează compilarea unor părți mai mici ale codului sursă, iar acest lucru îl face în timpul execuției sale. Astfel, folosind interpretorul, nu este nevoie să se compleze întregul program, ci complarea se face în părți, atunci când se execută.

Diferențele dintre compilare și interpretare sunt ilustrate în imaginea 1.10.



Imaginea 1.10. Diferența dintre modul de funcţionare a compilatorului și a interpretorului

După cum puteți vedea, prin procesul de compilare se efectuează generarea unui fișier complet nou care conține codul complet al mașinii programului compilat. Odată compilat, programul se poate executa de un număr arbitrar de ori pe diferite calculatoare. Pe de altă parte, prin procesul de interpretare nu se produce codul mașinii care reprezintă programul complet ce trebuie executat. Interpretarea se face în mai multe etape, instrucțiune după instrucțiune, astfel încât codul mașină se generează din părți, care este executat de calculator, imediat după compilare.

Procesul de interpretare permite pornirea inițială mai rapidă a programului, deoarece nu se face dintr-odată compilarea întregului cod sursă. Pe de altă parte, odată ce se efectuează procesul de compilare a codului sursă al limbajului de nivel înalt în cod maşină, fiecare pornire și execuție ulterioară va fi semnificativ mai rapidă, deoarece programul există deja în formatul mașină. Procesul de compilare mai are încă un avantaj față de interpretare. Deoarece codul sursă complet se compilează înainte de execuție, compilatorul are posibilitatea de a

verifica codul sursă scris. În cazul în care codul sursă conține o eroare de sintaxă, compilatorul nu va putea să compileze un astfel de program și, în acest fel, programul nu se va executa. În felul acesta, se previne execuția programelor care au o eroare în codul lor sursă. Pe de altă parte, detectarea erorilor de sintaxă nu este posibilă în timpul procesului de interpretare, deoarece interpretorul compilează codul din părți, instrucțiune după instrucțiune, în timpul execuției acestuia. Dacă codul sursă care se interpretează conține o eroare, va ajunge cu siguranță la ea, în timpul execuției programului în sine.

Notă

Limbajul de programare este doar o specificație

Limbajele de programare moderne nu pot fi împărțite în compilate și interpretate. Deoarece un anumit limbaj de programare este doar o specificație, procesul concret de implementare a limbajului se determină în modul în care va fi compilat în limbajul mașină. Acest lucru înseamnă practic că aproape fiecare limbaj poate fi interpretat și compilat și că metoda concretă de compilare depinde de cel ce va crea implementarea limbajului concret. Limbajul de programare nu este nimic altceva decât un set de anumite reguli care au fost convenite în prealabil. Un astfel de set de reguli determină sintaxa limbajului, care în final definește aspectul și structura codului pe care îl vor scrie programatorii. Pentru ca limbajul de programare de nivel înalt să devină într-adevăr viu, este necesar să se scrie o componentă a programului care va face compilarea codului unui astfel de limbaj în limbaj mașină. Atunci când se implementează o astfel de componentă, este nevoie să se realizeze concret toate acele reguli lingvistice ce au fost convenite anterior prin specificaţie, într-un limbaj de programare unificat.

Compilarea limbajelor moderne de programare de nivel înalt se face adesea prin combinarea celor două tehnici descrise, respectiv prin combinarea compilării și interpretării. Implementarea unor limbaje, pe de altă parte, se bazează exclusiv pe una dintre cele două metode de

compilare amintite a codului sursă. De exemplu, limbajul de programare `C` a fost conceput, în primul rând, pentru a fi compilat, ceea ce este şi cel mai frecvent caz în implementarea lui. Aceeași situație există și în limbajele de programare `Objective C` și `Swift`, limbaje care sunt utilizate pentru a crea programe pentru dispozitivele companiei Apple. Pe de altă parte, un limbaj de programare foarte popular de nivel înalt, numit `JavaScript`, este un exemplu de limbaj de programare interpretat. Limbajele cum sunt `C#`, `Java` și `PHP` sunt, în primul rând, compilate, însă procesul de compilare a lor include și unele proprietăți de interpretare.

Limbaje de programare pentru manipularea datelor

Limbajele de programare care sunt menționate la sfârșitul capitolului anterior pot fi considerate **limbaje de uz general** (engl. *general-purpose programming languages*). Acestea sunt limbajele care pot fi utilizate pentru a rezolva o gamă largă de probleme. Pe lângă aceste limbaje, există şi limbaje specifice domeniului (engl. *domain-specific languages*). Aceste limbaje sunt destinate să rezolve un anumit set de probleme atunci când se creează programe de calculator. Un exemplu ideal de astfel de limbaje sunt limbajele care se folosesc astăzi intens pentru manipularea datelor. Cel mai popular limbaj de acest tip este SQL (engl. *Structured Query Language*). Astăzi, acest limbaj există sub forma unui număr mare de implementări diferite (`MySQL`, `Microsoft SQL`, `PostgreSQL`, `Oracle SQL`, `SQLite`), deci se poate spune că este unul dintre cele mai semnificative limbaje specifice pentru manipularea datelor. Nu există persoană care să se considere astăzi programator și care să nu cunoască bine acest limbaj.

Limbajele moderne de programare cu utilizare largă

Până acum au fost menţionate unele dintre limbajele de programare de nivel înalt care astăzi sunt utilizate pe scară largă. O sarcină foarte dificilă este măsurarea popularității unui limbaj, deoarece o astfel de caracteristică poate fi observată prin prisma multor parametri diferiți

(numărul programelor scrise, mărimea comunității de programe, numărul locurilor de muncă noi disponibile etc.). De aceea, deocamdată, ne vom abține de la speculații despre clasificarea celui mai popular și celui mai folosit limbaj de programare, pentru că acest lucru nu este posibil să se stabilească. Cert este că putem spune care sunt astăzi toate limbajele utilizate intensiv, fără intenția de a clasifica oricare limbaj în vreun fel. O prezentare generală a tuturor celor mai populare limbaje de astăzi se află în tabelul 1.1.

Limbaj de programare	Descriere
C	Strămoșul celor mai moderne limbaje, C este un limbaj de programare care și astăzi, la 45 de ani de la prima prezentare, este folosit în principal pentru programele low-level, care necesită un control cât mai bun posibil al hardware-ului dispozitivelor calculatorului; C a influențat în mare măsură majoritatea limbajelor de programare enumerate în acest tabel; astăzi este folosit în special pentru programarea sistemică la nivel scăzut; de exemplu, baza sistemului de operare Linux (Linux kernel), ce este scrisă folosind în totalitate limbajul de programare C; de asemenea, o mare parte a sistemului de operare Windows este scris în acest limbaj; creatorul limbajului de programare C este Dennis Ritchie.
C++	C++ este încă unul dintre limbajele de generație mai veche, care își găsește aplicarea în vremurile moderne; a fost creat după reputația limbajului C, cu câteva extensii pe care limbajul C nu le avea; astăzi, este folosit în diferite domenii de programare, de la programare la nivel scăzut, până la aplicații cu un mediu de utilizare grafic; totuși, datorită complexității lui, pentru programarea la nivel mai înalt, programatorii folosesc cel mai adesea alte limbaje disponibile; a fost creat în 1983 de către Bjarne Stroustrup.
C#	C# este unul din limbajele familiei .NET, care a fost creat de Microsoft; este vorba de un limbaj comun,

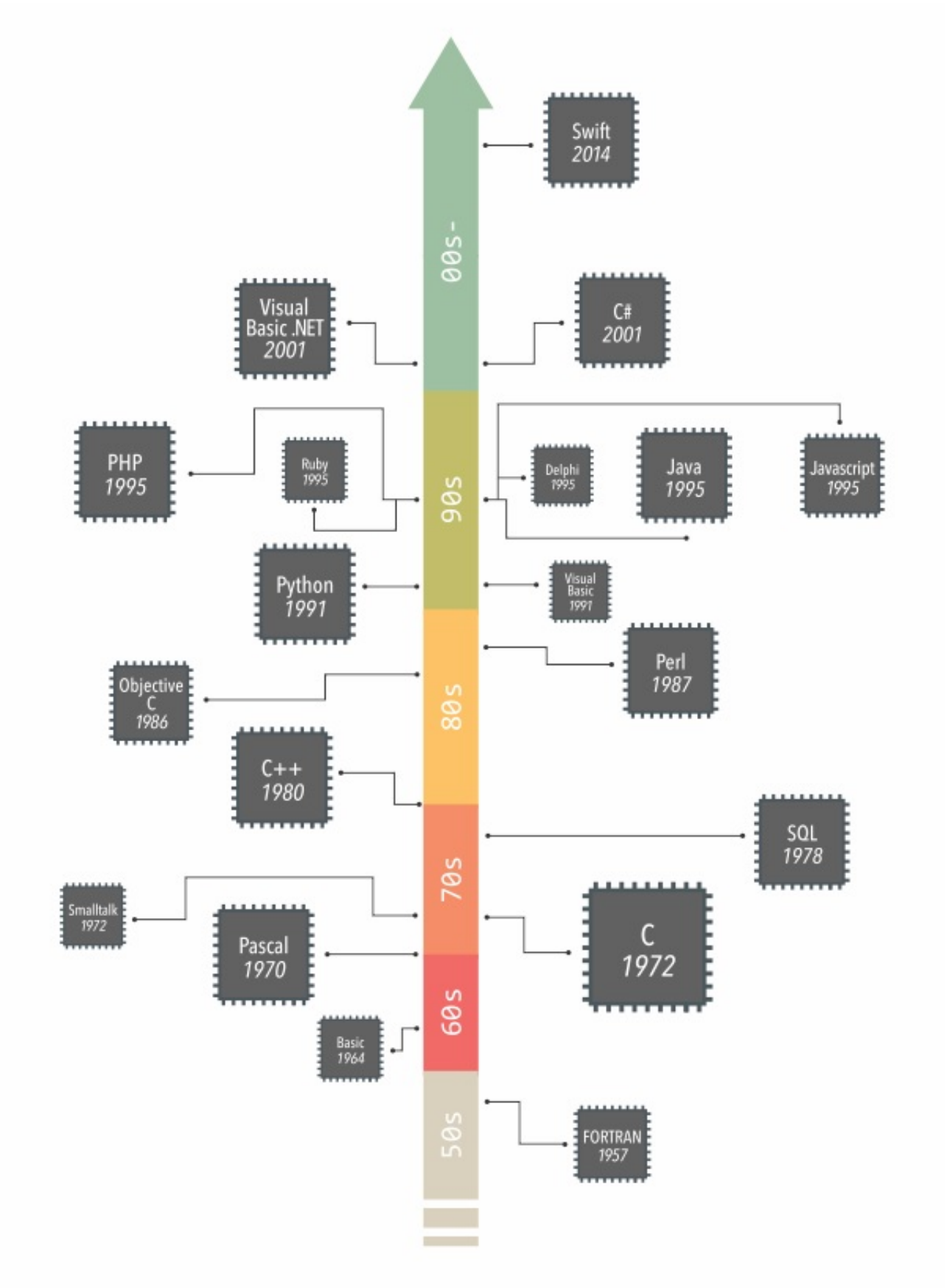
	care este utilizat astăzi foarte mult pentru crearea aplicațiilor desktop, web și mobile; a fost prezentat în anul 2000.
PHP	PHP este unul dintre cele mai populare limbaje de astăzi, dedicat în primul rând programării web; îl caracterizează o sintaxă simplă, astfel că aparține limbajelor care nu sunt dificil de învățat; este un exemplu ideal de creștere constantă a abstractizării programelor de calculator, adică o distragere tot mai mare a modului în care funcționează calculatorul, deoarece implementarea acestui limbaj este scrisă folosind limbajele C și C++; este un exemplu ideal de utilizare a limbajelor existente, pentru crearea unora noi de nivel mai înalt; a fost creat în 1995, iar creatorul său este Rasmus Lerdorf.
Java	Java, după mulți parametri, poate fi considerat cel mai popular limbaj de programare de astăzi; este destinat programării generale şi astfel poate fi folosit pentru a scrie aplicații pentru o gamă foarte largă de dispozitive; acesta este, de asemenea, şi cel mai mare avantaj al acestui limbaj, care a fost creat pentru a face posibilă execuția lui pe cât mai multe dispozitive de calcul diferite; totuși, astăzi Java se foloseşte cel mai des pentru programarea web și dezvoltarea aplicațiilor pentru dispozitivele alimentate de sistemul de operare Android; Java a fost creat în 1995, iar creatorul său este James Gosling și Sun Microsystems.
JavaScript	Limbajul de bază al programării client a aplicațiilor web, `JavaScript` este unul dintre blocurile de bază ale web-ului modern; se foloseşte pentru a face paginile web mai interactive și este, de asemenea, şi tehnologia de bază cu care se creează jocuri pe web și animație; este o parte integrantă a tuturor browserelor web moderne; pentru prima dată a apărut în 1995, ca parte integrantă a browserului web Netscape; creatorul lui este Brendan Eich.
SQL	Limbajul al cărui singur scop este de a manipula datele stocate în baze de date relaționale; astăzi există

	numeroase implementări ale acestui limbaj, create de companiile IT de top; este un limbaj care permite în primul rând citirea, scrierea, ștergerea și editarea datelor, dar și crearea de baze de date și tabele, precum și definirea unei logici simple după reputația limbajelor de programare generală; limbajul a fost creat în 1974 de către compania IBM.
Objective C / Swift	Limbaje folosite pentru crearea aplicațiilor pentru dispozitivele companiei Apple; Objective C este limbajul care inițial a fost utilizat pentru a programa aplicații pentru sistemele de operare OS X și iOS; Objective C a fost creat de Tom Love și Brad Cox în 1984, prin extinderea limbajului C, a unor funcții orientate pe obiecte ce au fost preluate din limbajul de programare Smalltalk; astfel, Objective C a fost creat prin combinarea acestor două limbaje; baza limbajului este absolut identică cu cea a C-ului, în timp ce anexa sub formă de suport orientat pe obiecte este absolut identică cu această funcție în limbajul Smalltalk; limbajul de programare Swift a fost prezentat în 2014 de către compania Apple ca o alternativă a limbajului Objective C; limbajul Swift se caracterizează prin folosirea unor abordări mult mai moderne, precum și a unei sintaxe simplificate.
Python	Python este un limbaj de programare cu scop general, ale cărui caracteristici de bază sunt simplitatea sintaxei, lizibilitatea și concizia; acest lucru practic înseamnă că, utilizând acest limbaj, unele probleme se pot rezolva cu mult mai puține linii de cod decât în cazul altor limbaje; acest limbaj se foloseşte în special pentru programarea web, ca alternativă a limbajelor PHP, Java sau C#; este un exemplu clasic al limbajului al cărui cod sursă se interpretează; a apărut pentru prima dată în 1991, iar creatorul lui este Guido van Rossum.
Perl	Un alt limbaj cu scop general, care se poate folosi pentru a scrie diferite tipuri de software; este unul dintre limbajele interpretate, iar interpretorul său

	poate fi integrat într-un număr mare de sisteme existente; cea mai mare aplicație o are pe web și este cunoscut în special pentru capacitățile sale bogate de procesare a textului; a fost creat în 1987 de Larry Wall.

Tabelul 1.1. Cele mai populare limbaje de programare de nivel înalt

Pentru a recapitula tot ce s-a spus în această lecție despre limbajele de programare, este dată imaginea 1.11., ce ilustrează evoluția lor în timp.


00s-
90s
80s
70s
60s
50s
Swift 2014
Visual Basic .NET 2001
C# 2001
PHP 1995
Ruby 1995
Delphi 1995
Java 1995
Javascript 1995
Python 1991
Visual Basic 1991
Perl 1987
Objective C 1986
C++ 1980
SQL 1978
Smalltalk 1972
Pascal 1970
C 1972
Basic 1964
FORTRAN 1957

Imaginea 1.11. Cronologia celor mai importante limbaje de programare de nivel înalt

Începutul unei călătorii captivante

Citind această lecție, puteţi anticipa doar o parte din emoţiile pe care le ascunde lumea programării. Desigur, după atâta cantitate de informații, probabil vă simțiți confuz. Un astfel de sentiment este complet normal și caracteristic pentru oricine intră în labirintul lumii programării. Totuşi, nu există motive de îngrijorare, căci acest program a fost conceput astfel încât să permită familiarizarea cu lumea programării într-o manieră sistematică și detaliată. Captivanta călătorie poate să înceapă!

---

1. Sursa: https://en.wikipedia.org/wiki/Ada_Lovelace