Simulacija rada priručne memorije

Dominik, Vitez

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Vitez

SIMULACIJA RADA PRIRUČNE MEMORIJE

ZAVRŠNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Dominik Vitez
Matični broj: 42843/14-R
Studij: Informacijski sustavi

SIMULACIJA RADA PRIRUČNE MEMORIJE

ZAVRŠNI RAD

Mentor:

Luka Milić, mag. ing. comp.

Varaždin, rujan 2018.

Dominik Vitez

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

1946. godine objavljen je jedan od najznačajnijih članaka na području arhitekture računala pod nazivom „Uvodna rasprava o logičkom oblikovanju elektroničkog računskog uređaja“ autora A. W. Burksa, H. H. Goldsteina i von J. Neumanna gdje je opisan „von Neumannov model“ računala koji se sastoji od pet funkcijskih jedinica. Jedna od tih jedinica je memorijska jedinica. U konceptu memorijske jedinice je priručna memorija o kojoj će biti riječi u ovome radu. U ovom radu opisana je priručna memorija i način njena djelovanja. Izrađena je aplikacija koja simulira rad priručne memorije s obzirom na razne ulazne parametre. Prikupljeni su rezultati, napravljena analiza te usporedba sa teorijskim očekivanjima. Kombinacijom raznih algoritama zamjene blokova i smještaja blokova u bločne priključke, uz veći kapacitet, moguće je poboljšati performanse priručne memorije.

Ključne riječi: priručna memorija; potpuno asocijativno preslikavanje; izravno preslikavanje; skupno asocijativno preslikavanje; FIFO; LFU; Random;

Sadržaj

1. Uvod

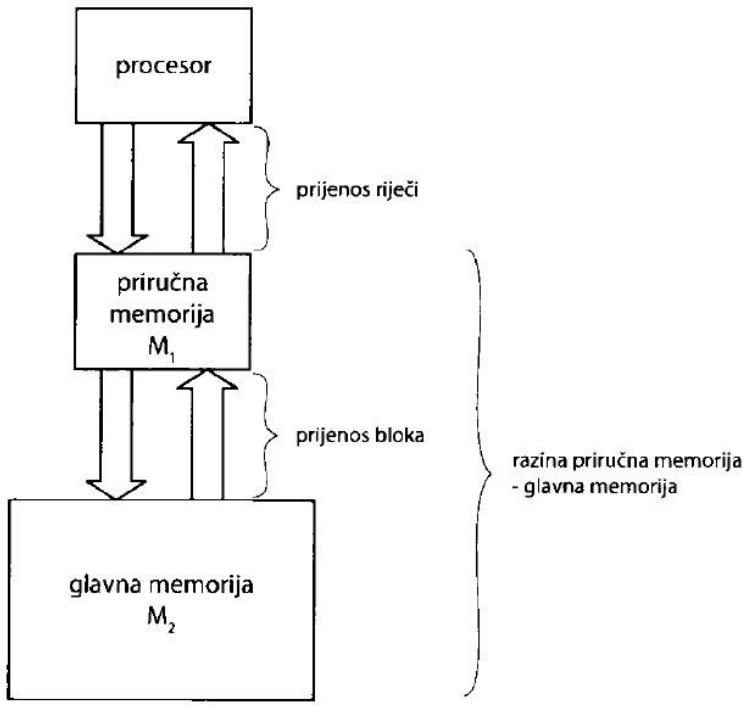
Informatika kao područje ljudskog djelovanja koje se bavi proučavanjem, razvojem i uporabom postupaka i uređaja za automatski prijenos i obradu podataka relativno je mlada znanost koja se počela razvijati 50-tih godina 20-tog stoljeća. Iako relativno novo područje ljudskog djelovanja, informatika je postala neizostavni dio ljudskog života. Svoj napredak informatika temelji na ubrzanom razvoju tehnologije koji je započeo u drugoj polovici 20-tog stoljeća.

Vrlo burna povijest razvoja računala, a s njime i informatike kao znanosti, podijeljena je u pet razdoblja s obzirom tehnologije koje su se rabile pri izradi računala. Od nulte generacije računala tj. mehaničkim računalima koja su se pojavila već u 17. stoljeću pa sve do današnjih računala sa tehnologijom vrlo visokog stupnja integracije (VLSI) bilo je mnogo značajnih godina no valja istaknuti 1946-tu godinu. Te godine je objavljen jedan od najznačajnijih članaka na području arhitekture računala pod nazivom "Uvodna rasprava o logičkom oblikovanju elektroničkog računskog uređaja" autora A. W. Burksa, H. H. Goldsteina i von J. Neumanna. U tom je članku opisan model računala koji se sastoji od četiri osnovne funkcijske jedinici : aritmetičko – logičke, upravljačke, memorijske i ulazno – izlazne jedinice. Takav model računala koristimo danas, a poznat nam je kao von Neumannovog model. Kod von Neumannovog modela računala svaka funkcijska jedinica ima svoju ulogu te ukoliko bi bilo koja funkcijska jedinica bila izostavljena, računalo ne bi funkcioniralo.

U ovom radu biti će prikazana memorijska jedinica računala, konkretnije priručna memorija. U prvom poglavlju biti će objašnjeno što je priručna memorija, njezina organizacija, kapacitet i način rada. U drugom ću poglavlju opisati načine smještaja blokova u bločne priključke priručne memorije. U trećem poglavlju biti će obrađeni algoritmi zamjene blokova. Četvrto poglavlje sadržavat će opis programske aplikacije. U petom poglavlju opisati ću prednosti i nedostatke algoritama organizacije priručne memorije te analizirati i usporediti rezultate simulacije s očekivanjima.

2. Priručna memorija

Priručna memorija (engl. *cache memory*) je memorija relativno malog kapaciteta i vrlo velike brzine u kojoj se nalazi kopija sadržaja dijela glavne memorije što najčešće čine tekući segmenti aktivnog programa i podataka. Njezina svrha je smanjenje latentnosti memorijske jedinice, odnosno umanjivanje jazza nastalog zbog različitih brzina procesora i memorije. (Ribić, 2011; Stallings, 2010)



Slika 1. Shematski prikaz dviju razina memorije (Izvor: Ribić 2011, str. 311)

Slika 1. prikazuje dvorazinsku memorijsku hijerarhiju koju dobivamo korištenjem priručne memorije (brže) i glavne memorije (sporije). Uobičajeni odnos brzina priručne memorije i glavne memorije je 5:1 što uvelike smanjuje latentnost memorije i povećava brzinu rada cijelog računalnog sustava. Valja napomenuti kako se podaci između glavne memorije i priručne memorije prenose u obliku bloka, dok se između procesora i priručne memorije podaci prenose u obliku riječi. (Ribić, 2011)

2.1. Organizacija priručne memorije

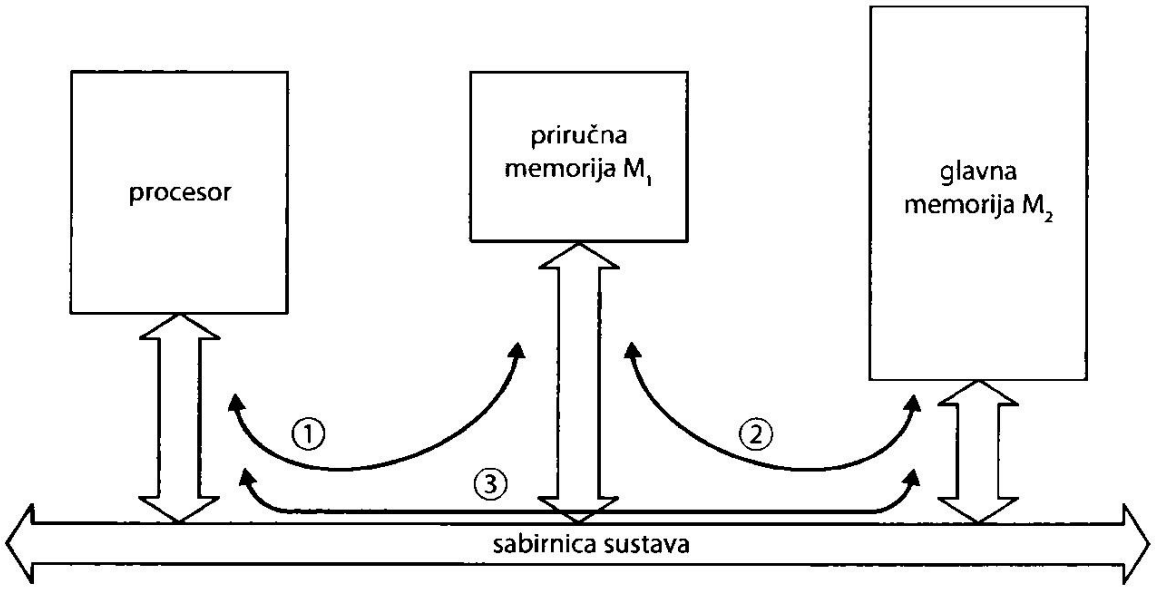
Priručna memorija sastoji se od 3 komponente : memorije za pohranu podataka (engl. *cache data memory*), memorije za pohranu značaka (engl. *cache tag memory*) i sklopova za upravljanje pristupom priručnoj memoriji. Memorija za pohranu podataka služi za pohranu

kopije sadržaja dijela glavne memorije. Ona je organizirana u male blokove koje nazivamo priručni blokovi ili linije. Svaki priručni blok ima značku u kojoj je zapisana njegova bločna adresa pomoću koje se određuje kojem dijelu glavne memorije odgovara odabrani blok. Valja napomenuti kako je glavna memorija podijeljena na blokove, čija veličina je jednaka blokovima priručne memorije. Značke se pohranjuju u memoriju za pohranu znački, a koristi ih sklop za upravljanje pristupom priručnoj memoriji pri utvrđivanju nalazi li se potrebna riječ u nekom od bločnih priključaka priručne memorije. (Ribić, 2011)

2.2. Uključivanje priručne memorije u sustav

Postoje dva glavna oblika uključivanja priručne memorije u sustav : „pogled sa strane“ (engl. *look-aside*) i „pogled kroz“ (engl. *look-through*).

Kod „pogleda sa strane“ priručna memorija i glavna memorija izravno su priključene na sabirnicu sustava kao što je prikazano na slici 2.



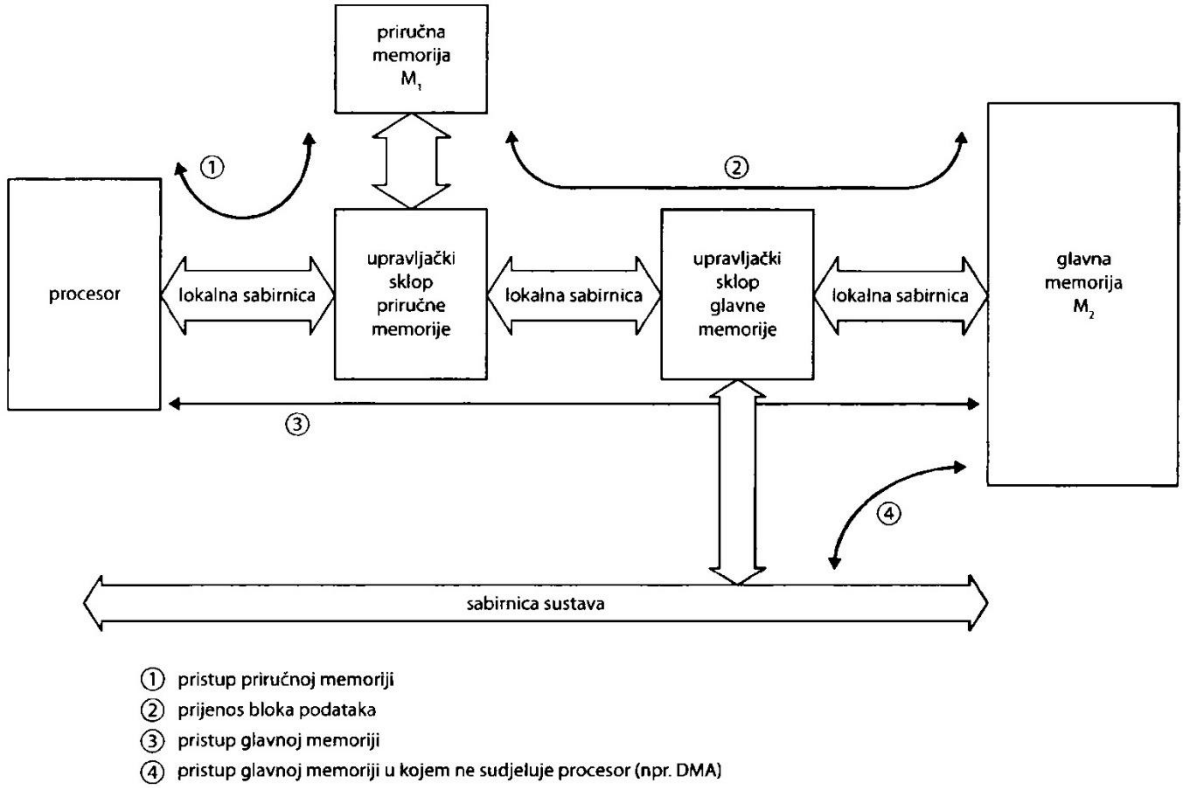
① pristup priručnoj memoriji
② prijenos bloka podataka (prijenos podataka u snopu)
③ pristup glavnoj memoriji

Slika 2. Način uključivanja priručne memorije „pogled sa strane“(Izvor: Ribić 2011, str. 315)

Procesor, kada treba pristupiti memoriji, postavlja adresu na adresnu sabirnicu te šalje upravljački signal R/W kako bi odredio vrstu sabirničkog ciklusa. Adresom koju procesor postavlja na adresnu sabirnicu uvijek se referencira lokacija u glavnoj memoriji. Potom, priručna memorija uspoređuje adresu na sabirnici sa adresama pohranjenim u memoriji za pohranu značaka. Ako dođe do poklapanja tj. pogotka, pristup memoriji završava pisanjem ili

čitanjem u priručnu memoriju bez sudjelovanja glavne memorije u sabirničkom ciklusu. Međutim, dođe li do promašaja, odnosno ne dogodi se preklapanje adrese postavljene na sabirnicu s adresama u memoriji za pohranu značaka, ostvaruje se izravni pristup glavnoj memoriji. Vrši se upis ili čitanje sadržaja sa referencirane memorijske lokacije te se podaci iz glavne memorije prenose u neki bločni priključak priručne memorije. Zajedno s podacima, prenosi se i adresa bloka koja se pohranjuje u adresu za pohranu značaka u odgovarajućem bločnom priključku. Valja napomenuti kako je blok koji se prenosi male veličine pa se prijenos odvija vrlo brzo jer se koristi prijenos podataka u snopu. (Ribić, 2011)

„Pogled kroz“ složeniji je način uključivanja priručne memorije u računarski sustav. Kod njega procesor ima izdvojenu sabirnicu preko koje komunicira s glavnim u priručnom memorijom.



Slika 3. Način uključivanja priručne memorije „pogled kroz“ (Izvor: Ribić 2011, str. 316)

Sabirnicu sustava koriste i druge računarske jedinice što omogućuje istodoban pristup priručnoj memoriji od strane procesora te glavnoj memoriji od strane neke druge jedinice. Pri ovoj organizaciji priručne memorije procesor ne šalje automatski zahtjev glavnoj memoriji nego tek kada se dogodi promašaj kod priručne memorije. Priručna memorija ovakve organizacije

omogućuje izvedbu široke lokalne sabirnice kojom se podaci brže prenose nego što je to slučaj sa organizacijom „pogled sa strane“.

S obzirom na svoju brzinu i način rada, ovakva organizacija priručne memorije zahtjeva složeniji upravljački sklop pa je cijena njezine izvedbe viša. Na slici 3. je prikazan način uključivanja priručne memorije „pogled kroz“. (Ribić, 2011)

2.3. Kapacitet i performanse priručne memorije

Omjer pogodaka je jedan od najbitnijih faktora koji direktno utječu na performanse priručne memorije. Što je više pogodaka tj. manje promašaja, priručna memorija radi brže što omogućuje smanjenje vremena obavljanja operacija na razini cijelog sustava.

Na omjer pogodaka direktno utječu kapacitet priručne memorije i veličina bloka. Veličina bloka podrazumijeva se broj bajtova ili riječi koji su pridruženi svakoj znački. Zahvaljujući lokalnosti programa, povećava li se veličina bloka, raste i omjer pogodaka, ali optimalna se veličina ne može odrediti. Veličina bloka priručne memorije može se kretati od malih blokova kapaciteta svega nekoliko bajtova pa sve velikih blokova veličine od 64 ili čak 128 bajta. Važno je istaknuti da, ako je preveliki kapacitet bloka, može doći do tzv. „onečišćenja priručne memorije“ što se očituje u tome da se neki podaci u bloku nikad ne koriste. (Ribić, 2011; Stallings, 2010)

Za priručnu memoriju vrijed slijedeće : ako ostali parametri ostanu nepromijenjeni, omjer pogodaka će rasti ako će se povećati kapacitet. Međutim, analize su pokazale da omjer pogodaka brzo rase do neke prijelomne vrijednosti kapaciteta priručne memorije kao što je 64KB, a zatim je rast omjera pogodaka usporen. To se događa zbog vremenske lokalnosti programa prema kojem program adresira određeni broj riječi u nekom vremenskom intervalu. Iz toga proizlazi da preveliki kapacitet priručne memorije nema utjecaj na performanse jer se neki podaci koji su smješteni u priručnu memoriju nikad neće koristiti. (Ribić, 2011; Stallings, 2010)

2.4. Razine priručne memorije

Priručnu memoriju možemo klasificirati prema broj razina koje zauzima u memorijskoj hijerarhiji. Pri tome se misli na jednorazinsku i višerazinsku priručnu memoriju.

Jednorazinska priručna memorija koristila se u ranijim generacijama računala. Priručna memorija tada je bila izvedena pomoću većeg broja čipova koji su sa procesorom komunicirali preko vanjske sabirnice. No, kako se tehnologija razvijala, omogućeno je implementiranje priručne memorije na samom procesorskom čipu. Priručna memorija na čipu (engl. *on-chip-cache*) uz to što je brža, u slučaju pogotka, uklonila je potrebu procesora za korištenjem

vanjske sabirnice. Takvu priručnu memoriju nazivamo još L1 priručna memorija. Druga razina priručne memorija (L2 priručna memorija) ima veći kapacitet od priručne memorije L1, a moguća je implementacija izvan procesorskog čipa ili na samom čipu. Razlog njenog uvođenja je da, u slučaju promašaja, procesor ne treba pristupati glavnoj memoriji preko vanjske sabirnice. Na taj se način smanjuju negativni efekti promašaja koji se pojavljuju u jednorazniskoj priručnoj memoriji. S ciljem smanjenja negativnih efekata koji nastaju promašajem u sekundarnoj priručnoj memoriji, uvedena je i treća razina priručne memorije (L3). Njezin kapacitet je nekoliko MB te je također implementirana na procesorski čip. (Ribić, 2011; Stallings, 2010)

2.5. Način smještaja blokova u bločne priključke

Glavni cilj priručne memorije je povećati brinu pristupa podacima koji se nalaze u glavnoj memoriji. To se ostvaruje na način da se dijelovi sadržaja glavne memorije kopiraju u bržu priručnu memoriju. Zbog veće brzine rada, priručna memorija ima višu cijenu izrade po jedinici kapaciteta od glavne memorije. Stoga, kako bi odnos cijene i kvalitete cijelog računala bio optimalan, kapacitet priručne memorije je znatno manji od kapaciteta glavne memorije. Zbog manjeg kapaciteta, svi blokovi iz glavne memorije ne mogu se smjestiti u priručnu memoriju. Taj problem je riješen uvođenjem algoritma koji određuju koji se blokovi iz glavne memorije mogu smjestiti u koje bločne priključke priručne memorije. Za smještanje blokova u bločne priključke priručne memorije postoje tri algoritma: izravno preslikavanje (engl. *direct mapping*), asocijativno preslikavanje (engl. *associative mapping*) i skupno asocijativno preslikavanje (engl. *set associative mapping*). Spomenuti algoritmi predstavljaju važan aspekt u organizaciji priručne memorije. (Ribić, 2011; Stallings, 2010)

2.5.1. Izravno preslikavanje

Izravno preslikavanje je najjednostavniji i najjeftiniji oblik preslikavanja koji ne zauzima previše prostora na čipu. Kod njega se svaki blok glavne memorije može smjestiti u samo jedan točno određeni bločni priključak priručne memorije. (Ribić, 2011; Stallings, 2010)

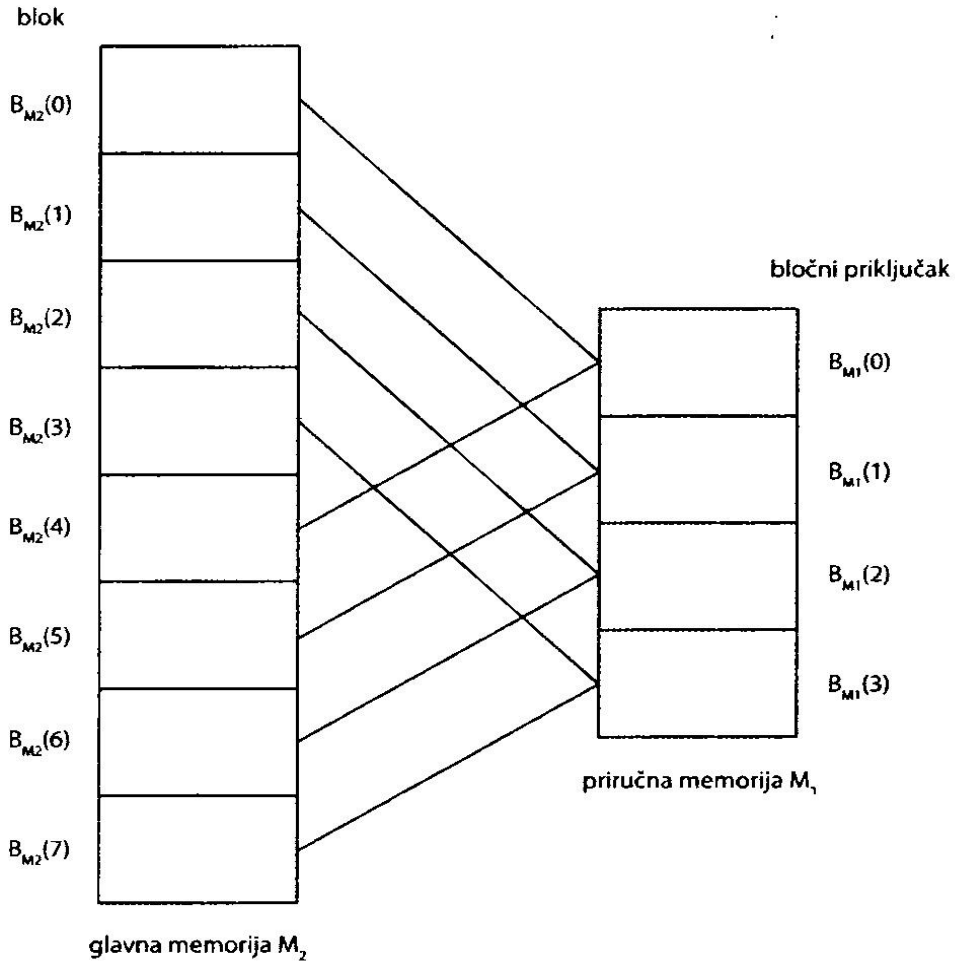
Uzmemo li da nam i označava broj priključnog bloka priručne memorije, j broj bloka glavne memorije, m ukupan broj bločnih priključaka priručne memorije i 2^s ukupan broj blokova glavne memorije, broj bločnog priključka priručne memorije u kojeg će se smjestiti blok iz glavne memorije dobivamo prema formuli:

$$i = j \ modulo \ m \qquad (1)$$

Koristeći spomenutu formulu, dobivamo tablicu 1. koja prikazuje smještanje blokova u bločne priključke. (Ribić, 2011; Stallings, 2010)

Bločni priključak	Blok glavne memorije
0	0, m, 2m, …, 2^s - m
1	1, m+1, 2m+1, …, 2^s – m+1
…	
m-1	m-1, 2m-1, 2m, …, 2^s - 1

Tablica 1. Smještanje blokova u bločne priključke



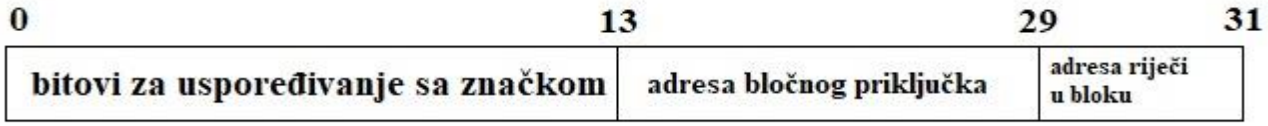
Slika 4. Pojednostavljeni prikaz izravnog smještanja blokova (Izvor: Ribić 2011, str. 324)

Slika 4. prikazuje izravan način smještanja blokova u bločne priključke koje smo dobili prema gore navedenoj formuli. Jasno možemo vidjeti da je $m = 4$, $2^s = 8$. Prema tome u svaki bločni priključak možemo smjestiti 2 bloka. U bločni priključak 0 biti će smješteni blokovi 0 i 4, u bločni priključak 1 blokovi 1 i 5, u bločni priključak 3 blokovi 2 i 6 dok će preostala dva bloka 3 i 7 biti smješteni u bločni priključak 3.

Kako imamo 2^s blokova glavne memorije, a svaki blok ima jedinstvenu adresu, proizlazi da je potrebno s bitova za određivanje adrese. Kod izravnog preslikavanja ta adresa je podijeljena na tri polja. x najmanje značajnih bitova adrese označava riječ u bloku glavne memorije. Ako

blok sadrži 2 riječi dovoljan je 1 bit, ali ako je u bloku 16 riječi potrebna su 4 bita za odabir riječi. Preostalih y bitova služi za referenciranje bloka glavne memorije. Tih y bitova podijeljeno je na dva dijela. z najznačajnijih bitova koristi se za usporedbu sa značkom, dok ostatak označava adresu bločnog priključka.

Uzmimo za primjer da je glavna memorija veličine 4 GB, priručna memorija 256 KB te veličina bloka 4 bajta. Tada dobijemo da je broj blokova glavne memorije 4GB / 4B = 1G odnosno 2^{30}, a broj bločnih priključaka 256 KB / 4B = 64K odnosno 2^{16}. Tako dobivamo da su potrebna 2 bita za odabir riječi u bloku, 16 bita za referenciranje bločnog priključka, dok je preostalih 12 najznačajnijih bitova potrebno za uspoređivanje sa značkom. Takva organizacija adrese prikazana je na slici 5.



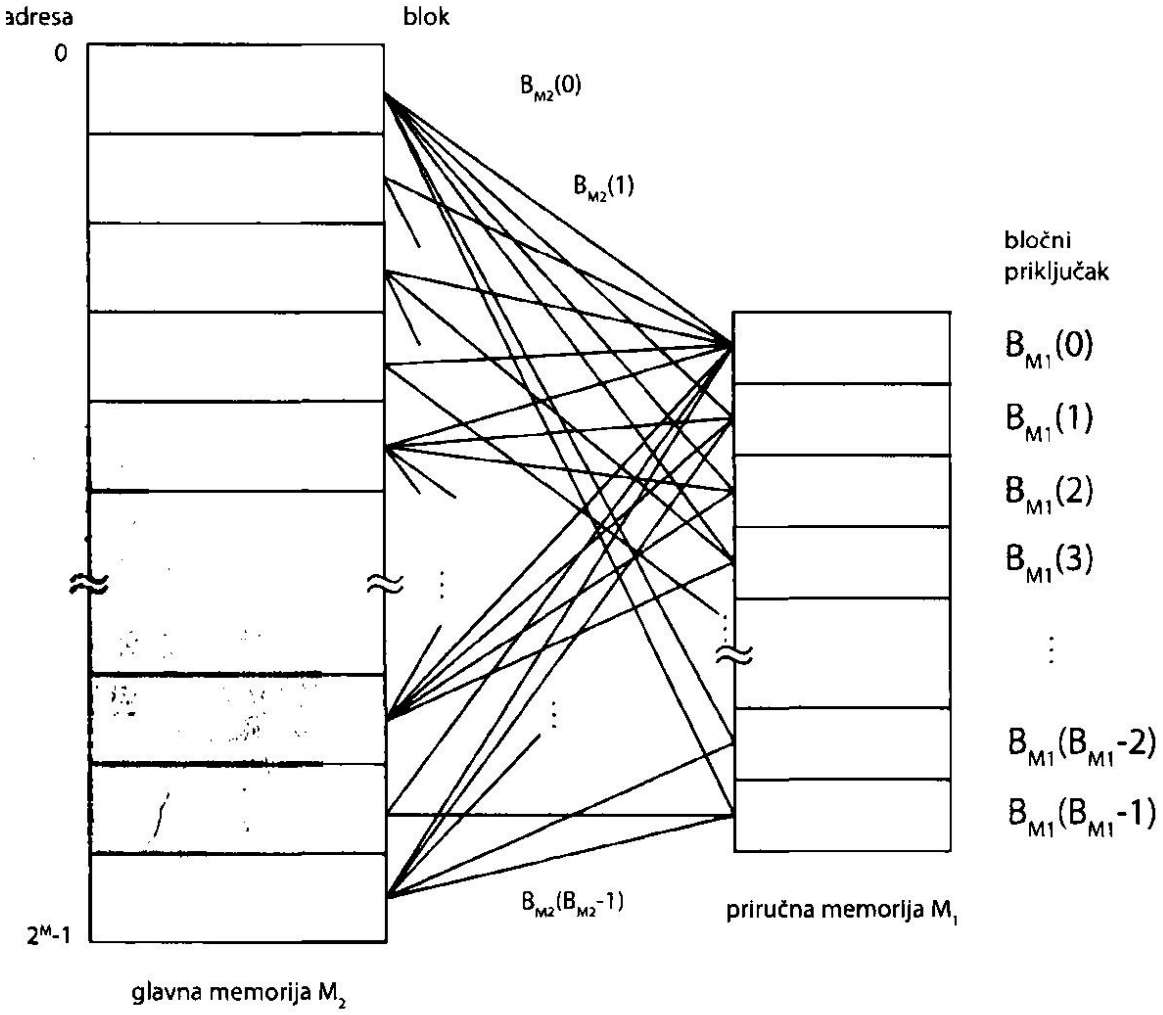
Slika 5. Organizacija 32-bitne adrese kod izravnog preslikavanja

Glavni nedostatak izravnog preslikavanja možemo uočiti primjenjujući formulu za određivanje bločnog priključka - iako imamo slobodne bločne priključke, blok iz glavne memorije mora se smjestiti točno u određeni priključak iako je on zauzet. Teoretski se može dogoditi da će cijelo vrijeme samo jedan bločni priključak biti zauzet dok će svi ostali biti slobodni. Kada bi se to dogodilo priručna memorija ne bi smanjila latentnost te bi tako izgubila svoju svrhu.

2.5.2. Potpuno asocijativno preslikavanje

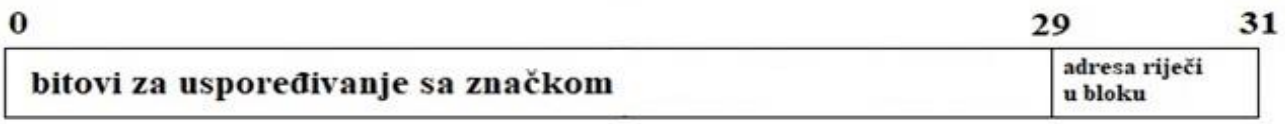
Priručna memorija s potpuno asocijativnim preslikavanje složeniji je oblik priručne memorije koji uklanja nedostatke izravnog preslikavanje. To se postiže tako da se svakom bloku glavne memorije omogućuje smještanje u bilo koji bločni priključak priručne memorije kao što je prikazano na *slici 6* Takvim načinom smještanja blokova u bločne priključke omogućena je velika prilagodljivost priručne memorije te osigurano da do zamjene bloka neće doći sve dok svi bločni priključci nisu popunjeni. (Ribić, 2011; Stallings, 2010)

Kako se svaki blok može smjestiti u bilo koji bločni priključak organizacija adrese za referenciranje memorije je drugačija nego kod izravnog preslikavanje. Kod potpuno asocijativnog preslikavanja adresa se dijeli na dva dijela. x najmanje značajnih bitova adrese označava riječ u bloku glavne memorije, dok se ostatak rabi za usporedbu sa značkom.



Slika 6. Pojednostavljeni prikaz asocijativnog smještanja blokova (Izvor: Ribić 2011, str. 322)

Uzmimo isti primjer koji smo imali kod izravnog preslikavanja .Neka je glavna memorija veličine 4 GB, priručna memorija 256 KB te veličina bloka 4 bajta. Tada dobijemo da je broj blokova glavne memorije 4GB / 4B = 1G odnosno 2^{30}. Veličina bloka je 4 bajta pa su potrebna 2 bita za odabir riječi u bloku. Kako se svaki blok može smjestit u bilo koji bločni priključak, adresa bločnog priključka nije potrebna već nam je značka veličine 30 bita. Takva organizacija adrese prikazana je na *slici 7*.



Slika 7. Organizacija 32-bitne adrese kod potpuno asocijativnog preslikavanja

Glavni nedostatak priručne memorije s potpuno asocijativnim preslikavanjem je složena izvedba koja zauzima veliku površinu na čipu. Zbog toga se takva priručna memorija koristi u izvedbama skupih procesora visokih performansi. (Ribić 2011)

2.5.3. Skupno asocijativno preslikavanje

Skupno asocijativno preslikavanje je kombinacija potpuno asocijativnog i izravnog preslikavanja kojim se pokušavaju riješiti problemi vezani za izravno preslikavanje i složeno sklopovlje kod potpuno asocijativnog preslikavanja. Pri priručnoj memoriji sa skupno asocijativnim preslikavanjem blokovi se smještaju u bločne priključke na sljedeći način : bločni priključci priručne memorije podijeljeni su u skupine k koje sadrže 2^n broj bločnih priključaka te se blok iz glave memorije može smjestiti u bilo koji bločni priključak iz skupine k kao što je prikazano na *slici 9*.

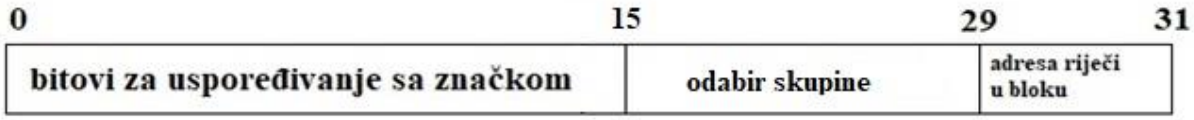
Uzmemo li da nam j označava broj bloka glavne memorije i v ukupan broj skupina bločnih priključaka priručne memorije broj skupine u kojeg će se smjestiti blok iz glavne memorije dobivamo prema formuli:

$$k = j \; modulo \; v \qquad (2)$$

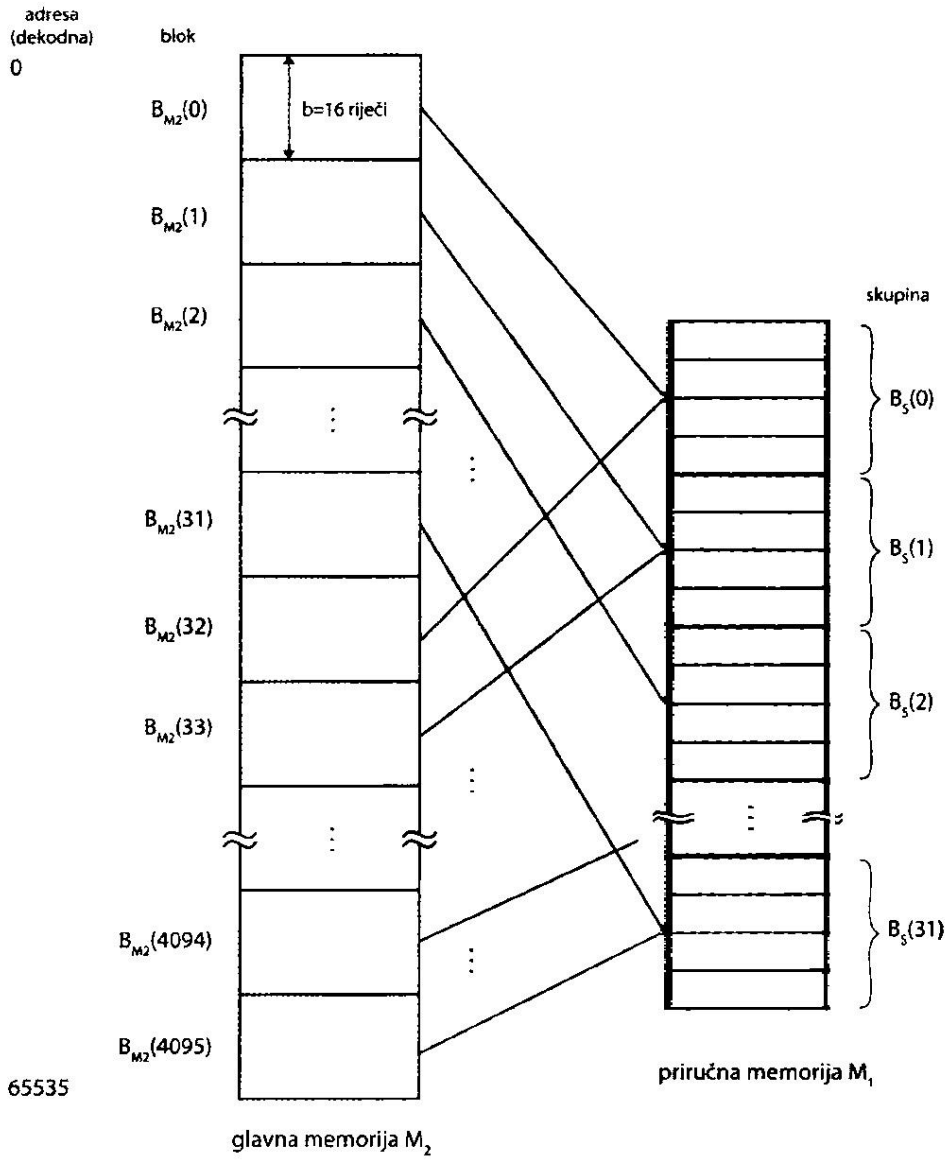
Koristeći gore navedenu formulu, blok iz glavne memorije izravnim preslikavanjem dodjeljujemo skupini bločnih priključaka, pa se zatim potpuno asocijativnim preslikavanjem smješta u slobodni bločni priključak odabrane skupine. (Ribić, 2011; Stallings, 2010)
Kako se svaki blok može smjestiti u bilo koji bločni priključak određene skupine blokova, organizacija adrese za referenciranje memorije je drugačija nego kod izravnog i potpuno asocijativnog preslikavanje. Kod skupnog asocijativnog preslikavanja adresa se dijeli na tri dijela. x najmanje značajnih bitova adrese označava riječ u bloku glavne memorije. Preostalih y bitova služi za referenciranje bloka glavne memorije. Tih y bitova podijeljeno je na dva dijela. z najznačajnijih bitova koristi se za usporedbu sa značkom, dok ostatak označava skupinu blokova u koju se blok smješta.

Uzmimo isti primjer koji smo imali kod izravnog i potpuno asocijativnog preslikavanja. Neka je glavna memorija veličine 4 GB, priručna memorija 256 KB te veličina bloka 4 bajta. Neka je priručna memorija sa 4-putnim skupnim asocijativnim preslikavanjem. Tada dobijemo da je broj blokova glavne memorije 4GB / 4B = 1G odnosno 2^{30}, a broj bločnih priključaka 256 KB / 4B = 64 K odnosno 2^{16}. Podijelimo li ukupna broj bločnih priključaka s 4 dobit ćemo ukupan broj skupina na koje je podijeljena priručna memorija. Tako u ovom slučaju dobivamo 64K/4 = 16K odnosno 2^{14} skupina. Veličina bloka je 4 bajta pa su potrebna 2 bita za odabir riječi u bloku. Tako dobivamo da su potrebna 2 bita za odabir riječi u bloku, 14 bita za

referenciranje skupine bločnih priključaka, dok je preostalih 16 najznačajnijih bitova potrebno za uspoređivanje sa značkom. Takva organizacija adrese prikazana je na *slici 8*.



Slika 8. Organizacija 32-bitne adrese kod 4-putnog skupnog asocijativnog preslikavanja



Slika 9. Pojednostavljeni prikaz 4-putne skupne asocijativne priručne memorije (Izvor: Ribić 2011, str. 328)

2-putna skupna asocijativna priručna memorija najčešće se rabi od skupnih asocijativnih priručnih memorija. Povećanjem kratnosti skupnih asocijativnih priručnih memorija raste omjer pogodaka tj. poboljšavaju se performanse. 4-putne skupne asocijativne memorije umjereno dodatno povećavaju performanse uz relativno malo povećanje cijene izrade, dok svako slijedeće povećanje broja bločnih priključaka po skupini ima mali učinak na performanse memorije. (Ribić, 2011)

2.6. Algoritmi zamjene blokova

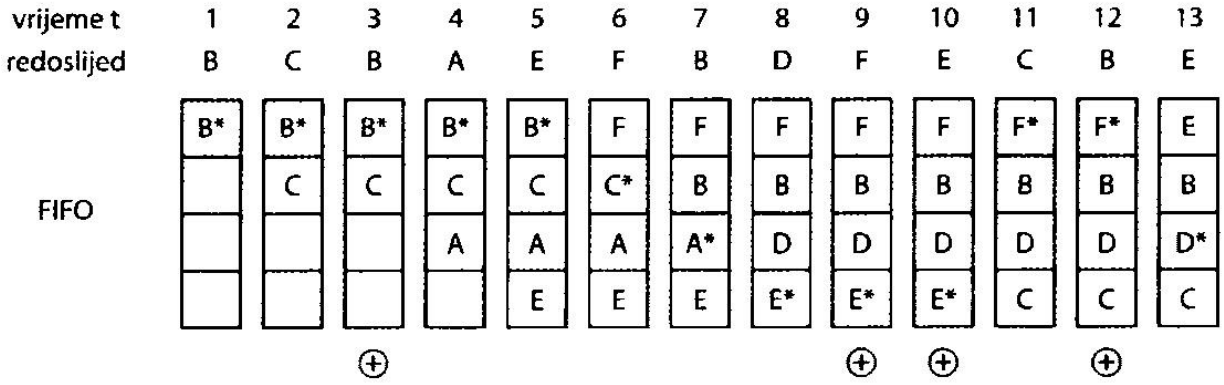
Jednom kada se priručna memorija popuni, a stigne novi blok iz glavne memorije kojeg bi trebalo smjestiti u neki bločni priključak, jedan stari blok treba zamijeniti. Ovisno o izvedbama priručne memorije donosi se odluka koji blok će biti zamijenjen. Kod priručne memorije s izravnim preslikavanjem ta odluka je unaprijed donesena s obzirom da postoji samo jedan točno određeni bločni priključak gdje se novi blok može smjestiti. No, to nije slučaj kod priručne memorije s asocijativnim preslikavanjem, bilo da je riječ o potpuno asocijativnom preslikavanju ili skupno asocijativnom preslikavanju. Kod takvih priručnih memorija uvedeni su algoritmi zamjene blokova. Izbor algoritma zamjene vrlo je bitan jer izravno utječe na performanse priručne memorije. Cilj algoritama zamjene je postizanje što većeg broja pogodaka, odnosno maksimiziranje vremenskog intervala između dva uzastopna promašaja. Valja napomenuti kako se radi postizanje velike brzine, algoritmi zamjene implementiraju sklopovski. (Ribić, 2011; Stallings, 2010)

Najčešće rabljeni algoritmi zamjene blokova su sljedeći : FIFO, LRU, NRU, LFU, Random, LIFO i MRU.

2.6.1. FIFO (First-In-First-Out)

FIFO (engl. *First-in-first-out*), „prvi-unutra-prvi-van“ je algoritam zamjene blokova pri kojemu je kandidat za zamjenu blok koji je najranije ušao u priručnu memoriji. Drugim riječima, zamjenjuje se blok koji je najdulje u priručnoj memoriji. (Ribić, 2011; Stallings, 2010)

Kako bismo prikazali rezultat algoritma FIFO uzmimo primjer sa šest različitih blokova koje označimo s A, B, C, D, E i F. Uzmemo da je slijed njihovih referenciranje B, C, B, A, E, F, B, D, F, E, C. Neka je sa zvjezdicom (*) označen kandidat za izmjenu, a s plusom (+) pogodak. *Slika 9.* prikazuje slijed zamjene za algoritam FIFO. (Ribić, 2011)



Slika 10. Primjer algoritma zamjene FIFO (Izvor: Ribić 2011, str. 330)

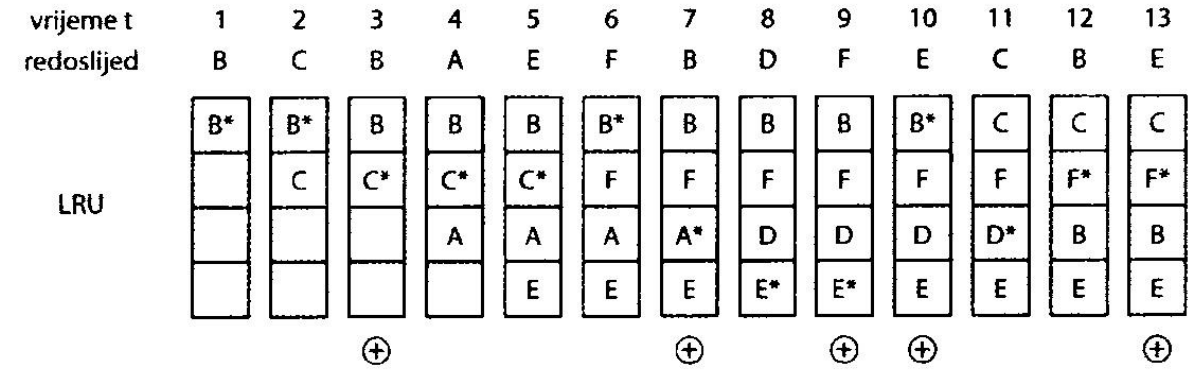
Za navedeni primjer vidimo kako imamo 4 pogotka od ukupno 13 referenciranja tako da je postotak pogodaka malo više od 30%.

2.6.2. LRU (Least Recently Used)

LRU (engl. *Least Recently Used*) je algoritam zamjene blokova pri kojem je kandidat za zamjenu onaj blok koji se nalazi u priručnoj memoriji najduže, a da nije referenciran. Ovaj algoritam bazira se na vremenskoj lokalnosti programa prema kojoj je vrlo mala vjerojatnost da će blok u budućnosti biti referenciran, ako nije bio referenciran u određenom prošlom vremenskom intervalu. (Ribić, 2011; Stallings, 2010)

Kako bismo prikazali rezultat algoritma LRU uzmimo isti primjer od šest različitih blokova koje označimo s A, B, C, D, E i F. Uzmemo da je slijed njihovih referenciranje B, C, B, A, E, F, B, D, F, E, C. Neka je sa zvjezdicom (*) označen kandidat za izmjenu, a s plusom (+) pogodak. *Slika 11.* prikazuje slijed zamjene za algoritam LRU.

Za navedeni primjer vidimo kako imamo 5 pogotka od ukupno 13 referenciranja tako da je postotak pogodaka nešto više od 38%.



Slika 11. Primjer algoritma zamjene LRU(Izvor: Ribić 2011, str. 330)

2.6.3. NRU (Not Recently Used)

NRU (engl. *Not Recently Used*) je aproksimacija algoritam LRU. Kod ovog algoritma zamjene blokova kandidat za zamjenu onaj blok koji u posljednje vrijeme nije bio referenciran. Definira se vremenski interval koji odgovara „posljednjem vremenu“ ovisno o samoj implementaciji. NRU algoritam često je korišten jer je jednostavniji za izvedbu od LRU. (Ribić, 2011)

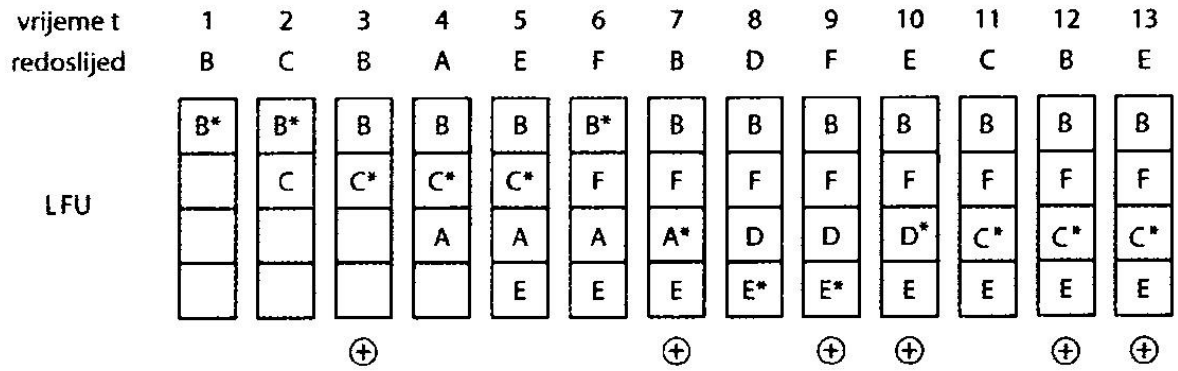
Kada bismo uzeli isti primjer kao do sada, slijed zamjene blokova bio bi isti kao i kod algoritma LRU, što je prikazano na *slici 11.*

2.6.4.LFU (Least Frequently Used)

LFU (engl. *Least Frequently Used*) je algoritam zamjene blokova pri kojem je kandidat za zamjenu onaj blok koji je u posljednje vrijeme bio najmanje puta referenciran. (Ribić, 2011)

Uzmimo isti primjer kao do sada šest različitih blokova A, B, C, D, E i F i slijedom referenciranje B, C, B, A, E, F, B, D, F, E, C. Zvjezdicom (*) označen kandidat za izmjenu, a s plusom (+) pogodak. *Slika 12.* prikazuje slijed zamjene za algoritam LFU.

Za navedeni primjer vidimo kako imamo 6 pogotka od ukupno 13 referenciranja tako da je postotak pogodaka nešto više od 41%.



Slika 12. Primjer algoritma zamjene LFU

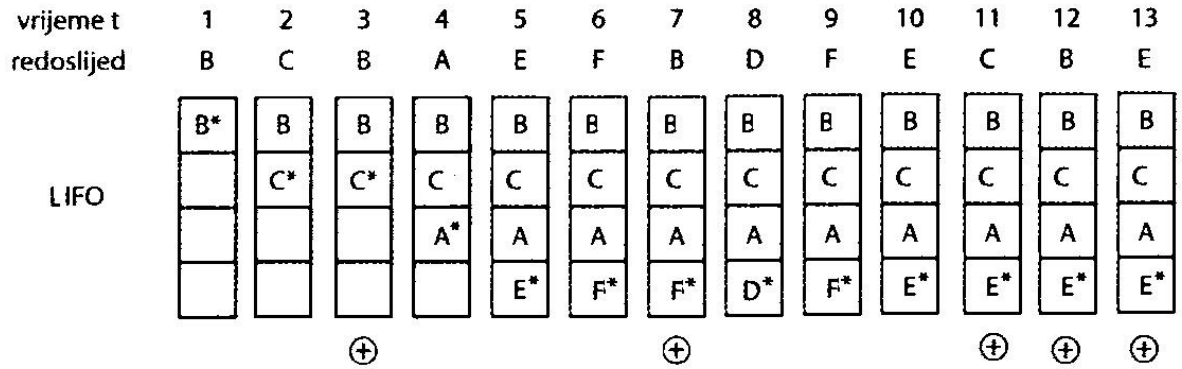
2.6.5. Random

Random je algoritam zamjene blokova pri kojem je kandidat za zamjenu odabran nasumično, bez pravila. Iako se odabir algoritma vrši slučajno, analize su pokazale kako algoritam daje nešto malo slabije rezultate od ostalih algoritama zamjene. (Ribić, 2011)

2.6.6. LIFO (Last-In-First-Out)

LIFO (engl. *Last-In-First-Out*) „zadnji-unutra-prvi-van“ je algoritam u potpunosti suprotan algoritmu FIFO. Ovaj algoritam zamjene blokova zamjenjuje blok koji je najkasnije ušao u priručnu memoriju. Drugim riječima, zamjenjuje se blok koji je najkraće u priručnoj memoriji.

Uzmimo isti primjer kao do sada šest različitih blokova A, B, C, D, E i F i slijedom referenciranje B, C, B, A, E, F, B, D, F, E, C. Zvjezdicom (*) označen kandidat za izmjenu, a s plusom (+) pogodak. *Slika 13*. prikazuje slijed zamjene za algoritam LIFO.



Slika 13. Primjer algoritma zamjene LIFO

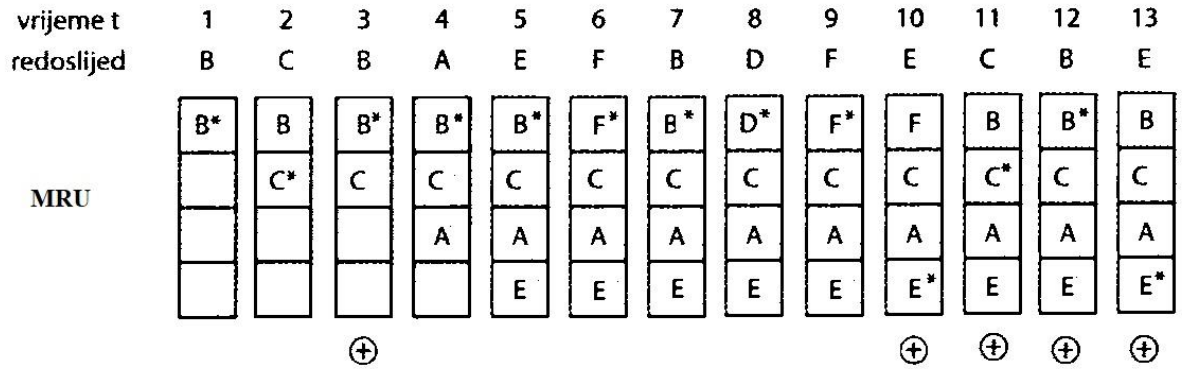
Za navedeni primjer vidimo kako imamo 5 pogotka od ukupno 13 referenciranja tako da je postotak pogodaka nešto više od 38%.

2.6.7. MRU (Most recently used)

MRU (engl. *Most recently used*) je algoritam u potpunosti suprotan algoritmu LRU. Ovaj algoritam zamjene blokova zamjenjuje onaj blok koji se nalazi u priručnoj memoriji najkraće, a da je referenciran najviše puta.

Uzmimo isti primjer kao do sada šest različitih blokova A, B, C, D, E i F i slijedom referenciranje B, C, B, A, E, F, B, D, F, E, C. Zvjezdicom (*) označen kandidat za izmjenu, a s plusom (+) pogodak. *Slika 14*. prikazuje slijed zamjene za algoritam MRU.

Za navedeni primjer vidimo kako imamo 5 pogotka od ukupno 13 referenciranja tako da je postotak pogodaka nešto više od 38%.



Slika 14. Primjer algoritma zamjene MRU

3. Aplikacija

U sklopu završnog rada, uz pisani dio izrađena je i programska aplikacija „Priručna memorija“. Programska aplikacija napisana je u alatu Microsoft Visual Studio 2015 programskim jezikom C#. Svrha programske aplikacije je grafički simulirati rad priručne memorije u nekom računalnom ustavu.

3.1. Opis funkcionalnosti

Aplikacija „Priručna memorija“ sadrži sljedeće funkcionalnosti:

Odabir algoritma zamjene – korisniku je omogućen odabir jednog od tri algoritma zamjene blokova priručne memorije (FIFO, LRU, Random) čiji rad želi simulirati. Odabir jednog algoritama je obavezan kako bi se simulacija izvršila.

Odabir načina smještaja blokova u bločne priključke – korisniku je omogućen odabir načina smještaja blokova u bločne priključke. Implementirani su sljedeći načini preslikavanja : potpuno asocijativno, izravno, skupno asocijativno X2, skupno asocijativno X4 i skupno asocijativno X8. Obavezan je odabir jednog načina preslikavanja kako bi se omogućila simulacija.

Unos broja blokova – korisnik može unesti broj blokova glavne memorije prema kojima želi izvršiti simulaciju. Minimalno je moguće odabrati 10 a maksimalno 1000000 blokova. Unos broja blokova je obavezan.

Odabir broja priključaka – korisnik može odabrati za koliko priključaka priručne memorije želi napraviti simulaciju. Minimalan broj priključaka je 2 a maksimalan 128. Ponuđeni broj priključaka varira ovisno o načinu smještaja blokova u priručnu memoriju. Odabir broja priključaka je obavezan.

Unos broja zahtjeva – korisniku može unesti za koliko zahtjeva za smještaj bloka u priručnu memoriju želi napraviti simulaciju. Minimalan broj zahtjeva je 10, a maksimalan 1000000. Unos broja zahtjeva je obavezan.

Generiranje redoslijeda – korisniku je omogućeno automatsko generirane redoslijeda po kojemu će se blokovi smještati u priručnu memoriju.

Ručno biranje blokova – korisniku je omogućen ručni odabir blokova te stvaranje redoslijeda pram kojemu će se blokovi smještati u priručnu memoriju.

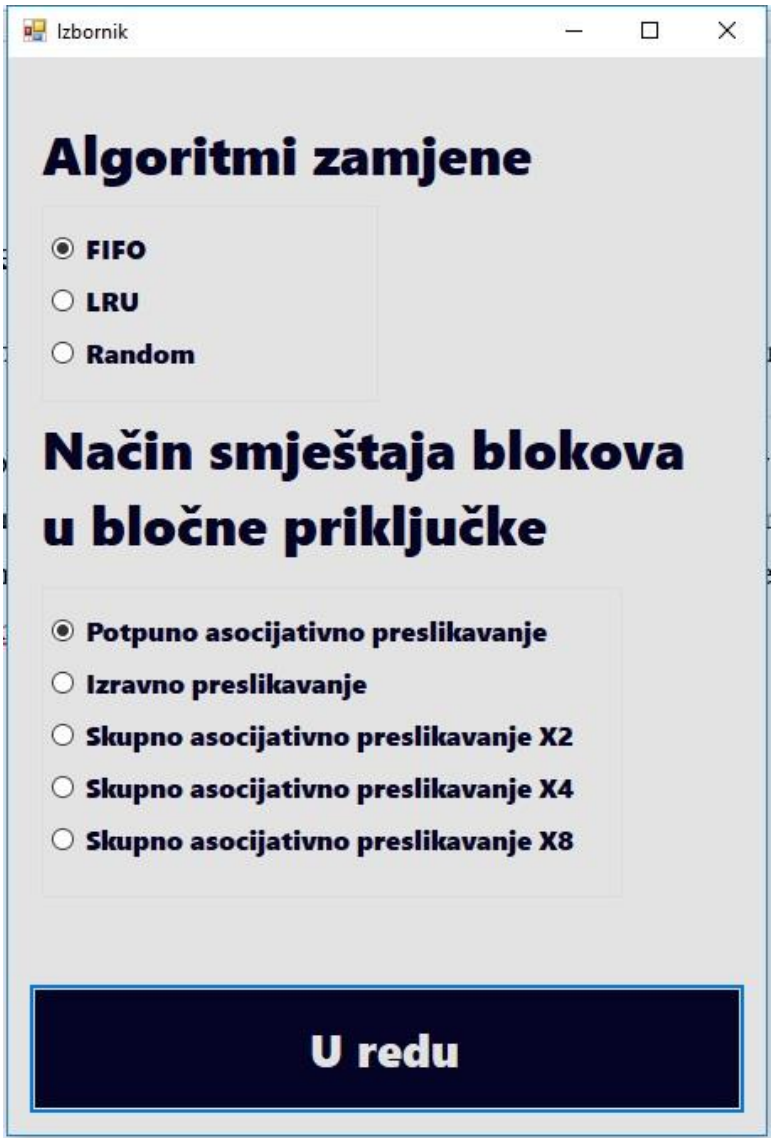
Simulacija – korisnik može pokrenuti simulaciju koja će se izvršiti do kraja.

Sljedeći korak – korisniku je omogućeno simuliranje korak po korak, tj. svakim novim klikom sljedeći blok ulazi u priručnu memoriju.

Inicijalizacija – korisniku je omogućeno, da nakon gotove simulacije, inicijalizira formu te odradi novu simulaciju.

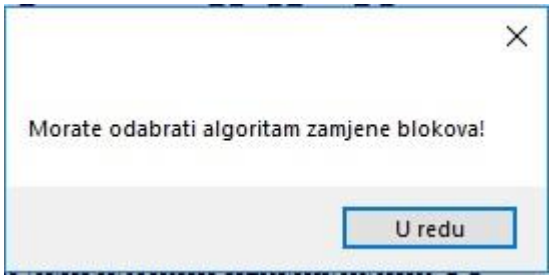
3.2. Opis sučelja

Pokretanjem programske aplikacije „*Priručna memorija*“ otvara se prozor „*Izbornik*“ kao na *slici 15*. Na njemu se nalaze odabir algoritma zamjene blokova, odabir načina smještaja blokova u bločne priključke i gumb za potvrdu odabira.



Slika 15. Izbornik aplikacije „Priručna memorija“

Izbori algoritama ostvareni su pomoću izbornih gumba (engl. *radio button*) te je potrebno izabrati po jedan algoritam iz obje izborne skupine. Ukoliko nije odabran barem jedan algoritam iz obje skupine javlja se poruka o grešci kao što je prikazano na *slici 16*.

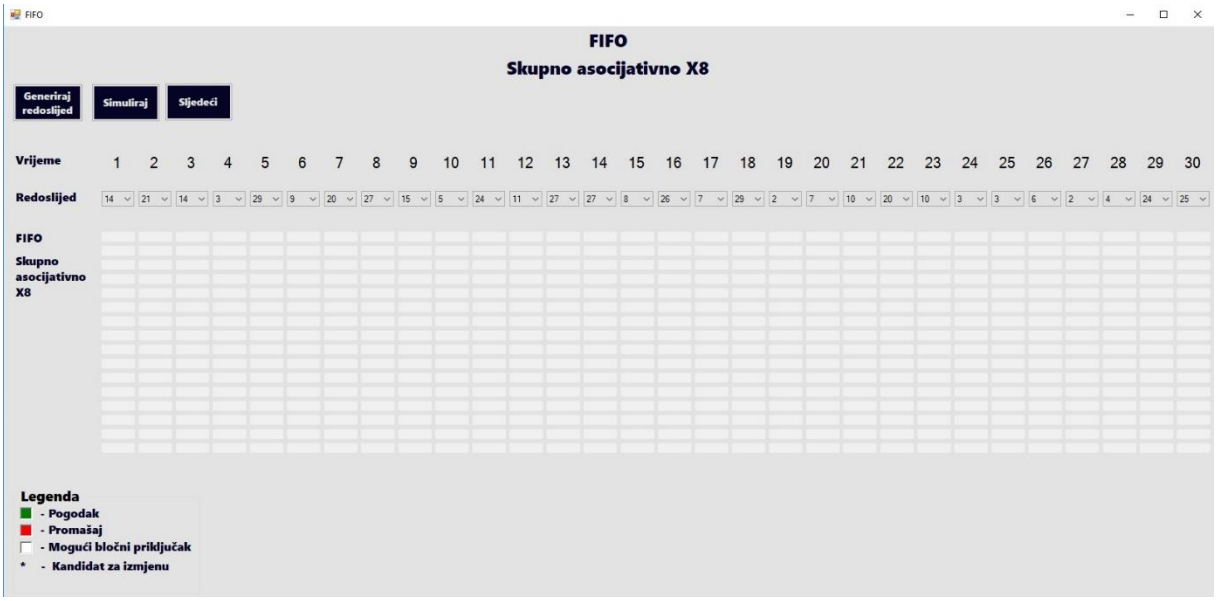


Slika 16. Poruka o grešci

Ukoliko nije odabran algoritam zamjene blokova poruka koja se prikazuje je „*Morate odabrati algoritam zamjene blokova!*“, no ukoliko nije odabran način smještaja blokova u priručnu memoriju poruka glasi „*Morate odabrati način smještaja blokova!*“. Ukoliko je odabran po jedan izborni gumb u svakoj izbornoj grupi otvara se novi prozor kao na *slici 17*.

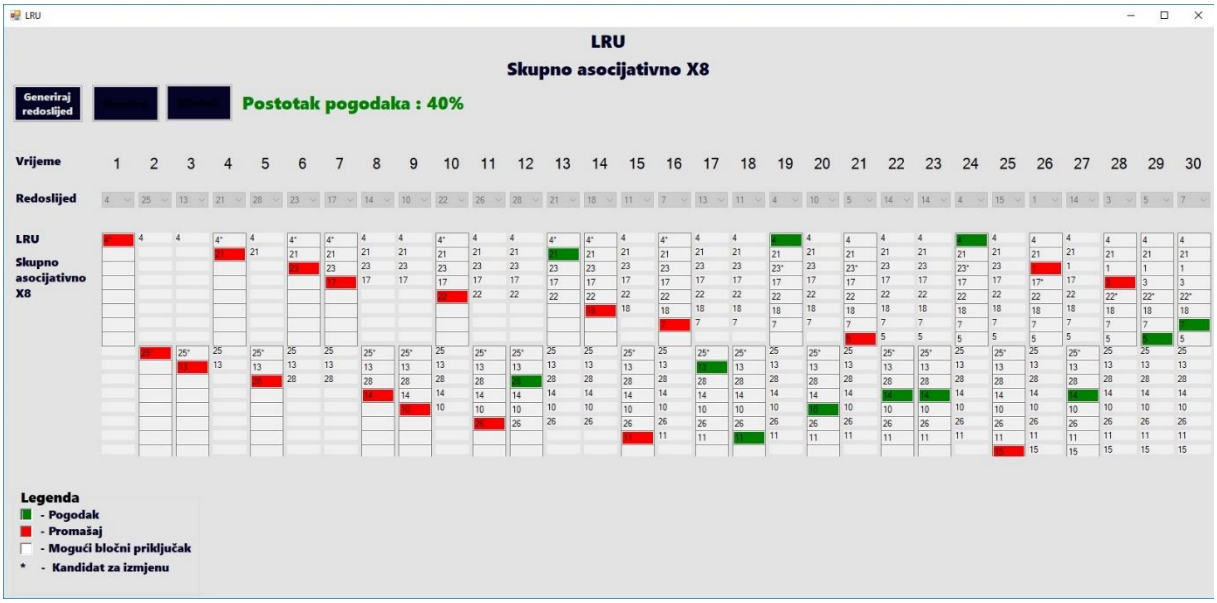
Slika 17. Prozor sa izborom parametara za simulaciju

Novootvoreni prozor omogućuje odabir parametara prema kojima želimo izvršiti simulaciju. Sve parametre je potrebno upisati, odnosno odabrati kako bi simulacija bila pokrenuta. Prvi parametar koji se nalazi na formi je broj blokova. On označava ukupan broj blokova na glavne memorije. Drugi parametar je broj priključaka. Taj parametra predstavlja ukupan broj priključaka priručne memorije u koje se mogu smjestiti blokovi iz glavne memorije. Zadnji parametar je broj zahtjeva. On prezentira koliko koraka će imati simulacija, drugim riječima, koliko će ukupno biti zahtjeva za smještanje blokova iz glavne memorije u neki od bločnih priključaka priručne memorije. Ukoliko broj blokova ili broj zahtjeva nisu upisani ili je upisana nedozvoljeni izraz, kao vrijednost parametara proslijedit će se zadana vrijednost koja iznosi 10. No ukoliko broj priključaka nije odabran otvorit će se prozor kao što je na *slici 16*. Poruka koja će se ispisati je „*Morate odabrati broj priključaka!*“. Kada su svi parametri ispravni otvara se prozor na kojemu se vrši simulacija. Prozor je prikazan na *slici 18*.



Slika 18. Prozor za simulaciju

Na vrhu prozora vidimo nazive dvaju algoritma prema kojima se vrši simulacija, algoritam zamjene blokova i način smještaja blokova u bločne priključke. Ispod naziva algoritama nalaze se tri gumba. Prvi gumb je za automatsko generiranje redoslijeda simulacije. Klikom na njega, ukoliko je simulacija završila kao što je prikazano na *slici 19*, možemo inicijalizirati formu, generirati novi redoslijed te ponoviti simulaciju. Inicijalizacijom forme prozor izgleda kao na *slici 18*. Drugi gumb koji se nalazi na prozoru je „*Simuliraj*“. Klikom na njega simulacija se odradi do kraja kao što je prikazano na *slici 19*.

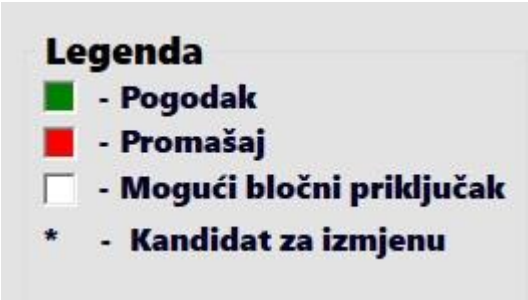


Slika 19. Završena simulacija

Završetkom simulacije, pokraj gumbi prikazuje se postotak pogodaka koji se dogodio tijekom simulacije. Treći gumb na prozoru je „*Sljedeći*“. Pomoću njega moguće je simulirati korak po korak. Svakim novim klikom, sjedeći u nizu blokova ulazi u priručnu memoriju. Nakon svakog koraka izračunava se postotak pogodaka. Ispod gumbi i labele koja označava postotak pogodaka nalazi se vrijeme. Ono označava broj koraka u simulaciji. Ispod vremena nalazi se „*Redoslijed*“. Redoslijed obilježava kojim će se redom blokovi iz glavne memorije smjestiti u bločni priključak priručne memorije. Redoslijed može biti generiran automatski klikom na gumb ili može biti odabran ručno kao što je prikazano na *slici 20*.

Slika 20. Ručni odabir redoslijeda

U sredini prozora nalazi se skup polja koja predstavljaju bločne priključke priručne memorije. Jedan stupac polja prikazuje bločne priključke priručne memorije u jednom koraku. Tako npr. Prvi stupac prikazuje priručnu memoriju nakon prvog koraka simulacije, dok 30 stupac prikazuje priručnu memoriju nakon 30 koraka simulacije. Svakim novim korakom simulacije mijenja se sadržaj priručne memorije. To je prikazano na *slici 19*. U donjem lijevom kutu nalazi se legenda. Prema legendi pogodak je obilježen zelenom pozadinom dok je promašaj obilježen crvenom, mogući bločni priključak za smještanje bloka kod skupnog asocijativnog preslikavanja naznačen je drugačijim obrubom polja, dok je sljedeći kandidat za izmjenu obilježen *. Legenda je prikazana na *slici 21*.



Slika 21. Legenda

4. Prednosti i nedostaci algoritama memorijska organizacije

Kada govorimo o algoritmima memorijske organizacije treba uzeti u obzir način smještaja blokova u priručnu memoriju i algoritam zamjene bloka. Svaki algoritam ima prednosti i nedostatke, a rezultati kombinacija algoritama biti će opisani u ovom poglavlju.

4.1. Teorijska očekivanja

Postotak pogodaka odnosno postotak promašaja, osim o kapacitetu priručne memorije, ovisi o algoritmima njene organizacije. Prema načinu smještaja blokova u bločne priključke postoje 3 algoritma : izravno preslikavanje, potpuno asocijativno preslikavanje i skupno-asocijativno preslikavanje.

Izravno preslikavanje je najjednostavniji oblik preslikavanje. Prednost te vrste preslikavanja je jednostavna izrada sklopa što za sobom povlači jeftiniju cijenu izvedbe. Iako je najjeftinija izvedba priručne memorije, zbog svojih nedostatka nije postala standard ugradnje u računalnoj industriji. Nedostatak izravnog preslikavanja je manji postotak pogodaka odnosno velik broj promašaja, što direktno utječe na performanse memorije. S obzirom na svoju izvedbu, gdje svaki blok ima unaprijed određen bločni priključak gdje se može smjestiti, izravno preslikavanje ne zahtjeva algoritam zamjene blokova, čiji izbor može dodatno poboljšati performanse priručne memorije. Zbog toga se, kod izravnog preslikavanja, može dogoditi da se cijelo vrijeme blokovi smještaju u svega par bločnih priključaka dok su ostali bločni priključci prazni. U tom slučaju koristilo bi se npr. 25% priručne memorije što drugim riječima znači da je tri četvrtine kapaciteta priručne memorije nekorišteno. (Ribić, 2011; Stallings, 2010; Mark D. Hill 1988)

Potpuno asocijativno preslikavanje je potpuna suprotnost direktnog preslikavanja. Kod ove vrste preslikavanja svaki blok može ući u bilo koji bločni priključak tako da se zamjena bloka vrši tek kada su svi bločni priključci priručne memorije popunjeni. Kako nema unaprijed određenog bločnog priključka u koji će se blok smjestiti veliku ulogu u performansama potpuno asocijativne priručne memorije imaju algoritmi zamjene blokova. Zbog toga je postotak pogodaka kod priručne memorije s potpuno asocijativnim preslikavanjem viši u odnosu na izravno preslikavanje. Nedostatak ove vrste preslikavanja je složeni sklop koji ima visoku cijenu implementacije, pa se zbog toga, ovakva organizacija priručne memorije koristi samo u super-računalima. (Ribić, 2011; Stallings, 2010;)

Skupno asocijativna priručna memorija je kompromisno rješenje između izravne priručne memorije s niskim postotkom pogodaka i potpuno asocijativne priručne memorije sa složenim sklopom. Kao kompromis, ovakva priručna memorija ima viši postotak pogodaka od izravne, a jednostavniji sklop od potpuno asocijativne priručne memorije. Implementacija sklopa za skupno asocijativno preslikavanje nije složena stoga ni cijena nije previsoka, a omjer pogodaka je relativno visok. Zbog toga je skupno asocijativna priručna memorija najbolji omjer cijene i kvalitete. Važno je istaknuti da kod skupno asocijativnog preslikavanja, postotak pogodaka raste kako raste kratnost memorije. Tako skupno asocijativna priručna memorija kratnosti dva ima manji postotak pogodaka od skupno asocijativne priručne memorije kratnosti npr. osam. Osim kratnosti, na postotak pogodaka utječi i izbor algoritma zamjene blokova. (Ribić, 2011; Stallings, 2010; Mark D. Hill 1988)

Tri najčešća algoritama zamjene blokova su : LFU, FIFO i Random. Prema analizama algoritma LFU je najefikasniji algoritam zamjene blokova. Njegova efikasnost se temelji na vremenskoj lokalnosti programa. Sljedeći po efikasnosti je FIFO. On zamjenjuje blok koji je najduže u priručnoj memoriji. Random algoritam bira slučajni blok. Prema analizama Random daje slabije rezultate od prethodna dva algoritma zamjene. (Ribić, 2011; Stallings, 2010)

4.2. Način ispitivanja

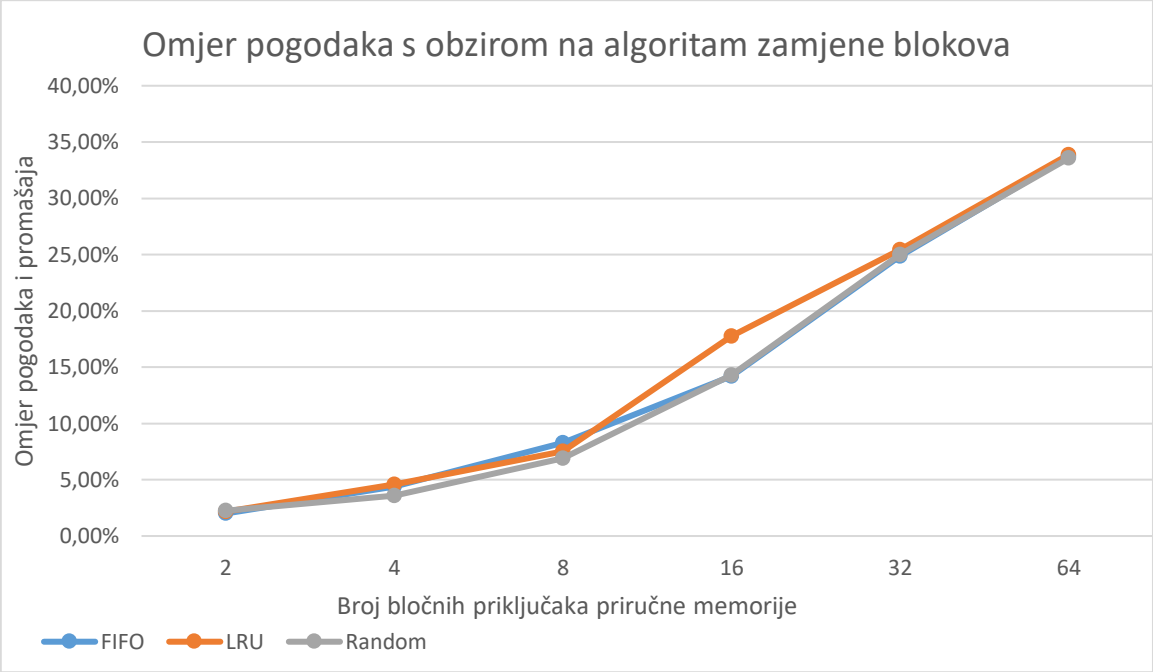
Kako bismo simulirali rad priručne memorije i ispitali teorijska očekivanja algoritama organizacije priručne memorije koristiti ćemo programsku aplikaciju „Priručna memorija“ koja je razvijena u sklopu završnog rada. Aplikacija nudi simulaciju sa pet algoritama preslikavanja: potpuno asocijativno, izravno, skupno asocijativno X2, skupno asocijativno X4 i skupno asocijativno X8 te tri algoritma zamjene blokova: FIFO, LRU i Random. Tako dobivamo ukupno 13 kombinacija zbog toga što izravno preslikavanje ne ovisi o algoritmu zamjene blokova.

Kao parametre simulacije uzeo sam glavu memoriju veličine 100 blokova, priručnu memoriju sa 2, 4, 8, 16, 32 i 64 bločna priključka te 100 zahtjeva za smještanje blokova u priručnu memoriju. Za skupno asocijativno X2 preslikavanje priručna memorija mora imati minimalno 4 priključna bloka, za skupno asocijativno X4 minimalno 8 dok za skupno asocijativno X8 preslikavanje priručna memorija ima minimalno 16 bločna priključka. Za svaki parametar bločnih priključaka priručne memorije odrađeno je 12 simulacija. Tako su kombinacijom svih ulaznih parametara odrađene ukupno 72 simulacije na temelju kojih su dobiveni i analizirani rezultati.

5. Rezultati simulacije algoritama memorijske organizacije

5.1. Analiza rezultata

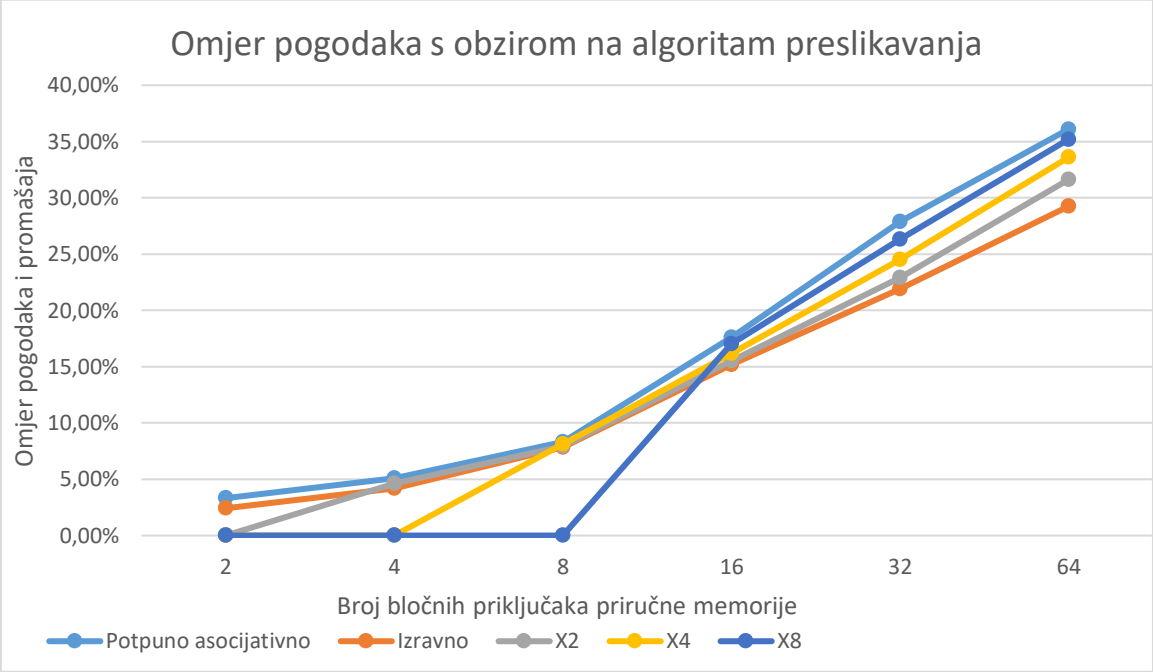
Uzevši u obzir sve simulacije svih načina smještaja, algoritmi zamjene blokova daju podjednake rezultate. Tako je uz zadane parametre simulacije za dva bločna priključka postotak pogodaka oko 2%, za 4 bločna priključka nešto manje od 5%, za 8 priključaka oko 15%, za 32 priključka 25% dok je za 64 bločna priključaka postotak pogodaka 34 %. Rezultati simulacije prikazani su na *slici 22*.



Slika 22. Omjer pogodaka s obzirom na algoritam zamjene blokova

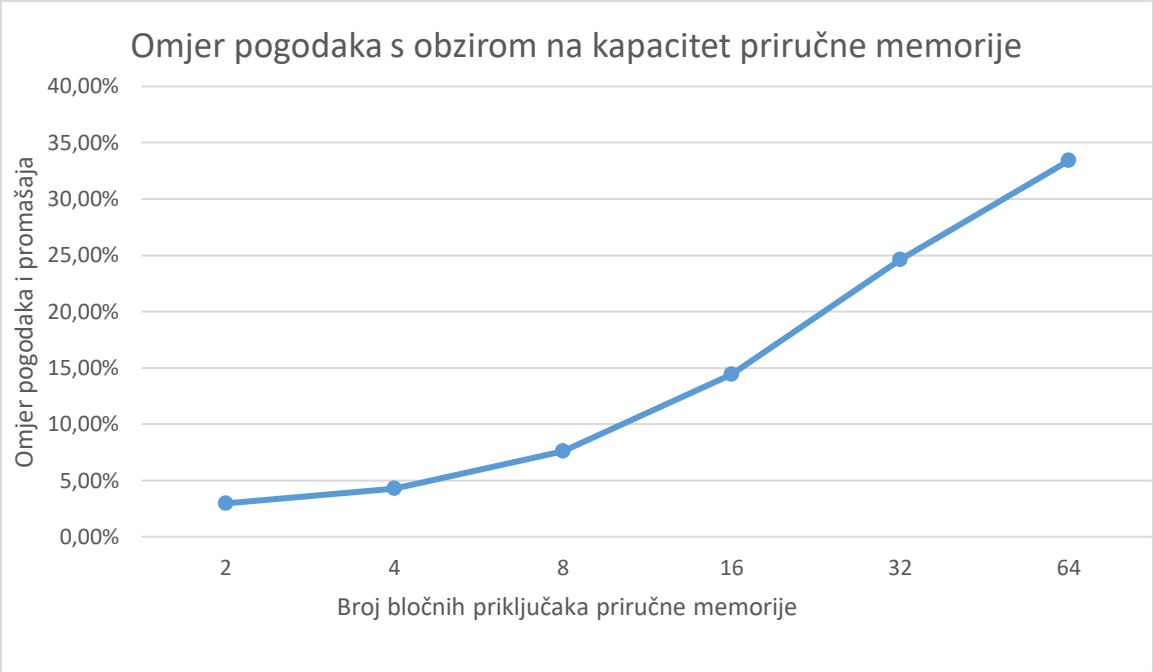
Na *slici 22*. možemo vidjeti kako ja za 16 priključnih blokova omjer pogodaka i promašaja za LRU algoritam zamjene blokova za 3% viši od ostalih algoritama dok je za sve druge brojeve bločnih priključaka omjer podjednak s omjerom ostalih algoritama zamjene blokova.

Simulacija priručne memorije s obzirom na način smještaja blokova dala je sjedeće rezultate. Najveći omjer pogodaka ima potpuno asocijativno preslikavanje, zatim su skupno asocijativna preslikavanja kod kojih se omjer pogodaka smanjuje kako se smanjuje broj blokova u skupini tj. kratnost. Izravno preslikavanje ima najslabiji omjer pogodaka i promašaja. Na *slici 23*. prikazani su rezultati simulacije rada priručne memorije s obzirom na algoritme preslikavanja.



Slika 23. Omjer pogodaka s obzirom na algoritam preslikavanja

S obzirom na broj bločnih priključaka simulacija je dala sljedeće rezultate. Tako je za priručnu memoriju sa 2 priključna bloka, uz zadane parametre simulacije, postotak pogodaka bio 3%, za priručnu memoriju s 4 priključna bloka omjer pogodaka i promašaja vio 5%, za priručnu memoriju s osam bločnih priključaka omjer je bio 8%, za 16 bločnih priključaka 15%, za 32 priključka 25% dok su 64 bločna priključka dala postotak pogodaka 33%. Rezultati simulacije prikazani su na *slici 24*.



Slika 24. Omjer pogodaka s obzirom na kapacitet priručne memorije

5.2. Usporedba rezultata simulacije s teorijskim očekivanima

Rezultati simulacije algoritama zamjene blokova priručne memorije dali su podjednake rezultate. Takvi rezultati nisu u skladu s teorijskim očekivanjima prema kojima je algoritam zamjene blokova LRU trebao dati najbolje rezultate, dok je algoritam zamjene blokova Random trebao imati najslabije rezultate simuliranja. Rezultata simulacije nisu se poklapali s teorijskim očekivanjima zbog toga što nije moguće simulirati vremensku i prostornu lokalnost koja se događa pri radu programa i na kojoj se temelje neki algoritmi zamjene blokova. Kada se parametar vremenske i prostorne lokalnosti izuzme iz simulacije, svi algoritmi zamjene blokova daju podjednake rezultate.

Rezultati simulacije algoritama načina smještaja blokova iz glavne memorije u bločne priključke priručne memorije poklapaju se s teorijskim očekivanjima. Valja napomenuti kako bi algoritmi načina smještaja za čiji rad su potrebni algoritmi zamjene blokova dali i bolje rezultate da se mogla simulirati vremenska i prostorna lokalnost programa.

Rezultati simuliranja rada priručne memorije s obzirom na kapacitet priručne memorije u potpunosti se slažu s teorijskim očekivanima te su potvrdili tezu koja govori kako omjer pogodaka raste kako se povećava kapacitet priručne memorije.

6. Zaključak

Priručna memorija, iako mala, uvelike utječe na rad računarskog sustava. Ona daje novu dimenziju memorijskoj jedinici te proširuje njezinu ulogu u sustavu. Priručna memorija određena je s dvije vrste algoritama: algoritmima smještaja blokova u bločne priključke i algoritmima zamjene blokova. Uz kapacitet, kombinacijom tih algoritama priručnoj memoriju su definirane performanse.

U ovom radu opisana je uloga priručne memorije u memorijskom sustavu. Opisani su algoritmi zamjene blokova kao i algoritmi prema kojima se blokovi smještaju u bločne priključke priručne memorije. Zatim je izrađena aplikacija koja simulira rad priručne memorije s obzirom na algoritme njezine organizacije. Opisana su teorijska očekivanja algoritama te je uz pomoć programa provedena simulacija. Konačno je napravljena analiza simulacije i usporedba s teorijskim očekivanjima.

Svakim danom, razvojem tehnologije, povećava se potreba za što većom i bržom priručnom memorijom. Kako kapacitet priručne memorije može samo donekle utjecati na performanse, važnu ulogu igraju algoritmi organizacije. Simulacijom raznih kombinacija algoritama može se izraditi optimalno rješenje koje bi omogućilo maksimalnu iskoristivost priručne memorije.

Popis literature

[1] Slobodan Ribarić, (2011), *Građa računala - arhitektura i organizacija računarskih sustava*, Algebra, Zagreb

[2] William Stallings, (2010), *Computer organization and architecture designing for performance - eighth edition*, Prentice Hall, Upper Saddle River, NJ

[3] Mark D Hill, (1988), *Aspects of Cache Memory and Instruction Buffer Performance*, preuzeto 1.9. 2018. s http://www.dtic.mil/dtic/tr/fulltext/u2/a604007.pdf

Popis slika

Popis tablica