Računalni programi i programski jezici

Eugen Rožić

ZTŠ Rudolf Perešin

Uvod

- Procesor izvodi (izvršava) naredbe (instrukcije) nad podatcima pomoću ostalih dijelova računala (memorije, ulazno/izlaznih jedinica, …)
- **Program** je niz naredbi (skupa s nekim podatcima) koji služi nekoj konkretnoj svrsi, dakle proizvest će neku konkretnu i smislenu posljedicu/efekt na računalu kada ga procesor izvrši
- Svaki program je *napisan* u nekom **programskom jeziku** – to se zove **izvorni kod**
- Procesor zna izvršavati samo programe, odnosno instrukcije napisane u **strojnom jeziku** (to je „jezik" procesora)
- Dakle, da bi procesor mogao izvršiti neki program on mora biti *preveden* u strojni jezik – to se zove **izvršni kod** (*executable* → .exe)

Programski jezik

- Programski jezici su formalni jezici, kao i ljudski jezici
  - Skup znakova, riječi i pravila za stvaranje (smislenih) tekstova, odnosno programa
    - **Sintaksa** – sastavljanje formalno ispravnih „rečenica” (naredbi), npr.
      - Hrvatski: „Ovo predavanje je baš zanimljivo.”,
        naspram „Ovo predavati biti zanimljivost.”
      - Python: „if rijec in rjecnik: print(rijec)
        else: print(„Ta riječ ne postoji!”)”
    - **Semantika** – smisao i smislenost „rečenica” (naredbi), npr.
      - Hrvatski: „Ovo predavanje je baš zanimljivo.”,
        naspram „Ovo predavanje je savršeno mokro i žarko crno.”
      - Python: „if rijec in rjecnik: print(book)
        else: cry()”

Strojni jezik

- Jezik procesora, ono što procesor „razumije”, dakle može izravno izvršavati
  - Zapisan samo pomoću nula i jedinica:
    - Kodovi za pojedinu elementarnu instrukciju (naredbu) procesoru, npr.
      - dodavanje, oduzimanje i ostale aritmetičke i logičke operacije,
      - spremanje i učitavanje iz memorije, …
    - Točne memorijske lokacije podataka zapisane binarno, itd.
  - Duljina jedne naredbe/instrukcije u bitovima → arhitektura (npr. 32 ili 64 bita)
- Svaki procesor (ili familija procesora) ima svoj strojni jezik
  - Zato se **izvršni kod programa** za jedan model/tip/klasu/familiju procesora (i operacijski sustav) ne može izvršavati na drugima…
- Ljudima je (praktički) nemoguće pisati u strojnom jeziku (bušene kartice, daaaavno), zato je potrebno imati druge jezike i posebne programe - **prevoditelje**

Zbirni jezik (*assembler*)

- Strojni jezik, ali zapisan ljudima razumljivijim znakovima:
  - riječi/kratice umjesto binarnih kodova za elementarne instrukcije, npr.
    - add, sub, mul, and, or, save, load, …
  - imena (labele) za pojedine memorijske lokacije i naredbe u programu kako bi čovjek lakše pratio što je gdje i razmišljao dok programira, itd.
- Također specifičan za svaki procesor, odnosno familiju procesora
- Prevođenje na strojni jezik (u **izvršni kod**) je trivijalno, odnosno izravno
  - Potrebne samo „tablice”, nikakva interpretacija
    - engl. *assemble* = sastaviti (od gotovih dijelova)
  - Ista logička **razina apstrakcije**, samo je jezik „jednostavniji” (ljudima)

Prevođenje programskih jezika

- Pretvaranje naredbi (teksta) programskog jezika u strojni kod
- Program koji to radi – prevoditelj (*compiler*)
  - engl. *compile* = sastaviti (iz informacija koje treba sakupiti i prilagoditi)
- Dakle, kada napišete neki program u nekom programskom jeziku morate ga *kompajlirati*
  - To znači „pozvati” prevoditelja za programski jezik u kojem ste napisali vaš program (i za ciljani model/tip/klasu/familiju procesora i operacijski sustav) nad tekstualnom datotekom koja sadrži vaš program (**izvorni kod**)
  - Rezultat toga je izvršna (executable -> .exe) datoteka, sa **izvršnim kodom**
- …ali, neki jezici ne funkcioniraju baš skroz tako
  - U principu je isto (jer u konačnici mora biti), ali je taj proces malo „sakriven” od korisnika/programera i izvodi se u drugo vrijeme i na malo drugačiji način

Viši programski jezici

- Svi oni jezici koji su na **višoj razini apstrakcije** od strojnog/zbirnog jezika
  - Dok programirate ne morate znati i nije vas briga na kojem i kakvom će se procesoru taj vaš program izvoditi nego računate na prevoditelja da će odraditi taj posao za vas
    - Jasno onda da će izvršna (.exe) datoteka koju prevoditelj stvori biti izvršiva samo na jednom modelu/tipu/klasi/familiji procesora (i operacijskom sustavu)
  - Znakovi, riječi i pravila bit će bliži ljudskom govoru, ljudskim jezicima i ljudskom logičkom razmišljanju
    - Bogatiji „vokabular” sa kompleksnim operacijama sažetima u jednu riječ
    - Primjeri: if, then, else; for, while; try, catch, finally; define, make, create, connect, …
  - Tehnički detalji toga kako će se te naredbe točno izvesti na procesoru su „sakriveni” i programer se njima ne zamara
    - Odnos efikasnosti u programiranju i u izvođenju (cijena rada naspram cijene opreme)

Viši programski jezici

- Uz apstrakciju jezika samog procesora i tehnologije računala na kojem će se program izvršavati dolazi do veće općenitosti i zajednice ljudi koji koriste neki jezik
  - Velike količine **biblioteka** gotovih programa, podprograma i drugih komada koda specifičnih namjena koji su iskoristivi
    - Nije nužno da programer sve piše sam iz nule, uzima gotova rješenja drugih za dijelove svojeg problema
  - Ponovno – odnos efikasnosti programiranja i izvođenja
    - Apollo - 145,000 LOC (lines of code) na 2Mhz, 4KB RAM, 72 KB ROM
    - Android – 15,000,000 LOC na 4/8 2+Ghz jezgre, 4GB RAM, 64+ GB flash
  - Sigurnost korištenja provjerenog i standardiziranog koda…

Primjeri (prevođenih) jezika

- **C** (1972; .c, .h)
  - Jako „blizu” računalnoj logici, niska razina apstrakcije, ali ipak jako moćan
  - Koristi se kada program treba biti maksimalno efikasan ili brz uz ograničene računalne resurse – znanost, ugradbeni sustavi (mikrokontroleri), operacijski sustavi, …
- **C++** (1985; .cpp, .hpp)
  - Srodan C-u, ali sa puno proširenja, za naprednije i kompleksnije primjene, ali isto kada je potrebna efikasnost i brzina
  - Grafika (igrice), kompleksne aplikacije sa grafikom i komunikacijom (npr. browseri), …
- **Basic** (1964), **Pascal** (1970)
  - Bivši pedagoški standardi, danas rijetko korišteni…
- **Fortran** (1957), **Cobol** (1959), **Go** (2009), **Rust** (2010) i mnogi mnogi mnogi drugi…

Interpretirani jezici

- Primjeri: **Python** (.py), **Ruby** (.rb), **PHP** (.php), …
- Jezici koji se ne *kompajliraju* – nema izvršne datoteke nego samo datoteka sa izvornim (*source*) kodom
  - Problem je što ne možete „sakriti” i intelektualno zaštititi izvorni kod
- Nad datotekom s izvornim kodom se pozove **interpreter**
  - Program koji u stvarnom vremenu prevodi izvorni kod i stvara izvršni kod, ali ga nigdje ne sprema za stalno, nego svaki put to radi iznova (nema .exe)
- Jezici za brzo programiranje jednostavnih stvari (tzv. skriptni jezici), tako je počelo barem
  - skripta = kratki programčić za neku specifičnu, zasebnu namjenu
- Danas se koriste puno za web development

Java – poseban slučaj

- Problem pisanja i/ili kompajliranja programa za sve moguće razne procesore, arhitekture računala i operacijske sustave, umjesto samo jednom
- Kako to Java rješava:
  - Izvorni kod (.java datoteke) se ne prevodi u izvršni kod nego u *bytecode* (.class, .jar datoteke)
  - *Bytecode* se izvodi na Java virtualnoj mašini (JVM)
    - JVM je program koji je kao interpreter za *bytecode* na računalu na kojem se nalazi
  - Dakle, da bi pokrenuli Java program vaše računalo mora imati instaliran JVM
- To je riješilo originalni problem – izvorni kod se piše i kompajlira samo jednom
  - Problematika izvođenja na raznim sustavima prebačena na programera JVM-ova
  - Brzina je manja nego C, ali puno veća od standardnih interpretiranih jezika
  - Koristi se jako puno u komunikacijama (između svakakvih sustava)

To je to!

Pitanja?