SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

ADRESIRANJE RADNE MEMORIJE

ALEN PEK
38273/09-R
U Varaždinu, 10.siječnja 2010.

Sadržaj

1. Uvod.. 1
2. Adresno polje. 2
3. Uloga centralnog procesora (CPU). 3
3.1. Programsko brojilo (PC). 3
3.2. Adresna i podatkovna sabirnica. 4
3.3. Registri 4
3.4. Dekoder. 4
4. Adresiranje radne memorije. 5
4.1. Direktno adresiranje. 5
4.2. Neposredno adresiranje. 6
4.3. Indirektno adresiranje. 8
4.4. Relativno adresiranje s obzirom na PC.. 9
4.5. Adresiranje registara i pomoću registara. 11
4.5.1. Adresiranje registara. 11
4.5.2. Registarsko indirektno adresiranje. 11
4.5.3. Registarsko relativno adresiranje. 12
4.6. Indeksirano adresiranje. 13

4.7. Adresiranje po stranicama. 15
5. Zaključak. 17
6. Literatura. 18

1. Uvod

Na početku je potrebno reći što je u stvari memorija, čemu služi te koje vrste memorije postoje. Važno je i napomenuti da se svi primjeri adresiranja odnose na 8-bitno računalo. Danas postoje i 32 i 64 bitna računala, no princip adresiranja je manje više isti. Naravno da se pojavom „više bitnih“ računala dobivaju i dodatne mogućnosti koje nema 8–bitno računalo, no za početak je cilj objasniti adresiranje na način kako to rade 8-bitna računala.

Dakle, što je to memorija ? Memorija je u pravilu niz digitalnih skolopova (registara) koji služe za zapis podataka u binarnom obliku. Registri su ustvari lokacije za zapis digitalnih brojeva. Lokacija u pravilu sadrži adresu koju prikazuje memoriji svaki put kada se želi raditi sa sadržajem te lokacije. Lokacija se dakle sastoji od sadržaja i adrese (na kojoj se sadržaj nalazi i po kojem ga memorija raspoznaje). Nije nužno da se upotrijebi svaka lokacija koju je moguće adresirati, već onoliko koliko ih za neku konfiguraciju stvarno treba. Svaka od tih lokacija ima svoju određenu jedinstvenu adresu po kojoj se razlikuje od drugih adresa (razlikuje se i sadržajem), a ona je u stvari binarni broj od „n“ bita. Tom se adresom može adresirati 2 na „n“ riječi, što bi značilo da za 2 na 8 (ukoliko se radi o 8-bitnom računalu sa jednom rječju od 8 bita) memorija ima 256 lokacija. To je dakako nedovoljno za ozbiljniju obradu podataka te se zbog toga koriste dvije 8-bitne riječi, tj. 16 bita, u konačnici 65536 lokacija i to je već zadovoljavajući broj za ozbiljniju obradu. To je i glavni razlog zašto 8-bitna računala koriste 2 riječi od 8 bita. Radi se o riječima veće i manje težine, (koje će kasnije u primjerima biti detaljnije objašnjene).

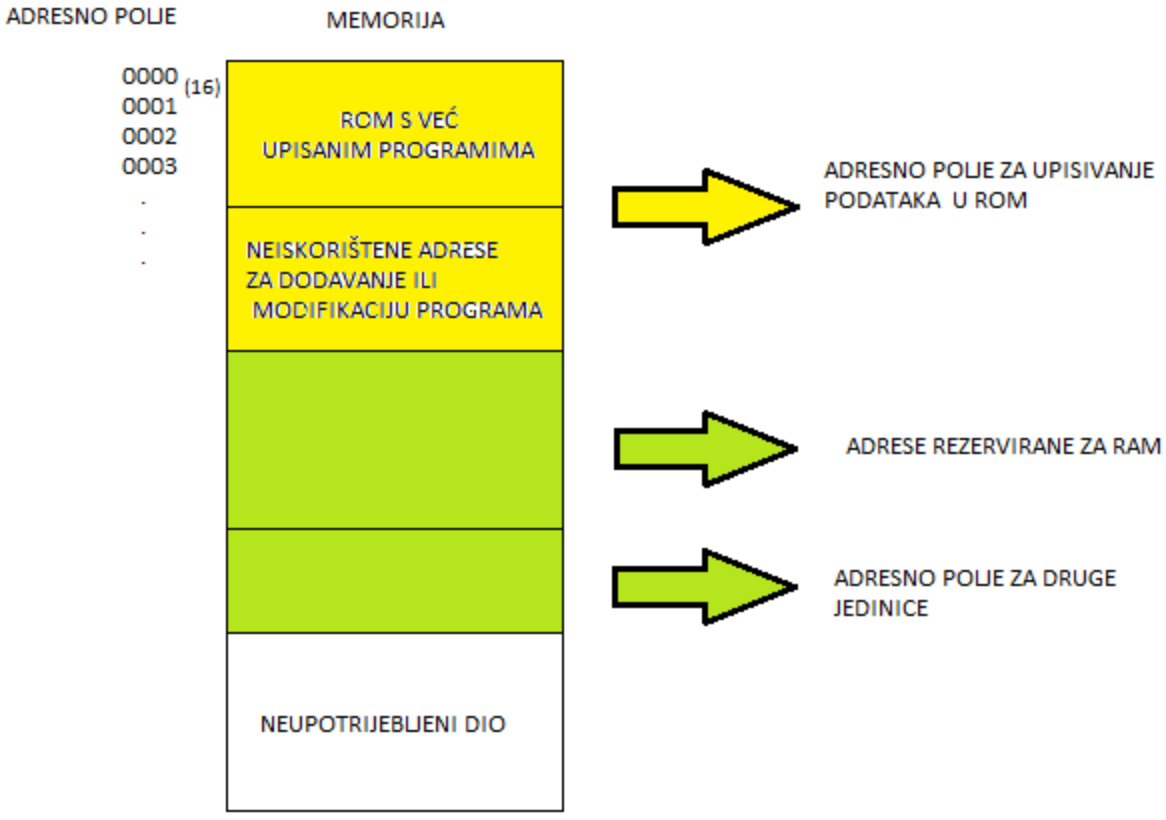
Razlikujemo dvije osnovne vrste memorije, to su ROM (memorija samo za čitanje, odnosno za zapisivanje programa) i RAM (memorija sa mogučnošću slučajnog pristupa svakoj lokaciji, upisno-ispisna memorija).

Što se adresiranja memorije tiče, nije važno gdje se operand (podatak nad kojim želimo vršiti neku operaciju) nalazi. To može biti i ROM i RAM, ili bilo koja druga nememorijska adresa. Potrebno je određenim brojem zahvata doći do njega i izvršiti ga. U nastavku slijedi objašnjenje pojmova, te dijelova mikroračunala koji se koriste u postupku samog adresiranja.

.

2. Adresno polje

Adresno polje čini svaka moguća adresa koju procesorska jedinica može adresirati ukoliko za to postoji potreba. To je omogućeno pomoću instrukcije koja se može sastojati od jedne, dviju ili tri riječi.
Prva je riječ operacijski kod (operacijski kod osim užeg koda koji sadrži vrstu operacije, sadrži i to o kakvom je adresiranju riječ), druga i treća riječ najčešće označavaju adresu operanda (operand predstavlja vrijednost nad kojim djeluje instrukcija, tj. nad kojim se vrši nekakav proces odnosno obrada, a zapisan je u memoriji). Ukoliko je traženi podatak zapisan u registrima opće namjene, rad je ubrzan jer su adrese tih registara dio tzv. proširenog operacijskog koda. Adresa se u tom slučaju nalazi u prvoj riječi instrukcije pa je operand odmah dostupan. Kao što je rečeno u uvodu, veličina adresnog polja 8-bitnih računala iznosi 64k lokacije, tj. za adresiranje je potrebno 16 bitova, odnosno dvije 8-bitne riječi (manje i veće težine, bitne za sam postupak adresiranja). Adresno polje može biti za različite namjene (slika 2.1).

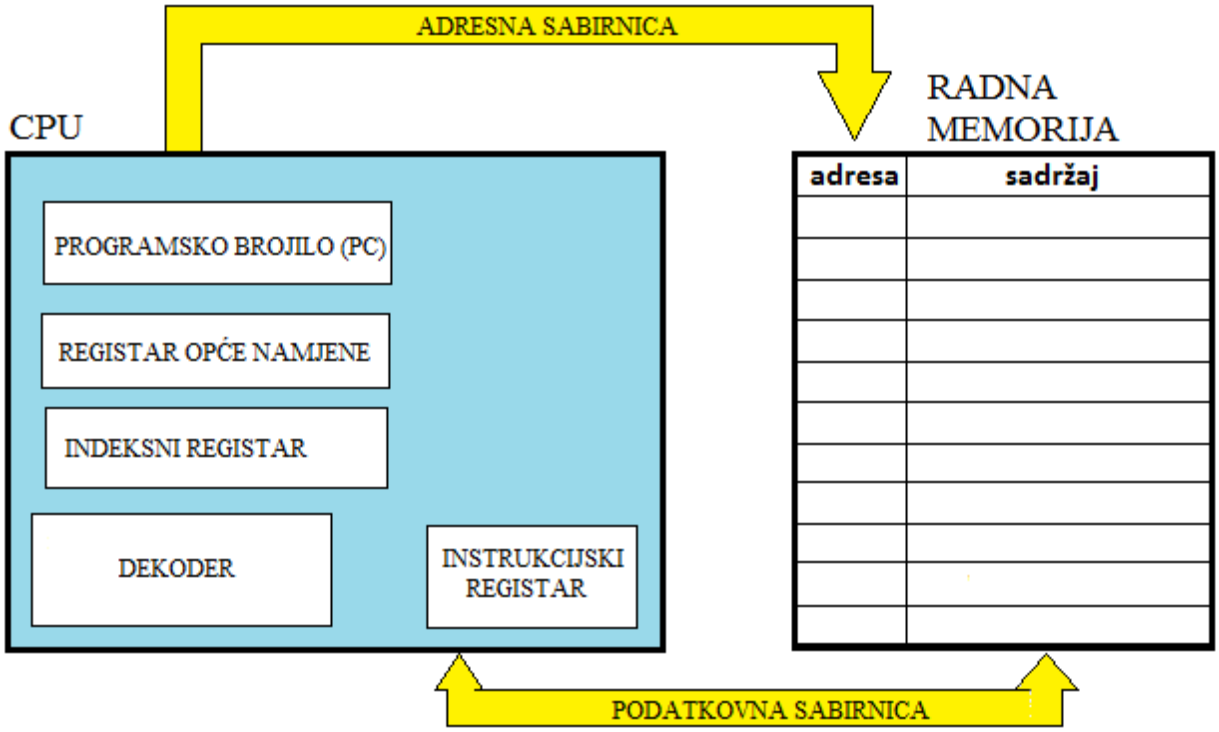


Slika 2.1. Prikaz podjele adresnog polja za različite namjene

Kao što vidimo na slici, adresirati se može i ROM i RAM memorija, te bilo koja druga nememorijska adresa. Razlika je u tome što je ROM moguće samo čitati, dok se u adrese namjenjene RAM-u može i upisivati.

3. Uloga centralnog procesora (CPU)

Kao i svaka druga instrukcija, i adresiranje je nemoguće bez rada centralnog procesora odnosno njegovih najbitnijih dijelova. U nastavku sam pojasnio koji su to bitni dijelovi te ulogu svakog od njih u adresiranju.



Slika 3.1. Prikaz centralnog procesora povezanog sa radnom memorijom putem sabirnica

3.1. Programsko brojilo (PC)

Programsko brojilo sadrži adresu lokacije memorije čiji sadržaj ukazuje na sljedeću instrukciju koju treba izvesti. Prvo je potrebno naredbu iz memorije dovesti u procesor, zatim se sadržaj iz programskog brojila preko adresne sabirnice prenosi u memoriju, te pokazuje na lokaciju s operacijskim kodom. Naredbe se uzimaju po redu, pa se zbog toga za adresiranje koristi PC a ne registri, jer se PC automatski povećava za 1.

3.2. Adresna i podatkovna sabirnica

Sabirnice su „krvotok“ računala, preko njih se prenose podaci, adrese i upravljački signali između mikroprocesora i ostalih komponenti računala, u ovom slučaju memorije. Adresna sabirnica prenosi adresu registra koja određuje izvor ili odredište podataka poslanih po sabirnici podataka. Adresnu sabirnicu čini skup jednosmjernih izlaznih linija iz mikroprocesora. Standardna mikroračunala imaju 16-bitnu sabirnicu, što omogućuje izbor registra u memorijskom prostoru od 64k riječi. Podatkovna sabirnica omogućuje dvosmjerni tok podataka (od mikroprocesora prema memoriji i obratno), no ne i istovremeno. Standardno računalo ima 8-bitnu podatkovnu sabirnicu.

3.3. Registri

Instrukcijski registar zna da podatak koji dolazi treba upisati u brojilo podataka, odnosno mjesto gdje se pohranjuje adresa operanda. Indeksni registar služi za indeksirano adresiranje, na način da se adresa dobije kombiniranjem sadržaja indeks registra s odgovarajućom veličinom koja sačinjava dio instrukcije. (Više u poglavlju „Indeksirano adresiranje“). Registar opće namjene ubrzava rad računala jer je pristup do njega brži od onih u memoriji (zahvat u memoriju nije potreban). U mikroračunalu najčešće ih je 1 ili 2 a može ih biti i 8 do 16.

3.4. Dekoder

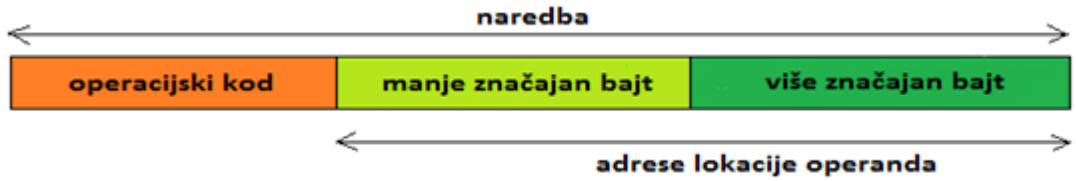
Upotrebom dekodera moguće je izvesti različite načine adresiranja i pokrenuti razne akcije kojima je ishodište odgovarajući broj na registru. Tako je iz razloga jer je često potrebno samo na jednom vodu rezultirati stanjem 1 kako bi se pokrenula neka akcija, dok je na svim ostalim stanje 0. Zbog toga i mora postojati onoliko vodova koliko je mogućih stanja.

4. Adresiranje radne memorije

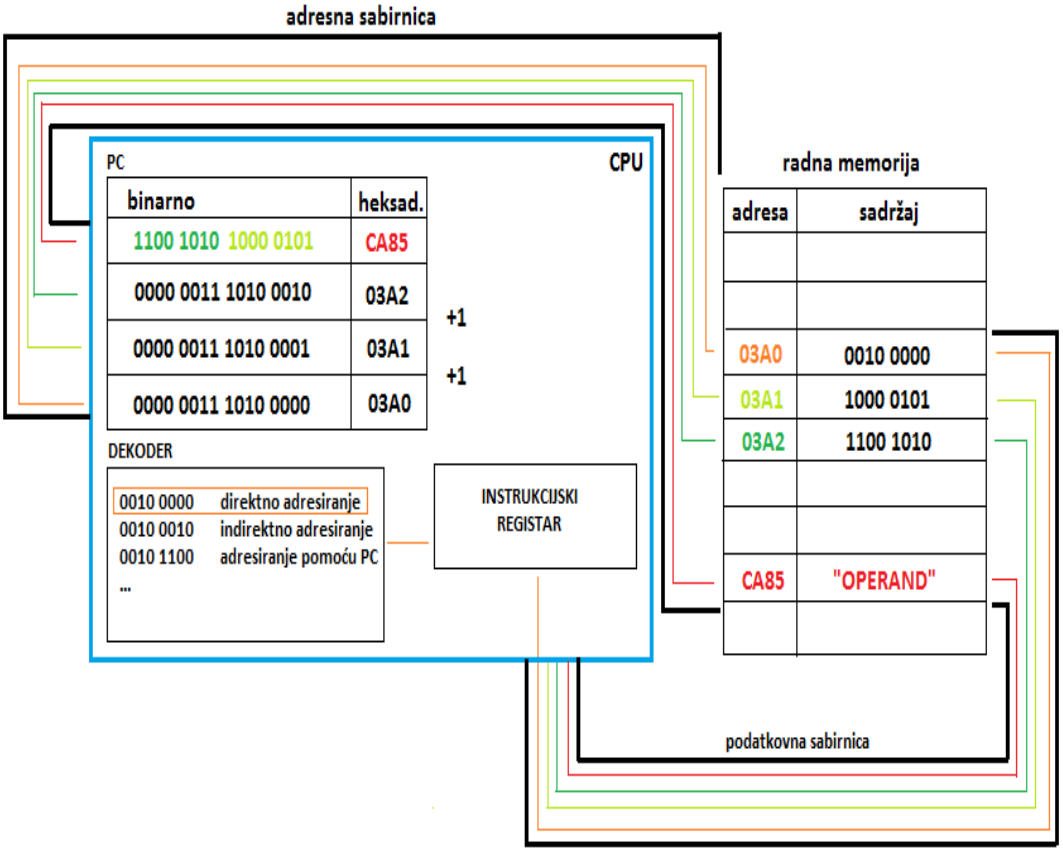
Nakon objašenjenja svih pojmova i uloga dijelova centralnog procesora, prelazim na objašnjenje različitih načina adresiranja. Adresiranje možemo definirati kao postupak oblikovanja efektivne adrese operanda, a pri tome se misli na konačnu adresu pomoću koje se pristupa podatku ili odredištu na kojem se podatak pohranjuje.

4.1. Direktno adresiranje

Što se direktnog adresiranja tiče, adresa lokacije operanda nalazi se u drugom i trećem bajtu naredbe. To saznajemo tek nakon čitanja operacijskog koda koji govori da se radi o direktnom adresiranju. Prvi bajt predstavlja operacijski kod (Slika 4.1.).



Slika 4.1. Prikaz naredbe u direktnom adresiranju memorije



Slika 4.2. Prikaz direktnog adresiranja memorije

Na slici 4.2. možemo svidjeti prikaz direktnog adresiranja. Važno je napomenuti da sam prikaz modificirao zbog mogućnosti što boljeg objašnjenja. Neke dijelove sam izostavio (tipa registara opće namjene i indeksnog registra) i prikazao samo one najbitnije u direktnom adresiranju. Što se tiče „operanda“ na lokaciji CA85, zapis je moguć samo u binarnom obliku. Programsko

brojilo inače ne može sadržavati više binarnih brojeva, već se ono automatski povećava za 1 i sadrži samo jedan broj. Na početku je to (od dolje prema gore) adresa 0000 0011 1010 0000 (označava adresu lokacije s operacijskim kodom) koji je u hekasadecimalnom brojevnom sustavu broj 03A0. Preko adresne sabirnice taj se broj prenosi u radnu memoriju te nalazi lokacija sa tom adresom. Na toj se lokaciji nalazi operacijski kod koji se preko podatkovne sabirnice šalje u CPU, sprema u instrukcijskom registru, te dekodira u dekoderu. Dekoder nam „pokazuje“ da se radi o direktnom adresiranju. Zatim programsko brojilo (nakon što je poznato s kakvim se tipom adresiranja radi) se uvećava za 1 i pokazuje adresu manje značajnog bajta stvarne adrese operanda, a to je 1000 0101. Čita ga i prebacuje preko podatkovne sabirnice u CPU te privremeno pamti u jednom od registara. Programsko brojilo se po drugi put uvećava za 1 i ovaj put pokazuje adresu značajnijeg bajta stvarne adrese operanda. Taj se binarni broj također šalje u memoriju pomoću adresne sabirnice, te čita sadržaj lokacije, a to je u ovom slučaju 1100 1010. Nakon toga se prenosi preko podatkovne sabirnice u CPU. Programsko brojilo sada sadrži i riječ manje i veće težine tako da je stvarna adresa lokacije operanda 1100 1010 1000 0101, a to je u heksadecimalnom brojevnom zapisu CA85. Ta se adresa prenosi preko adresne sabirnice i ukazuje na lokaciju sa operandom kojeg tražimo. Nedostatak je ovog načina taj što su potrebna dva zahvata u memoriju samo kako bi dobili adresu operanda, što bi sa čitanjem operacijskog koda i operanda značilo i veći broj zahvata zahvata, a to je relativno sporo. Toliko o direktnom adresiranju.

4.2. Neposredno adresiranje

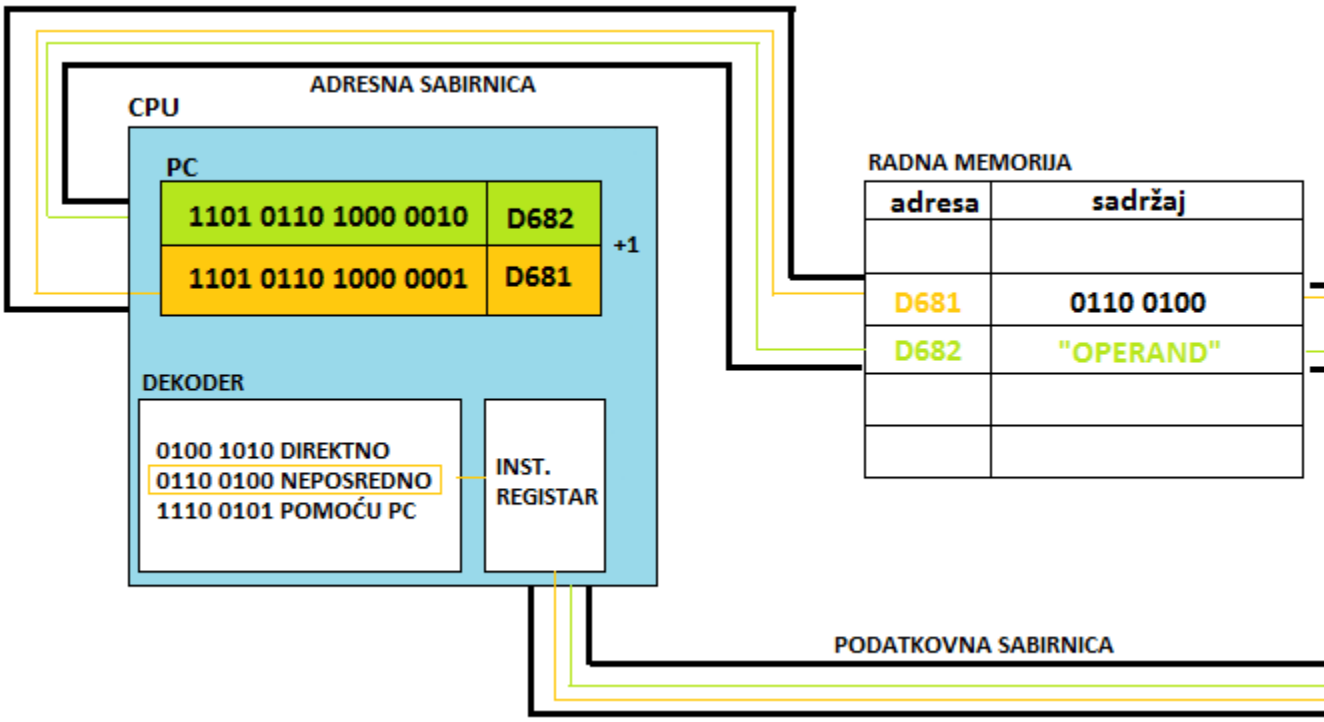
Kada govorimo o neposrednom adresiranju, tada sama naredba sadrži podatak, a ne adresu kao što je to slučaj direktnim adresiranjem (slika 4.3.).

Slika 4.3. Prikaz naredbe u neposrednom adresiranju memorije

Na slici 4.4. možemo vidjeti prikaz neposrednog adresiranja, u ovom slučaju operand je podatak od 8 bita. Dakle, u programskom brojilu se nalazi adresa koja u heksadecimalnom zapisu ima vrijednost D681. Ta se adresa adresnom sabirnicom šalje u radnu memoriju, nalazi lokacija sa tom adresom, čita njen sadržaj (operacijski kod) te šalje podatkovnom sabirnicom u CPU. Sprema se u instrukcijskom registru i dekodira u dekoderu koji nam kazuje da je riječ o neposrednom adresiranju. PC se uvećava za 1 i pokazuje adresu na kojoj se nalazi operand (kod direktnog bi to tek bila adresa manje značajnog bajta stvarne adrese operanda). Operand se čita, prenosi u CPU i dalje se s njim obavljaju željene opercacije.

Podatak može biti jedne ili dvije riječi, ovisno dali je podatak 8-bitni ili 16-bitni. U slučaju da

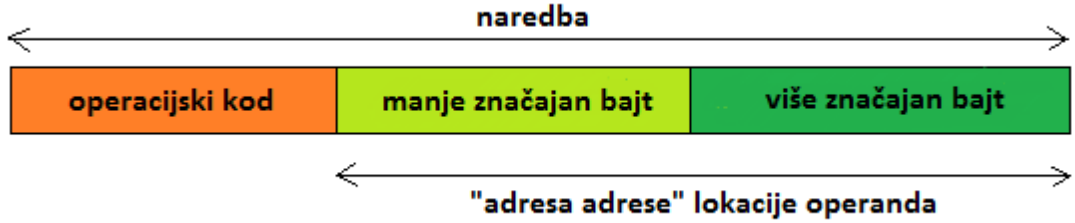
je u navedenom primjeru traženi podatak bio 16 bitni, drugi bajt bi se nalazio na adresi D683. Prednost ovakvog načina adresiranja je taj što podatak nije potrebno tražiti u memoriji, već se on dovodi u procesor kao sastavni dio naredbe, a to ujedno znači i manje utrošenog vremena za izvođenje naredbe. Potrebna su samo dva zahvata (uključujući i čitanje operacijskog koda) u memoriju i po tome je ovaj način adresiranja brži od direktnog.



Slika 4.4. Prikaz neposrednog adresiranja

4.3. Indirektno adresiranje

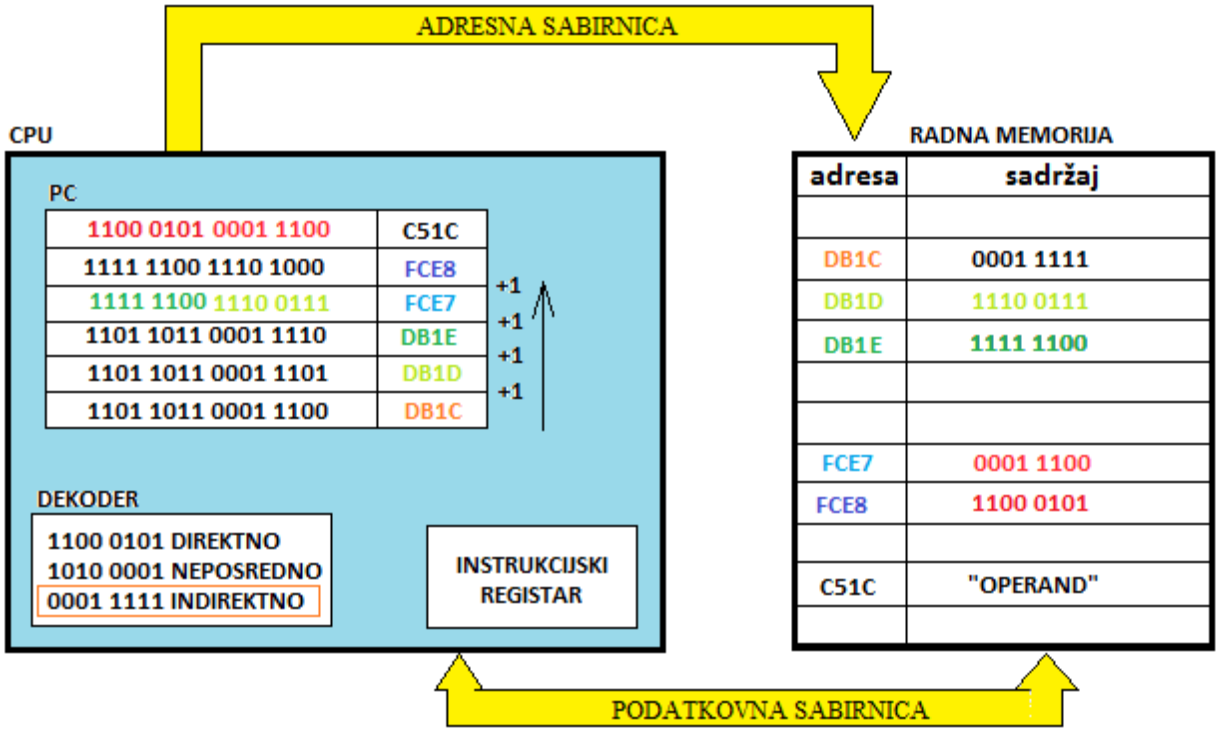
Naredba u indirektnom adresiranju u drugom i trećem bajtu ne sadrži niti adresu operanda kao kod direktnog, niti sam operand kao kod neposrednog adresiranja, već „adresu adrese“ operanda (slika 4.5), tj adresu lokacije (pokazivačka adresa) na kojoj se nalazi adresa operanda. Nakon trećeg zahvata u memoriju (nakon operacijskog koda, te manje i više bitne riječi adrese lokacije adrese operanda) dobivamo pokazivačku adresu (pointer). To znači da su potrebna još dva zahvata, pa tek onda sam operand. To je prilično velik broj zahvata u memoriju. Zbog toga je ovaj način adresiranja prilično neracionalan.



Slika 4.5. Prikaz naredbe u indirektnom adresiranju

Na slici 4.6. možete vidjeti prikaz takvog adresiranja. No ovaj put nisam koristio smjerove svakog bajta zbog preglednosti, tako da ce se svaki smjer bajtova označiti samo bojom (što bajt određuje). Kao i u prethodna dva primjera adresiranja, napominjem da PC može prikazati samo jedan binarni podatak, te da je slika improvizirana.

Dakle, krenimo na primjer. U programskom se brojilu nalazi adresa lokacije koja sadrži operacijski kod, a ta adresa je u ovom slučaju DB1C. Prenosi se putem adresne sabirnice u radnu memoriju i sa lokacije čita operacijski kod. On se prenosi putem podatkovne sabirnice u CPU, sprema u instrukcijskom registru i dekodira u dekoderu. Dekoder u ovom slučaju pokazuje da se radi o indirektnom adresiranju. Programsko brojilo se uvećava za 1 i pokazuje broj DB1D, a on ujedno označuje manje značajan bajt adrese adrese operanda. Riječ „adrese“ nije greškom napisana dva puta, već je riječ o upravo tome, manje značajnom bajtu adrese koja će nas odvesti na adresu stvarnog operanda. Taj se bajt prenosi dakako u CPU i privremeno pohranjuje u registru, a PC se ponovno uvećava za 1. Sada prikazuje adresu značajnijeg bajta „adrese adrese“ tj. pokazivačke adrese. To je u ovom slučaju DB1E koji se također prenosi u CPU, ono sada raspolaže i manje bitnom i više bitnom rječju, tj. prikazuje FCE7. Taj FCE7 sada označava adresu manje značajnog bajta stvarne adrese operanda. Programsko brojilo šalje tu adresu u radnu memoriju gdje se čita sadržaj njene lokacije, tj. manje značajan bajt stvarne adrese operanda, a to je 0001 1100. Taj se bajt prebacuje podatkovnom sabirnicom u CPU i privremeno pohranjuje. PC se opet uvećava za 1 i ovaj put prikazuje adresu više značajnog bajta stvarne adrese operanda. On u ovom slučaju iznosi 1100 0101, te se prenosi u CPU. Sada PC sadrži i manje i više značajan bajt te pokazuje na adresu C51C, što ustvari predstavlja stvarnu adresu operanda.

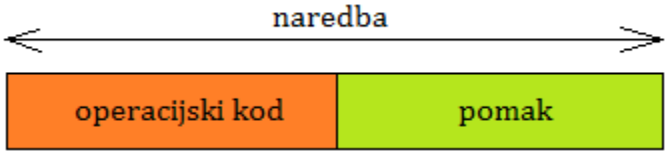


Slika 4.6. Prikaz indirektnog adresiranja memorije

Kao što sam rekao u uvodu, ovaj je način adresiranja prilično neracionalan i neekonomičan jer je potreban velik broj zahvata u memoriju, čak 6, a to ga ujedno čini i najsporijim od do sada viđenih načina. Prednost je što se stvarna adresa u pokazivačku lokaciju ne mora pisati unaprijed, već za vrijeme samog izvođenja programa. Primjenjuje se u moćnijim mikroračunalima zbog složenosti.

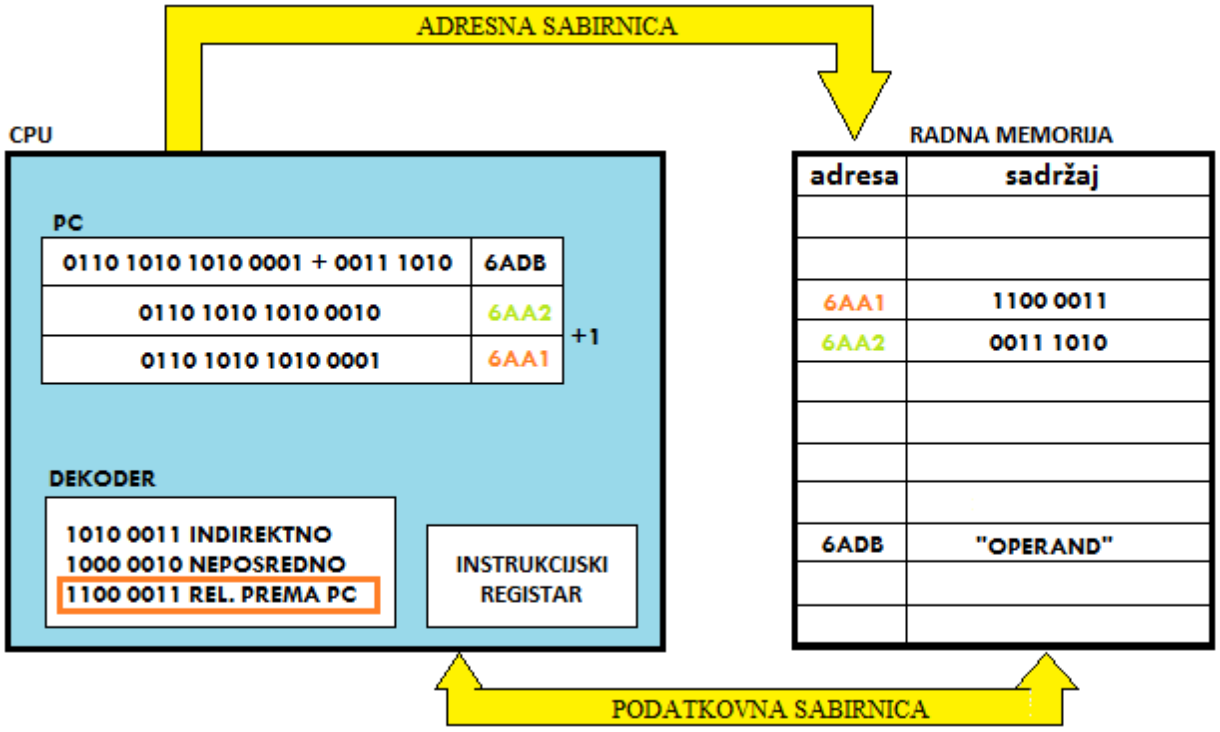
4.4. Relativno adresiranje s obzirom na PC

Prilikom adresiranja uz pomoć programskog brojila, naredba ne sadrži niti jednu adresu (ni stvarnu ni pokazivačku) niti manje i više značajne bajtove, već bajt kojeg nazivamo „pomak“. Na slici 4.7. možemo vidjeti kako izgleda naredba u ovom načinu adresiranja. Navedeni se pomak dodaje sadržaju programskog brojila, te dobiveni označava adresu operanda.



Slika 4.7. Prikaz naredbe u relativnom adresiranju prema PC

Pomak je ustvari binarni broj u aritmetici dvojnog komplementa. Pomoću njega je moguće predočiti sve brojeve u dekadskom sustavu od -128 do 127, a to ujedno znači i da se operand može nalaziti u tom intervalu. Ako je pomak pozitivan, tada se adresira adresa koja je viša od adrese naredbe, a u slučaju da je negativan, tada je obnuto, dakle adresira se adresa niža od adresne naredbe. Slijedi primjer uz ilustraciju relativnog adresiranja prema PC (Slika 4.8.). Dijelovi CPU-a poput registra opće namjene i indeksnog registra su izostavljeni zbog preglednosti.



Slika 4.8. Prikaz realtivnog adresiranja uz pomoć programskog brojila

U programskom brojilu se nalazi adresa operacijskog koda, koja je u ovom slučaju 0110 1010 1010 0001, tj. 6AA1. Ta se adresa adresnom sabirnicom prenosi u radnu memoriju te čita sadržaj lokacije pod tom adresom, tj. operacijski kod. On je u binarnom obliku 1100 0011, te se prenosi podatkovnom sabirnicom u CPU, sprema u instrukcijski registar i dekodira u dekoderu. Vidi se iz dekodera da je riječ o relativnom adresiranju uz pomoć programskog brojila. PC se uvećava za 1 i pokazuje na adresu „pomaka“. Sa adrese se iz radne memorije čita sadržaj lokacije (0011 1010) i prenosi podatkovnom sabirnicom u CPU, a zatim se zbraja sa prvobitnom adresom u

programskom brojilu, dakle 0110 1010 1010 0001 + 0011 1010. U heksadecimalnom obliku zapisa to bi značilo 6ADB, što ujedno predstavlja adresu opranda. Operand se s te adrese čita i prenosi u CPU. Ovaj način adresiranja je prilično brz jer naredbu čine 2 bajta i nužan je samo jedan zahvat u memoriju da se dođe do operanda. Nedostatak bi bio ograničenost granicama dohvata kojim ovaj način adresiranja raspolaže.

4.5. Adresiranje registara i pomoću registara

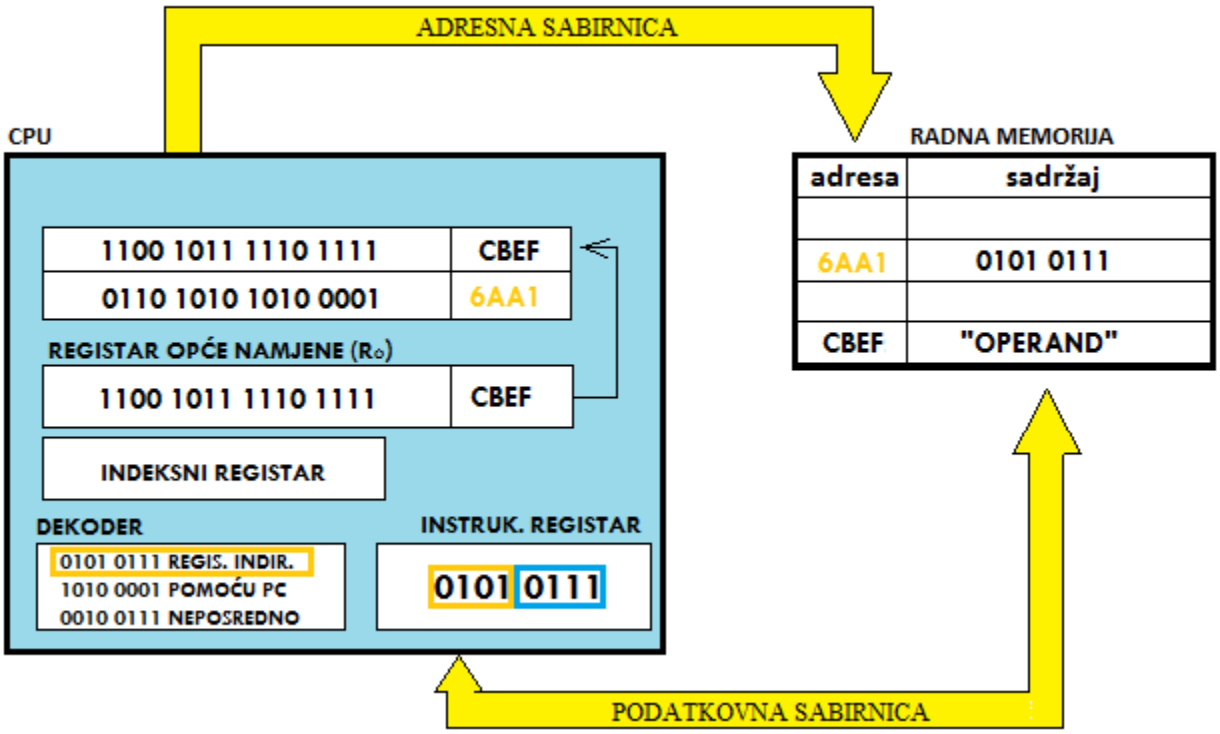
Kad govorimo o adresiranju registara ili adresiranju pomoći njih, tada mislimo na 3 različita načina adresiranja. Prvo od njih je adresiranje samih registara.

4.5.1. Adresiranje registara

Važno napomenuti da se ti registri nalaze u CPU što uvelike ubrzava proces, a iz memorije je potrebno uzeti samo operacijski kod. Operand nije potrebno dovoditi iz memorije jer se on nalazi u CPU, točnije u jednom od registara. Nakon što se pročita operacijski kod, poznato je o kakvom se adresiranju radi te točno na kojem se registru nalazi operand. To se saznaje nakon što se u instrukcijskom registru operacijski kod raščlani na 2 djela. U tom slučaju prvi dio označava o kakvom se tipu adresiranja radi, a drugi adresu registra u kojem se nalazi operand. Nakon toga operacijski kod pokazuje na registar gdje se nalazi operand.

4.5.2. Registarsko indirektno adresiranje

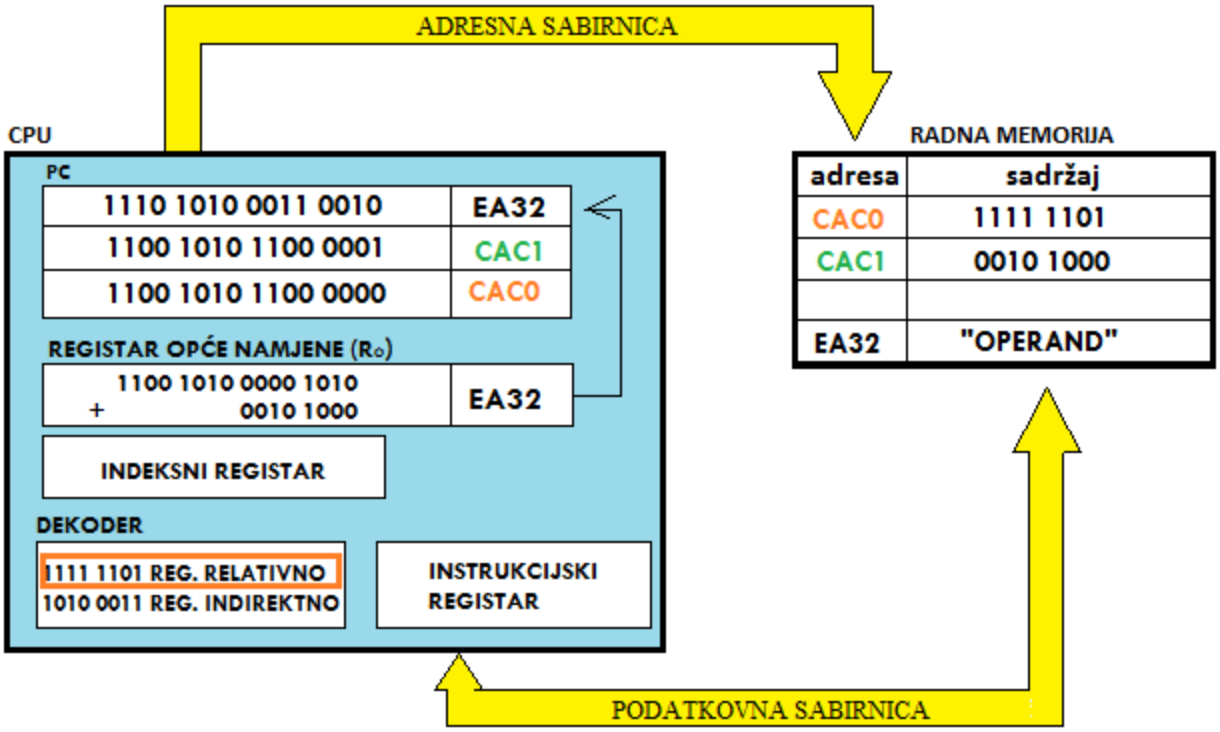
Druga dva načina adresiranja odvijaju se pomoću registara. U prvom slučaju registar se naziva pokazivački registar, a koristi se za indirektno adresiranje lokacije memorije (slika 4.9.). On tada služi kao pokazivač, a njegov sadržaj sačinjava stvarnu adresu operanda. Znatno je brže od indirektnog adresiranja preko memorijske lokacije jer je potrebno samo adresirati registar koji već sadrži adresu operanda (kod indirektnog treba prvo dobiti pokazivačku adresu, pa manje i više značajan bajt stvarne adrese i tek se onda čita operand).
U primjeru vidimo da se u programskom brojilu nalazi adresa operacijskog koda, koja se putem adresne sabirnice prenosi u radnu memoriju i čita sadržaj te lokacije, tj. sam operacijski kod. On se prenosi podatkovnom sabirnicom u CPU i dekodira u dekoderu iz kojeg vidimo da je riječ o indirektnom adresiranju pomoću registara. Nakon toga taj se operacijski kod u instrukcijskom registru raščlanjuje, prvi dio označava o kakvom se adresiranju radi, a drugi predstavlja adresu registra na kojoj se nalazi operand. Nakon toga pronalazi se registar s tom adresom, (u primjeru je to registar R0, može biti i do Rn) i čita se stvarna adresa operanda, koja je u ovom slučaju CBEF. Ta se adresa prenosi u PC, te putem adresne sabirnice šalje u radnu memoriju gdje se traži lokacija sa tom adresom. Na toj se lokaciji nalazi sam operand, koji se nakon toga prenosi u CPU.



Slika 4.9. Prikaz registarskog indirektnog adresiranja

4.5.3. Registarsko relativno adresiranje

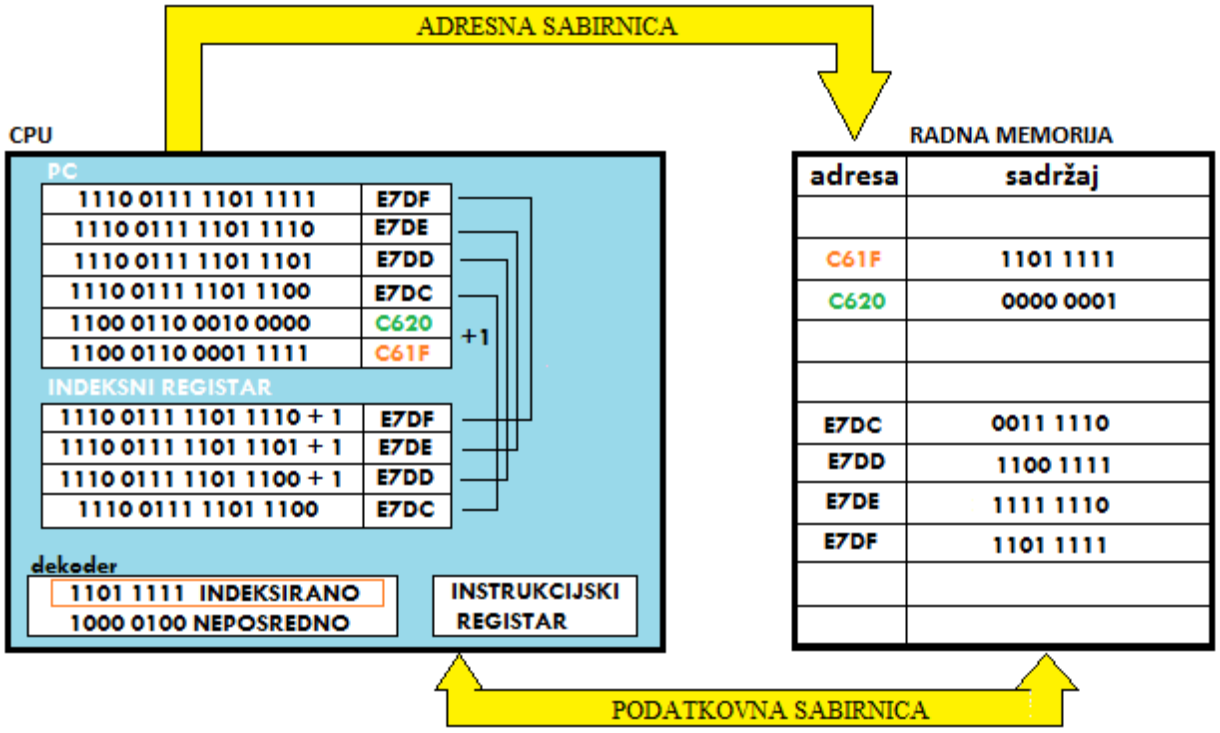
Treći način adresiranja je relativno adresiranje s ozbirom na registar. Slično je kao i adresiranje uz pomoć programskog brojila po tome što u naredbi također sadrži „pomak“. No u ovom se tipu adresiranja taj pomak dodaje sadržaju registra, a ne programskog brojila.
U navedenom primjeru (slika 4.10.) vidimo da se nakon čitanja operacijskog koda PC uvećava za jedan te ukazuje na adresu pomaka. Adresa se prenosi adresnom sabirniom u radnu memoriju, čita se pomak te se prenosi podatkovnom sabirnicom u CPU. U registru opće namjene se zbraja sa prijašnjim sadržajem toga registra (u ovom slučaju je to 1100 1010 0000 1010, a inače može biti bilo koji drugi binarni podatak, ovisi što je zapisano u tom registru prije postupka adresiranja) te se dobiva adresa na kojoj se nalazi operand, u ovom slučaju ta adresa u heksadecimalnom obliku jest EA32. Ta se adresa prenosi u PC, a zatim preko adresne sabirnice u radnu memoriju. Ondje se nalazi lokacija te adrese te se čita operand i prenosi podatkovnom sabirnicom u CPU.



Slika 4.10. Prikaz registarskog relativnog adresiranja memorije

4.6. Indeksirano adresiranje

Ovaj način adresiranja koristi indeks registar. Sadržaj toga registra se kombinira sa pomakom koji se najčešće dodaje, ali se može i oduzimati od sadržaja indeksnog registra. Posebnom se naredbom prije početka adresiranja upisuje adresa prvog po redu podatka s koji se obavi operacija u prvom izvođenju, a automatski se sadržaj registra povećava za 1 i tako za svaki sljedeći podatak. Primjer ovakovog tipa adresiranja slijedi u nastavku (slika 4.11.).



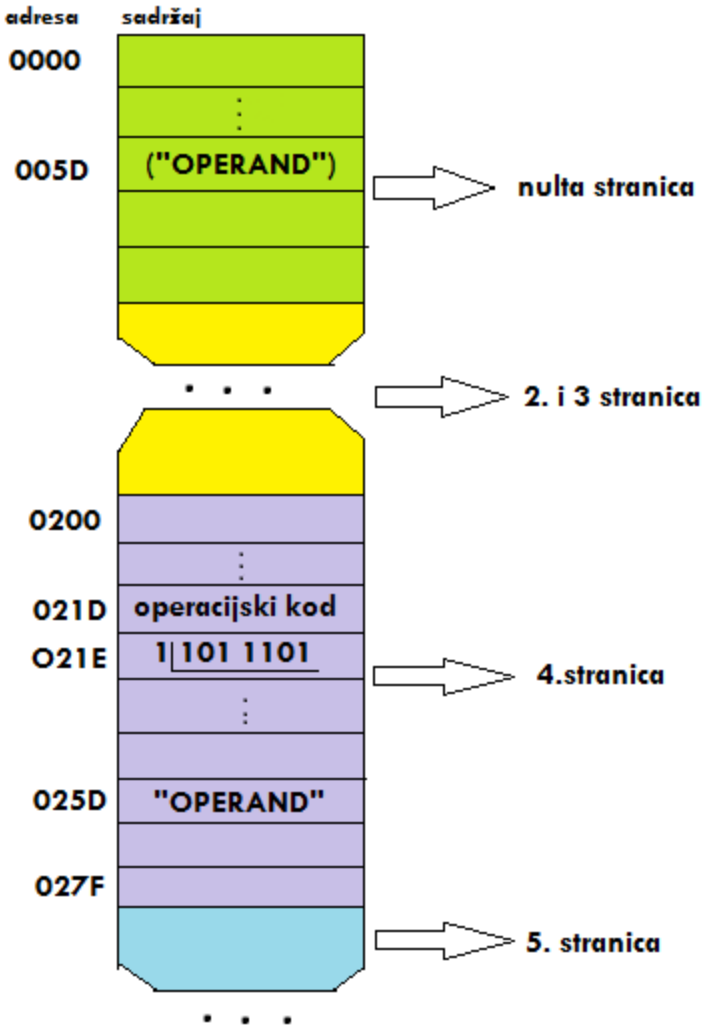
Slika 4.11. Prikaz indeksiranog adresiranja memorije

Kao što je običaj, naredbe u programskom brojilu čitamo od dolje prema gore, tako da se „na dnu“ programskog brojila nalazi adresa operacijskog koda koja u heksadecimalnom obliku zapisa ima oblik C61F. Prenosi se adresnom sabirnicom u radnu memoriju, čita iz memorije i prenosi u CPU. Sprema se u instrukcijskom registru i dekodira u dekoderu prema kojem vidimo da je riječ o indeksiranom adresiranju. PC se uvećava za jedan i sad pokazuje adresu na kojoj se nalazi „pomak“. Nakon čitanja, pomak se prenosi u CPU i čuva, a u indeksni registar prikazuje adresu prvog podatka kojeg tražimo (ta adresa je zapisana u registru prije postupka indirektnog adresiranja), a to je E7DC. Prenosi se u PC koje dalje preko adresne sabirnice u memoriji traži tu adresu, te čita podatak na toj lokaciji i podatkovnom sabirnicom ga prenosi u CPU te ga sprema. Nakon toga vrijednost indeksnog registra se uvećava za 1 (ovisno koliki je pomak) i ukazuje na adresu drugog podatka kojeg tražimo, prenosi ga u PC i tako redom ovisno koliko je podatak velik, on može biti i više od 4 bajta kao što je dano u primjeru.

Bitno je naglasiti da ovakav način adresiranja omogućuje rad s nizom operanada koji su u memoriji smješteni jedan iza drugoga. Operandi su ovdje u biti svi podaci na lokacijama od E7DC do E7DF. Vrlo je slično adresnom relativnom adresiranju koje također sadrži pomak u drugom bajtu naredbe.

4.7. Adresiranje po stranicama

Kod ovakvog načina adresiranja memorija se dijeli na stranice (nulta, prva, druga, treća...). Na jednoj se stranici može adresirati 128 lokacija. Kao i prethodno spomenuti načini adresiranja, također sadrži pomak, no on u ovom slučaju ima nešto drukčiju ulogu. Naime, kao što je i uobičajeno, pomak se nalazi iza operacijskog koda. Kod adresiranja po stranicama taj se pomak gleda na način da prva znamenka (najveće težine) predstavlja stranicu na kojoj se nalazi adresirana lokacija. U slučaju da je najznačajniji bit pomaka nula, to znači da se adresirana lokacija nalazi na nultoj stranici, a ako je to jedinica, onda znači da se adresirana lokacija nalazi na istoj stranici gdje i instrukcija.



Slika 4.12. Prikaz adresiranja memorije po stranicama

U primjeru možemo vidjeti da se operacijski kod nalazi na lokaciji 021D, a pomak odmah do njega tj. na 021E. Oba se nalaze na 4. stranici u memoriji. Kao što vidimo, najznačajniji bit pomaka je 1,a to znači da se adresirana lokacija nalazi na istoj stranici gdje i instrukcija, dakle na četvrtoj. Zatim se preostalih sedam znamenaka pomaka (101 1101 tj. 5D) zbraja sa početnom adresnom stranice, koja je u ovom slučaju 0200. Dobivamo adresu operanda koja iznosi 025D. U slučaju da je najznačajniji bajt pomaka bio 0, to bi značilo da se operand nalazi na nultoj stranici, a vrijednost 7-bitnog pomaka (ukoliko bi sve ostale znamenke ostale iste) bi se zbrajala sa početnom vrijednošću nulte stranice, dakle sa 0000. Dobili bi adresu 005D na kojoj se nalazi operand (u zagradi).
Najveći nedostatak ovog načina adresiranja je što se mogu adresirati samo nulta stranica i ona na kojoj je zapisan podatak što znači da su ostale stranice izvan dohvata. Riješenje toga problema je da se operandi nalaze na istoj stranici gdje i naredbe.

5. Zaključak

Kao što je i ranije rečeno, adresiranje možemo definirati kao postupak oblikovanja efektivne adrese operanda, a da se pri tome misli na konačnu adresu pomoću koje se pristupa podatku ili odredištu na kojem se podatak pohranjuje. Stoga smo vidjeli da postoje različiti načini adresiranja, direktno adresiranje, neposredno, indirektno, relativno s obzirom na programsko

brojilo, adresiranje registara i pomoći njih, indeksirano i adresiranje po stranicama. Oni se razlikuju po nekim komponentama. Kao prvo, to su sadržaji naredbi, neki sadrže pravu adresu operanda (direktno) u samoj naredbi, neki pokazivačku adresu (indirektno), zatim pomak (relativno obzirom na PC, adresiranje registara, indeksirano i adresiranje stranica). Daljnja podjela bi mogla biti po učestalosti korištenju programskog brojila odnosno registara, te na kraju dijeljenja memorije, tj. stranica. Svaki od ovih načina ima određene prednosti i nedostatke. Najvažnije je da tip adresiranja bude što brži (neposredno, relativno uz pomoć PC) , a prednost onih sporijih može biti da adresa ne mora biti zapisana unaprijed, već ju je moguće zapisati i tijekom odvijanja programa (indirektno adresiranje).

Sljedeća komponenta po kojoj se mjeri brzina adresiranja je svakako broj zahvata u memoriju. Što je veći broj zahvata u memoriju adresiranje je sporije (npr indirektno). Neki načini kao što je rečeno sadrže adresu u samoj instrukciji što dodatno ubrzava adresiranje jer zahvat u memoriju nije potreban.

Sigurno je da će se pojavom novih arhitektura pojavljivati i novi načini adresiranja, tako da već danas imamo procesore od 32 bita (MC 68040) koji imaju 18 različitih načina adresiranja. Procesori od 64 bita sigurno nude i više načina adresiranja, a što nas tek čeka u budućnosti, pokazat će vrijeme.

6. Literatura

Knjige:

Smiljanić G., *Mikroračunala,* Školska knjiga, Zagreb 1991.

Ribarić S., *Arhitektura računala,* Školska knjiga, Zagreb 1996.

Uroš Peruško, Vlado Glavinić, *Digitalni sustavi,* Školska knjiga, Zagreb 2005.

Internet:

http://www.informatika.buzdo.com/s130.htm

http://elektrotehnika.tvz.hr/php/skini_repoz.php?id=15294&id1=8&id2=1

Ostvareni bodovi : 7/7