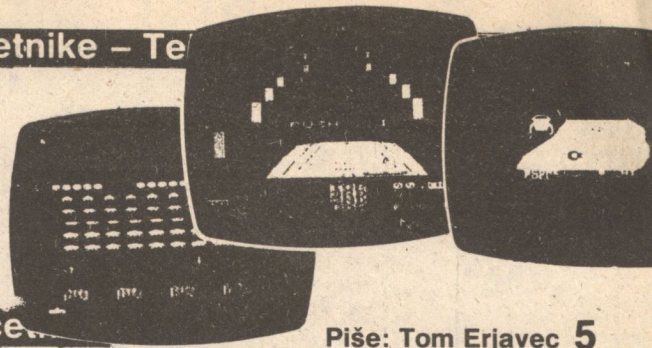


# OSNOVE RAČUNALNIŠTVA



Piše: Tom Erjavec 5

**M**EMO: Zadnjič smo spregovorili o arhitekturi centralne procesne enote. Spoznali smo pojme, kot so aritmetično logična enota, notranje vodilo, zunanje podatkovno in naslovno vodilo, programski in podatkovni števniki. Skušali smo si zapomniti imena in funkcije registrov, da bi danes lahko spregovorili o naboru inštrukcij, to je ukazih, ki jih razume in izvršuje CPE.

## Nabor inštrukcij

Nabor inštrukcij je skupina nalog, ki jih razume in izpolnjuje CPE. Ker je med jezikom, ki ga razume stroj, in človeškim jezikom velik prepad, je bilo potrebno poiskati nekaj vmesnega, kar lahko razumeta človek in stroj.

Strojni jezik, ki je vsakemu stroju lasten in ga drug stroj navadno ne razume, je skupina številčnih kod. Vsaka koda ima svoj pomen. Denimo, da imamo CPE, ki mu številka 85 pomeni »seštej števili iz akumulatorjev A in B ter postavi rezultat v A«. Ko koda 85 pride v CPE, ta še ne ve, kaj naj naredi. Zato jo pošlje v inštrukcijski dekodec. Ta lahko sproži za kodo 85 »mikroprogram«, shranjen v izredno hitrem pomnilniku. Mikroprogram narekuje je ALE, naj vzame številko iz A ter ji prišteje številko iz B in rezultat shrani nazaj v A; pri tem naj zbrise prejšnjo vsebino A, hkrati pa naj v PSW zapiše vrsto rezultata operacije.

V mikroprogramu so zapisana navodila za vsako inštrukcijsko kodo, ki jo inštrukcijski dekodec prepozna.

Zdaj lahko na vse to pozabimo, saj se nam s takimi mikrodogodivčinami ni potrebno ukvarjati. Vse bo opravil CPU, ko bo prepoznal eno od inštrukcijskih kod. In to zelo hitro. Mikroprogram za eno kodo je opravljen v manj kot mikrosekundi (torej jih lahko izvede več kot 1.000.000 na sekundo).

Vrnimo se na raven inštrukcijske kode. Številka

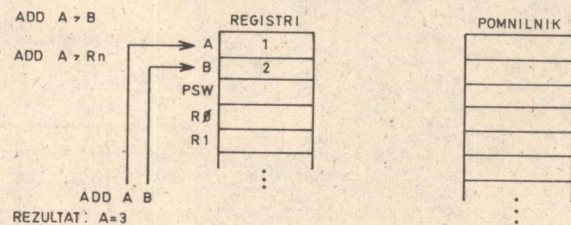
85 nam kaj malo pomeni. Jutri bi jo zagotovo pozabili, posebno, če bi morali pomniti še vseh drugih 130 kod, ki jih (denimo) pozna naš mikroprocesor. Zato so si izmislili jezik, ki mu pravimo zbirni jezik (angl. assembly language). Njegovim besedam pravijo MNE-MONIKI. Mnemonik je beseda, ki nas na nekaj spominja. Praviloma so vsi zbirni jeziki (vsak stroj ima navadno svojega) sestavljeni iz okrajšanih angleških besed. Tako bi se ukaz »odštej vsebini registrov A in B ter shrani rezultat v A, pri tem pa postavi statusno besedo PSW« glasil v zbirnem jeziku npr. »SUB A, B«, pri čemer je SUB okrajšava za angleško besedo SUBTRACT (odštej), A in B pa sta imeni registrov, ki sodelujeta v operaciji.

Gotovo nam ADD A, B (»add« pomeni v angleščini »seštej«) pove mnogo več kot številka 85. Toda stroj tega ne razume. Zato mora program, ki je napisan v zbirnem jeziku, prevesti v strojno kodo. To težaško delo spreminjanja mnemonikov v številke opravlja ZBIRNIK (angl. ASSEMBLER), to je prevajalnik za zbirni jezik. Zbirnik je program. Ko naleti na zapis ADD A, B, namesto njega zapiše številko 85. O zbirniku in zbirnem jeziku se bomo podrobneje pogovarjali prihodnjič.

Zdaj smo se že izognili skoraj vsemu zamotanemu, kar se godi v mikroprocesorju in kar preprost človek kaj težko razume. Zato si oglejmo, kako deluje nabor inštrukcij. Vse inštrukcije namreč niso tako preproste kot ADD A, B. Vsaka je sestavljena iz kode, ki pove, kaj se bo zgodilo z operandoma, in iz operandov, ki sta števili, sodelujoči v operaciji. Včasih pa ni podano neposredno število, ampak naslov ali vsaj pot do tja, kjer je število. Ker do operandov lahko pridemo na več načinov, pravimo temu naslavljanje operandov.

Na skupini inštrukcij ADD si bomo ogledali načine naslavljanja.

**ADD A, B**  
**ADD A, Rn**

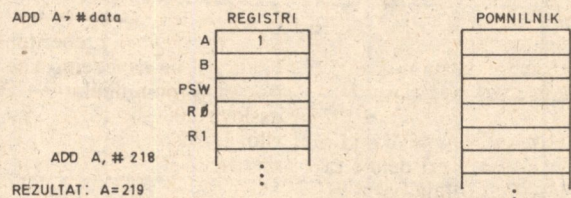


V zgornjem primeru naslavljammo registre. A, B in Rn (kjer je n številka od 0 do 7 – tako smo definirali arhitekturo našega mikroprocesorja zadnjič) so znani registri in jih je preprosto

nasloviti. To imenujemo naslavljanje registrov.

**ADD A,#data**

Zgoraj pomeni # data takojšen podatek. V primeru ADD A,#218 bi se vredno



sti v A prištel kar številka 218. Primerjaj s spodnjim primerom! Tako naslavljanje se imenuje naslavljanje takojšnjega podatka.

**ADD A, dir**

V zgornji inštrukciji »dir« pomeni naslov pomnilnika. To je številka »škatlice« v pomnilniku, v kateri je operand. Za naslov 218 bi bila inštrukcija takale: ADD A,218, kar bi pomenilo, da bi vsebino škatlice na naslovu 218 prištel k vsebini v registru A. Temu naslavljanju pravimo direktno.

**ADD A,\*dir**  
**ADD A,\*Rn**

Pri zgornjem naslavljanju bomo izračunali pravi naslov operanda iz baznega naslova in odmika. Odmik

## Vaše vprašanje, strokovnjakov odgovor

V naši rubriki bomo nekaj prostora namenili tudi vprašanjem bralcev. Pišite nam na naslov **Uredništvo Teleksa, Titova 35, 61000 Ljubljana** (z oznako: **Za osnove računalništva**), izbrali bomo najzanimivejša vprašanja, takšna, ki bi utegnila zanimati zlasti najširši krog začetnikov. Vprašanja so lahko teoretična, skušali pa bomo posredovati tudi druge informacije, npr. naslove klubov, specializiranih revij.

### Kako poteka komunikacija računalnika s terminalom? J. P., Ljubljana

Kot je splošno znano, je terminal sestavljen iz ekrana in tastature. Mnogi si predstavljajo vse skupaj kot celoto, vendar to pravzaprav ni res. To sta v resnici dve ločeni periferni enoti, ena izhodna in ena vhodna, ki sta medsebojno neodvisni. Terminal in računalnik sta povezana z najmanj tremi žicami. To so ničelna (ali zemlja), oddajna, po kateri tastatura pošilja sporočila računalniku, in sprejemna, po kateri ekran sprejema sporočila od računalnika. Sporočila, o katerih govorimo, so znaki (torej črke) oziroma natančneje kode znakov. Kode so standardizirane in ne glede na proizvajalca pomeni ista koda vedno isto črko. Najbolj znane so kode ASCII. Znaki oziroma njihove kode obsegajo vse velike in male

črke angleške abecede, številke, ločila (kot so vejica, pika, klicaj itd.), matematične znake (plus, zvezdica, ki pomeni množenje, enačaja itd.) in se nekatere znake. Poleg tega pa še kontrolne znake, ki (kot pove že ime) kontrolirajo obliko zapisa na ekranu (ali tiskalniku). Taki znaki se ne izpišejo, ampak pomenijo določeni ukazi ekranu. Na primer, premik na začetek vrstice zahteva znak CR (carriage return), premik eno vrstico navzdol zahteva znak LF (line feed). Vseh znakov ASCII je 128, prvih 32 je kontrolnih, drugi pa so navadni. Črka A ima, na primer, kodo 65, plus ima kodo 43, CR ima kodo 13 itd.

Kaj se torej zgodi, ko pritisnemo na kako tipko tastature? Tastatura (z ustreznim elektronskim vezjem) določi kodo pritisnjene tipke in jo pošlje računalniku. Ne glede na to,

kaj je računalnik tedaj počel, prekine trenutni posel in ugotovi, kaj je to prekinitelj povzročilo. Ugotovi, da je prejel kodo od tastature in kaj mora zaradi tega narediti. Če je to koda navadne črke, jo pošlje ekranu in jo spravi nekam v spomin (v buffer), da jo bo kasneje obdelal. Če pa je prejel kodo kontrolnaga znaka, to pomeni, da mora izvršiti neki ukaz. Pritisk na tipko torej šele posredno sproži izpis ustreznega znaka na ekranu. Temu pravimo tudi odmev (ali echo). Ko je računalnik na kratko obdelal pritisk tipke, nadaljuje prekinjeni posel. Ukazi so večinoma sestavljeni iz več znakov, zato imamo na tastaturi posebno tipko, s katero ukaz zaključimo. To je tipka RETURN, ki ima kodo CR. Šele ko računalnik sprejme to kodo, iz bufferja vzame prejete znake, ugotovi, kaj ukaz zahteva in ga izpolni. V omejenem času računalnik pričakuje samo omejeno število ukazov, ki jih lahko pravilno razume. Če vtipkamo nepravilen oziroma nepričakovan ukaz, ga računalnik ne razume in njegovo nadaljnje delovanje je odvisno od programa, ki se izvaja. Lah-