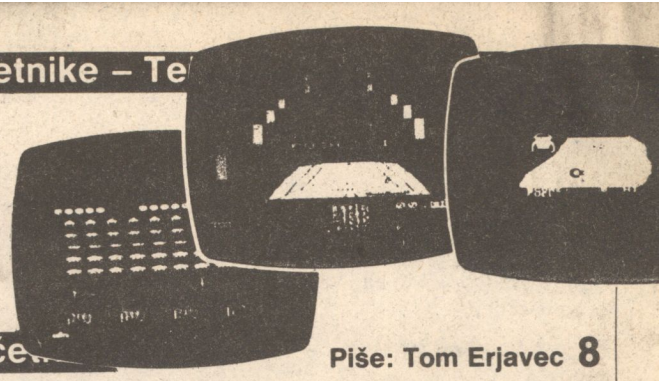
OSNOVE RAČUNALNIŠTVA



Teleksov tečaj za začetnike – Teleksov tečaj za zače...

Piše: Tom Erjavec 8

MEMO: Zadnjič je pogovor tekel o višjih programskih jezikih. Nekaj smo jih našteli, spoznali smo pojme prevajanja in interpretiranja, ogledali smo si, kaj so podprogram, strukturirano programiranje in programski moduli. Pri programiranju uporabljamo različne podatkovne strukture. Danes bomo omenili nekaj zanimivejših.

Podatki v računalniku

Vsi podatki v računalniku so zapisani v obliki binarnih števil. To pa še ne pomeni, da lahko uporabnik manipulira zgolj s števili. Uporablja lahko tudi znake (črke), nize znakov, polja, sezname, vrste, sklade itd. Če jih hočemo predstaviti, je to nekoliko bolj zapleteno kot npr. predstaviti naravna ali cela števila. Poskusimo si na hitro ogledati nekaj osnovnih vrst podatkov.

Naravna števila (npr. 3, 124, 1095, ...) predstavimo najbolj enostavno. Odvisno od dolžine računalniške besede lahko zapišemo v eno besedo različno velika največja števila. Če je beseda dolga 8 bitov, je največje možno naravno število 255. Na splošno velja, da lahko v besedo, ki je dolga n bitov, zapišemo največje število 2 na potenco n minus 1 (2**n−1). V primeru 8 bitov je to 2**8−1, to je 256−1=255. Za 16-bitno besedo bi bilo največje število 2**16−1 = 65536. Z osmimi biti lahko torej zapišemo kaj majhno število, s šestnajstimi že bistveno večje. Kaj če bi hoteli zapisati še večja števila, pa bi bila računalniška beseda prekratka? Tedaj lahko uporabimo dvojno besedo. Če imamo 16-bitno besedo in uporabimo dvojno natančnost, lahko zapišemo število v 32 bitov (štirje zlogi) in tako lahko zapišemo največje število 2**32−1=4.294.967.296, kar pa je že sila veliko število. Vendar mora pri takem zapisu program vedeti, da število ni zapisano samo v eni besedi, ampak v dveh. To programu povemo z definicijo podatkovnega tipa.

Do tu znamo zapisati samo naravna števila, torej negativnega števila sploh ne moremo zapisati. Pač. To so računalnikarji uredili tako, da so se domenili, kako lahko zapišejo negativna števila: najvišji bit računalniške besede uporabijo za zapis predznaka. Program mora natanko vedeti, da je kak podatek definiran s predznakom, ker bi sicer prebral povsem napačno številko. Govorimo o kodiranem podatku. Če je najvišji bit enak 1 (npr. 1××××××, kjer je ××××××× binarni zapis števila), je število negativno, sicer (npr. 0×××××××) pa pozitivno. Negativna števila so zapisana v dvojiškem komplementu, a to naj povemo le informativno. V osem bitov lahko zapišemo kot največje pozitivno število s predzankom +127 in najmanjše −128. Za šestnajst bitov bi bili ti vrednosti +32767 in −32768. Če uporabljamo zapis z dvojno besedo, potem uporabimo za zapis predznaka le najvišji bit zgornje besede, medtem ko vsi drugi biti obeh besed pomenijo število.

V računalniku lahko zapišemo tudi decimalna števila, tako da jih binarno kodiramo. To so potem binarno kodirana decimalna števila (angl. Binary Coded Decimal – BCD). Pri kodiranju uporabimo za zapis ene decimalne številke štiri bite. V štiri bite lahko zapišemo števila do 15. A vse vrednosti nad 9 so prepovedane. Torej pomenijo vsaki štirje biti od najnižjega navzgor po eno številko zapisa decimalnega števila. Npr. število 15 (DEC) bi zapisali z binarnim kodiranjem kot 0001 0101, pri čemer pomeni 0001 enico in 0101 petico.

Na nekoliko bolj zapletene načine se da predstaviti tudi števila z decimalno vejico, fiksno ali plavajočo, a se tokrat v podrobnosti ne bomo spuščali. Poglejmo si raje, kakšni podatki so nam dostopni v programskem jeziku, prevajalnik pa naj skrbi, da jih bo stroj razumel.

Najbolj enostaven podatek je podatek Boolovega tipa, to je podatek »da« ali »ne«, 1 ali 0, »pravilno« ali »napačno« (angl. true ali false). Program in programer se tako sporazumeta o kakem dogodku ali dejstvu, ki ga »true« potrdi, »false« pa ovrže.

Drugi enostaven podatkovni tip je celoštevilska spremenljivka (angl. integer), ki ima lahko npr. vrednosti −793, 0, 12573 itd.

Nekateri jeziki definirajo »realna števila« (angl. Real), drugi »števila z decimalno vejico« (angl. Fixed-point ali Floating point – fiksna ali plavajoča vejica). Števila s plavajočo vejico so lahko predstavljena kot realna števila z decimalno vejico (npr. 3,749) ali pa so zapisana v eksponentnem zapisu (npr. 3.1415E-2 pomeni isto kot 0.031415). Število za oznako E pomeni, za koliko mest se mora premakniti decimalna vejica glede na število, ki je zapisano pred E. Če E sledi negativno število, se vejica premika levo, sicer desno.

Zapisovanje znakov

Naj bo dovolj o številih. Poglejmo, kako stroj zapisuje znake. Ker stroj razume le številke, moramo znake kodirati vanje. Za kodiranje znakov obstaja več mednarodnih standardov. Težave so zaradi različnih narodov, ki uporabljajo v svojih abecedah različne znake. Omenimo najbolj razširjeni standard ASCII. Kratica pomeni American Standard Code for Information Interchange (ameriška standardna koda za izmenjavo informacij). Obsega kode za 128 različnih znakov (tudi kontrolne znake za upravljanje V/I naprav). V ASCII kodi bi denimo besedo TRAVA zapisali s številskim nizom v HEX zapisu: 54 52 41 56 41.

Od drugih razširjenih kod velja omeniti še EBCDIC kodo, ki obsega 256 znakov (s kontrolnimi znaki vred). Za iste znake uporabljata ASCII in EBCDIC povsem različne kode.

V programih lahko zapisujemo iz znakov tudi cele besede, ali celo stavke. V ta namen moramo definirati polja znakov. Vsako tako polje vsebuje informacijo, koliko znakov je polje dolgo in koliko znakov od vseh možnih je zares zapolnjenih z besedilom. Tako lahko računalnik manipulira z besedili. V pomnilniku pa vsak znak zapisa zavzame en zlog (osem bitov).

Možnosti za zapis tekstovnih polj v pomnilniku je več, a oglejmo si eno. V našem primeru bomo zapisali najprej definicijo tekstovnega polja in s tem prihranili v pomnilniku prostor za besedilo. Tekstovno polje imenujemo BESEDA in naj bo tako dolgo, da bomo lahko vanj zapisali besedo z 10 znaki:

BESEDA TEXT(10)

Takoj po definiranju bi bilo polje BESEDA v pomnilniku videti takole (števila so zapisana v HEX):

0A 00 40 40 40 40 40 40 40 40 40 40

0A pomeni 10(DEC) in povezuje dolžino polja (10 znakov). Druga štvilka (00) pomeni dolžino podatka v polju. Ker v polju BESEDA ni zapisano še nič, je dolžina podatka 0.40 (HEX) v EBCDIC kodi pomeni kodo za presledek. Torej je tekstovno polje ob definiranju zapolnjeno s presledki. Če bi v polje BESEDA zapisali besedo TRAVA, bi dobilo v pomnilniku naslednjo obliko (zapis je v HEX):

0A 05 E3 D9 C1 E5 C1 40 40 40 40 40 0A

še vedno pomeni dolžino polja (10). Dolžina podatkov pa se je spremenila, saj polje ni več

Popravki popravkov

Čeprav smo v zadnji številki objavili popravke k predzadnji številki, je zaradi malomarnosti znova prišlo do napak v popravkih. Prvi popravek bi moral biti zapisan takole:

LABELA	MNEMONIK	OPERANDI	KOMENTAR
P2	ADD	A,B	seštej registra A in B

Razvidno bi moralo biti, da gre za štiri stolpce, a je bilo besedilo zapisano zvezno.

Podobna napaka je bila narejena pri četrtem popravku, kjer besedilo ni bilo zapisano v stolpcih. Naslednji zapis v dveh stolpcih je pravilen:

Instrukcija	koda (HEX)
MOV dir,A	OF14 (16-bitni naslov operanda 1)

s čimer smo hoteli povedati, da se zapis MOV dir,A prevede v dve števili: OF14(HEX) in naslov operanda v (HEX) zapisu.

Sedaj pa še popravki besedila iz zadnje številke. V drugem naslovu je pisalo »Komplikatorji in interpretorji«, pisati pa bi moralo:

Kompilatorji in interpretorji

Upamo, da ni kdo pomislil, da kompilatorji res kompliciraj o!

Druga napaka je bila pri zapisu stavčnega konstrukta IF v poglavju »Stavčni konstrukti«. Zgradba bi morala biti namesto v preprostem stolpcu zapisana takole:

```
IF (pogoj), THEN
    blok programa, če je
    bil pogoj izpolnjen
  ELSE
    blok programa, če pogoj
    ni bil izpolnjen
ENDIF
```

Tak zapis (z zamiki) daje resnični občutek strukture.

Tretja napaka je bila v poglavju Podprogrami za primerom funkcije. Za procedure je pisalo, da »spremljajo parametre od glavnega programa«, pisati pa bi moralo, da »sprejemajo parametre od glavnega programa«.

V imenu postavljalcev besedila in kontrole se vam za napake vljudno opravičujem.

Avtor