# 1. Memorije

## *1.1 Keš memorija*

### Zadatak 1.1.1

Operativna memorija računara je kapaciteta 1 Mbajt, a širina reči iznosi 1 bajt.

a) Nacrtati strukturu keš memorije, označiti širinu u bitovima svih relevantnih delova i ukratko opisati njihovu namenu za keš memoriju sa 1024 bloka, blokom veličine četiri bajta i asocijativnim preslikavanjem pretpostavljajući da je širina reči dela keš memorije u kome se čuva sadržaj jedan bajt.

b) Objasniti kako se za slučaj čitanja bajtovske veličine utvrđuje da li se sadržaj sa generisane adrese nalazi u keš memoriji i kako se vrši samo očitavanje traženog sadržaja.

c) Sadržaj relevantnih lokacija operativne memorije je dat na slici **Error! Reference source not found.**. Četiri operacije čitanja iz operativne memorije sa lokacija 5122d, 1025d, 8192d i 4099d se izvode u datom redosledu. Navesti vrednosti svih relevantnih delova keš memorije posle ove četiri operacije, pretpostavljajući da je keš memorija na početku bila prazna.

| **adresa** | … | 1024 | 1025 | 1026 | 1027 | … | 4096 | 4097 | 4098 | 4099 | … | 5120 | 5121 | 5122 | 5123 | … | 8192 | 8193 | 8194 | 8195 | … |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **sadržaj** | … | 0 | 1 | 0 | 0 | … | 0 | 0 | 0 | 2 | … | 0 | 0 | 3 | 0 | … | 4 | 0 | 0 | 0 | … |

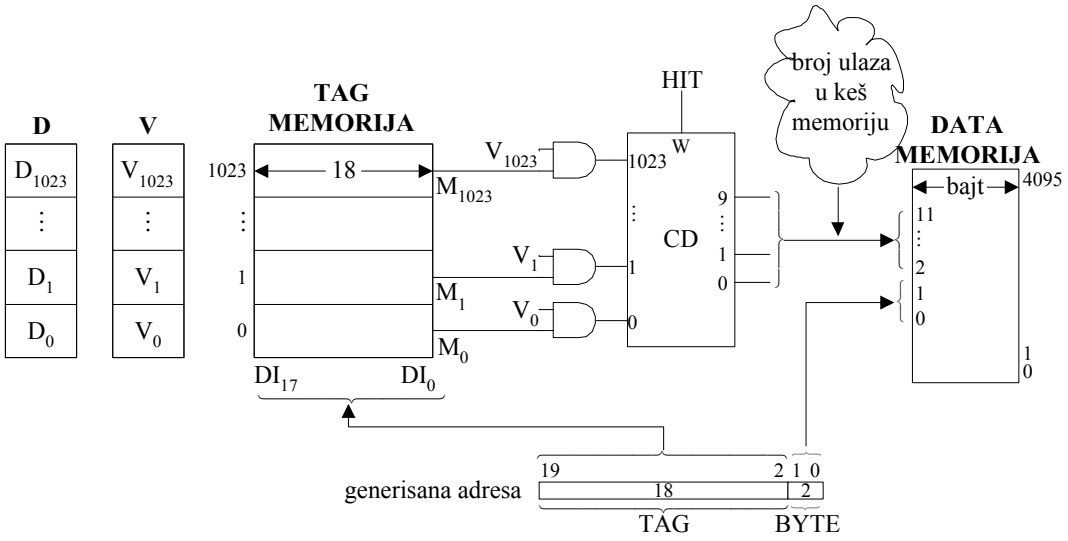
**Slika 1.1.1.** Sadržaj relevantnih lokacija operativne memorije

**Rešenje:**

a) S obzirom da je kapacitet operativne memorije 1 Mbajt ($2^{20}$ bajta) i da je veličina bloka četiri bajta ($2^2$ bajta), operativna memorija sadrži $2^{18}$ blokova. Stoga je struktura generisane adrese kao na slici 1.1.2 i sastoji se iz:

- TAG (18 bita)—broj bloka operativne memorije i
- BYTE (2 bita)—adresa bajta u bloku.

Na osnovu ovakve strukture generisane adrese i zadatih karakteristika keš memorije dolazi se do strukture keš memorije prikazane na istoj slici.



**Slika 1.1.2.** Struktura keš memorije sa asocijativnim preslikavanjem

Keš memorija se sastoji iz sledećih delova:

- $D_{0...1023}$ (dirty bitovi)—1024 flip-flopa,
- $V_{0...1023}$ (valid bitovi)—1024 flip-flopa,
- TAG MEMORIJA—asocijativna memorija kapaciteta 1024 reči širine 18 bita,
- CD—koder 1024/10 i
- DATA MEMORIJA—RAM memorija kapaciteta 1024 bloka.

Dirty bitovi označavaju za svaki od 1024 ulaza keš memorije da li je bilo upisa u odgovarajući blok DATA MEMORIJE od kada je blok doveden u taj ulaz.

Valid bitovi označavaju za svaki od 1024 ulaza keš memorije da li je odgovarajući ulaz TAG MEMORIJE važeći.

TAG MEMORIJA služi za čuvanje 1024 TAG polja blokova sadržaja koji se nalaze u odgovarajućim ulazima DATA MEMORIJE i generisanje aktivnih vrednosti signala saglasnosti $\mathbf{M_{0...1023}}$ ukoliko postoji saglasnost TAG polja generisane adrese i sadržaja odgovarajućeg ulaza TAG MEMORIJE.

CD služi za generisanje aktivne vrednosti signala saglasnosti **HIT** i broja ulaza u keš memoriju za koji je otkrivena saglasnost.

DATA MEMORIJA služi za čuvanje 1024 bloka sadržaja.

b) TAG bitovi (18) iz generisane adrese se porede sa sadržajima sva 1024 ulaza u TAG MEMORIJI. Ako se sadržaj bilo kojeg ulaza slaže sa TAG bitovima generisane adrese i odgovarajući V bit je postavljen, signal saglasnosti HIT postaje aktivan.

Bitovi (10) koji označavaju broj ulaza u keš memoriju gde je otkrivena saglasnost dobijeni sa izlaza kodera i BYTE (2) bitovi generisane adrese se koriste kao adresa u DATA MEMORIJI i sadržaj se čita.

c) Struktura generisane adrese i vrednosti polja TAG i BYTE za četiri zadate generisane adrese su dati na slici 1.1.3.

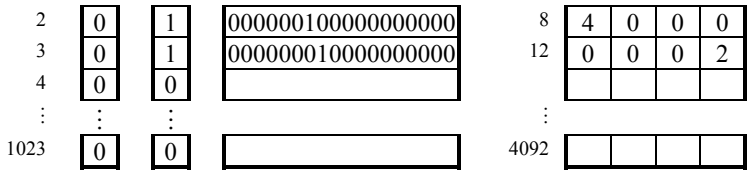
| | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1025 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4099 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 5122 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 8192 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | ←——————————————————————tag——————————————————————→ | | | | | | | | | | | | | | | | | | ←byte→ | |

**Slika 1.1.3.** Struktura generisanih adresa

Postoji 256 K blokova u operativnoj memoriji i 1 K blokova u keš memoriji. Bilo koji blok operativne memorije se može smestiti u bilo koji blok keš memorije. Adrese operativne memorije 5122d, 1025d, 8192d i 4099d su u blokovima 500h, 100h, 800h i 400h operativne memorije, respektivno. Ovi blokovi se smeštaju u blokove 0, 1, 2 i 3 keš memorije, respektivno. Vrednosti svih relevantnih delova keš memorije posle četiri operacije čitanja prikazani su na slici **Error! Reference source not found.**.

| broj ulaza | D | V | TAG MEMORIJA |
|---|---|---|---|
| 0 | 0 | 1 | 000000010100000000 |
| 1 | 0 | 1 | 000000000100000000 |

| adresa | DATA MEMORIJA | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 3 | 0 |
| 4 | 0 | 1 | 0 | 0 |




**Slika 1.1.4.** Sadržaj relevantnih delova keš memorije

## Zadatak 1.1.2

Operativna memorija računara je kapaciteta 1 Mbajt, a širina reči iznosi 1 bajt.

a) Nacrtati strukturu keš memorije, označiti širinu u bitovima za sve relevantne delove i ukratko opisati njihovu namenu za keš memoriju sa 1024 bloka, blokom veličine četiri bajta i direktnim preslikavanjem pretpostavljajući da je širina reči dela keš memorije u kome se nalazi sadržaj jedan bajt.

b) Isto kao pod 0.1b.

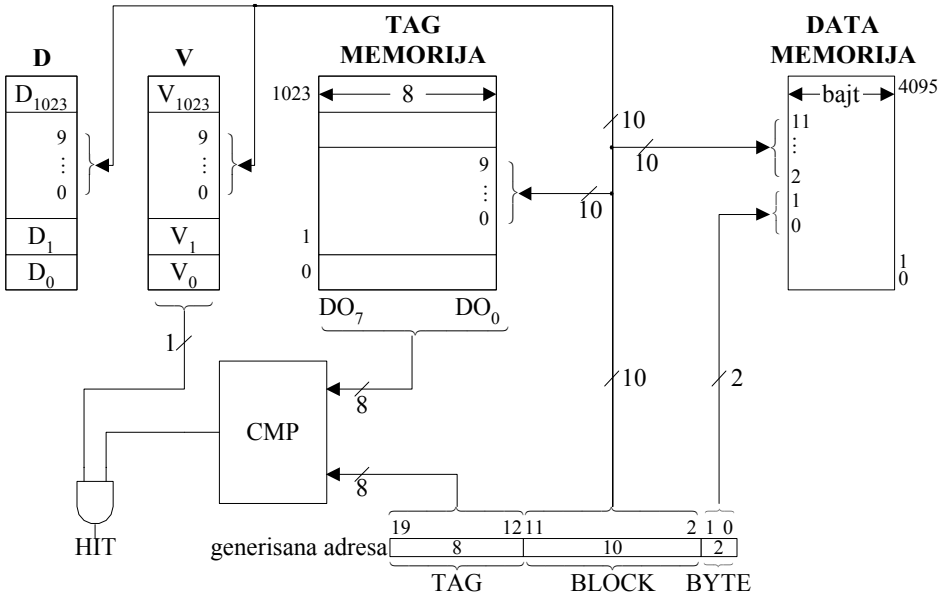
c) Isto kao pod 0.1c.

**Rešenje:**

a) S obzirom da je kapacitet operativne memorije 1 Mbajt ($2^{20}$ bajta) i da je veličina bloka četiri bajta ($2^2$ bajta), operativna memorija sadrži $2^{18}$ blokova. Kako u keš memoriju mogu da se smeste 1024 bloka ($10^{10}$ blokova), to se kod direktnog preslikavanja uzima da je operativna memorija podeljena na 256 grupa ($2^8$ grupa) veličine 1024 bloka ($10^{10}$ blokova). Stoga je struktura generisane adrese kao na slici 1.1.5 i sastoji se iz:

- TAG (8 bita)—broj grupe operativne memorije,
- BLOCK (10 bita)—broj bloka unutar grupe i broj bloka u keš memoriji i
- BYTE (2 bita)—adresa bajta unutar bloka.

Na osnovu ovakve strukture generisane adrese i zadatih karakteristika keš memorije dolazi se do strukture keš memorije prikazane na istoj slici.




**Slika 1.1.5.** Struktura keš memorije sa direktnim preslikavanjem

Keš memorija se sastoji iz sledećih delova:

- $D_{0...1023}$ (dirty bitovi)—1024 flip-flopa,
- $V_{0...1023}$ (valid bitovi)— 1024 flip-flopa,
- TAG MEMORIJA—RAM memorija kapaciteta 1024 reči širine 8 bita,
- CMP—komparator i
- DATA MEMORIJA—RAM memorija kapaciteta 1024 bloka.

Dirty bitovi označavaju za svaki od 1024 ulaza keš memorije da li je bilo upisa u odgovarajući blok DATA MEMORIJE.

Valid bitovi označavaju za svaki od 1024 ulaza keš memorije da li je odgovarajući ulaz TAG MEMORIJE važeći.

TAG MEMORIJA služi za čuvanje 1024 TAG polja blokova sadržaja koji se nalaze u odgovarajućim ulazima DATA MEMORIJE.

CMP služi za generisanje aktivne vrednosti signala saglasnosti **HIT**. Ovaj signal se aktivira ako:

- TAG polje generisane adrese je isto kao sadržaj TAG MEMORIJE adresiran poljem BLOCK generisane adrese i
- V bit adresiran poljem BLOCK generisane adrese je postavljen.

DATA MEMORIJA služi za čuvanje 1024 bloka sadržaja.

b) BLOCK bitovi (10) iz generisane adrese se koriste kao adresa za TAG MEMORIJU. TAG bitovi očitani iz TAG MEMORIJE sa porede sa TAG bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i ako je V bit, adresiran BLOCK bitovima generisane adrese, postavljen, aktivira se signal HIT.

BLOCK bitovi (10) i BYTE bitovi (2) generisane adrese se koriste kao adresa za DATA MEMORIJU i sadržaj se čita.

c) Struktura generisane adrese i vrednosti polja TAG, BLOCK i BYTE generisanih adresa dati su na slici 1.1.6.

| | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1025 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4099 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 5122 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 8192 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | ←tag→ | | | | | | | | ←block→ | | | | | | | | | | ←byte→ | |

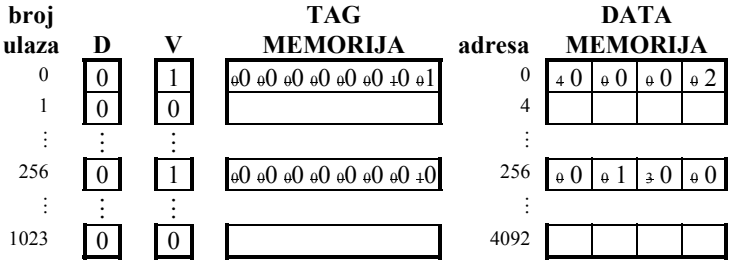
**Slika 1.1.6.** Struktura generisanih adresa

Postoji 256 K blokova u operativnoj memoriji i 1 K blokova u keš memoriji. Blokovi operativne memorije su organizovani u 256 grupa sa 1 K blokova po grupi. Na osnovu toga se utvrđuje da je:

- adresa operativne memorije 5122d u bloku 100h i grupi 1,
- adresa operativne memorije 1025d u bloku 100h i grupi 0,
- adresa operativne memorije 8192d u bloku 0h i grupi 2 i
- adresa operativne memorije 4099d u bloku 0h i grupi 1.

Blok 100h keš memorije se prvo napuni blokom 100h iz grupe 1 operativne memorije (adresa 5122d), a zatim se preko njega prepiše blok 100h iz grupe 0 (adresa 1025d). Blok 0h

keš memorije se prvo napuni blokom 0h iz grupe 2 operativne memorije (adresa 8192d), a zatim se preko njega prepiše blok 0h iz grupe 1 (adresa 4099d). Vrednosti svih relevantnih delova keš memorije posle četiri operacije čitanja prikazani su na slici **Error! Reference source not found.**.




**Slika 1.1.7.** Sadržaj relevantnih delova keš memorije

## Zadatak 1.1.3

Operativna memorija računara je kapaciteta 1 Mbajta, a širina reči iznosi 1 bajt.

a) Nacrtati strukturu keš memorije, označiti širinu u bitovima za sve relevantne delove i ukratko opisati njihovu namenu za:

a1) keš memoriju sa 1024 bloka, blokom veličine četiri bajta i set-asocijativnim preslikavanjem sa dva bloka po setu i

a2) keš memoriju sa 1024 bloka, blokom veličine četiri bajta i set-asocijativnim preslikavanjem sa četiri bloka po setu,

pretpostavljajući da je širina reči dela keš memorije u kome se nalazi sadržaj jedan bajt

b) Objasniti kako se za oba tipa keš memorije za slučaj čitanja bajtovske veličine utvrđuje da li se sadržaj sa generisane adrese nalazi u keš memoriji i kako se vrši samo očitavanje traženog sadržaja.

c) Isto kao pod 0.1c.

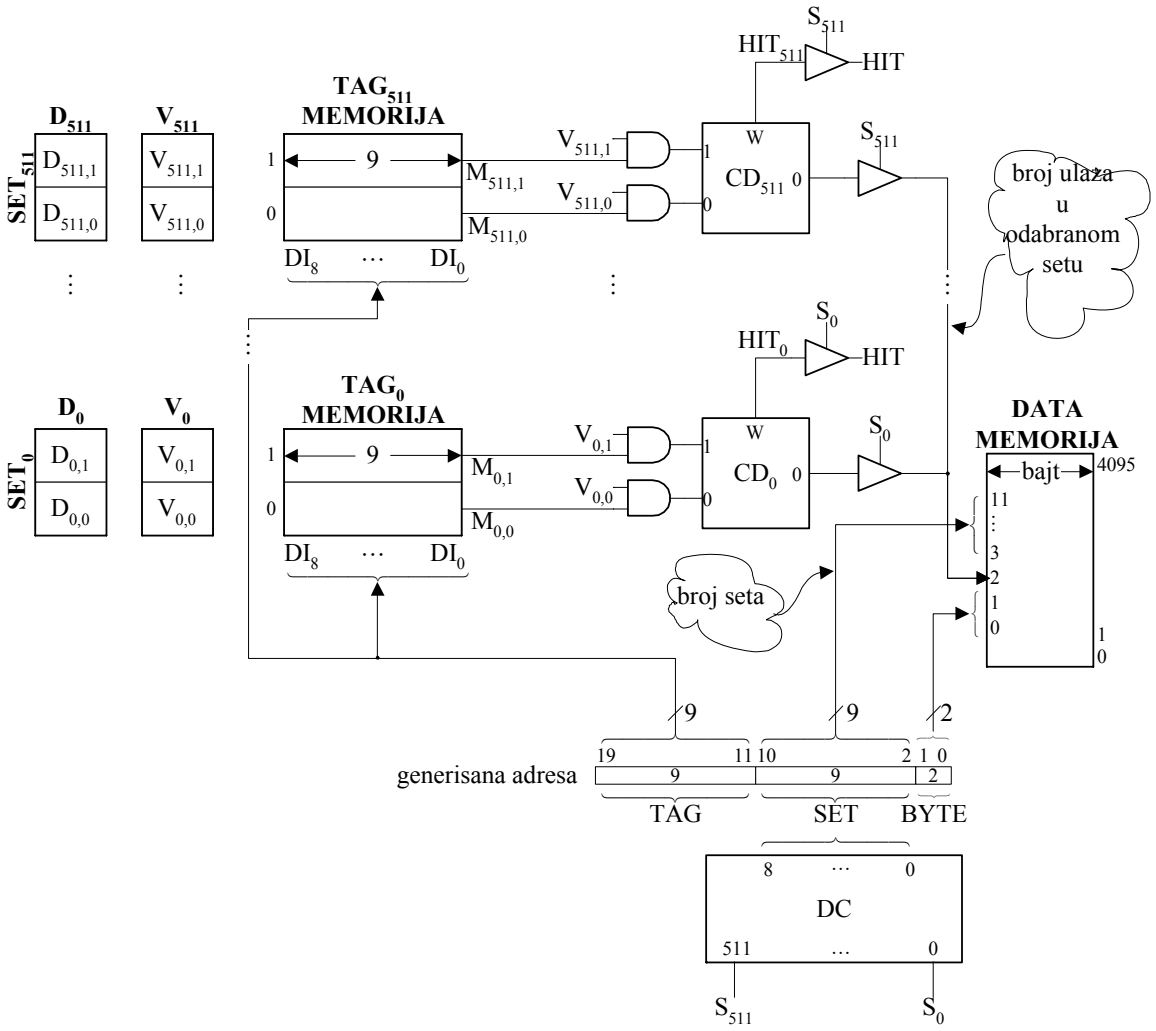
**Rešenje:**

a) Strukture keš memorija sa set-asocijativnim preslikavanjem sa dva i četiri bloka po setu su date u posebnim odeljcima.

a1) S obzirom da je kapacitet operativne memorije 1 Mbajt ($2^{20}$ bajta) i da je veličina bloka četiri bajta ($2^2$ bajta), operativna memorija sadrži $2^{18}$ blokova. Kako u keš memoriju mogu da se smeste 1024 bloka ($10^{10}$ blokova) organizovanih set-asocijativno sa dva bloka po setu, to keš memorija sadrži 512 setova ($2^9$ setova). Stoga se ovde uzima da je operativna memorija podeljena na 512 grupa ($2^9$ grupa) veličine 512 blokova ($2^9$ blokova). Stoga je struktura generisane adrese kao na slici 1.1.8 i sastoji se iz:

- TAG (9 bita)—broj grupe operativne memorije,
- SET (9 bita)—broj bloka unutar grupe i broj seta unutar keš memorije i
- BYTE (2 bita)—adresa bajta u bloku.

Na osnovu ovakve strukture generisane adrese i zadatih karakteristika keš memorije dolazi se do strukture keš memorije prikazane na istoj slici.




**Slika 1.1.8.** Struktura keš memorije sa set-asocijativnim preslikavanjem sa dva bloka po setu

Keš memorija se sastoji iz sledećih delova:

- $D_{0,0}$, $D_{0,1}$, $D_{1,0}$, $D_{1,1}$, …, $D_{511,0}$, $D_{511,1}$ (dirty bitovi)—1024 flip-flopa organizovanih u 512 setova sa po dva flip-flopa po setu,
- $V_{0,0}$, $V_{0,1}$, $V_{1,0}$, $V_{1,1}$, …, $V_{511,0}$, $V_{511,1}$ (valid bitovi)—1024 flip-flopa organizovanih u 512 setova sa po dva flip-flopa po setu,
- $TAG_0$ MEMORIJA, $TAG_1$ MEMORIJA, …, $TAG_{511}$ MEMORIJA—512 asocijativnih memorija svaka kapaciteta dve reči širine 9 bita,
- $CD_0$, $CD_1$, …, $CD_{511}$—512 kodera 2/1,
- DC—dekoder 9/512 i
- DATA MEMORIJA—RAM memorija kapaciteta 1024 bloka.

Dirty bitovi seta *i* označavaju za svaki od dva ulaza seta *i* keš memorije da li je bilo upisa u odgovarajući blok DATA MEMORIJE.

Valid bitovi seta *i* označavaju za svaki od dva ulaza seta *i* keš memorije da li je odgovarajući ulaz TAG MEMORIJE važeći.

$TAG_i$ MEMORIJA služi za čuvanje dva TAG polja seta *i* blokova sadržaja koji se nalaze u odgovarajućim ulazima DATA MEMORIJE i generisanje aktivnih vrednosti signala

saglasnosti $\mathbf{M_{i,0}}$ i $\mathbf{M_{i,1}}$ ukoliko postoji saglasnost TAG polja generisane adrese i sadržaja odgovarajućeg ulaza TAG MEMORIJE.

$CD_i$ služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT_i}$ i broja ulaza u keš memoriju za koji je otkrivena saglasnost.

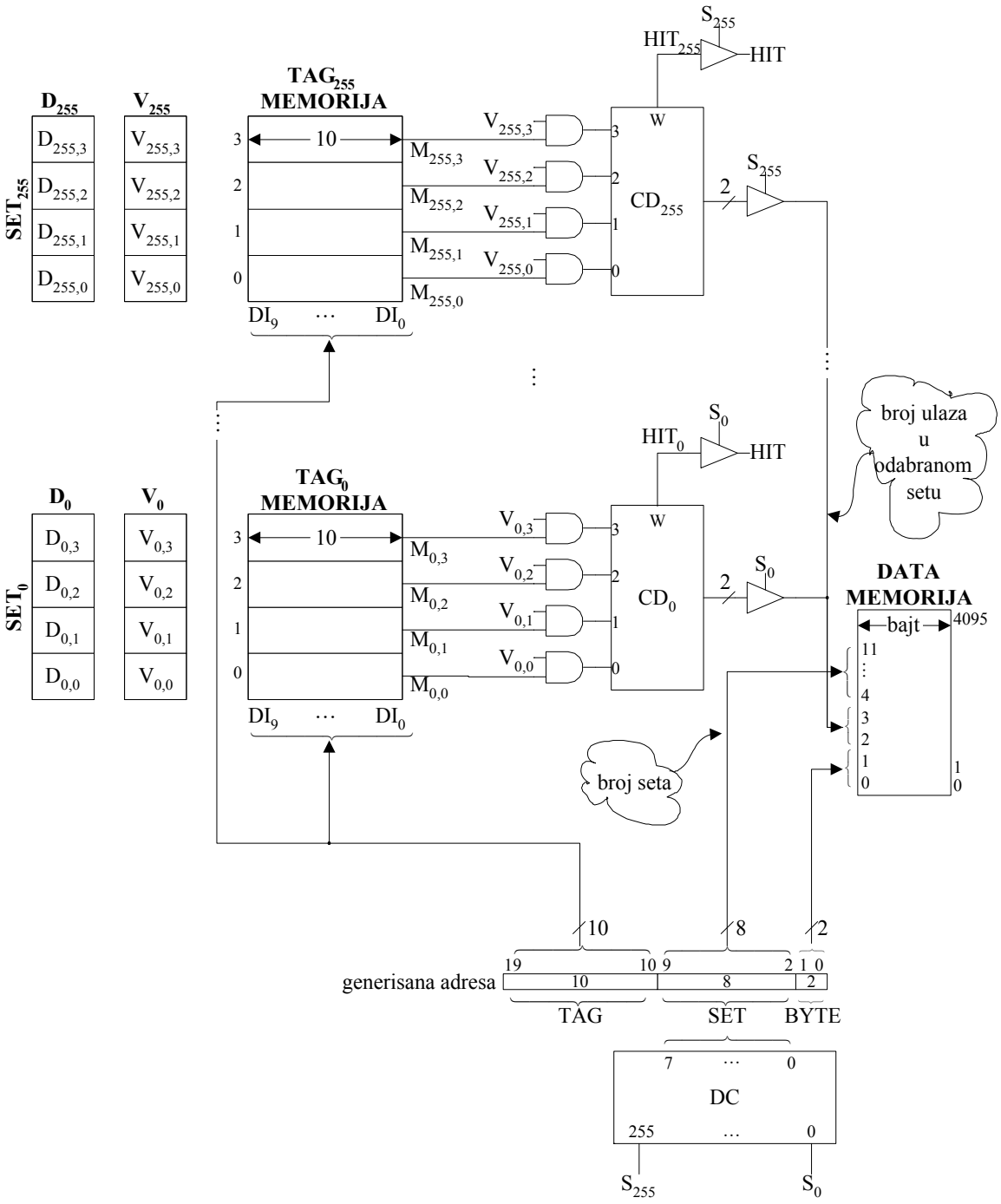
DC služi za generisanje signala selekcije setova $\mathbf{S_{0...511}}$ na osnovu polja SET generisane adrese.

DATA MEMORIJA služi za čuvanje 1024 bloka sadržaja. Blokovi 0 i 1 seta 0 smeštaju se u blokove 0 i 1 DATA MEMORIJE, blokovi 0 i 1 seta 1 smeštaju se u blokove 2 i 3 DATA MEMORIJE, i tako redom do blokova 0 i 1 seta 511 koji se smešaju u blokove 1022 i 1023 DATA MEMORIJE.

a2) S obzirom da je kapacitet operativne memorije 1 Mbajt ($2^{20}$ bajta) i da je veličina bloka četiri bajta ($2^2$ bajta), operativna memorija sadrži $2^{18}$ blokova. Kako u keš memoriju mogu da se smeste 1024 bloka ($10^{10}$ blokova) organizovanih set-asocijativno sa četiri bloka po setu, to keš memorija sadrži 256 setova ($2^8$ setova). Stoga se ovde uzima da je operativna memorija podeljena na 1024 grupe ($2^{10}$ grupa) veličine 256 blokova ($2^8$ blokova). Stoga je struktura generisane adrese kao na slici 1.1.9 i sastoji se iz:

- TAG (10 bita)—broj grupe unutar operativne memorije,
- SET (8 bita)—broj bloka unutar grupe i broj seta u keš memoriji i
- BYTE (2 bita)—adresa bajta unutar bloka.

Na osnovu ovakve strukture generisane adrese i zadatih karakteristika keš memorije dolazi se do strukture keš memorije prikazane na istoj slici.




**Slika 1.1.9.** Struktura keš memorije sa set-asocijativnim preslikavanjem sa četiri bloka po setu

Keš memorija se sastoji iz sledećih delova:

- $D_{0,0}$, $D_{0,1}$, $D_{0,2}$, $D_{0,3}$, $D_{1,0}$, $D_{1,1}$, $D_{1,2}$, $D_{1,3}$, …, $D_{255,0}$, $D_{255,1}$, $D_{255,2}$, $D_{255,3}$, (dirty bitovi)— 1024 flip-flopa organizovanih u 256 setova sa po četiri flip-flopa po setu,
- $V_{0,0}$, $V_{0,1}$, $V_{0,2}$, $V_{0,3}$, $V_{1,0}$, $V_{1,1}$, $V_{1,2}$, $V_{1,3}$, …, $V_{255,0}$, $V_{255,1}$, $V_{255,2}$, $V_{255,3}$, (valid bitovi)— 1024 flip-flopa organizovanih u 256 setova sa po četiri flip-flopa po setu,

- $TAG_0$ MEMORIJA, $TAG_1$ MEMORIJA, …, $TAG_{255}$ MEMORIJA—256 asocijativnih memorija svaka kapaciteta četiri reči širine 10 bita,
- $CD_0$, $CD_1$, …, $CD_{255}$—256 kodera 4/2,
- DC—dekoder 8/256 i
- DATA MEMORIJA—RAM memorija kapaciteta 1024 bloka.

Dirty bitovi seta *i* označavaju za svaki od četiri ulaza seta *i* keš memorije da li je bilo upisa u odgovarajući blok DATA MEMORIJE.

Valid bitovi seta *i* označavaju za svaki od četiri ulaza seta *i* keš memorije da li je odgovarajući ulaz TAG MEMORIJE važeći.

$TAG_i$ MEMORIJA služi za čuvanje četiri TAG polja seta *i* blokova sadržaja koji se nalaze u odgovarajućim ulazima DATA MEMORIJE i generisanje aktivnih vrednosti signala saglasnosti $\mathbf{M_{i,0}}$, $\mathbf{M_{i,1}}$, $\mathbf{M_{i,2}}$ i $\mathbf{M_{i,3}}$ ukoliko postoji saglasnost TAG polja generisane adrese i sadržaja odgovarajućeg ulaza TAG MEMORIJE.

$CD_i$ služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT_i}$ i broja ulaza u keš memoriju za koji je otkrivena saglasnost.

DC služi za generisanje signala selekcije setova $\mathbf{S_{0...255}}$ na osnovu polja SET generisane adrese.

DATA MEMORIJA služi za čuvanje 1024 bloka sadržaja. Blokovi 0, 1, 2 i 3 seta 0 smeštaju se u blokove 0, 1, 2 i 3 DATA MEMORIJE, blokovi 0, 1, 2 i 3 seta 1 smeštaju se u blokove 4, 5, 6 i 7 DATA MEMORIJE, i tako redom do blokova 0, 1, 2 i 3 seta 255 koji se smešaju u blokove 1020, 1021, 1022 i 1023 DATA MEMORIJE.

b) TAG bitovi generisane adrese se porede sa sadržajima svih ulaza TAG MEMORIJE svuh setova. Signali saglasnosti i broja ulaza u kome je otkrivena saglasnost se formiraju za svaki set posebno kao i u slučaju asocijativnog preslikavanja. Od njih se dalje dekodovanim signalima seta selektuje signal saglasnosti HIT i broj ulaza u setu za datu keš memoriju.

SET bitovi iz generisane adrese, selektovani bitovi broja ulaza u setu i BYTE bitovi iz generisane adrese se koriste kao adresa za DATA MEMORIJU i sadržaj se čita.

c) Struktura generisanih adresa i vrednosti polja TAG, SET i BYTE za set-asocijativna preslikavanja sa dva i četiri bloka po setu dati su na slici 1.1.10.

| | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1025 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4099 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 5122 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 8192 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | ←tag→ | | | | | | | | | ←block→ | | | | | | | | ←byte→ | | 2 bloka po setu |
| | ←tag→ | | | | | | | | | | ←block→ | | | | | | | ←byte→ | | 4 bloka po setu |

**Slika 1.1.10.** Struktura generisanih adresa

Sadržaj relevantnih lokacija keš memorije posle četiri operacije čitanja za keš memorije sa set-asocijativnim preslikavanjima sa dva i četiri bloka po setu dati su u posebnim odeljcima.

c1) Set-asocijativno preslikavanje sa 2 bloka po setu

Postoje 256 K blokova operativne memorije i 1 K blokova keš memorije. Ovi blokovi operativne memorije su organizovani u 512 grupa sa 512 blokova po grupi. 1 K blokova keš memorije su organizovani u 512 setova sa 2 bloka po setu. *i*-ti blok operativne memorije

($0 \le i \le 511$) iz bilo koje od 512 grupa može se preslikati samo na bilo koji od dva bloka u $i$-tom setu keš memorije ($0 \le i \le 511$). Na osnovu toga se utvrđuje da je:

- adresa operativne memorije 5122d u bloku 100h i grupi 2,
- adresa operativne memorije 1025d u bloku 100h i grupi 0,
- adresa operativne memorije 8192d u bloku 0h i grupi 4 i
- adresa operativne memorije 4099d u bloku 0h i grupi 2.

Blok 0h u setu 100h keš memorije se napuni blokom 100h iz grupe 2 operativne memorije (adresa 5122d). Blok 1h u setu 100h keš memorije se napuni blokom 100h iz grupe 0 operativne memorije (adresa 1025d). Blok 0h u setu 0h keš memorije se napuni blokom 0h iz grupe 4 operativne memorije (adresa 8192d). Blok 1h u setu 0h keš memorije se napuni blokom 0h iz grupe 2 operativne memorije (adresa 4099d). Vrednosti svih relevantnih delova keš memorije posle četiri operacije čitanja prikazani su na slici **Error! Reference source not found.**.

| broj ulaza | D | V | TAG MEMORIJA | adresa | DATA MEMORIJA | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 000000100 | 0 | 4 | 0 | 0 | 0 | **blok 0** | **$SET_0$** |
| 1 | 0 | 1 | 000000010 | 4 | 0 | 0 | 3 | 0 | **blok 1** | |
| | | | ⋮ | ⋮ | | | | | | |
| 0 | 0 | 1 | 000000010 | 2048 | 0 | 0 | 3 | 0 | **blok 0** | **$SET_{256}$** |
| 1 | 0 | 1 | 000000000 | 2052 | 0 | 1 | 0 | 0 | **blok 1** | |
| | | | ⋮ | ⋮ | | | | | | |
| 0 | 0 | 0 | | 4088 | | | | | **blok 0** | **$SET_{511}$** |
| 1 | 0 | 0 | | 4092 | | | | | **blok 1** | |

**Slika 1.1.11.** Sadržaj relevantnih delova keš memorije.

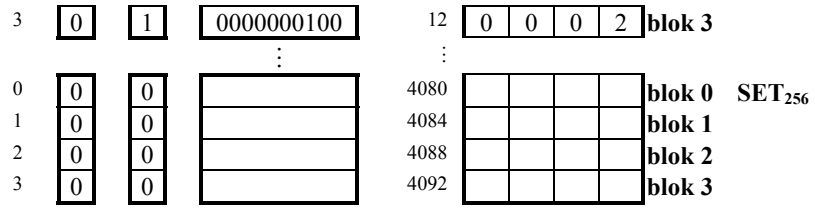
c1) Set-asocijativno preslikavanje sa 4 bloka po setu

Postoji 256 K blokova operativne memorije i 1 K blokova keš memorije. Ovi blokovi operativne memorije su organizovani u 1 K grupa sa 256 blokova po grupi. 1 K blokova keš memorije su organizovani u 256 setova sa 4 bloka po setu. $i$-ti blok operativne memorije ($0 \le i \le 255$) iz bilo koje od 1 K grupa može se preslikati samo na bilo koji od četiri bloka u $i$-tom setu keš memorije ($0 \le i \le 255$). Na osnovu toga se utvrđuje da je:

- adresa operativne memorije 5122d u bloku 0h i grupi 5,
- adresa operativne memorije 1025d u bloku 0h i grupi 1,
- adresa operativne memorije 8192d u bloku 0h i grupi 8 i
- adresa operativne memorije 4099d u bloku 0h i grupi 4.

Blok 0h u setu 0h keš memorije se napuni blokom 0h iz grupe 5 operativne memorije (adresa 5122d). Blok 1h u setu 0h keš memorije se napuni blokom 0h iz grupe 1 operativne memorije (adresa 1025d). Blok 2h u setu 0h keš memorije se napuni blokom 0h iz grupe 8 operativne memorije (adresa 8192d). Blok 3h u setu 0h keš memorije se napuni blokom 0h iz grupe 4 operativne memorije (adresa 4099d). Vrednosti svih relevantnih delova keš memorije posle četiri operacije čitanja prikazani su na slici **Error! Reference source not found.**.

| broj ulaza | D | V | TAG MEMORIJA | adresa | DATA MEMORIJA | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0000000101 | 0 | 0 | 0 | 3 | 0 | **blok 0** | **$SET_0$** |
| 1 | 0 | 1 | 0000000001 | 4 | 0 | 1 | 0 | 0 | **blok 1** | |
| 2 | 0 | 1 | 0000001000 | 8 | 4 | 0 | 0 | 0 | **blok 2** | |




**Slika 1.1.12.** Sadržaj relevantnih delova keš memorije.

## Zadatak 1.1.4

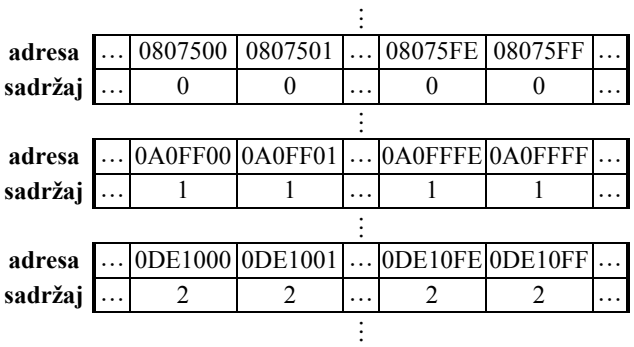
Operativna memorija računara je kapaciteta 512 Mbajta, a širina reči iznosi 16 bita.

a) Nacrtati strukturu keš memorije, označiti širinu u bitima svih relevantnih delova i kratko objasniti funkcije svih delova keš memorije za sledeće slučajeve:

- keš memorije sa 64 K blokova, veličinom bloka 256 reči i asocijativnim preslikavanjem,
- keš memorije sa 64 K blokova, veličinom bloka 256 reči i direktnim preslikavanjem,
- keš memorije sa 64 K blokova, veličinom bloka 256 reči i set-asocijativnim preslikavanjem, sa 16 blokova po setu,

pretpostavljajući da je širina reči dela keš memorije u kome se čuvaju sadržaji 16 bita.

b) Sadržaj relevantnih lokacija operativne memorije je dat na slici **Error! Reference source not found.** pri čemu su sve vrednosti prikazane heksadecimalno. Uzeti da se, najpre, realizuju dve operacije čitanja sa adresa 0A0FF00h i 0A0FFFFh, zatim dve operacije upisa vrednosti 0003h i 0004h u lokacije na adresama 0DE1000h i 0DE10FFh, respektivno i na kraju dve operacije čitanja sa adresa 0807500h i 08075FFh. Operacije čitanja i upisa se isključivo realizuju nad sadržajima keš memorije. Ukoliko se u slučaju gornjih operacija čitanja i upisa utvrdi da nema saglasnosti dovlači se odgovarajući blok sadržaja iz operativne u keš memoriju. Blokovi keš memorije čiji je sadržaj modifikovan vraćaju se u operativnu memoriju.




**Slika 1.1.13.** Sadržaj relevantnih lokacija operativne memorije

Za sve slučajeve keš memorije, dati sadržaje relevantnih delova keš memorije posle izvršenja navedenih operacija, pretpostavljajući da je keš memorija na početku bila prazna.

**Rešenje:**

Kapacitet operativne memorije je 512 Mbajta ($2^{29}$ bajta). Izražen u adresibilnim 16-bitnim rečima kapacitet operativne memorije je 256 Mreči ($2^{28}$ reči). Na osnovu toga sledi da je širina adresne reči 28 bita. Veličina bloka u sve tri realizacije keš memorije je 256 reči, pa je

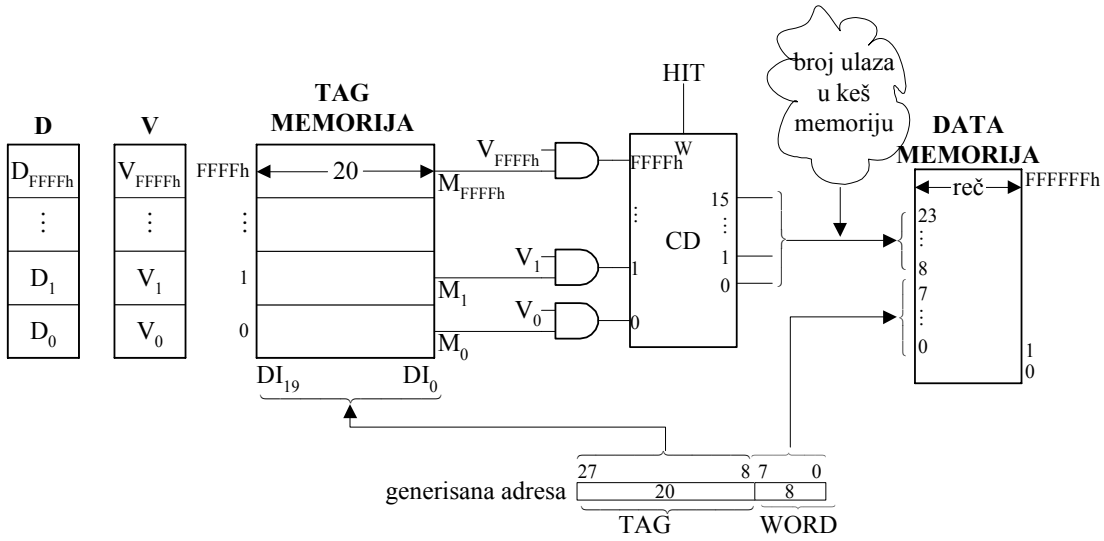
polje u adresnoj reči koje određuje adresu reči unutar bloka u sva tri slučaja širine osam bita, što znači da je polje u adresnoj reči koje određuje redni broj reči unutar bloka uvek širine 8 bita.

a) Strukture keš memorija su date u posebnim odeljcima.

a1) U slučaju keš memorije sa asocijativnim preslikavanjem utvrđuje se da je struktura generisane adrese kao na slici 1.1.14, i sastoji se iz:

- TAG (20 bita)—redni broj bloka operativne memorije i
- WORD (8 bita)—redni broj reči unutar bloka.

Na osnovu ovakve strukture generisane adrese i zadatih karakteristika keš memorije dolazi se do strukture keš memorije prikazane na istoj slici.




**Slika 1.1.14.** Struktura keš memorije sa asocijativnim preslikavanjem

Keš memorija se sastoji iz sledećih delova:

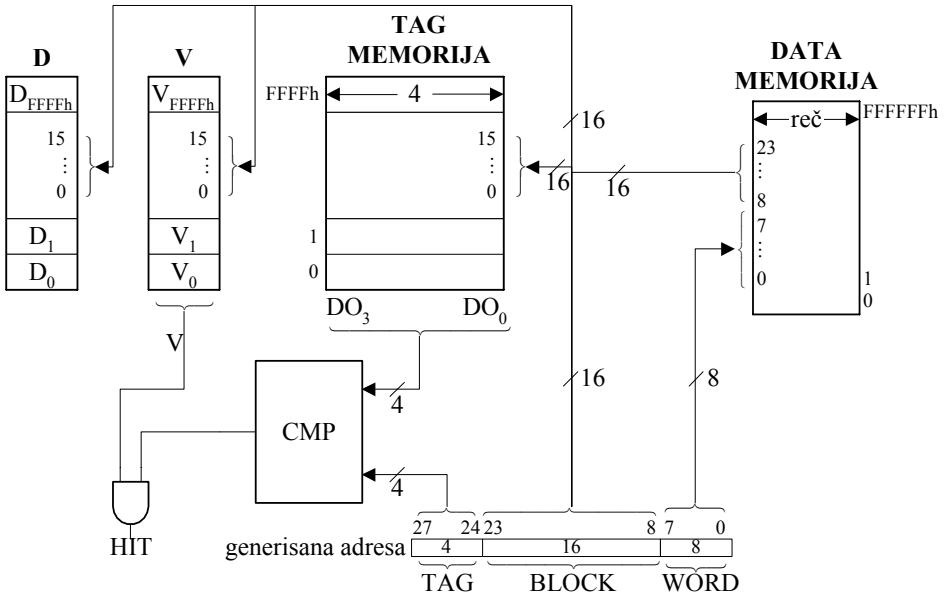
- $D_{0...FFFFh}$ (dirty bitovi)—65536 flip-flopova,
- $V_{0...FFFFh}$ (valid bitovi)—65536 flip-flopova,
- TAG MEMORIJA—asocijativna memorija kapaciteta 64 K reči širine 20 bita,
- CD—koder 64 K/16 i
- DATA MEMORIJA—RAM memorija kapaciteta 64 K blokova.

Njihova funkcija je opisana u zadatku 0.1.

a2) U slučaju keš memorije sa direktnim preslikavanjem utvrđuje se da je struktura generisane adrese kao na slici 1.1.15, i sastoji se iz:

- TAG (4 bita)—redni broj grupe blokova operativne memorije,
- BLOCK (16 bita)—redni broj bloka unutar grupe i redni broj bloka u keš memoriji i
- WORD (8 bita)—redni broj reči unutar bloka.

Na osnovu ovakve strukture generisane adrese i zadatih karakteristika keš memorije dolazi se do strukture keš memorije prikazane na istoj slici.




**Slika 1.1.15.** Struktura keš memorije sa direktnim preslikavanjem

Keš memorija se sastoji iz sledećih delova:

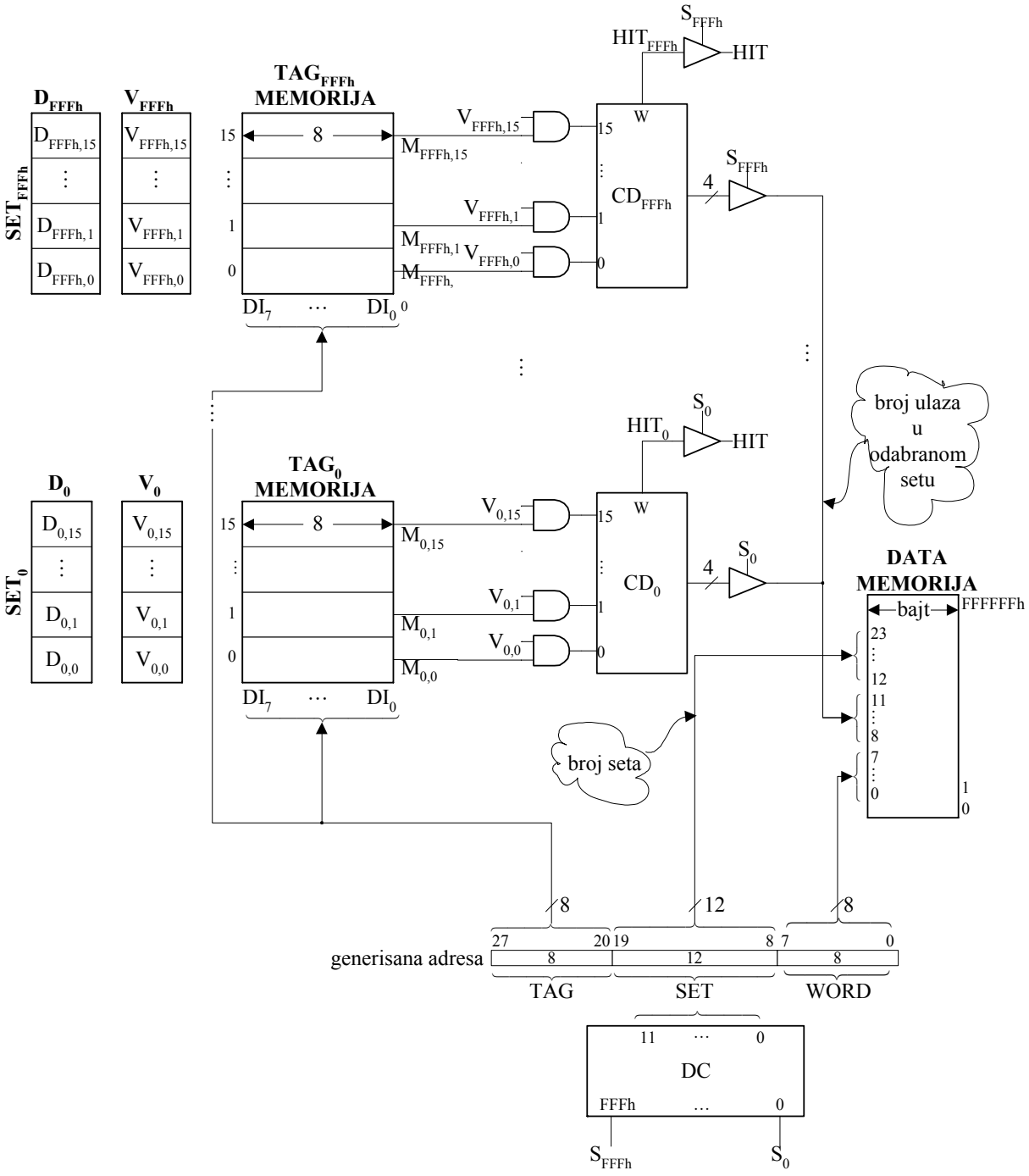
- $D_{0...FFFFh}$ (dirty bitovi)—65536 flip-flopova,
- $V_{0...FFFFh}$ (valid bitovi)—65536 flip-flopova,
- TAG MEMORIJA—RAM memorija kapaciteta 64 K reči širine 4 bita,
- CMP—komparator i
- DATA MEMORIJA—RAM memorija kapaciteta 64 K blokova.

Njihova funkcija je data u zadatku 0.2.

a3) U slučaju keš memorije sa set-asocijativnim preslikavanjem utvrđuje se da je struktura generisane adrese kao na slici 1.1.16 i sastoji se iz:

- TAG (8 bita)—redni broj grupe blokova operativne memorije,
- SET (12 bita)—redni broj bloka unutar grupe i redni broj seta u keš memoriji i
- WORD (8 bita)—redni broj reči unutar bloka.

Na osnovu ovakve strukture generisane adrese i zadatih karakteristika keš memorije dolazi se do strukture keš memorije prikazane na istoj slici.




**Slika 1.1.16.** Struktura keš memorije sa set-asocijativnim preslikavanjem

Keš memorija se sastoji iz sledećih delova:

- $D_{0,0}$, $D_{0,1}$, …, $D_{0,15}$, $D_{1,0}$, $D_{1,1}$, …, $D_{1,15}$, $D_{4095,0}$, $D_{4095,1}$, …, $D_{4095,15}$ (dirty bitovi)—64 K flip-flopa organizovanih u 4096 setova sa po 16 flip-flopa po setu,
- $V_{0,0}$, $V_{0,1}$, …, $V_{0,15}$, $V_{1,0}$, $V_{1,1}$, …, $V_{1,15}$, $V_{4095,0}$, $V_{4095,1}$, …, $V_{4095,15}$ (valid bitovi)—64 K flip-flopa organizovanih u 4096 setova sa po 16 flip-flopa po setu,
- $TAG_0$ MEMORIJA, $TAG_1$ MEMORIJA, …, $TAG_{4095}$ MEMORIJA—4096 asocijativnih memorija svaka kapaciteta 16 reči širine 8 bita,

- $CD_0$, $CD_1$, …, $CD_{4095}$—4096 kodera 16/4,
- DC—dekoder 12/4096 i
- DATA MEMORIJA—RAM memorija kapaciteta 64 K bloka.

Njihova funkcija je data u zadatku 0.3.

b) Strukture generisanih adresa za tri načina preslikavanja i vrednosti polja TAG i WORD za asocijativno preslikavanje, TAG, BLOCK i WORD za direktno preslikavanje i TAG, SET i WORD za set-asocijativno preslikavanje su dati na slici 1.1.17. Kako je u sva tri slučaja keš memorije svako polje adresne reči širine jednake umnošku od 4 bita, najjednostavnije je polja adrese posmatrati u heksadecimalnom sistemu. Sadržaji relevantnih delova keš memorije za tri tipa keš memorije biće prikazani u posebnim odeljcima.

| | 27…24 | 23…20 | 19…16 | 15…12 | 11…8 | 7…4 | 3…0 |
|---|---|---|---|---|---|---|---|
| 0A0FF00h | 0 | A | 0 | F | F | 0 | 0 |
| 0A0FF01h | 0 | A | 0 | F | F | F | F |
| 0DE1000h | 0 | D | E | 1 | 0 | 0 | 0 |
| 0DE1001h | 0 | D | E | 1 | 0 | F | F |
| 0807564h | 0 | 8 | 0 | 7 | 5 | 0 | 0 |
| 0807580h | 0 | 8 | 0 | 7 | 5 | F | F |

tag — word — asocijativno
tag — block — word — direktno
tag — set — word — set-asocijativno

**Slika 1.1.17.** Struktura generisanih adresa

b1) Za slučaj asocijativnog preslikavanja prema objašnjenjima iz zadatka 0.1, strukturi generisane adrese i vrednostima polja TAG i WORD, u ulaze 0, 1 i 2 keš memorije se redom upisuju blokovi 0A0FFh, 0DE10h i 08075h. Izgled relevantnih delova keš memorije je dat na slici **Error! Reference source not found.**.

| broj ulaza | D | V | TAG MEMORIJA | adresa | DATA MEMORIJA | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0A0FF | 000000h | 1 | 1 | … | 1 | 1 |
| 1 | 1 | 1 | 0DE10 | 000100h | 3 | 2 | … | 2 | 4 |
| 2 | 0 | 1 | 08075 | 000200h | 0 | 0 | … | 0 | 0 |
| ⋮ | ⋮ | ⋮ | | ⋮ | | | | | |
| FFFFh | 0 | 0 | | FFFF00h | | | … | | |

**Slika 1.1.18.** Sadržaj relevantnih delova keš memorije

Popunjavanje ulaza keš memorije se realizuje na sledeći način.

**Operacija čitanja sa adrese 0A0FF00h operativne memorije.** Utvrđuje se da nema saglasnosti. Adresa je iz bloka 0A0FFh operativne memorije, pa se ovaj blok dovlači i smešta u ulaz 0 DATA MEMORIJE, jer je ulaz 0 prvi slobodan ulaz. Kao rezultat toga na lokacijama 000000h do 0000FFh DATA MEMORIJE pojavljuju se sada vrednosti 0001h. U isti ulaz TAG MEMORIJE se upisuje vrednost TAG polja generisane adrese. U ulaz 0 V flip-flopova se upisuje vrednost 1. Potom se utvrđuje da ima saglasnosti u ulazu 0 keš memorije i sa adrese 000000h DATA MEMORIJE čita sadržaj 0001h.

**Operacija čitanja sa adrese 0A0FFFFh operativne memorije.** Utvrđuje se da ima saglasnosti u ulazu 0 keš memorije i sa adrese 0000FFh DATA MEMORIJE čita sadržaj 0001h.

**Operacija upisa vrednosti 0003h u lokaciju na adresi 0DE1000h operativne memorije.** Utvrđuje se da nema saglasnosti. Adresa je iz bloka 0DE10h operativne memorije, pa se ovaj blok dovlači i smešta u ulaz 1 DATA MEMORIJE, jer je ulaz 1 prvi slobodan ulaz. Kao rezultat toga na lokacijama 000100h do 0001FFh DATA MEMORIJE pojavljuju se sada vrednosti 0002h. U isti ulaz TAG MEMORIJE se upisuje vrednost TAG polja generisane adrese. U ulaz 1 V flip-flopova se upisuje vrednost 1. Potom se utvrđuje da ima saglasnosti u ulazu 1 keš memorije pa se u lokaciju DATA MEMORIJE sa adrese 000100h upisuje vrednost 0003h i u ulaz 1 D flip-flopova upisuje vrednost 1.

**Operacija upisa vrednosti 0004h u lokaciju na adresi 0DE10FFh operativne memorije.** Utvrđuje se da ima saglasnosti u ulazu 1 keš memorije pa se u lokaciju DATA MEMORIJE sa adrese 0001FFh upisuje vrednost 0004h.

**Operacija čitanja sa adrese 0807500h operativne memorije.** Utvrđuje se da nema saglasnosti. Adresa je iz bloka 08075h operativne memorije, pa se ovaj blok dovlači i smešta u ulaz 2 DATA MEMORIJE, jer je ulaz 2 prvi slobodan ulaz. Kao rezultat toga na lokacijama 000200h do 0002FFh DATA MEMORIJE pojavljuju se sada vrednosti 0000h. U isti ulaz TAG MEMORIJE se upisuje vrednost TAG polja generisane adrese. U ulaz 2 V flip-flopova se upisuje vrednost 1. Potom se utvrđuje da ima saglasnosti u ulazu 2 keš memorije i sa adrese 000200h DATA MEMORIJE čita sadržaj 0000h.

**Operacija čitanja sa adrese 08075FFh operativne memorije.** Utvrđuje se da ima saglasnosti u ulazu 2 keš memorije i sa adrese 0002FFh DATA MEMORIJE čita sadržaj 0000h.

b2) Za slučaj direktnog preslikavanja prema objašnjenjima iz zadatka 0.2, strukturi generisane adrese i vrednostima polja TAG, BLOCK i WORD, u ulaze A0FFh, DE10h i 8075h keš memorije se redom upisuju blokovi 0A0FFh, 0DE10h i 08075h operativne memorije. Izgled relevantnih delova keš memorije je dat na slici **Error! Reference source not found.**:

| broj ulaza | D | V | TAG MEMORIJA | adresa | DATA MEMORIJA |
|---|---|---|---|---|---|
| 0000h | 0 | 0 | | 000000h | … |
| 8075h | 0 | 1 | 0 | 807500h | 0 0 … 0 0 |
| A0FFh | 0 | 1 | 0 | A0FF00h | 1 1 … 1 1 |
| DE10h | 1 | 1 | 0 | DE1000h | 3 2 … 2 4 |
| FFFFh | 0 | 0 | | FFFF00h | … |

**Slika 1.1.19.** Sadržaj relevantnih delova keš memorije

Popunjavanje ulaza keš memorije se realizuje na sledeći način.

**Operacija čitanja sa adrese 0A0FF00h operativne memorije.** Utvrđuje se da u ulazu A0FFh keš memorije nema saglasnosti. Adresa je iz bloka 0A0FFh operativne memorije, pa se ovaj blok dovlači i smešta u ulaz A0FFh DATA MEMORIJE. Kao rezultat toga na lokacijama A0FF00h do A0FFFFh DATA MEMORIJE pojavljuju se sada vrednosti 0001h. U isti ulaz TAG MEMORIJE se upisuje vrednost TAG polja generisane adrese. U ulaz A0FFh V flip-flopova se upisuje vrednost 1. Potom se utvrđuje da ima saglasnosti u ulazu A0FFh keš memorije i sa adrese A0FF00h DATA MEMORIJE čita sadržaj 0001h.

**Operacija čitanja sa adrese 0A0FFFFh operativne memorije.** Utvrđuje se da ima saglasnosti u ulazu A0FFh keš memorije i sa adrese A0FFFFh DATA MEMORIJE čita sadržaj 0001h.

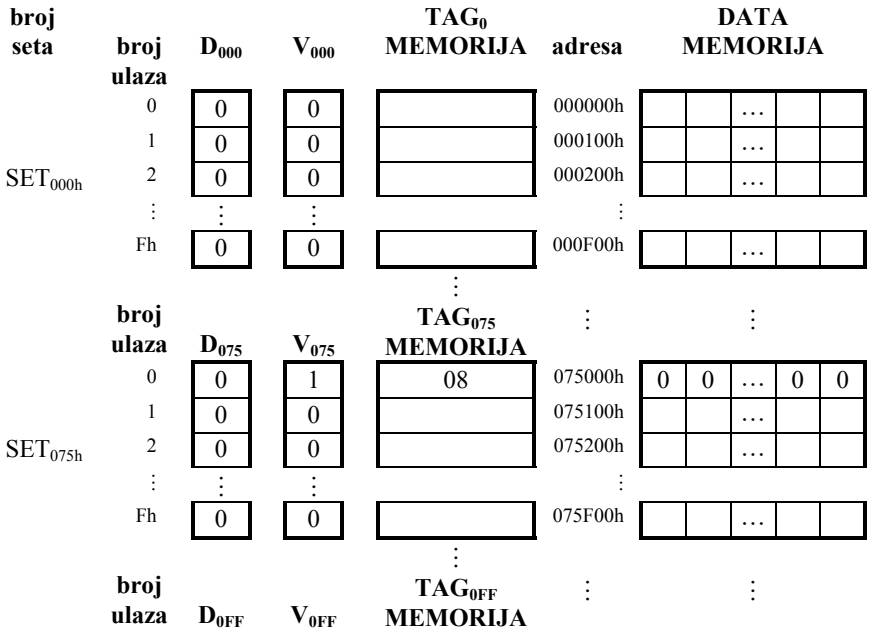
**Operacija upisa vrednosti 0003h u lokaciju na adresi 0DE1000h operativne memorije.** Utvrđuje se da u ulazu DE10h keš memorije nema saglasnosti. Adresa je iz bloka 0DE10h operativne memorije, pa se ovaj blok dovlači i smešta u ulaz DE10h DATA MEMORIJE. Kao rezultat toga na lokacijama DE1000h do DE10FFh DATA MEMORIJE pojavljuju se sada vrednosti 0002h. U isti ulaz TAG MEMORIJE se upisuje vrednost TAG polja generisane adrese. U ulaz DE10h V flip-flopova se upisuje vrednost 1. Potom se utvrđuje da ima saglasnosti u ulazu DE10h keš memorije pa se u lokaciju DATA MEMORIJE sa adrese DE1000h upisuje vrednost 0003h i u ulaz DE10h D flip-flopova upisuje vrednost 1.

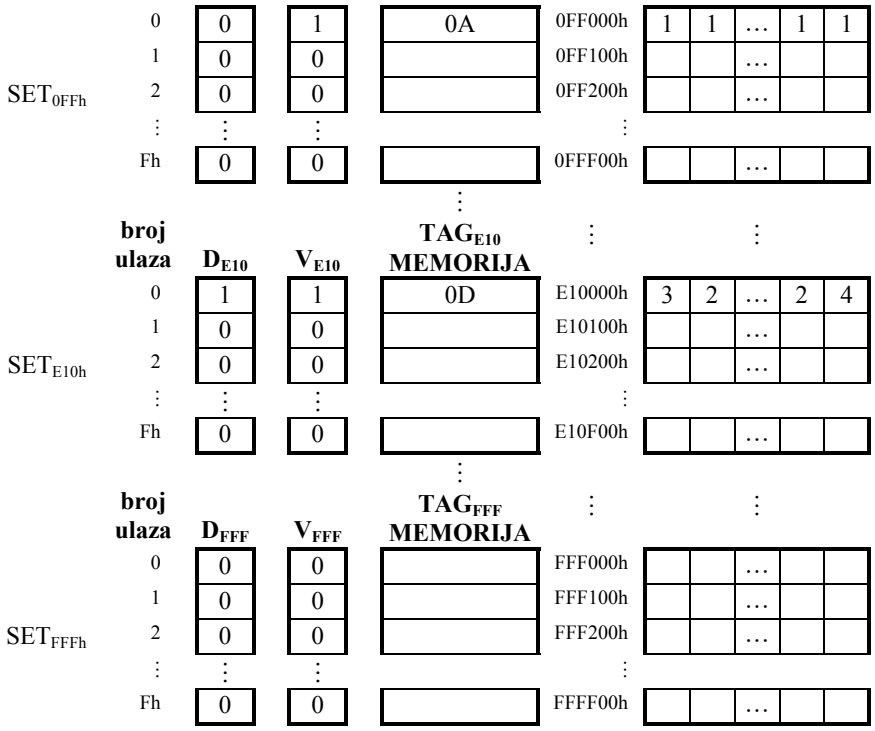
**Operacija upisa vrednosti 0004h u lokaciju na adresi 0DE10FFh operativne memorije.** Utvrđuje se da ima saglasnosti u ulazu DE10h keš memorije pa se u lokaciju DATA MEMORIJE sa adrese DE10FFh upisuje vrednost 0004h.

**Operacija čitanja sa adrese 0807500h operativne memorije.** Utvrđuje se da u ulazu 8075h keš memorije nema saglasnosti. Adresa je iz bloka 08075h operativne memorije, pa se ovaj blok dovlači i smešta u ulaz 8075h DATA MEMORIJE. Kao rezultat toga na lokacijama 807500h do 8075FFh DATA MEMORIJE pojavljuju se sada vrednosti 0000h. U isti ulaz TAG MEMORIJE se upisuje vrednost TAG polja generisane adrese. U ulaz 8075h V flip-flopova se upisuje vrednost 1. Potom se utvrđuje da ima saglasnosti u ulazu 8075h keš memorije i sa adrese 807500h DATA MEMORIJE čita sadržaj 0000h.

**Operacija čitanja sa adrese 08075FFh operativne memorije.** Utvrđuje se da ima saglasnosti u ulazu 8075 keš memorije i sa adrese 8075FFh DATA MEMORIJE čita sadržaj 0000h.

b3) Za slučaj set-asocijatovnog preslikavanja prema objašnjenjima iz zadatka 0.3, strukturi generisane adrese i vrednostima polja TAG, SET i WORD u ulaze 0 setova 0FFh, E10h i 075h keš memorije se redom upisuju blokovi 0A0FFh, 0DE10h i 08075h operativne memorije. Izgled relevantnih delova keš memorije je dat na slici **Error! Reference source not found.**.






**Slika 1.1.20.** Sadržaj relevantnih delova keš memorije

**Operacija čitanja sa adrese 0A0FF00h operativne memorije.** Utvrđuje se da u setu 0FFh keš memorije nema saglasnosti. Adresa je iz bloka 0A0FFh operativne memorije, pa se ovaj blok dovlači i smešta u ulaz 0 seta 0FFh DATA MEMORIJE. Kao rezultat toga na lokacijama 0FF000h do 0FF0FFh DATA MEMORIJE pojavljuju se sada vrednosti 0001h. U isti ulaz $TAG_{0FFh}$ MEMORIJE se upisuje vrednost TAG polja generisane adrese. U ulaz 0 $V_{0FFh}$ flip-flopova se upisuje vrednost 1. Potom se utvrđuje da ima saglasnosti u ulazu 0 seta 0FFh keš memorije i sa adrese 0FF000h DATA MEMORIJE čita sadržaj 0001h.

**Operacija čitanja sa adrese 0A0FFFFh operativne memorije.** Utvrđuje se da ima saglasnosti u ulazu 0 seta 0FFh keš memorije i sa adrese 0FF0FFh DATA MEMORIJE čita sadržaj 0001h.

**Operacija upisa vrednosti 0003h u lokaciju na adresi 0DE1000h operativne memorije.** Utvrđuje se da u setu E10h keš memorije nema saglasnosti. Adresa je iz bloka 0DE10h operativne memorije, pa se ovaj blok dovlači i smešta u ulaz 0 seta E10h DATA MEMORIJE. Kao rezultat toga na lokacijama E10000h do E100FFh DATA MEMORIJE pojavljuju se sada vrednosti 0002h. U isti ulaz $TAG_{E10h}$ MEMORIJE se upisuje vrednost TAG polja generisane adrese. U ulaz 0 $V_{E10h}$ flip-flopova se upisuje vrednost 1. Potom se utvrđuje da ima saglasnosti u ulazu 0 seta E10h keš memorije pa se u lokaciju DATA MEMORIJE sa adrese E10000h upisuje vrednost 0003h i u ulaz 0 $D_{E10h}$ flip-flopova upisuje vrednost 1.

**Operacija upisa vrednosti 0004h u lokaciju na adresi 0DE10FFh operativne memorije.** Utvrđuje se da ima saglasnosti u ulazu 0 seta E10h keš memorije pa se u lokaciju DATA MEMORIJE sa adrese E100FFh upisuje vrednost 0004h.

**Operacija čitanja sa adrese 0807500h operativne memorije.** Utvrđuje se da u ulazu0 seta 075h keš memorije nema saglasnosti. Adresa je iz bloka 08075h operativne memorije, pa se ovaj blok dovlači i smešta u ulaz 0 seta 075h DATA MEMORIJE. Kao rezultat toga na

lokacijama 075000h do 0750FFh DATA MEMORIJE pojavljuju se sada vrednosti 0000h. U isti ulaz $TAG_{075h}$ MEMORIJE se upisuje vrednost TAG polja generisane adrese. U ulaz 0 $V_{075h}$ flip-flopova se upisuje vrednost 1. Potom se utvrđuje da ima saglasnosti u ulazu 0 seta 075h keš memorije i sa adrese 075000h DATA MEMORIJE čita sadržaj 0000h.

**Operacija čitanja sa adrese 08075FFh operativne memorije.** Utvrđuje se da ima saglasnosti u ulazu 0 seta 075h keš memorije i sa adrese 0750FFh DATA MEMORIJE čita sadržaj 0000h.

### Zadatak 1.1.5

U računaru postoji operativna memorija kapaciteta 16 Mbajta. Adresiranje operativne memorije je bajtovsko, a operacije obraćanja memoriji (čitanje i upis) su za bajtovske veličine.

a) Nacrtati, označiti širinu u bitovima svih relevantnih delova keš memorije pretpostavljajući da je obraćanje delu keš memorije u kome se nalaze sadržaji bajtovsko i objasniti njihovu funkciju za:

a1) keš memoriju sa 256 ulaza realizovanu u asocijativnoj tehnici preslikavanja na nivou bloka ako je veličina bloka 16 bajtova,

a2) keš memoriju realizovanu u tehnici direktnog preslikavanja na nivou bloka, ako je veličina bloka 16 bajta i kapacitet dela keš memorije u kojoj se nalaze sadržaji 32 Kbajta.

b) Objasniti kako se za oba tipa keš memorije za slučaj čitanja bajtovske veličine utvrđuje da li se sadržaj sa generisane adrese nalazi u keš memoriji i kako se vrši samo očitavanje željenog sadržaja.

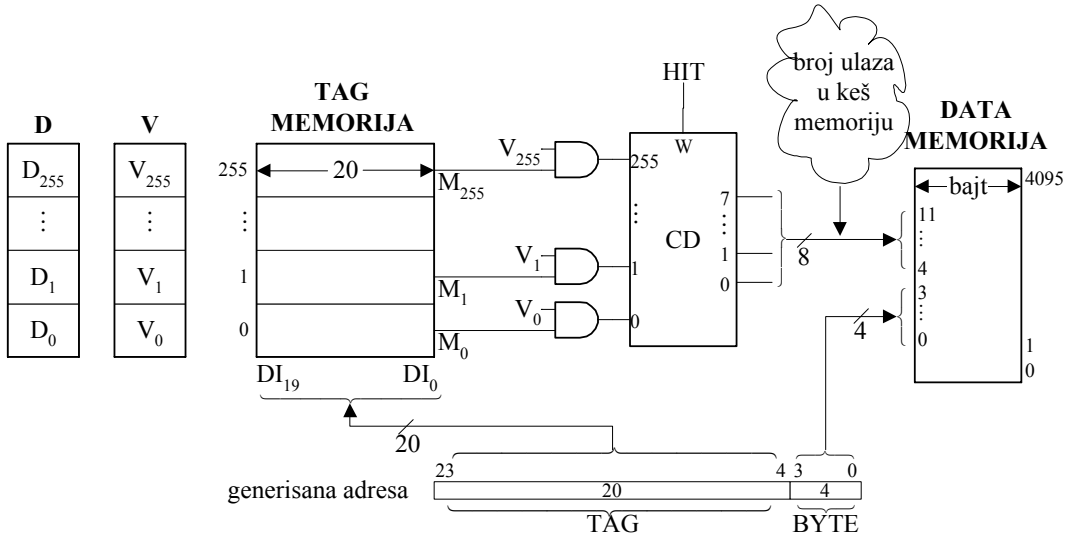
c) U nekim situacijama sadržaj sa određenog ulaza keš memorije treba najpre vratiti u operativnu memoriju pa tek onda blok iz operativne memorije dovući u određeni ulaz keš memorije. Objasniti za oba tipa keš memorije:

- kako se generišu adrese operativne memorije u koje treba najpre vratiti sadržaj iz određenog ulaza keš memorije i
- šta se sve radi i kako se generišu sve potrebne adrese kod dovlačenja bloka iz operativne u keš memoriju?

**Rešenje:**

a) Struktura keš memorija sa asocijativnim i direktnim preslikavanjem je data u posebnim odeljcima.

a1) Struktura keš memorije sa asocijativnim preslikavanjem je data na slici 1.1.21.




**Slika 1.1.21.** Struktura keš memorije sa asocijativnim preslikavanjem

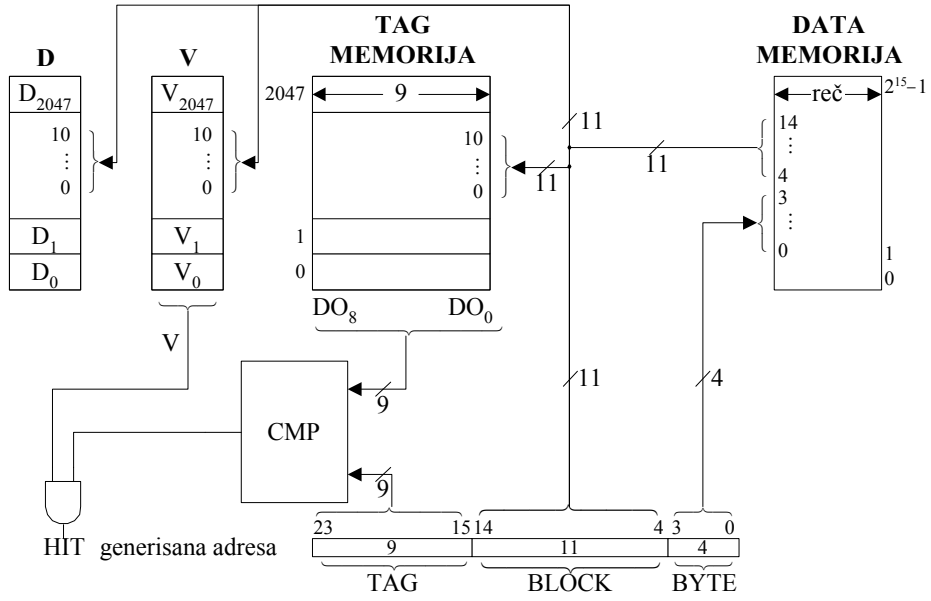
Kako je kapacitet operativne memorije 16 Mbajta ($2^{24}$ bajta), a veličina bloka 16 bajtova ($2^4$ bajta), ona sadrži $2^{20}$ blokova. Prema tome, struktura generisane adrese je:

- TAG (20 bita)—broj bloka u operativnoj memoriji i
- BYTE (4 bita)—adresa bajta u bloku.

Struktura keš memorije je:

- $D_{0\ldots255}$—dirty bitovi,
- $V_{0\ldots255}$—valid bitovi,
- TAG MEMORIJA—asocijativna memorija gde se nalazi 256 TAG polja širine 20 bita,
- CD—koder 256/8 i
- DATA MEMORIJA—RAM memorija gde se nalazi 256 blokova sadržaja.

a2) Struktura keš memorije sa direktnim preslikavanjem je data na slici 1.1.22.




**Slika 1.1.22.** Struktura keš memorije sa direktnim preslikavanjem

Pošto keš memorija za sadržaj ima kapacitet 32 Kbajta ($2^{15}$ bajta), a veličina bloka je 16 bajtova ($2^4$ bajta), ona ima $2^{11}$ blokova i isto toliko ulaza. Glavna memorija je podeljena na $2^{24}/(2^{11}\cdot 2^4) = 2^9$ grupa. Zato je struktura generisane adrese:

- TAG (9 bita)—broj grupe u operativnoj memoriji,
- BLOCK (11 bita)—broj bloka unutar grupe i broj bloka u keš memoriji i
- BYTE (4 bita)—adresa bajta unutar bloka.

Struktura keš memorije:

- $D_{0\ldots2047}$—dirty bitovi,
- $V_{0\ldots2047}$—valid bitovi,
- TAG MEMORIJA—RAM memorija gde se nalaze 2048 TAG polja širine 9 bita,
- CMP—komparator i
- DATA MEMORIJA—RAM memorija gde se nalaze 2048 bloka sadržaja.

b) Utvrđivanje da li se sadržaj nalazi u keš memoriji i čitanje sadržaja za dva tipa keš memorije su prikazani u posebnim odeljcima.

b1) Asocijativno preslikavanje

TAG bitovi (20) iz generisane adrese se porede sa sadržajem svih 256 ulaza u TAG MEMORIJI. Ako se sadržaj bilo kojeg ulaza slaže sa TAG bitovima generisane adrese i odgovarajući V bit je postavljen, signal HIT postaje aktivan.

Bitovi (8) koji označavaju broj ulaza u keš memoriji na kome je otkrivena saglasnost dobijeni sa izlaza kodera i BYTE (4) bitovi generisane adrese se koriste kao adresa u DATA MEMORIJI i sadržaj se čita.

b2) Direktno preslikavanje

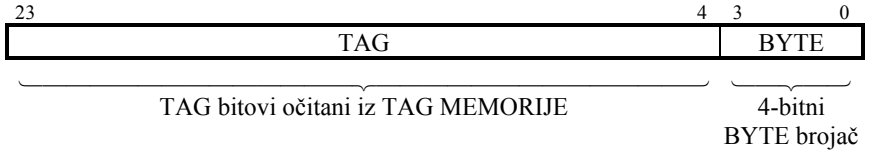
BLOCK bitovi (11) iz generisane adrese se koriste kao adresa za TAG MEMORIJU. TAG bitovi očitani iz TAG MEMORIJE se porede sa TAG bitovima iz generisane adrese. Ako se slažu i ako je bit V, adresiran BLOCK bitovima generisane adrese, postavljen, signal HIT postaje aktivan.

BLOCK bitovi (11) i BYTE bitovi (4) generisane adrese se koriste kao adresa za DATA MEMORIJU i sadržaj se čita.

c) Generisanje potrebnih adresa je prikazano u posebnim odeljcima.

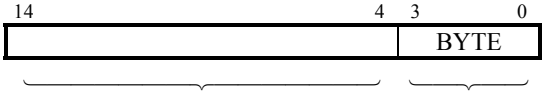
c1) Generisanje potrebnih adresa prilikom vraćanja bloka iz keš memorije u operativnu memoriju za slučaj asocijativnog preslikavanja je dat u daljem tekstu.

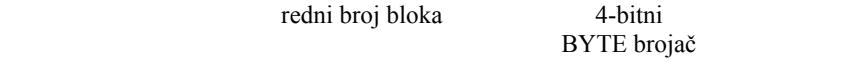
Adresa operativne memorije je:




Redni broj bloka odabranog za zamenu se koristi kao adresa TAG MEMORIJE. 20-bitno polje TAG daje 20 najstarijih bitova adrese operativne memorije, a 4-bitni BYTE brojač daje najmlađa četiri bita.

Adresa DATA MEMORIJE je:

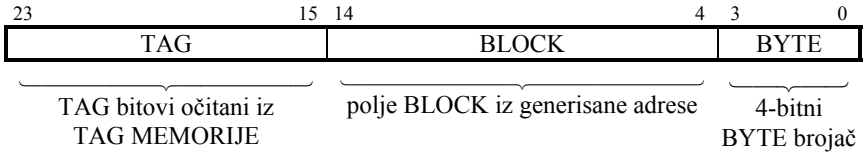





Redni broj bloka odabranog za zamenu se koristi kao 11 najstarijih bita adrese DATA MEMORIJE, a 4-bitni BYTE brojač daje četiri najmlađa bita.

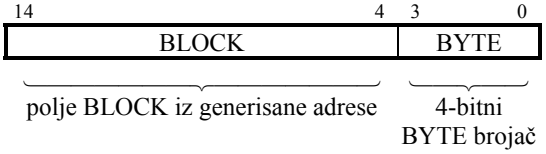
Generisanje potrebnih adresa prilikom vraćanja bloka iz keš memorije u operativnu memoriju za slučaj direktnog preslikavanja je dat u daljem tekstu.

Adresa operativne memorije je:




Broj bloka iz generisane adrese se koristi kao adresa TAG MEMORIJE. 9-bitno polje TAG daje devet najstarijih bitova adrese operativne memorije, polje BLOCK iz generisane adrese daje sledećih 11 bitova, a 4-bitni BYTE brojač daje najmlađa četiri bita.

Adresa DATA MEMORIJE je:

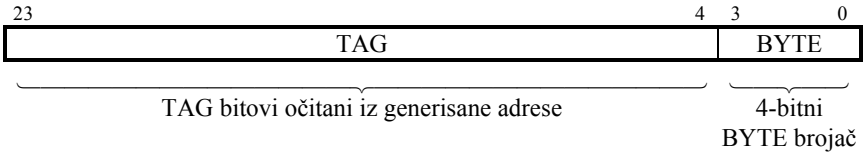



Polje BLOCK iz generisane adrese daje najstarijih 11 bitova, a 4-bitni BYTE brojač daje četiri najmlađa bita.

c2)

Generisanje potrebnih adresa prilikom porenosa bloka iz operativne memorije u keš memoriju za slučaj asocijativnog preslikavanja je dato u daljem tekstu.

Adresa operativne memorije je:

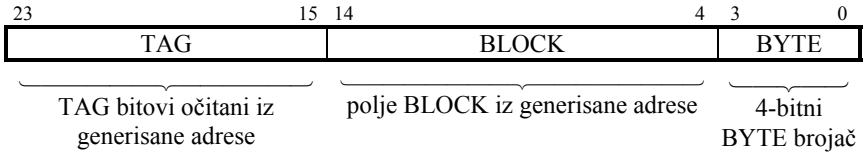



Adresa DATA MEMORIJE je ista kao u tački c1).

TAG bitovi iz generisane adrese se upisuju u TAG MEMORIJU u ulaz specifikovan brojem bloka koji se zamenjuje. Valid bit adresiran brojem bloka koji se zamenjuje se postavlja.

Generisanje potrebnih adresa prilikom porenosa bloka iz operativne memorije u keš memoriju za slučaj direktnog preslikavanja je dato u daljem tekstu.

Adresa operativne memorije je:




Adresa DATA MEMORIJE je ista kao u tački c2).

TAG bitovi iz generisane adrese se upisuju u TAG MEMORIJU u ulaz specifikovan poljem BLOCK iz generisane adrese. Valid bit adresiran poljem BLOCK iz generisane adrese se setuje.

### Zadatak 1.1.6

Operativna memorija računara ima kapacitet 2 Mbajta, širina reči iznosi 16 bita i vreme pristupa memorije iznosi $T_m$. Procesor ima razdvojene keš memorije za instrukcije i podatke. Instrukcijska keš memorija ima kapacitet DATA dela 4 Kbajta, preslikavanje je na nivou bloka, veličina bloka je 128 bajta, širina reči je 16 bita i vreme pristupa iznosi $T_k$. Pretpostaviti da su sve instrukcije širine 16 bitova, i da se adrese na koje ukazuje registar PC odnose na 16-bitne veličine.

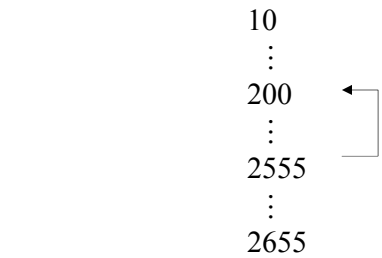
a) Nacrtati strukturu svih relevantnih delova keš memorije sa:

- direktnim preslikavanjem,
- asocijativnim preslikavanjem i
- set-asocijativnim preslikavanjem sa dva bloka po setu,

i ukratko opisati funkciju svakog dela. Nacrtati strukturu adrese koju generiše procesor i objasniti kako se koriste grupe bitova u sva tri slučaja da se:

- proveri da li je tražena instrukcija u keš memoriji i
- da se očita instrukcija iz keš memorije.

b) Izračunati ukupno vreme potrebno za čitanje instrukcija iz keš memorije, za sva tri tipa keš memorije, kada se izvršava sledeći program:




Prva očitana instrukcija se nalazi na adresi 10d, a poslednja na adresi 2655d. Petlja se izvrši dva puta. Keš memorija je bila prazna na početku. Kada se računa vreme:

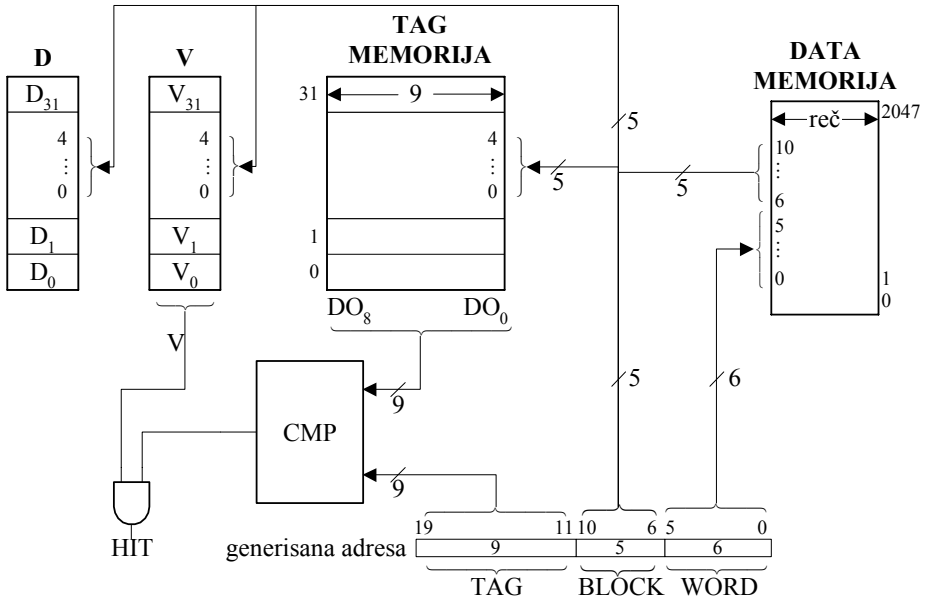
- treba pretpostaviti da se prvo prenese ceo blok iz operativne memorije u keš memorije, pa se tek onda instrukcija očita i
- treba uzeti u obzir samo vremena pristupa $T_m$ i $T_k$ i zanemariti vremena potrebna za ostale aktivnosti.

**Rešenje:**

Kapacitet operativne memorije je 1 M 16-bitnih reči ($2^{20}$ reči), a veličina bloka je 64 reči ($2^6$ reči), pa operativna memorija sadrži $2^{14}$ (16 K) blokova. Kapacitet DATA dela keš memorije je 2 K reči ($2^{11}$ reči), pa DATA deo keš memorije sadrži 32 bloka ($2^5$ blokova).

a) Strukture keš memorije za sve tri vrste preslikavanja date su u posebnim odeljcima.

a1) Struktura keš memorije sa direktnim preslikavanjem je prikazana na slici 1.1.23.

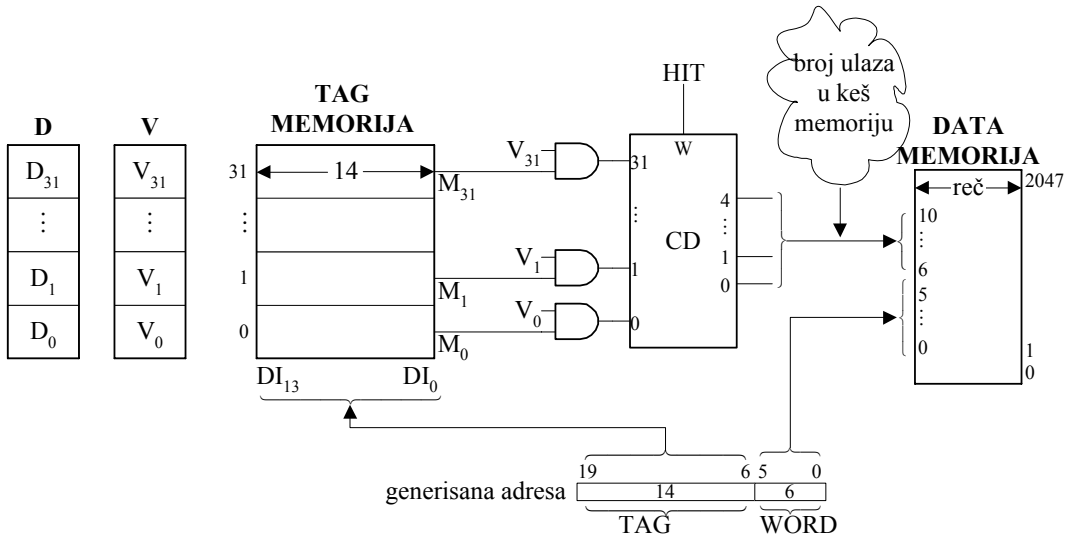



**Slika 1.1.23.** Struktura keš memorije sa direktnim preslikavanjem

BLOCK bitovi (5) iz generisane adrese se koriste kao ulaz za TAG MEMORIJU. TAG bitovi očitani iz TAG MEMORIJE se porede sa TAG bitovima iz generisane adrese. Ako se slažu i ako je V bit, takođe adresiran BLOCK bitovima generisane adrese, postavljen, aktivira se signal HIT.

BLOCK bitovi (5) i WORD bitovi (6) generisane adrese se koriste kao adresni bitovi DATA MEMORIJE i instrukcija se čita.

a2) Struktura keš memorije sa asocijativnim preslikavanjem je prikazana na slici 1.1.24.

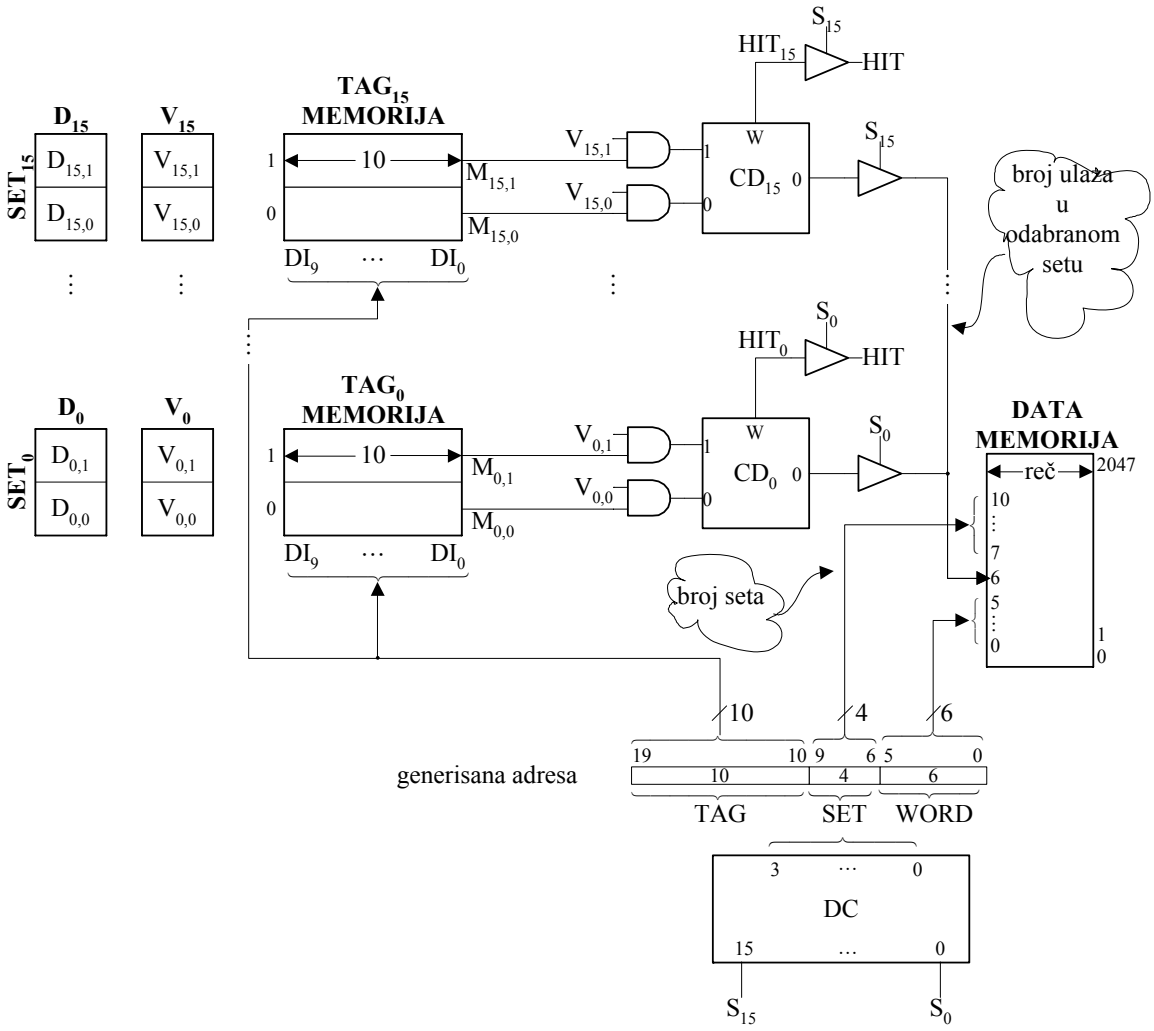



**Slika 1.1.24.** Struktura keš memorije sa asocijativnim preslikavanjem

TAG bitovi (14) generisane adrese se porede sa sadržajem svih 32 ulaza TAG MEMORIJE (asocijativne). Ako se sadržaj bilo kog ulaza slaže sa TAG bitovima iz generisane adrese i odgovarajući V bit je postavljen, aktivira se signal HIT.

Bitovi (5) koji kodiraju broj ulaza u keš memoriju (ulaz gde je bio HIT) dobijeni sa izlaza kodera i WORD bitovi (6) iz generisane adrese se koriste kao adresa DATA MEMORIJE i instrukcija se čita.

a3) Struktura keš memorije set-asocijativnim preslikavanjem sa dva bloka po setu je prikazana na slici 1.1.25.




**Slika 1.1.25.** Struktura keš memorije sa set-asocijativnim preslikavanjem sa dva bloka po setu

TAG bitovi (10) generisane adrese se porede sa sadržajima dva ulaza u tag memorijama (asocijativnim) TAG-0 MEMORIJA do TAG-15 MEMORIJA u 16 setova. Ako se sadržaj bilo kojeg od dva ulaza u setu slaže sa TAG bitovima generisane adrese i odgovarajući V bit je postavljen, aktivira se signal HIT u tom setu. Ovaj signal se koristi kao HIT signal za celu keš memoriju.

SET bitovi (4) generisane adrese, broj ulaza bloka (1) gde je pronađen HIT za odabrani set i WORD bitovi (6) iz generisane adrese se koriste kao adresa za DATA MEMORIJU i instrukcija se čita.

b)

Relevantne adrese se sastoje od:

- broja bloka i adrese reči za asocijativno preslikavanje,
- broja grupe, broja bloka i adrese reči za direktno preslikavanje i
- broja grupe, broja bloka i adrese reči za set-asocijativno preslikavanje sa dva bloka po setu.

Ove adrese su date na slici 1.1.26 (sve vrednosti su decimalne).

| | **asocijativno** | | **direktno** | | | **set-asocijativno** | | |
|---|---|---|---|---|---|---|---|---|
| **adresa** | **blok** | **reč** | **grupa** | **blok** | **reč** | **grupa** | **blok** | **reč** |
| 10 | 0 | 10 | 0 | 0 | 10 | 0 | 0 | 10 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 200 | 3 | 8 | 0 | 3 | 8 | 0 | 3 | 8 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| | 15 | | | 15 | | 0 | 15 | |
| | 16 | | | 16 | | 1 | 0 | |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| | 31 | | 0 | 31 | | 1 | 15 | |
| | 32 | | 1 | 0 | | 2 | 0 | |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 2555 | 39 | 59 | 1 | 7 | 59 | 2 | 7 | 59 |
| ⋮ | 40 | | | | | | | |
| 2655 | 41 | 31 | 1 | 9 | 31 | 2 | 9 | 31 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| | 47 | | | | | | 15 | |

**Slika 1.1.26.** Struktura relevantnih adresa za sva tri tipa preslikavanja

b1) U slučaju direktnog preslikavanja operacije čitanja blokova operativne memorije se izvode sledećim redosledom:

- blokovi 0 do 31 iz grupe 0,
- blokovi 0 do 7 iz grupe 1,
- blokovi 3 do 31 iz grupe 0, and
- blokovi 0 do 9 iz grupe 1.

Ovi blokovi se smeštaju u keš memoriju na sledeći način:

- blokovi 0 do 31 iz grupe 0—oni se dovlače u blokove 0 do 31 ($32T_b$),
- blokovi 0 do 7 iz grupe 1— oni se dovlače u blokove 0 do 7 ($8T_b$),
- blokovi 3 do 31 iz grupe 0—blokovi 3 do 7 iz grupe 0 se dovlače u blokove 3 do 7 ($5T_b$), dok se blokovi 8 do 31 već nalaze u blokovima 8 do 31 keš memorije i
- blokovi 0 do 9 iz grupe 1—blokovi 0 do 2 iz grupe 1 su već u blokovima 0 do 2, a blokovi 3 do 9 iz grupe 1 se dovlače u blokove 3 do 9 ($7T_b$).

Način smeštanja i sadržaji svakog ulaza TAG MEMORIJE prikazani su na slici **Error! Reference source not found.**.

| **ulaz** | **direktno** | | | |
|---|---|---|---|---|
| 0 | 0 | 1 | | |
| 1 | 0 | 1 | | |
| 2 | 0 | 1 | | |
| 3 | 0 | 1 | 0 | 1 |
| 4 | 0 | 1 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| 7 | 0 | 1 | 0 | 1 |
| 8 | 0 | | | 1 |
| 9 | 0 | | | 1 |
| 10 | 0 | | | |
| 11 | 0 | | | |
| 12 | 0 | | | |
| 13 | 0 | | | |
| 14 | 0 | | | |
| 15 | 0 | | | |
| 16 | 0 | | | |
| 17 | 0 | | | |
| 18 | 0 | | | |
| 19 | 0 | | | |
| 20 | 0 | | | |
| 21 | 0 | | | |
| 22 | 0 | | | |
| 23 | 0 | | | |
| 24 | 0 | | | |
| 25 | 0 | | | |
| 26 | 0 | | | |
| 27 | 0 | | | |
| 28 | 0 | | | |
| 29 | 0 | | | |
| 30 | 0 | | | |
| 31 | 0 | | | |

**Slika 1.1.27.** Sadržaji svih ulaza TAG MEMORIJE u slučaju direktnog preslikavanja

Vreme dohvatanja bloka je:

$T_b = 64T_m$.

Ukupno vreme dohvatanja svih blokova je:

$T_{b\text{all}} = 32T_b + 8T_b + 5T_b + 7T_b = 52T_b$.

Ukupno vreme svih pristupa keš memoriji je:

$T_{k\text{all}} = ((199 - 10 + 1) + (2555 - 200 + 1)\cdot 2 + (2655 - 2556 + 1))T_k = 5002T_k$.

Ukupno vreme je:

$T = T_{b\text{all}} + T_{k\text{all}} = 52\cdot 64\cdot T_m + 5002T_k$.

b2) U slučaju asocijativnog preslikavanja operacije čitanja blokova operativne memorije se izvode sledećim redosledom:

- blokovi 0 do 39 i
- blokovi 3 do 41.

Ovi blokovi se smeštaju u keš memoriju na sledeći način:

- blokovi 0 do 39—blokovi 0 do 31 se dovlače u ulaze 0 do 31 ($32T_b$), a blokovi 32 do 39 se dovlače u ulaze 0 do 7 ($8T_b$) i
- blokovi 3 do 41—blokovi 3 do 26 se dovlače u ulaze 8 do 31 ($24T_b$), a blokovi 9 do 41 se dovlače u ulaze 0 do 14 ($15T_b$).

Način smeštanja i sadržaji svakog ulaza TAG MEMORIJE prikazani su na slici **Error! Reference source not found.**.

| ulaz | asocijativno | | |
|---|---|---|---|
| 0 | 0 | 32 | 27 |
| 1 | 1 | 33 | 28 |
| 2 | 2 | 34 | 29 |
| 3 | 3 | 35 | 30 |
| 4 | 4 | 36 | 31 |
| 5 | 5 | 37 | 32 |
| 6 | 6 | 38 | 33 |
| 7 | 7 | 39 | 34 |
| 8 | 8 | 3 | 35 |
| 9 | 9 | 4 | 36 |
| 10 | 10 | 5 | 37 |
| 11 | 11 | 6 | 38 |
| 12 | 12 | 7 | 39 |
| 13 | 13 | 8 | 40 |
| 14 | 14 | 9 | 41 |
| 15 | 15 | 10 | |
| 16 | 16 | 11 | |
| 17 | 17 | 12 | |
| 18 | 18 | 13 | |
| 19 | 19 | 14 | |
| 20 | 20 | 15 | |
| 21 | 21 | 16 | |
| 22 | 22 | 17 | |
| 23 | 23 | 18 | |
| 24 | 24 | 19 | |
| 25 | 25 | 20 | |
| 26 | 26 | 21 | |
| 27 | 27 | 22 | |
| 28 | 28 | 23 | |
| 29 | 29 | 24 | |
| 30 | 30 | 25 | |
| 31 | 31 | 26 | |

**Slika 1.1.28.** Sadržaji svih ulaza TAG MEMORIJE u slučaju asocijativnog preslikavanja

Vreme dohvatanja bloka je:

$T_b = 64T_m$.

Ukupno vreme dohvatanja svih blokova je:

$T_{b\text{all}} = 32T_b + 8T_b + 24T_b + 15T_b = 79T_b$.

Ukupno vreme svih pristupa keš memoriji je:

$T_{k\text{all}} = ((199 - 10 + 1) + (2555 - 200 + 1)\cdot 2 + (2655 - 2556 + 1))T_k = 5002T_k$.

Ukupno vreme je:

$T = T_{b\text{all}} + T_{k\text{all}} = 79\cdot 64\cdot T_m + 5002T_k$.

b3) U slučaju set-asocijativnog preslikavanja sa dva bloka po setu operacije čitanja blokova operativne memorije se izvode sledećim redosledom:

- blokovi 0 do 15 iz grupe 0,
- blokovi 0 do 15 iz grupe 1,
- blokovi 0 do 7 iz grupe 2,
- blokovi 3 do 15 iz grupe 0,

- blokovi 0 do 15 iz grupe 1 i
- blokovi 0 do 9 iz grupe 2.

Ovi blokovi se smeštaju u keš memoriju na sledeći način:

- blokovi 0 do 15 iz grupe 0—oni se dovlače u ulaze 0 setova 0 do 15 ($16T_b$),
- blokovi 0 do 15 iz grupe 1—oni se dovlače u ulaze 1 setova 0 do 15 ($16T_b$),
- blokovi 0 do 7 iz grupe 2—oni se dovlače u ulaze 0 setova 0 do 7 ($8T_b$),
- blokovi 3 do 15 iz grupe 0—blokovi 3 do 7 iz grupe 0 se dovlače u ulaze 1 setova 3 do 7 ($5T_b$), dok su blokovi 8 do 15 iz grupe 0 već u ulazima 0 setova 8 do 15,
- blokovi 0 do 15 iz grupe 1—blokovi 0 do 2 iz grupe 2 su već u ulazima 1 za setove 0 do 2, blokovi 3 do 7 iz grupe 1 se dovlače u ulaze 0 setova 3 do 7 ($5T_b$), a blokovi 8 do 15 iz grupe 1 su već u ulazima 1 setova 8 do 15 i
- blokovi 0 do 9 iz grupe 2—blokovi 0 do 2 iz grupe 2 su već u ulazima 0 za setove 0 do 2, blokovi 3 do 7 iz grupe 2 se dovlače u ulaze 1 setova 3 do 7 ($5T_b$), a blokovi 8 i 9 iz grupe 2 se dovlače u ulaze 0 setova 8 i 9 ($2T_b$).

Način smeštanja i sadržaji svakog ulaza TAG MEMORIJE prikazani su na slici **Error! Reference source not found.**.

| | **set** | **grupa** | | | |
|---|---|---|---|---|---|
| | 0 | 0 | 2 | | |
| | 1 | 0 | 2 | | |
| | 2 | 0 | 2 | | |
| | 3 | 0 | 2 | 1 | |
| | 4 | 0 | 2 | 1 | |
| | 5 | 0 | 2 | 1 | |
| | 6 | 0 | 2 | 1 | |
| **ulaz 0** | 7 | 0 | 2 | 1 | |
| | 8 | 0 | | | 2 |
| | 9 | 0 | | | 2 |
| | 10 | 0 | | | |
| | 11 | 0 | | | |
| | 12 | 0 | | | |
| | 13 | 0 | | | |
| | 14 | 0 | | | |
| | 15 | 0 | | | |
| | 0 | 1 | | | |
| | 1 | 1 | | | |
| | 2 | 1 | | | |
| | 3 | 1 | 0 | 2 | |
| | 4 | 1 | 0 | 2 | |
| | 5 | 1 | 0 | 2 | |
| | 6 | 1 | 0 | 2 | |
| **ulaz 1** | 7 | 1 | 0 | 2 | |
| | 8 | 1 | | | |
| | 9 | 1 | | | |
| | 10 | 1 | | | |
| | 11 | 1 | | | |
| | 12 | 1 | | | |
| | 13 | 1 | | | |
| | 14 | 1 | | | |
| | 15 | 1 | | | |

**Slika 1.1.29.** Sadržaji svih ulaza TAG MEMORIJE u slučaju set-asocijativnog preslikavanja

Vreme dohvatanja bloka je:

$T_b = 64T_m$.

Ukupno vreme dohvatanja svih blokova je:

$T_{b\text{all}} = 16T_b + 16T_b + 8T_b + 5T_b + 5T_b + 5T_b + 2T_b = 57T_b$.

Ukupno vreme svih pristupa keš memoriji je:

$T_{k\text{all}} = ((199 - 10 + 1) + (2555 - 200 + 1)\cdot 2 + (2655 - 2556 + 1))T_k = 5002T_k$.

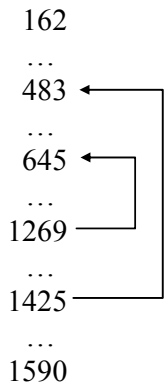
Ukupno vreme je:

$T = T_{b\text{all}} + T_{k\text{all}} = 57\cdot 64\cdot T_m + 5002T_k$.

## Zadatak 1.1.7

Posmatra se računar sa adresnim prostorom veličine 1 MW (mega reči). Adresibilna jedinica je 16-bitna reč. Procesor poseduje dve odvojene keš memorije, za instrukcije i za podatke. Posmatra se keš memorija za instrukcije. Kapacitet dela keš memorije za sadržaje je 1 KW, a preslikavanje je na nivou bloka veličine 8 reči. Memorijska reč ovog dela keš memorije je širine 16 bita, a vreme pristupa je $T_k$. Memorijska reč operativne memorije je širine 16 bita, a vreme pristupa je $T_m$. Pretpostavlja se da su sve instrukcije širine 16 bita.

Procesor izvršava sledeći program (adrese instrukcija su date decimalno):




Prva instrukcija je sa adrese 162, a poslednja sa adrese 1590. Spoljašnja petlja izvršava se 2 puta, a unutrašnja po 3 puta u svakoj iteraciji spoljašnje petlje. Keš memorija je prazna na početku. Pretpostaviti da procesor najpre dovuče ceo blok iz operativne u keš memoriju, pa tek onda očitava reč iz keš memorije.

Posmatraju se tri realizacije keš memorije:

- sa asocijativnim preslikavanjem
- sa direktnim preslikavanjem
- sa set-asocijativnim preslikavanjem, sa po dva ulaza u setu.

a) Dati strukturu adrese za sve tri realizacije keš memorije i napisati vrednosti polja adrese koja se pojavljuju pri izvršavanju datog programa.

b) Dati stanje relevantnih delova keš memorije (osim dela koji čuva sadržaje), za sve realizacije.

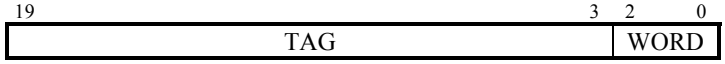
c) Izračunati ukupno vreme potrebno za očitavanje instrukcija datog programa za sve tri realizacije i uporediti ih.

**Rešenje:**

a) Kako je adresni prostor računara 1 MW = $2^{20}$ W, a adresibilna jedinica je reč, širina adresne reči je 20 bita. Blok je veličine 8 reči, pa je polje WORD koje određuje redni broj reči unutar bloka širine 3 bita, u sva tri slučaja realizacije keš memorije. Deo keš memorije za sadržaj ima kapacitet 1 KW = $2^{10}$ W. Prema tome, operativna memorija ima 128 K blokova ($2^{17}$ blokova), a keš memorija sadrži 128 blokova ($2^7$ blokova).

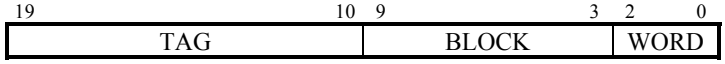
a1) Asocijativno preslikavanje

Značenje relevantnih polja adrese objašnjeno je u prethodnim zadacima. Struktura adrese je sledeća:



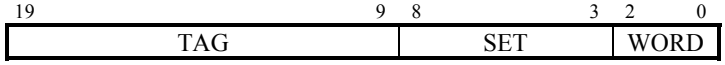

a2) Direktno preslikavanje

Značenje relevantnih polja adrese objašnjeno je u prethodnim zadacima. Kako keš memorija sadrži $2^7$ blokova, struktura adrese je sledeća:




a3) Set-asocijativno preslikavanje

Značenje relevantnih polja adrese objašnjeno je u prethodnim zadacima. Kako keš memorija sadrži $2^7$ blokova i 2 bloka po setu, postoji $2^6$ setova, pa je struktura adrese sledeća:




Vrednosti odgovarajućih polja relevantnih adresa za sva tri slučaja, pri izvršavanju datog programa date su na slici 1.1.30 (sve vrednosti su decimalne).

| | **asocijativno** | | **direktno** | | | **set-asocijativno** | | |
|---|---|---|---|---|---|---|---|---|
| **adresa** | **blok** | **reč** | **grupa** | **blok** | **reč** | **grupa** | **blok** | **reč** |
| 162 | 20 | 2 | 0 | 20 | 2 | 0 | 20 | 2 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 483 | 60 | 3 | 0 | 60 | 3 | 0 | 60 | 3 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 645 | 80 | 5 | 0 | 80 | 5 | 1 | 16 | 5 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1269 | 158 | 5 | 1 | 30 | 5 | 2 | 30 | 5 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1425 | 178 | 1 | 1 | 50 | 1 | 2 | 50 | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1590 | 198 | 6 | 1 | 70 | 6 | 3 | 6 | 6 |

**Slika 1.1.30.** Vrednosti polja relevantnih adresa za sva tri tipa preslikavanja

b) U posebnim odeljcima za svaki tip preslikavanja su prikazani poredak čitanja blokova, način smeštanja u keš memoriju i sadržaj TAG MEMORIJE.

b1) Asocijativno preslikavanje

Operacije čitanja se vrše nad blokovima operativne memorije po sledećem redosledu:

- Blokovi 20 do 158
- Blokovi 80 do 158
- Blokovi 80 do 178

•Blokovi 60 do 158
•Blokovi 80 do 158
•Blokovi 80 do 198

Deo TAG MEMORIJA keš memorije ima 128 adresnih ulaza ($2^7$ ulaza). Navedeni blokovi se smeštaju u keš memoriju na sledeći način:

•Blokovi 20 do 147 u ulaze 0 do 127 ($128T_b$)
•Blokovi 148 do 158 u ulaze 0 do 10 ($11T_b$)
•Blokovi 159 do 178 u ulaze 11 do 30 ($20T_b$)
•Blokovi 179 do 198 u ulaze 31 do 50 ($20T_b$)

Stanje TAG MEMORIJE na kraju izvršenja programa dato je na slici **Error! Reference source not found.**.

| Ulaz | Sadržaj |
|---|---|
| 0 | 148 |
| ⋮ | ⋮ |
| 50 | 198 |
| 51 | 71 |
| ⋮ | ⋮ |
| 127 | 147 |

**Slika 1.1.31.** Sadržaj TAG MEMORIJE na kraju programa za asocijatovno preslikavanje

b2) Direktno preslikavanje

Operacije čitanja se vrše nad blokovima operativne memorije po sledećem redosledu:

•Blokovi 20 do 127 grupe 0
•Blokovi 0 do 30 grupe 1
•Blokovi 80 do 127 grupe 0
•Blokovi 0 do 30 grupe 1
•Blokovi 80 do 127 grupe 0
•Blokovi 0 do 50 grupe 1
•Blokovi 60 do 127 grupe 0
•Blokovi 0 do 30 grupe 1
•Blokovi 80 do 127 grupe 0
•Blokovi 0 do 30 grupe 1
•Blokovi 80 do 127 grupe 0
•Blokovi 0 do 70 grupe 1

Ovi blokovi se smeštaju u keš memoriju na sledeći način:

•Blokovi 20 do 127 grupe 0 se dohvataju i smeštaju u blokove 20 do 127 ($108T_b$)
•Blokovi 0 do 30 grupe 1 se dohvataju i smeštaju u blokove 0 do 30 ($31T_b$)
•Blokovi 31 do 50 grupe 1 se dohvataju i smeštaju u blokove 31 do 50 ($20T_b$)
•Blokovi 51 do 70 grupe 1 se dohvataju i smeštaju u blokove 51 do 70 ($20T_b$)

Stanje TAG MEMORIJE na kraju izvršenja programa dato je na slici **Error! Reference source not found.**.

| Blok | Sadržaj |
|---|---|
| 0 | 1 |
| ⋮ | ⋮ |

| | |
|---|---|
| 70 | 1 |
| 71 | 0 |
| ⋮ | ⋮ |
| 127 | 0 |

**Slika 1.1.32.** Sadržaj TAG MEMORIJE na kraju programa za direktno preslikavanje

b3) Set-asocijativno preslikavanje

Slika 1.1.33 prikazuje sekvence blokova operativne memorije koji se adresiraju i njihove grupe, kao i broj promašaja u keš memoriji:

| Grupa | Blok | Broj promašaja |
|---|---|---|
| 0 | 20–63 | 44 |
| 1 | 0–63 | 64 |
| 2 | 0–30 | 31 |
| 1 | 16–63 | 0 |
| 2 | 0–30 | 0 |
| 1 | 16–63 | 0 |
| 2 | 0–50 | 20 |
| 0 | 60–63 | 0 |
| 1 | 0–63 | 0 |
| 2 | 0–30 | 0 |
| 1 | 16–63 | 0 |
| 2 | 0–30 | 0 |
| 1 | 16–63 | 0 |
| 2 | 0–63 | 13 |
| 3 | 0–6 | 7 |

**Slika 1.1.33.** Poredak pristupa sekvencama blokova prikazan u okviru odgovarajućih grupa

Slika **Error! Reference source not found.** prikazuje način smeštanja ovih blokova operativne memorije u setove i ulaze keš memorije, kao i krajnje stanje TAG MEMORIJE:

Ulaz 0:

| | | |
|---|---|---|
| 0 | 1 | 3 |
| ⋮ | ⋮ | ⋮ |
| 6 | 1 | 3 |
| 7 | 1 | |
| ⋮ | ⋮ | ⋮ |
| 19 | 1 | |
| 20 | 0 | 2 |
| ⋮ | ⋮ | ⋮ |
| 30 | 0 | 2 |
| 31 | 0 | 2 |
| ⋮ | ⋮ | ⋮ |
| 50 | 0 | 2 |
| 51 | 0 | 2 |
| ⋮ | ⋮ | ⋮ |
| 63 | 0 | 2 |

Ulaz 1:

| | |
|---|---|
| 0 | 2 |
| ⋮ | ⋮ |
| 19 | 2 |
| 20 | 1 |

| | |
|---|---|
| ⋮ | ⋮ |
| 63 | 1 |

| Blok | Ulaz 0 | Ulaz 1 |
|---|---|---|
| 0 | 3 | 2 |
| ⋮ | ⋮ | ⋮ |
| 6 | 3 | 2 |
| 7 | 1 | 2 |
| ⋮ | ⋮ | ⋮ |
| 19 | 1 | 2 |
| 20 | 2 | 1 |
| ⋮ | ⋮ | ⋮ |
| 63 | 2 | 1 |

**Slika 1.1.34.** Način smeštanja blokova po setovima i ulazima keš memorije kao i krajnje stanje TAG MEMORIJE

c) Ukupno vreme za bilo koju realizaciju iznosi:

$T = N_{\text{miss}} \cdot T_b + N \cdot T_k$,

gde je $N_{\text{miss}}$ broj promašaja u keš memoriji, $N$ ukupni broj operacija, a $T_b$ vreme dovlačenja jednog bloka iz operativne memorije:

$T_b = 8 \cdot T_m$

Kako su svi ostali parametri jednaki za sve tri realizacije, jedino $N_{\text{miss}}$ opisuje efikasnost pojedine realizacije. Ovaj broj iznosi:

- za asocijativno preslikavanje: $N_{\text{miss}} = 128 + 11 + 20 + 20 = 179$;
- za direktno preslikavanje: $N_{\text{miss}} = 108 + 31 + 20 + 20 = 179$;
- za set-asocijativno preslikavanje: $N_{\text{miss}} = 44 + 64 + 31 + 20 + 13 + 7 = 179$.

Dakle, za dati primer su sve tri tehnike (sasvim slučajno!) dale iste rezultate.

### Zadatak 1.1.8

Potpuno tačan LRU aor oct 94 (30)

Posmatra se računar sa sledećim karakteristikama. Operativna memorija je kapaciteta $2^{16}$ reči a širina reči je 16 bita. Procesor generiše adrese koje se odnose na 16-bitne adrese

Procesor ima keš memoriju sa asocijativnim preslikavanjem na nivou bloka sa četiri ulaza.Veličina bloka je 16 reči. Koristi se LRU algoritam zamene blokova i write-back algoritam za ažuriranje sadržaja operativne memorije. LRU algoritam je realizovan sa četiri brojača po modulu 4, svaki uz svoj ulaz keš memorije. Ulaz odabran za zamenu je onaj čiji brojač ima vrednost 0. Na početku je keš memorija bila prazna a svi brojači su imali vrednost 0. Vrednost brojača se ažurira na isti način kada je keš memorija puna i kada je prazna. Procesor generiše sledeću sekvencu adresa sa tipom operacije naznačenim u zagradi posle svake adrese (R = read, W = write):

734Fh (R), DE38h (R), FF03h (W), 7350h (R), 7351h (R), DE48h (W), 7352h (R), 7348h (R).

(10) Navesti sadržaje asocijativnog (TAG) dela (A0, A1, A2 i A3) za ulaze 0 do 3 keš memorije nakon date sekvence adresa.

(5) Navesti vrednosti valid bitova (V0, V1, V2 i V3) i dirty bitova (D0, D1, D2 i D3) za ulaze 0 do 3 keš memorije nakon date sekvence adresa.
(5) Navesti vrednosti LRU brojača C1 nakon svake generisane adrese u zadatoj sekvenci adresa.
(10) Pretpostaviti da je zadata keš memorija sa direktnim preslikavanjem na nivou bloka. Veličina bloka je 256 reči. Kapacitet DATA dela keš memorije je 1 K reči. Navesti sekvencu vrednosti TAG polja za zadatu sekvencu adresa.

**Rešenje:**

Kako je u zadatku rečeno da se za zamenu bira ulaz sa vrednošću LRU brojača 0, to znači da ova vrednost označava najdavnije korišćeni ulaz. Jasno je tako da najveća vrednost 3 označava najskorije korišćeni ulaz. Algoritam ažuriranja LRU brojača pri referisanju nekog ulaza je, zbog toga, sledeći:

1) dekrementirati sve vrednosti LRU brojača koje su veće od vrednosti brojača ulaza koji se referiše; na ovaj način se svi ostali ulazi koji su skorije korišćeni od referisanog označavaju kao "davnije korišćeni za jedan korak"; brojači ulaza koji imaju manju vrednost od referisanog se ne menjaju (oni su ionako davnije korišćeni);
2) postaviti LRU borač referisanog ulaza na najveću vrednost (binarno sve jedinice); na ovaj način se referisani ulaz označava kao najskorije korišćen.

Na primer, posmatrajmo sledeće stanje LRU brojača:

| Ulaz | LRU brojač (dec) |
|---|---|
| 0 | 0 |
| 1 | 3 |
| 2 | 1 |
| 3 | 2 |

Pretpostavimo da se dogodilo obraćanje ulazu broj 2. Sprovodeći opisani algoritam, stanje posle ovog obraćanja je sledeće:

| Ulaz | LRU brojač (dec) |
|---|---|
| 0 | 0 |
| 1 | 2 |
| 2 | 3 |
| 3 | 1 |

Ostaje pitanje kako se sprovodi algoritam inicijalno, posle samog reseta keš memorije (tzv. "hladni start"). Prilikom implementacije LRU algoritma, najjednostavnije je da se algoritam sprovodi potpuno istovetno kao što se sprovodi i u "stacionarnom" stanju, onako kako je to opisano. U zadatku je rečeno da su inicijalno sve vrednosti LRU brojača 0 (posle reseta), pa se algoritam sprovodi tako što se na početku vrednosti 0 ne menjaju (jer nisu veće od vrednosti brojača referisanog ulaza), dok se brojač referisanog ulaza postavlja na 3. Donja tabela daje rezultate sprovođenja ovog algoritma po koracima. Treba primetiti da na početku postoji nekoliko ulaza sa vrednostima LRU brojača jednakim 0, pa se bilo koji od njih može izabrati za "zamenu" (odnosno popunu), da bi se i algoritam izbora ulaza za popunu sprovodio na isti način. Može se izabrati bilo koji od njih, na primer onaj sa najnižim brojem (što se jednostavnim hardverom rešava). Posle popune svih ulaza, uvek samo jedan ulaz ima vrednost brojača 0, pa je izbor jednoznačan.

Sledeća tabela sadrži vrednosti svih relevantnih delova keš memorije posle svake adrese u sekvenci adresa:

| adresa | A0 | A1 | A2 | A3 | C0 | C1 | C2 | C3 | V0 | V1 | V2 | V3 | D0 | D1 | D2 | D3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 734Fh | 734h | — | — | — | 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DE38h | 734h | DE3h | — | — | 2 | 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| FF03h | 734h | DE3h | FF0h | — | 1 | 2 | 3 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 7350h | 734h | DE3h | FF0h | 735h | 0 | 1 | 2 | 3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| 7351h | 734h | DE3h | FF0h | 735h | 0 | 1 | 2 | 3 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| DE48h | DE4h | DE3h | FF0h | 735h | 3 | 0 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 7352h | DE4h | DE3h | FF0h | 735h | 2 | 0 | 1 | 3 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 7348h | DE4h | 734h | FF0h | 735h | 1 | 3 | 0 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |

Prema tome, odgovori na postavljena pitanja su:

**a)** Sadržaji asocijativnog dela su: DE4h, 734h, FF0h, 735h.
**b)** Vrednosti valid bitova su 1, 1, 1, 1, a vrednosti dirty bitova su 1, 0, 1, 0.
**c)** Vrednosti brojača C1 su 0, 3, 2, 1, 1, 0, 0, 3.
**d)** Prema objašnjenjima iz zadatka **Error! Reference source not found.**, sekvenca vrednosti TAG polja je 011100b, 110111b, 111111b, 011100b, 011100b, 110111b, 011100b, 011100b.

## Zadatak 1.1.9

Približan LRU

Memorijski adresni prostor računara je 4 GB, a adresibilna jedinica je bajt. Keš memorija je sa set-asocijativnim preslikavanjem na nivou bloka. Blok je veličine 256 B, a keš memorija ima kapacitet memorije podataka 4 MB. Svaki set ima 4 ulaza.

Algoritam zamene je približan LRU. Ovaj algoritam se sprovodi na sledeći način. Svakom ulazu pridružen je jedan tzv. LRU pomerački 4-bitni registar koji ima inicijalnu vrednost 0. Kada se pristupa ulazu *i* nekog seta, LRU registri svih ulaza tog seta se pomeraju udesno za jedno mesto, s tim da se u registar ulaza *i* upisuje sleva 1, a u sve ostale 0. Kada se bira ulaz za zamenu, bira se onaj ulaz koji ima najmanju vrednost LRU registra, posmatranu kao neoznačeni broj. Taj odabrani registar se resetuje, a zatim se sprovodi prethodno opisani algoritam ažuriranja registara.

Procesor generiše sledeću sekvencu adresa (sve vrednosti su heksadecimalne):

12FF0A, 22FF0B, 30C0C, 32FF00, 42FF01, 130C01, 22FF00, 52FFFF.

a) Prikazati logičku strukturu adrese, označiti relevantna polja i objasniti njihovo značenje.

b) Napisati sekvencu vrednosti polja *Tag* i polja *Set* koje se referišu u datoj sekvenci.

c) Prikazati sadržaje ulaza setova koji su referisani datom sekvencom (označiti te setove), posle završetka date sekvence.

d) Zašto je opisani algoritam samo približan LRU, a ne potpuno tačan? Prikazati na primeru.

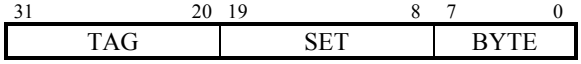
**Rešenje:**

*Diskusija:*

Potpuno tačan algoritam LRU je relativno složen za hardversku implementaciju. Zbog toga se u realnim sistemima najčešće primenjuju razne aproksimacije LRU algoritma koje imaju jednostavnu hardversku implementaciju, a daju vrlo približnu logiku zamene najdavnije referisanog ulaza. Jednostavnost hardverske implementacije treba da obuhvati dve komponente: 1) jednostavno ažuriranje podataka koji se koriste pri sprovođenju algoritma i 2) jednostavno pronalaženje ulaza za zamenu. Približni algoritmi ne moraju da zadovolje sasvim strogi poredak ulaza po hronologiji njihovog korišćenja, već da imaju manju ili veću korelaciju sa ovom hronologijom. Jedan ovakav algoritam je opisan u ovom zadatku.

*Rešenje:*

a) Kako je adresibilna jedinica bajt, a adresni prostor je veličine 4 GB, širina adrese je 32 bita. Sa druge strane, kapacitet keš memorije je $4\,\text{MB} = 2^{22}\,\text{B}$. Veličina bloka je $256\,\text{B} = 2^8\,\text{B}$, a svaki set ima $4 = 2^2$ ulaza, što znači da keš poseduje $2^{12}$ setova. Zbog toga je logička struktura adrese sledeća:




b) Prema rezultatima prethodne tačke, vrednosti polja TAG i SET za datu sekvencu su sledeće (sve vrednosti su heksadecimalne):

| Adresa | TAG | SET |
|---|---|---|
| 0012FF0A | 001 | 2FF |
| 0022FF0B | 002 | 2FF |
| 00030C0C | 000 | 30C |
| 0032FF00 | 003 | 2FF |
| 0042FF01 | 004 | 2FF |
| 00130C01 | 001 | 30C |
| 0022FF00 | 002 | 2FF |
| 0052FFFF | 005 | 2FF |

c) Kao što se vidi iz prethodne tačke, jedina dva seta koja su referisana su setovi 2FFh i 30Ch. Sledeće tabele daju pregled stanja svih ulaza ovih setova tokom izvršavanja date sekvence.

Posle šest prvih adresa:

| Ulaz | Tag | LRU Reg (bin) |
|---|---|---|
| 0 | 001 | 0001 |
| 1 | 002 | 0010 |
| 2 | 003 | 0100 |
| 3 | 004 | 1000 |

Set 2FF

| Ulaz | Tag | LRU Reg (bin) |
|---|---|---|
| 0 | 000 | 0100 |
| 1 | 001 | 1000 |
| 2 | | 0000 |
| 3 | | 0000 |

Set 30C

Dalje se set 30Ch više ne referiše, pa se njegovo stanje ne menja. Posle sedme adrese koja referiše ulaz 1 seta 2FFh (dogodio se pogodak), LRU registri se ažuriraju prema opisanom algoritmu, pa je stanje seta 2FFh sledeće:

| Ulaz | Tag | LRU Reg (bin) |
|---|---|---|
| 0 | 001 | 0000 |
| 1 | 002 | 1001 |
| 2 | 003 | 0010 |
| 3 | 004 | 0100 |

Set 2FF

Kada se generiše poslednja adresa iz sekvence, događa se promašaj u setu 2FFh. Za zamenu se bira ulaz 0, jer on ima najmanju vrednost LRU brojača. U ovom specijalnom slučaju, on je i najdavnije korišćeni ulaz, pa je opisani algoritam zapravo dao isti rezultat kao i potpuno tačan LRU. Stanje posle date sekvence je, dakle:

| Ulaz | Tag | LRU Reg (bin) |
|---|---|---|
| 0 | 005 | 1000 |
| 1 | 002 | 0100 |
| 2 | 003 | 0001 |
| 3 | 004 | 0010 |

Set 2FF

| Ulaz | Tag | LRU Reg (bin) |
|---|---|---|
| 0 | 000 | 0100 |
| 1 | 001 | 1000 |
| 2 | | 0000 |
| 3 | | 0000 |

Set 30C

d) Opisani algoritam koristi zapravo pomeračke registre za pamćenje istorije obraćanja ulazima. Svaki bit u svim registrima predstavlja jedno obraćanje iz prošlosti. Kako je veličina registara ograničena, i "pamćenje" prošlosti je ograničeno, pa se može dogoditi da se "zaboravi" razlika između korišćenja neka dva ulaza koja su dovoljno davno referisana. Na primer, posmatrajmo sledeću sekvencu obraćanja sa sledećim brojem ulaza u nekom setu: 1, 0, 2, 2, 2, 2. Posle ove sekvence, stanja LRU registara će biti:

| Ulaz | LRU Reg (bin) |
|---|---|
| 0 | 0000 |
| 1 | 0000 |
| 2 | 1111 |
| 3 | 0000 |

Dakle, izgubljena je informacija o razlici korišćenja ulaza 0 i 1, pa se za zamenu bira bilo koji od njih, recimo 0, iako je on skorije korišćen. Jasna su dva zaključka: 1) opisani efekat "zaboravljanja" ima sve manje značaja što su LRU pomerački registri duži, odnosno što je pamćenje prošlosti dalekosežnije; 2) kada su LRU registri dovoljno dugački, zapravo i nije bitno što se opisani efekat ispoljio, jer su neka dva ulaza čiji se registri ne razlikuju (jednaki su 0) oba dovoljno davno korišćena, pa je zapravo sve jedno koji će od njih biti zamenjen.

## Zadatak 1.1.10

LRU i486

Kod mikroprocesora Intel 80486 koristi se sledeća aproksimacija LRU algoritma. Svaki set poseduje 4 ulaza. Ta četiri ulaza se dele da dve grupe (grupa 0 i grupa 1) od po dva ulaza (ulaz 0 i ulaz 1 u grupi). Za sprovođenje algoritma koriste se tri bita pridružena svakom setu. Jedan bit (bit B2) govori kojoj se grupi ulaza najskorije pristupalo, a druga dva bita (biti B1 i B0) govore kom se ulazu iz svake grupe skorije pristupalo.

a) Definisati algoritam i hardver kojim se ažuriraju opisana tri bita pri jednom pristupu nekom ulazu.

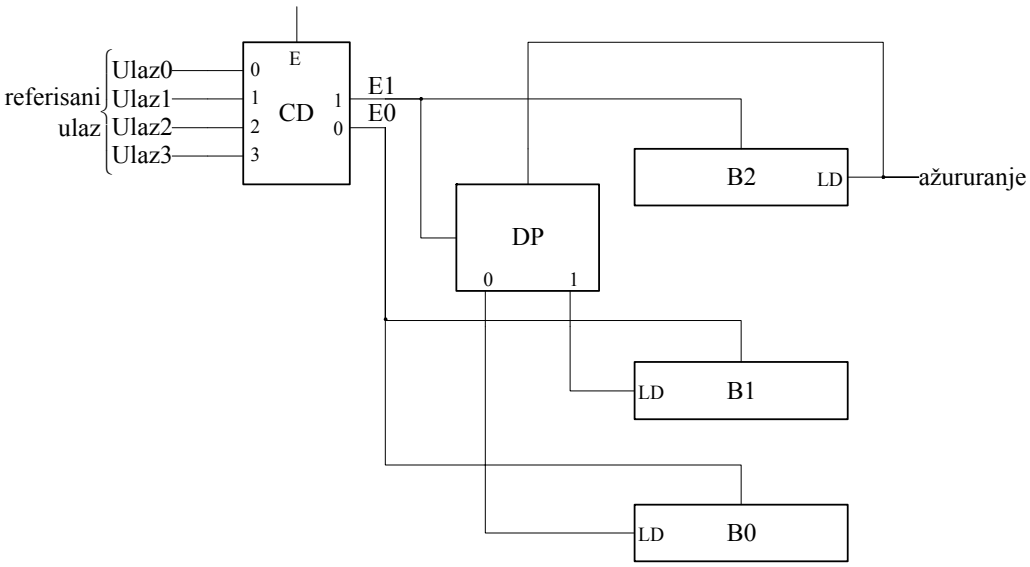
b) Definisati algoritam i hardver kojim se vrši izbor ulaza za zamenu.

c) Za keš memoriju i sekvencu adresa iz zadatka 0, dati vrednosti opisanih bita seta 2FFh posle završetka date sekvence.

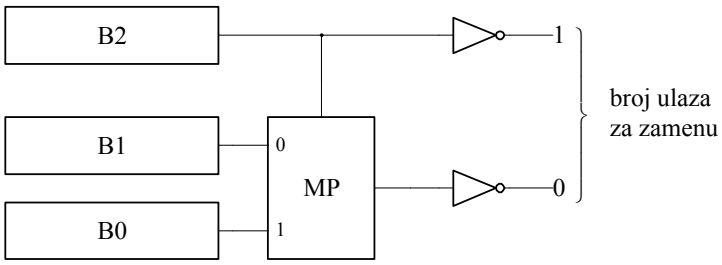
**Rešenje:**

a) Neka je broj ulaza (0 do 3) kome se vrši obraćanje kodovan sa 2 bita E1 i E0. Pri svakom obraćanju treba ažurirati opisane bite na sledeći način: u bit B2 treba upisati broj grupe ulaza kojoj se pristupa; to je zapravo vrednost bita E1. Ako je to grupa 0, onda u bit B0 treba upisati kom se ulazu iz grupe pristupalo, a to je zapravo vrednost

E0; ako je to grupa 1, onda u bit B1 treba upisati ovu vrednost E0. Prema tome, potrebni hardver izgleda ovako:




b) Izbor ulaza za zamenu vrši se po sledećem algoritmu: bira se za zamenu ulaz koji je davnije korišćen iz one grupe koja je davnije korišćena. Na primer, ako je vrednost bita B2 jednaka 0, bira se grupa 1; pri tom, ako je vrednost bita B1 jednaka 0, bira se ulaz 1 iz te grupe. Hardver izgleda ovako:




c) Data sekvenca predstavlja traženje sledećih vrednosti TAG u setu 2FFh: 1, 2, 3, 4, 2, 5. Stanja ulaza i bita B ovog seta tokom izvršavanja ove sekvence su sledeća:

| TAG | Ulaz0 | Ulaz1 | Ulaz2 | Ulaz3 | bitovi B2…0 |
|---|---|---|---|---|---|
| 1 | 1 | | | | 000 |
| 2 | 1 | 2 | | | 001 |
| 3 | 1 | 2 | 3 | | 101 |
| 4 | 1 | 2 | 3 | 4 | 111 |
| 2 | 1 | 2 | 3 | 4 | 011 |
| 5 | 1 | 2 | 5 | 4 | 101 |

Upoređivanjem konačnog stanja seta sa onim iz zadatka 0, može se primetiti da za dati primer algoritam iz ovog zadatka daje lošiju odluku pri zameni ulaza 2, koji je skorije korišćen nego ulaz 0.

## Zadatak 1.1.11

Upoređenje algoritama zamene

Posmatraju se sledeća četiri algoritma zamene: FIFO, potpuno tačni LRU opisan u zadatku 0, približni LRU opisan u zadatku 0, sa širinom pomeračkih registara od po 3 bita, i približni LRU procesora i80486 opisan u zadatku 0. U sva četiri slučaja posmatra se jedan set keš memorije sa četiri ulaza.

a) Uporediti četiri opisana algoritma sa stanovišta hardverske implementacije, i to sa dva aspekta: mehanizma ažuriranja informacija potrebnih za sprovođenje algoritma, pri svakom obraćanju nekom ulazu, i mehanizma izbora ulaza za zamenu.

b) Uporediti četiri opisana algoritma sa stanovišta efikasnosti na primeru sledeće sekvence vrednosti TAG polja adresa koje pripadaju istom posmatranom setu: 1, 2, 3, 4, 1, 2, 2, 2, 5, 4, 4, 1.

**Rešenje:**

a) Za implementaciju FIFO algoritma potreban je samo jedan brojač po modulu broja ulaza u setu (u ovom slučaju 4). Ovaj brojač inkrementira se pri svakoj *zameni* nekog ulaza (a ne pri svakom obraćanju). Ulaz za zamenu je onaj ulaz na koga ukazuje ovaj brojač.

Za implementaciju potpuno tačnog LRU algoritma potrebno je onoliko brojača koliko ima ulaza u setu, za svaki set. Kao što je opisano u zadatku 0, svi ovi brojači imaju $\log_2 n$ bita, gde je $n$ broj ulaza po setu. Kada se vrši obraćanje nekom ulazu, postupak je sledeći: svi brojači koji imaju veću (u dualnom rešenju manju) vrednost od vrednosti brojača ulaza kome se obraća se smanjuju (dualno: povećavaju) za jedan; ostali brojači se ne menjaju; zatim se brojač ulaza kome se obraća postavlja na najveću (dualno: najmanju) vrednost, tj. sve jedinice (dualno: nule). Hardverska implementacija ovog postupka je relativno složena: potrebno je komparatorima upoređivati vrednosti različitih brojača, a zatim je potrebno vršiti dekrementiranje (inkrementiranje) izabranih brojača. Za zamenu se, veoma jednostavno, bira onaj ulaz čiji brojač ima najmanju (dualno: najveću) vrednost, tj. sve nule (dualno: sve jedinice). Ovaj deo algoritma je jednostavan za implementaciju, jer se zahteva samo prepoznavanje svih nula (jedinica) u registru.

Za implementaciju približnog LRU algoritma iz zadatka 0, kao što je prikazano, potrebno je onoliko pomeračkih registara koliko ima ulaza u setu, za svaki set. Širina ovih registara je proizvoljna (što je veća, algoritam je bliži tačnom LRU). Pri obraćanju nekom ulazu, svi registri se pomeraju za jedno mesto udesno, a sleva se u njih upisuje 0, osim u registar ulaza kome se obraćalo, u koji se sleva upisuje 1. Za zamenu se bira registar koji ima najmanju binarnu vrednost (tj, najstariju jedinicu na najlakšem mestu). Kao što se vidi, mehanizam ažuriranja je jednostavan, dok je za izbor ulaza za zamenu potrebna nešto složenija logika. U svakom slučaju, ovaj algoritam je jednostavniji za implementaciju od potpuno tačnog LRU.

Za implementaciju približnog LRU algoritma procesora i80486 potrebno je samo tri bita za četiri ulaza po setu. Mehanizam ažuriranja ovih bita i izbora ulaza za zamenu detaljno je opisan u zadatku 0. Sasvim je jasno da je implementacija ovog algoritma jednostavnija od oba prethodna LRU algoritma.

Kao što se vidi, implementacija FIFO algoritma je jednostavna, znatno jednostavnija od implementacije tačnog LRU algoritma. Što se tiče LRU algoritama, složenost njihove implementacije opada kako je algoritam "dalji" od tačnog: implementacija tačnog LRU je složena, dok je implementacija LRU algoritma procesora i80486 najjednostavnija, jer je i ovaj algoritam najmanje tačan.

b)

b1) FIFO algoritam

U tablici je dat redosled stanja četiri ulaza posmatranog seta po koracima, kao i FIFO brojača *posle* zamene ulaza, za datu sekvencu:

| TAG | Ulaz 0 | Ulaz 1 | Ulaz 2 | Ulaz 3 | FIFO brojač | Promašaj |
|---|---|---|---|---|---|---|
| 1 | 1 | | | | 1 | * |
| 2 | 1 | 2 | | | 2 | * |
| 3 | 1 | 2 | 3 | | 3 | * |
| 4 | 1 | 2 | 3 | 4 | 0 | * |
| 1 | 1 | 2 | 3 | 4 | 0 | |
| 2 | 1 | 2 | 3 | 4 | 0 | |
| 2 | 1 | 2 | 3 | 4 | 0 | |
| 2 | 1 | 2 | 3 | 4 | 0 | |
| 5 | 5 | 2 | 3 | 4 | 1 | * |
| 4 | 5 | 2 | 3 | 4 | 1 | |
| 4 | 5 | 2 | 3 | 4 | 1 | |
| 1 | 5 | 1 | 3 | 4 | 2 | * |

Kako je broj promašaja najvažniji parametar pomoću koga možemo, za datu sekvencu, uporediti efikasnost različitih algoritama zamene, posmatraćemo ovaj broj za sve algoritme. Za FIFO algoritam broj promašaja iznosi 6.

b2) Tačni LRU algoritam

U tablici je dat redosled stanja četiri ulaza posmatranog seta po koracima, kao i LRU brojača *posle* obraćanja, za datu sekvencu:

| TAG | Ulaz 0 | Ulaz 1 | Ulaz 2 | Ulaz 3 | LRU 0 | LRU 1 | LRU 2 | LRU 3 | Promašaj |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | | 3 | 0 | 0 | 0 | * |
| 2 | 1 | 2 | | | 2 | 3 | 0 | 0 | * |
| 3 | 1 | 2 | 3 | | 1 | 2 | 3 | 0 | * |
| 4 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | * |
| 1 | 1 | 2 | 3 | 4 | 3 | 0 | 1 | 2 | |
| 2 | 1 | 2 | 3 | 4 | 2 | 3 | 0 | 1 | |
| 2 | 1 | 2 | 3 | 4 | 2 | 3 | 0 | 1 | |
| 2 | 1 | 2 | 3 | 4 | 2 | 3 | 0 | 1 | |
| 5 | 1 | 2 | 5 | 4 | 1 | 2 | 3 | 0 | * |
| 4 | 1 | 2 | 5 | 4 | 0 | 1 | 2 | 3 | |
| 4 | 1 | 2 | 5 | 4 | 0 | 1 | 2 | 3 | |
| 1 | 1 | 2 | 5 | 4 | 3 | 0 | 1 | 2 | |

Broj promašaja iznosi 5.

b3) Približni LRU algoritam sa pomeračkim registrima

U tablici je dat redosled stanja četiri ulaza posmatranog seta po koracima, kao i LRU pomeračkih registara (binarno) *posle* obraćanja, za datu sekvencu:

| TAG | Ulaz 0 | Ulaz 1 | Ulaz 2 | Ulaz 3 | LRU 0 | LRU 1 | LRU 2 | LRU 3 | Promašaj |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | | 100 | 000 | 000 | 000 | * |
| 2 | 1 | 2 | | | 010 | 100 | 000 | 000 | * |
| 3 | 1 | 2 | 3 | | 001 | 010 | 100 | 000 | * |
| 4 | 1 | 2 | 3 | 4 | 000 | 001 | 010 | 100 | * |
| 1 | 1 | 2 | 3 | 4 | 100 | 000 | 001 | 010 | |
| 2 | 1 | 2 | 3 | 4 | 010 | 100 | 000 | 001 | |
| 2 | 1 | 2 | 3 | 4 | 001 | 110 | 000 | 000 | |
| 2 | 1 | 2 | 3 | 4 | 000 | 111 | 000 | 000 | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 2 | 3 | 4 | 100 | 011 | 000 | 000 | * |
| 4 | 5 | 2 | 3 | 4 | 010 | 001 | 000 | 100 | |
| 4 | 5 | 2 | 3 | 4 | 001 | 000 | 000 | 110 | |
| 1 | 5 | 1 | 3 | 4 | 000 | 100 | 000 | 011 | * |

Broj promašaja iznosi 6.

b4) Približni LRU algoritam procesora i80486

U tablici je dat redosled stanja četiri ulaza posmatranog seta po koracima, kao i LRU bita (binarno) *posle* obraćanja, za datu sekvencu:

| TAG | Ulaz 0 | Ulaz 1 | Ulaz 2 | Ulaz 3 | LRU | Promašaj |
|---|---|---|---|---|---|---|
| 1 | 1 | | | | 000 | * |
| 2 | 1 | 2 | | | 001 | * |
| 3 | 1 | 2 | 3 | | 101 | * |
| 4 | 1 | 2 | 3 | 4 | 111 | * |
| 1 | 1 | 2 | 3 | 4 | 010 | |
| 2 | 1 | 2 | 3 | 4 | 011 | |
| 2 | 1 | 2 | 3 | 4 | 011 | |
| 2 | 1 | 2 | 3 | 4 | 011 | |
| 5 | 1 | 2 | 5 | 4 | 101 | * |
| 4 | 1 | 2 | 5 | 4 | 111 | |
| 4 | 1 | 2 | 5 | 4 | 111 | |
| 1 | 1 | 2 | 5 | 4 | 010 | |

Broj promašaja iznosi 5.

Važno je naglasiti da se iz navedenog može samo steći osećaj o delovanju pojedinih algoritama i uporediti ti algoritmi samo za datu sekvencu. *Nikako* se ne mogu izvući opšti zaključci o strogom poretku efikasnosti navedenih algoritama u proizvoljnom slučaju. Naime, efikasnost algoritama se može izmeriti za neku *dovoljno veliku* posmatranu sekvencu (uzorak) koji se dobija kao trag generisanih adresa skupa aplikacija za koju se projektuje procesor i keš memorija. Odluka o izboru algoritma koji će se implemetirati donosi se na osnovu tako dobijenih rezultata i hardverske složenosti opisanih algoritama (tzv. *cost-benefit* analiza).

## Zadatak 1.1.12

aor sep 95 (30)

Posmatra se procesor sa bezadresnim formatom instrukcija. Procesor ima razdvojen aritmetički i kontrolni stek. Aritmetički stek se koristi kao implicitni izvor i odredište za operande u aritmetičkim, logičkim, pomeračkim i instrukcijama prenosa podataka. Kontrolni stek se koristi za čuvanje odgovarajućih vrednosti pri skokovima na potprograme i prekidne rutine, i za njihovo restauriranje prilikom povratka iz potprograma i prekidnih rutina. Registar SPA ukazuje na vrh aritmetičkog steka, a registar SPC ukazuje na vrh kontrolnog steka. Oba steka rastu ka višim adresama u memoriji. Vrh steka je prva slobodna lokacija. Kada se prave skokovi na potprograme ili prekidne rutine na kontrolnom steku se čuvaju prvo PC pa zatim PSW.

Kapacitet operativne memorije je $2^{16}$ reči, širina reči je 16 bita, a procesor generiše adrese koje se odnose na 16-bitne veličine. Operacije se izvode samo nad 16-bitnim celobrojnim veličinama.

Procesor koristi keš memoriju sa set-asocijativnim preslikavanjem i četiri ulaza po setu, veličinom bloka od 64 reči i *write-back* algoritam ažuriranja operativne memorije. Kapacitet DATA dela keš memorije je 32 Kbajta. Na početku je stek bio prazan. Vrednosti registara SPA i SPC su 4FFFh i 7F30h, respektivno. Izvršava se sledeća sekvenca instrukcija:

```
adresa:     instrukcija:
FF00        PUSH   imm(2)
FF02        PUSH   abs(2255h)
FF04        ADD
FF05        POP    abs(2255h)
FF07        JSR    1224h          ; poziv potprograma
```

(10) Nacrtati strukturu adrese operativne memorije, označiti širinu svakog polja i označiti njihovu svrhu.
(10) Napisati sekvencu vradnosti za polja TAG i BLOCK za adrese koje se generišu prilikom izvršavanja navedene sekvence instrukcija.
(10) Napisati vrednosti svih relevantnih TAG polja, zajedno sa vrednostima odgovarajućih V (valid) i D (*dirty*) bitova nakon izvršenja navedene sekvence instrukcija.

**Rešenje:**

a) Kako je širina adresne reči 16 bita, veličina bloka $64 = 2^6$ reči, a keš memorija poseduje 4 ulaza po setu i kapacitet $32\ KB = 16\ KW = 2^6 \cdot 64 \cdot 4\ W$, postoji $2^6$ setova, pa je struktura adresne reči sledeća:


b) Sekvenca adresa koje generiše procesor prilikom izvršavanja datog programa, kao i vrednosti odgovarajućih polja date su u sledećoj tabeli:

| dares | TAG | BLOCK | operacija |
|---|---|---|---|
| FF00 | F | 3C | R |
| FF01 | F | 3C | R |
| 4FFF | 4 | 3F | W |
| FF02 | F | 3C | R |
| FF03 | F | 3C | R |
| 2255 | 2 | 09 | R |
| 5000 | 5 | 00 | W |
| FF04 | F | 3C | R |
| 5000 | 5 | 00 | R |
| 4FFF | 4 | 3F | R |
| 4FFF | 4 | 3F | W |
| FF05 | F | 3C | R |
| FF06 | F | 3C | R |
| 4FFF | 4 | 3F | R |
| 2255 | 2 | 09 | W |
| FF07 | F | 3C | R |
| FF08 | F | 3C | R |
| 7F30 | 7 | 3C | W |
| 7F31 | 7 | 3C | W |

c) U sledećoj tabeli dati su sadržaji relevantnih setova keš memorije, posle izvršavanja date sekvence.

| | | | | |
|---|---|---|---|---|
| ulaz 0 | 5, D=1 | 2, D=1 | F, D=0 | 4, D=1 |
| ulaz 1 | | | 7, D=1 | |
| ulaz 2 | | | | |
| ulaz 3 | | | | |
| | set 00h | set 09h | set 3Ch | set 3Fh |

## Zadatak 1.1.13

aor jan 95

Procesor je troadresni i ima 8 registara opšte namene, R0 do R7, svi su 32-bitni. Memorijske adrese su širine 32 bita, širina magistrale podataka je 32 bita, a adresiranje je na nivou 32-bitnih reči. Procesor operiše samo sa 32-bitnim celobrojnim veličinama (u daljem tekstu *reč* označava 32-bitnu veličinu).

Procesor poseduje posebnu instrukciju za rad sa nizovima reči u memoriji (*string* instrukcija). To je troadresna instrukcija MOVS R*dst*, R*src*, R*cnt*. Ona kopira blok memorijskih reči sa jednog mesta na drugo. Adresa početka izvorišnog bloka je u registru R*src*, adresa početka odredišnog bloka je u registru R*dst*, a dužina bloka je u registru R*cnt*, gde su R*dst*, R*src*, R*cnt* registri opšte namene (R0 do R7). Sadržaj registra R*cnt* može biti i nula. Ova instrukcija menja vrednosti ovih registara. Na kraju izvršavanja ove instrukcije, R*src* i R*dst* ukazuju na prvu memorijsku reč iza bloka izvorišta/odredišta, R*cnt* ima vrednost nula, a indikator Z se postavlja na 1.

Između opisanog procesora i magistrale računara vezana je keš memorija organizovana asocijativno na nivou bloka od 16 MW (mega reči), sa 8 ulaza u asocijativnu memoriju i *write-back* tehnikom ažuriranja operativne memorije. Na početku je keš bio prazan. Posmatra se sledeća sekvenca instrukcija koje procesor izvršava:

```
adresa      instrukcija
FFFFFF00    MOV   R0,#10000000h  ; ovde ima 1 i sedam nula
FFFFFF02    MOV   R1,#0
FFFFFF04    MOV   R2,#10000000h  ; ovde ima 1 i sedam nula
FFFFFF06    MOVS  R2,R1,R0
FFFFFF07    MOV   R0,#FFFFFFFh  ; ovde ima sedam cifara F
FFFFFF09    MOVS  R2,R1,R0
```

a)(10) Dati niz vrednosti polja TAG adresa koje se generišu pri izvršavanju datog programa. Ako se neka sekvenca vrednosti polja TAG ponavlja više puta, koristiti notaciju (*x*, *y*, *z*, …)*n*, koja označava da se niz vrednosti *x*, *y*, *z*, … ponavlja *n* puta.

b)(10) Dati sadržaj svih ulaza asocijativne memorije, kao i vrednosti V (*Valid*) i D (*Dirty*) bita, posle završetka prve MOVS, kao i posle završetka druge MOVS instrukcije, ako je algoritam zamene FIFO.

c)(10) Isto kao pod b), samo je algoritam zamene LRU.

**Rešenje:**

a) Prema opisu MOVS instrukcije, vidi se da se u oba slučaja vrši prenos po FFFFFFh reči. Kako je širina adresne reči 32 bita, a veličina bloka 16 MW = $2^{24}$ W, polje TAG zauzima viših 8 bita adresne reči. Sekvenca vrednosti polja TAG adresa koje generiše procesor tokom izvršavanja datog programa je tako (sve vrednosti su heksadecimalne, *n*=FFFFFFh):

(FF)7, (00,10)*n*, (01,11)*n*, (02,12)*n*, …., (0F,1F)*n*, (FF)3, (10,20)*n*, (11,21)*n*, (12,22)*n*, …., (1F,2F)*n*.

b) Posle prve MOVS instrukcije, sadržaj asocijativnog dela keš memorije je sledeći:

| Ul. | TAG | V | D |
|---|---|---|---|
| 0 | 1F | 1 | 1 |
| 1 | 0C | 1 | 0 |
| 2 | 1C | 1 | 1 |
| 3 | 0D | 1 | 0 |
| 4 | 1D | 1 | 1 |
| 5 | 0E | 1 | 0 |
| 6 | 1E | 1 | 1 |
| 7 | 0F | 1 | 0 |

Posle druge MOVS instrukcije, sadržaj asocijativnog dela keš memorije je sledeći:

| Ul. | TAG | V | D |
|---|---|---|---|
| 0 | 1F | 1 | 0 |
| 1 | 2F | 1 | 1 |
| 2 | 1C | 1 | 0 |
| 3 | 2C | 1 | 1 |
| 4 | 1D | 1 | 0 |
| 5 | 2D | 1 | 1 |
| 6 | 1E | 1 | 0 |
| 7 | 2E | 1 | 1 |

c) FIFO algoritam deluje na osnovu informacije o najdavnije *zamenjenom* ulazu, dok LRU deluje na osnovu informacije o najdavnije *korišćenom* ulazu. Kako se u datom primeru redosled zamene i redosled korišćenja ulaza poklapaju, rezultat za LRU algoritam je potpuno isti kao u tački pod b).

## Zadatak 1.1.14

aor jan 96

Adresna reč procesora široka je 32 bita, a adresibilna jedinica je 32-bitna reč. Između magistrale procesora i memorije računara vezana je keš memorija organizovana set-asocijativno na nivou bloka od 16 MW (mega reči), sa 16 setova, dva ulaza po setu i *store-through* tehnikom ažuriranja operativne memorije. Na početku je keš bio prazan. Posmatra se sledeća sekvenca adresa koje procesor generiše (sve vrednosti su heksadecimalne):

02000000, 32001245, 17000112, 02000001, 22004545, 27001112, 12000440, 37006886, 27001288, 22001223, 02011212, 17000112, 07000000

a)(10) Dati niz vrednosti polja TAG date sekvence adresa.

b)(10) Dati sadržaj oba ulaza svih setova koji su referisani u datoj sekvenci, kao i vrednosti V (*Valid*) bita, posle završetka date sekvence, ako je algoritam zamene FIFO. Označiti setove.

c)(10) Isto kao pod b), samo je algoritam zamene LRU.

**Rešenje:**

a) Kako je veličina bloka 16 MW = $2^{24}$ W, keš memorija sadrži 16 = $2^4$ setova, a adresna reč je široka 32 bita, polje TAG zauzima 4 najviša bita, a polje SET 4 srednja bita adresne reči. Tako je sekvenca vrednosti polja TAG sledeća:

0, 3, 1, 0, 2, 2, 1, 3, 2, 2, 0, 1, 0

Niz vrednosti polja SET je sledeći:

2, 2, 7, 2, 2, 7, 2, 7, 7, 2, 2, 7, 7

b) Sadržaj relevantnih setova keš memorije posle date sekvence, za FIFO algoritam zamene je sledeći:

| Ulaz | Tag | V |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

Set 2

| Ulaz | Tag | V |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

Set 7

c) Sadržaj relevantnih setova keš memorije posle date sekvence, za LRU algoritam zamene je sledeći:

| Ulaz | Tag | V |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 2 | 1 |

Set 2

| Ulaz | Tag | V |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 1 |

Set 7

## Zadatak 1.1.15

Procesor je jednoadresni i ima samo jedan registar podataka (akumulator, Acc) i 16 adresnih registara, X0 do X15, svi su 32-bitni. Pored ovih registara, postoji registar PSW sa uobičajenim značenjem, koji ima indikatore N, Z, C i V. Memorijske adrese su širine 32 bita, širina magistrale podataka je 32 bita, adresiranje je na nivou 32-bitnih reči. Procesor operiše samo sa 32-bitnim celobrojnim veličinama (u daljem tekstu *reč* označava 32-bitnu veličinu).

Između opisanog procesora i magistrale vezan je keš kontroler, organizovan set-asocijativno na nivou bloka. Blok je veličine 4 K reči, postoje četiri ulaza po setu, a kapacitet dela keš memorije za sadržaj je 256 K reči. Algoritam zamene je LRU, a strategija ažuriranja operativne memorije je *store-through*. Procesor izvršava sledeći program (ispred instrukcije je njena adresa, heksadecimalno):

```
1FFA    LOAD    X0,2            ; X0:=2
1FFC    LOAD    X1,44001200H    ; X1:=44001200h
1FFE    ADD     X1              ; Acc:=Acc+Mem[X1]
1FFF    STORE   X1              ; Mem[X1]:=Acc
2000    MOV     A,X0            ; Acc:=X0
2001    DEC                     ; Acc:=Acc-1
2002    MOV     X0,A            ; X0:=Acc
2003    JNZ     1FFE            ; if Z=0 goto 1FFE
```

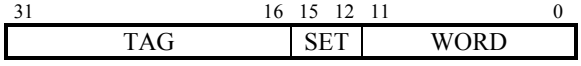
a)(10) Prikazati logičku strukturu adrese i označiti značenje i dužinu svakog polja.

b)(10) Prikazati sadržaj relevantnih setova *tag* memorije (koji su referencirani) posle izvršavanja datog dela programa (keš je na početku bio prazan).

c)(10) Izračunati broj ciklusa čitanja i upisa (posebno) na magistrali, koji se obave tokom izvršavanja datog dela programa.

**Rešenje:**

a) Kako je veličina bloka $4\,\mathrm{K} = 2^{12}$ reči, a keš poseduje 4 ulaza po setu i memoriju podataka kapaciteta $256\,\mathrm{K} = 2^{18} = 2^4 \cdot 4 \cdot 2^{12}$ reči, sledi da je polje SET širine 4 bita, a polje WORD 12 bita, pa je struktura adresne reči sledeća:




b) Sekvenca adresa koje se generišu tokom izvršavanja datog programa je sledeća (sve vrednosti su heksadecimalne):

1FFA, 1FFB, 1FFC, 1FFD, 1FFE, 44001200, 1FFF, 44001200, 2000, 2001, 2002, 2003,

1FFE, 44001200, 1FFF, 44001200, 2000, 2001, 2002, 2003

Sekvenca vrednosti polja TAG je tako:

0, 0, 0, 0, 0, 4400, 0, 4400, 0, 0, 0, 0, 0, 4400, 0, 4400, 0, 0, 0, 0.

Sekvenca vrednosti polja SET je:

1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2.

Sadržaj relevantnih setova keš memorije je na kraju:

| | Set 1 | Set 2 |
|---|---|---|
| ulaz 0 | 0 | 0 |
| ulaz 1 | 4400 | |
| ulaz 2 | | |
| ulaz 3 | | |

c) Ukupan broj promašaja u kešu pri izvršavanju date skevence je: *Nmiss*=3.

Ukupan broj ciklusa čitanja je: *Nread* = *Nmiss*·BlockSize =3·4 K = 12 K = 12288.

Ukupan broj operacija upisa jednak je broju ciklusa upisa (jer je tehnika *store-through*): *Nwrite* = 2.

## Zadatak 1.1.16

Razmatrani multiprocesorski sistem na slici ?? se sastoji od tri procesora P1, P2 i P3 koji su preko zajedničke magistrale povezani na zajedničku memoriju M. Svaki od procesora ima svoju privatnu keš memoriju (C1, C2 i C3) za podatke koje najčešće koristi. Neka se u početku podatak X sa vrednošću 0 nalazi u memoriji M, a ne nalazi se ni u jednoj od privatnih keš memorija. Pretpostavlja se sekvenca obraćanja ova tri procesora podatku X u sledećem poretku

1. P1: Read X
2. P2: Read X
3. P1: Write X, 1
4. P2: Read X
5. P3: Read X

a) Ako je usvojena strategija upisa u keš memoriju *write-back,* koji se logički problem može prepoznati pri obraćanju različitih procesora ovom podatku?

b) Da li se ovaj problem može rešiti usvajanjem strategije upisa *write-through*?

c) Predložite i obrazložite način za rešenje ovog problema.

**Rešenje:**

Kod *write-back* strategije pri pogotku upis se obavlja samo u keš memoriji, a glavna memorija se ne ažurira. Prema tome, vrednosti promenljive X u glavnoj memoriji i njenih kopija u privatnim keš memorijama posle izvršenja svake od instrukcija su date u Tabeli

| | C1 | C2 | C3 | M |
|---|---|---|---|---|
| | — | — | — | 0 |
| 1. | 0 | — | — | 0 |

| | | | | |
|---|---|---|---|---|
| 2. | 0 | 0 | — | 0 |
| 3. | 1 | 0 | — | 0 |
| 4. | 1 | 0 | — | 0 |
| 5. | 1 | 0 | 0 | 0 |

Kao posledica promašaja pri čitanju u prvoj i drugoj instrukciji vrednost 0 ce biti dohvaćena iz glavne memorije i upisana u privatne keš memorije C1 i C2. Trećom instukcijom procesor P1 menja vrednost promenljive X, pa, kako je došlo do pogotka i primenjen je write-back, upisuje se samo kopija u C1 dok memorija ostaje neažurna, kao i kopija podatka X u C2. Zbog toga naredna instrukcija P2 kojom se opet čita podatak X rezultira pogotkom u C2 i pristupa neažurnoj vrednosti. Peta instrukcija kojom P3 čita podatak X, pošto izaziva promašaj u C3, prihvata neažurnu vrednost iz memorije i takođe prouzrokuje nekorektno izvršavanje programa. Ovakav logički problem se naziva problem koherencije ili konzistencije keš memorija. Kao sto se iz gornjeg primera može videti, najčešći uzrok ovog problema je deljenje podataka sa mogućnošću upisa. Ovaj problem takođe može nastati i zbog migracije procesa, kao i zbog ulazno/izlaznih aktivnosti.

b) U slučaju da je primenjena strategija upisa write-through, kod treće instrukcije upis od strane procesora P1 bi promenio vrednost podatka i u keš meoriji C1 i u glavnoj memoriji. Tako bi peta instrukcija (čitanje od strane P3) sada vratila ažurnu vrednost. Međutim, četvrta instrukcija (čitanje od strane P2) bi opet vratila pogrešnu vrednost jer ona rezultira pogotkom i vraća vrednost iz C2 koja je ostala neažurna.

c) Ovaj problem bi mogao da se reši na dva načina:

1. Pre upisa u svoju kopiju deljenog podatka u privatnoj keš memoriji, procesor P1 treba da preko magistrale pošalje signal kojim ce poništiti sve ostale kopije istog podatka u keš memorijama (u ovom slučaju u C2). Posle toga, kopija podatka u C1 postaje jedina važeća kopija u sistemu. Zato naredno čitanje procesora P2 (četvrta instrukcija) ne nalazi podatak u C2 (promašaj) i izbacuje zahtev za čitanjem podatka na magistralu. Tada kontroler keš memorije C1 prepoznaje da se radi o podatku čiju jedinu važeću kopiju poseduje, pa izdaje signal kojim sprečava memoriju da odgovori, već sam izbacuje ažurnu vrednost trazenog podatka na magistralu, odakle ga prihvata keš meorija C2. Ovom istom prilikom može i da se ažurira glavna memorija, tako da i peta instrukcija kasnije dobije pravu vrednost.

2. Drugi način se sastoji u tome sto pri upisu u svoju kopiju deljenog podatka, procesor P1 izbacuje upisivanu vrednost (1) na magistralu tako da sve kopije istog podatka i u keš memorijama i u glavnoj memoriji mogu da se ažuriraju istovremeno.

Očigledno da kontroleri keš memorija treba da implementiraju dodatne funkcije kojim se rešava ovaj problem. Skup precizno definisanih akcija pri upisu i čitanju deljenih podataka se obično naziva protokolom za održanje koherencije keš memorije. Kako se prvo rešenje zasniva na poništenju (invalidaciji) zastarelih kopija, takvi protokoli se nazivaju invalidacionim protokolima (write-invalidate). Drugi pristup se zasniva na distribuiranom upisu, pa se takva rešenja nazivaju ažurirajucim protokolima (write-update). Protokoli za koherenciju mogu biti vrlo različiti, a pitanje izbora je vrlo složeno, veoma utiče na efikasnost sistema i mnogo zavisi kako od strukture samog sistema, tako i od karakteristika programa koji se izvršavaju.

## Zadatak 1.1.17

Postoje dva pristupa adresiranju keš memorije u računarskom sistemu: preko fizičkih i virtuelnih adresa.

a) Objasniti i skicirati postupak obraćanja u jednom i u drugom slučaju, na sledećem primeru. Virtuelna adresa ima 32 bita, fizička adresa 24 bita, veličina stranice je 1Kbajta, keš memorija je set-acocijativno mapirana sa dva bloka po setu, veličina keš memorije je 8 Kbajta, a veličina bloka 16 bajtova.

b) Uporediti ova dva pristupa u pogledu brzine i složenosti.

c) Koji uslov treba da je zadovoljen da bi brzine ova dva pristupa bile jednake?

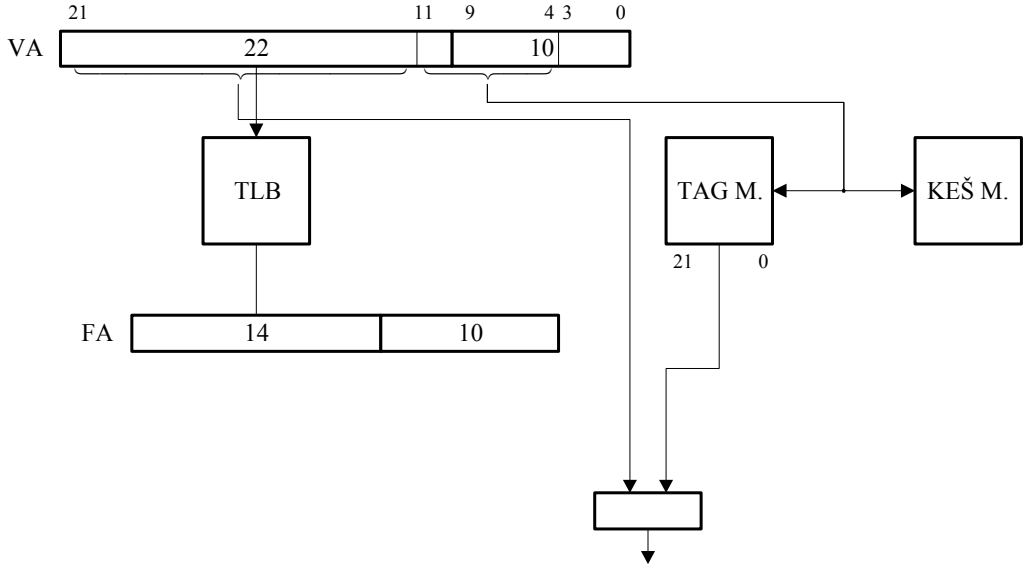
d) Kako bi se proces obraćanja kod fizički adresiranih keš memorija mogao ubrzati?

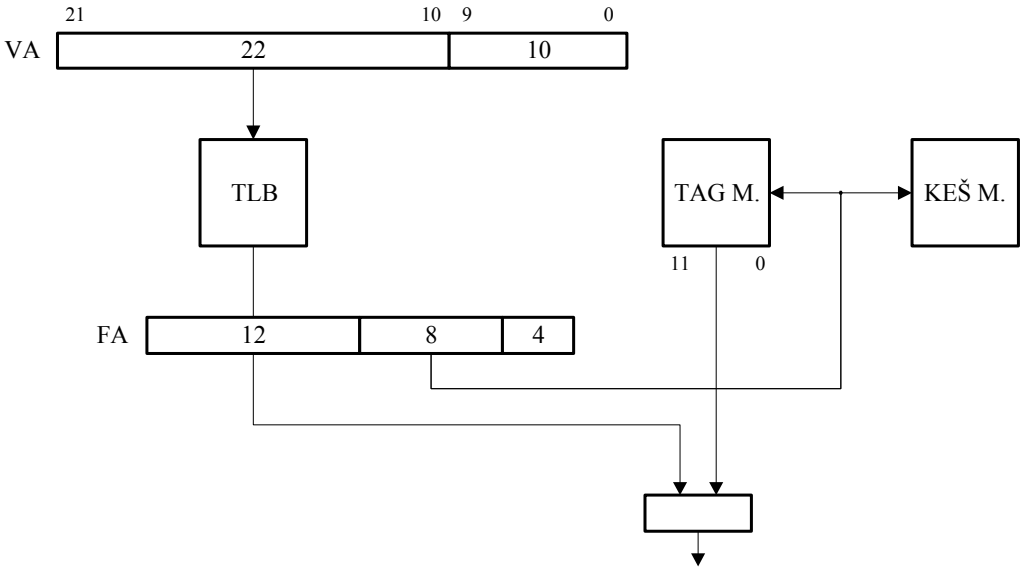
**Rešenje:**

a) Postupak pristupa keš memoriji u slučaju obraćanja sa fizičkim adresama dat je na slici 5.??a. Uzimajući u obzir zadate parametre, struktura fizičke adrese je:

4 bita za blok
8 bita za set
12 bita za tag.

Pošto su bitovi kojima se indeksira keš memorija (broj seta) u opštem slučaju različiti u fizičkoj i virtuelnoj adresi (samo donjih 10 bitova su u ovom primeru isti), mora se prvo izvršiti prevođenje virtuelne adrese zadate od strane procesora. Tek onda se uzimaju bitovi 0-11 fizičke adrese i pristupa se keš memoriji.






Slika 5.??b opisuje postupak obraćanja keš memoriji preko virtuelnih adresa. U tom slučaju obraćanje keš memoriji može da počne odmah sa donjih 12 bitova virtuelne adrese. Adresni deo keš memorije (tagovi) u ovom slučaju ima širinu 20 bitova. Fizička adresa je ovde potrebna jedino ako dođe do promašaja u keš memoriji pa je neophodno obraćanje glavnoj memoriji.

b) Pošto kod fizički adresiranih memorija pristupu keš memoriji u opstem slučaju mora da prethodi prevođenje adrese, vreme obraćanja im je duže od virtuelno adresiranih keš memorija kod kojih pristup može da počne odmah i da se odvija konkurentno sa prevođenjem adrese. U ovom drugom slučaju fizička adresa je potrebna samo ako dođe do promašaja i potrebe obraćanja glavnoj memoriji.

Sto se tiče složenosti, kako su virtuelne adrese obično veće od fizičkih, kod virtuelno adresiranih keš meorija tagovi sadrže više bitova, pa je potrebno više memorije za njihovo smeštanje. Dodatni problem se javlja kada različiti procesi koriste keš memoriju. U tom slučaju se može desiti da iste virtuelne adrese kod dva različita procesa odgovaraju različitim fizičkim adesama, kao i obratno, da ista fizička adresa može da odgovara različitim virtuelnim adresama u različitim procesima. Zato pri promeni konteksta treba poništiti sadržaj keš memorije (cache flush) ili uz virtualne adrese voditi identifikator procesa.

c) Proces obraćanja keš memoriji može započeti odmah, ne čekajuci prevođenje, samo ako su za pristup dovoljni bitovi adrese koji se pri prevođenju ne menjaju (bitovi za adresiranje u okviru stranice). Prema tome, može se izvesti potreban uslov koji kaže da veličina stranice mora da bude veća ili jednaka od količnika veličine keš memorije i stepena asocijativnosti. Prema tome, u gornjem primeru bi trebalo smanjiti keš memoriju na 2Kbajta ili povećati stepen asocijativnosti na osam.

d) U slučaju da gornji uslov nije zadovoljen, ubrzanje pristupa fizički adresiranoj keš memoriji se može postici jednostavnom kontrolom da li se tekuće obraćanje nalazi u okviru iste stranice kao i prethodno obraćanje. Ovo je je vrlo često slučaj zbog karakteristične prostorne i vremenske lokalnosti obraćanja. U tom slučaju preslikavanje je isto kao i u prethodnom slučaju, pa se može odmah koristiti zapamćena vrednost prethodnog fizičkog broja stranice.

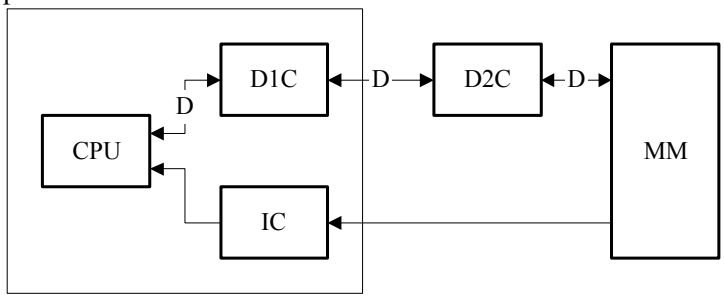
## Zadatak 1.1.18

Efikasnost korišćenja keš memorije u računarskom sistemu je prvenstveno određena njenom brzinom, kapacitetom i propusnim opsegom (?). Imajuci u vidu ove pokazatelje, predložite i obrazložite jednu organizaciju sistema keš memorije koja teži da optimizuje performansu zadovoljavajući gornje kriterijume. Treba posebno diskutovati pojedine projektne odluke oko mehanizma mapiranja, dužine bloka, strategije upisa, itd.

**Rešenje:**

U najvećem broju sistema ranije je korišćena jedna keš memorija i za instrukcije i za podatke kao zajednički resurs. Ovo je bilo povoljno sa stanovišta faktora uspešnosti i efikasnog načina korišćenja zajedničkog prostora. Međutim, u poslednje vreme zahtevi za povećanjem propusnog opsega keš memorije su porasli pogotovo u organizacijama sa protočnom obradom. Kod njih se cesto dešava da pristup keš memoriji postane izvor strukturnog hazarda, jer se istovremeno zahteva prihvatanje naredne instrukcije i pristup podatku od neke prethodne instrukcije. Zato se sve češće uvode posebne keš memorije za instrukcije i podatke, čime se otklanja ovaj hazard i udvostručava propusni opseg keš memorije.

Ovo razdvajanje ima i druge prednosti. Razdvojene keš memorije se smeštaju na čipu upravo fizički blisko onim funkcionalnim jedinicama koji ih koriste što omogućava brzi pristup. One se takođe mogu posebno optimizovati po veličini same keš memorije, veličini bloka, načinu mapiranja, načinu prihvatanja podataka, itd., jer su i karakteristike obraćanja instrukcijama i podacima različite.




Keš memorija za instrukcije može da bude prostija jer se obično ne dozvoljava upisivanje u područje koda. Zbog dosta izrazene prostorne lokalnosti ona može da ima veću dužinu bloka i da koristi dohvatanje unapred (*prefetching*).

Što se tiče keš memorije za podatke pred nju se takođe postavljaju primarni zahtevi za većom bržinom i faktorom uspešnosti. Ovi zahtevi su kontradiktorni jer je faktor uspešnosti može osetno povećati porastom kapaciteta keš memorije, ali tada se brzina pristupa smanjuje. Pored toga, prostor na čipu je veoma ograničen. Zato se pribegava rešenju da se uvedu dva (ili više) nivoa keš memorije. Pritom, prvi nivo je na čipu, ima manji kapacitet, prostiju organizaciju (obično direktno mapiran i *write-through*) i zato ima veliku brzinu pristupa. Problem povećanja faktora uspešnosti se rešava drugim nivoom keš memorije koji se nalazi izvan čipa (ali na istoj ploči) i ima mnogo veći kapacitet. Pristup drugom nivou je naravno sporiji ali još uvek dosta brži od pristupa glavnoj memoriji. Projektne odluke za drugi nivo keš memorije mogu biti različite (npr. veća dužina bloka, *write-back* strategija upisa, itd.). Vrlo često se poštuje princip inkluzije koji nameće da se čitav sadzaj keš memorije nižeg nivoa nalazi i u keš memoriji višeg nivoa, što je od koristi pogotovo u multiprocesoskim sistemima.

Imajući u vidu gore navedenu diskusiju, tražena organizacija bi izgledala kao na slici 5.???

Zadatak 1.1.19 (Keš memorije, AOR Sep 96):

Adrese su širine 16 bita, a adresiranje je na nivou 16-bitnih reči. Između procesora i magistrale računara vezana je keš memorija organizovana set-asocijativno na nivou bloka od 2 KB, dva ulaza po setu, kapacitetom memorije podataka 16 KB, FIFO algoritmom zamene i *store-through* tehnikom ažuriranja operativne memorije. Keš je inicijalno prazan. Procesor generiše sledeću sekvencu adresa:

07FAh, B510h, 060Fh, 1875h, 0511h, A500h.

a) Dati sadržaj svih ulaza set-asocijativne memorije posle date sekvence (pisati heksadecimalno).

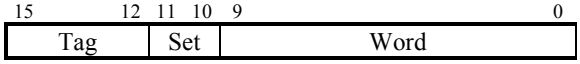
b) Koliki je broj promašaja u kešu *Nmiss* za prethodni slučaj?

c) Dati sadržaj svih ulaza ako je keš sa direktnim preslikavanjem na nivou bloka od 2 KB i kapacitetom memorije podataka od 8 KB (pisati heksadecimalno).

d) Koliki je broj promašaja u kešu *Nmiss* za prethodni slučaj?

.1.1 Rešenje:

a) i b) U ovom slučaju, blok je veličine 1 K = $2^{10}$ reči, a keš sadrži 4 seta. Struktura adrese izgleda ovako:




Prema tome, parovi (Tag, Set) koji se pojavljuju u datoj sekvenci su redom:

(0,1), (B,1), (0,1), (1,2), (0,1), (A,1)

Sadržaj set-asocijativne memorije je na kraju sledeći:

| | | | | |
|---|---|---|---|---|
| Ulaz 1 | | B | | |
| Ulaz 0 | | A | 1 | |
| Set | 0 | 1 | 2 | 3 |

Broj promašaja je 4.

c) i d) U ovom slučaju blok je iste veličine, keš je dva puta manjeg kapaciteta, ali ima samo jedan ulaz po setu (direktno preslikavanje je granični slučaj set-asocijativnog, kada je broj ulaza po setu jednak 1). Zato je struktura adrese ista, kao i sekvenca polja koja se pojavljuju. Sadržaj je na kraju:

| | | | | |
|---|---|---|---|---|
| | | A | 1 | |
| Set | 0 | 1 | 2 | 3 |

Broj promašaja je sada 5.

## *1.2 Virtuelna memorija*

### Zadatak 1.2.1

Posmatra se računar sa straničnom organizacijom virtuelne memorije. Virtuelni adresni prostor korisnika je 16 Mbajta i podeljen je na stranice veličine 1 Kbajt. Realni adresni prostor je 1 Mbajt i podeljen je na blokove veličine 1 Kbajt. Virtuelnom i realnom adresom se adresiraju reči dužine 1 bajt.

Uraditi sledeće:

a) Označiti dužine u bitovima svih delova virtuelne i realne adrese.

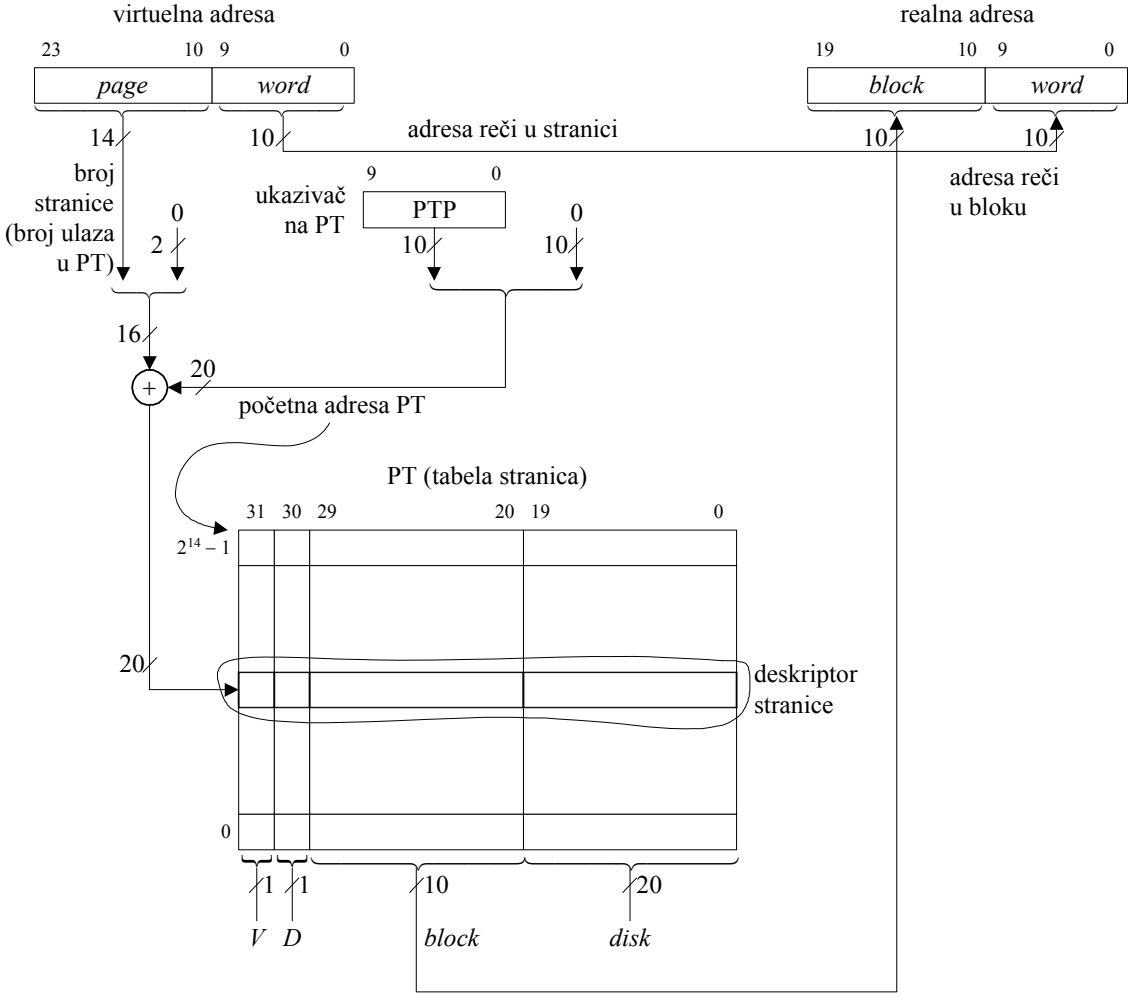
b) Nacrtati tabelu stranica, objasniti kako se pristupa tabelama stranica različitih korisnika i dati funkciju pojedinih polja deskriptora stranice.

c) Objasniti kako se vrši preslikavanje virtuelne u realnu adresu za slučaj da je stranica u memoriji i navesti šta se od toga radi hardverski a šta softverski.

d) Objasniti šta se radi za slučaj da stranica nije u memoriji i navesti šta se od toga radi hardverski a šta softverski.

**Rešenje:**

a) Kod stranične organizacije virtuelne memorije virtuelna adresa ima dva polja: broj stranice (*page*) i adresa reči u stranici (*word*). Kako je veličina virtuelnog adresnog prostora 16 Mbajta ($2^{24}$ bajta), a stranice 1 Kbajt ($2^{10}$ bajta), to su veličine polja *page* i *word* 14 i 10 bita, respektivno. Kod stranične organizacije virtuelne memorije realna adresa ima dva polja: broj bloka (*block*) i adresa reči u bloku (*word*). Kako je veličina realnog adresnog prostora 1 Mbajta ($2^{20}$ bajta), a bloka 1 Kbajt ($2^{10}$ bajta ), to su veličine polja *block* i *word* 10 i 10 bita, respektivno. Struktura virtuelne i realne adrese i dužine u bitovima delova virtuelne i realne adrese su prikazani na slici **Error! Reference source not found.**.



**Slika 1.2.1.** Virtuelna adresa, realna adresa i tabela stranica za straničnu organizaciju virtuelne memorije

b) Tabela stranica (PT) jednog korisnika je data na slici **Error! Reference source not found.**. Tabela stranica ima poseban ulaz za svaku stranicu korisnika. U njima se nalaze informacije neophodne za preslikavanje stranica virtuelnog adresnog prostora korisnika u blokove fizičke memorije. Te informacije se nazivaju deskriptori stranica. S obzirom da u virtuelnom adresnom prostoru korisnika ima $2^{14}$ stranica, to i tabela stranica ima $2^{14}$ ulaza. Početna adresa tabele stranica sadržana je u posebnom registru procesora koji se naziva ukazivač na PT (PTP). Na osnovu sadržaja ukazivača na PT i broja stranice, jedinica za preslikavanje virtuelnih adresa u realne adrese dolazi do deskriptora stranice, a time i do informacija neophodnih za preslikavanje.

U operativnoj memoriji postoji posebna tabela stranica svakog korisnika. Početne adrese tabela stranica svih korisnika čuva operativni sistem. Prilikom prebacivanja procesora sa korisnika na korisnika operativni sistem, najpre, čuva kontekst korisnika kome se oduzima procesor, a potom, restaurira kontekst korisnika kome se dodeljuje procesor. Tom prilikom se u ukazivač na PT upisuje početna adresa tabele stranica korisnika kome se dodeljuje procesor. Time se obezbeđuje da jedinica za preslikavanje pristupa tabeli stranica korisnika kome je dodeljen procesor.

Polja deskriptora stranice su:

- *V* (1bit)—stranica u memoriji,
- *D* (1bit)—stranica modifikovana,
- *block* (10 bita)—broj bloka i
- *disk* (20 bita)—adresa na disku.

Polje *V* označava da li je data stranica u operativnoj memoriji. Postavlja ga operativni sistem prilikom dovlačenja date stranice sa diska u operativnu memoriju. Koristi ga jedinica za preslikavanje i to ako

- je postavljen, da formira realnu adresu i
- nije postavljen, da generiše prekid.

Polje *D* označava da li je data stranica modifikovana. Postavlja ga jedinica za preslikavanje ako je bilo operacija upisa u datu stranicu. Koristi ga operativni sistem onda kada odabere datu stranicu za zamenu i to ako

- je postavljen, da datu stranicu vrati na disk
- nije postavljen, da datu stranicu ne vraća na disk.

Polje *block* označava broj bloka operativne memorije u kome se nalazi data stranica. Vrednost u ovom polju ima smisla jedino ako je polje *V* postavljeno. Polje *block* postavlja operativni sistem prilikom dovlačenja date stranice sa diska u operativnu memoriju. Koristi ga jedinica za preslikavanje, i to ako je polje V postavljeno, da formira realnu adresu.

Polje *disk* označava adresu date stranice na disku. Polje *disk* postavlja operativni sistem prilikom formiranja tabele stranice datog korisnika. Koristi ga samo operativni sistem i to radi lociranja date stranice na disku prilikom

- dovlačenja stranice sa diska u odabrani blok operativne memorije
- vraćanja modifikovane stranice iz bloka operativne memorije odabranog za zamenu na disk.

Uzeto je proizvoljno, jer to nije predmet razmatranja u ovom zadatku, pa stoga nije ni definisano, da je veličina adrese stranice na disku 20 bitova.

c) Preslikavanje virtuelne u realnu adresu za slučaj da je stranica u memoriji realizuje se kompletno hardverski i to pomoću jedinice za preslikavanje. Ovo preslikavanje se realizuje u sledećim koracima:

1. Polje *page* virtuelne adrese predstavlja broj ulaza u tabelu stranica u kome se nalazi deskriptor date stranice. S obzirom na to da je adresiranje fizičke memorije bajtovsko i da deskriptor stranice zauzima četiri bajta potrebno je broj ulaza u tabelu stranica pretvoriti u pomeraj u odnosu na početak tabele stranica. To se realizuje pomeranjem ulevo za dva mesta sadržaja polja *page* vitruelne adrese.

2. Sadržaj registra ukazivač na PT predstavlja početnu adresu tabele stranica. Pomeraj u odnosu na početak tabele stranica formiran u prethodnom koraku se sabira sa sadržajem registra ukazivač na PT i time dobija adresa deskriptora date stranice. Počev od formirane adrese treba očitati dva bajta, jer se u njima nalaze polja *V* i *block* koja koristi jedinica za preslikavanje.

3. Polje *V* deskriptora ukazuje na to da li je data stranica u memoriji. Stoga se, najpre, sa adrese formirane u prethodnom koraku čita prvi bajt u kome se nalazi polje *V* deskriptora i vrši provera da li je ovo polje postavljeno. Za slučaj kada je stranica u memoriji utvrdiće se da je polje V postavljeno, pa se produžava sa sledećim koracima.

4. Polje *block* deskriptora sadrži broj bloka u kome se nalazi data stranica. Ovo polje sa nalazi delom u prvom a delom u drugom bajtu deskriptora. Stoga se sa prve sledeće adrese čita i drugi bajt deskriptora. Konkatinacijom polja *block* iz deskriptora stranice i polja *word* iz virtuelne adrese formira se realna adresa.

d) U slučaju da stranica nije u memoriji realizuju se samo prva tri od četiri koraka za slučaj kada je stranica u memoriji. U koraku tri se u ovom slučaju utvrđuje da polje *V* nije postavljeno, pa se generiše prekid. Svi koraci do generisanja signala prekida, uključujući i njegovo generisanje, realizuju se hardverski pomoću jedinice za preslikavanje. Sve ostalo, što za ovaj slučaj potom treba uraditi, radi se softverski i to radi poseban deo operativnog sistema. Ovo se realizuje u sledećim koracima:

1. Oduzima se procesor datom korisniku, njegov kontekst čuva i korisnik stavlja u red blokiranih korisnika..

2. Organizuje se dovlačenje date stranice sa diska u neki od blokova operativne memorije.

3. Dodeljuje se procesor nekom od radnosposobnih korisnika. Tada se restaurira njegov kontekst. U okviru toga se u programski brojač upisuje nova vrednost i time kontrola predaje tom novom korisniku.

Dovlačenje stranice sa diska u neki od blokova operativne memorije se realizuje u sledećim koracima:

—Traži se blok u koji će se smestiti data stranica. Pri tome se izvršava jedan od sledeća dva koraka:

- Ukoliko ima slobodnih blokova po nekom algoritmu se odlučuje koji blok treba dodeliti datoj stranici.
- Ukoliko nema slobodnih blokova po nekom algoritmu se odlučuje koju stranicu treba izbaciti iz operativne memorije da bi se blok u kome se ona nalazi oslobodio i dodelio datoj stranici. Ukoliko je stranica odabrana za izbacivanje modifikovana mora se vratiti

na disk pre nego što se blok u kome se ona nalazi koristi da se u njega dovuče nova stranica. Adresa na disku stranice odabrane za izbacivanje dobija se iz polja *disk* deskriptora date stranice. Po završenom vraćanju date stranice na disk, u polje *V* deskriptora stranice koja je vraćena na disk upisuje se nula.

—Dovlači se data stranica sa diska u odabrani blok operativne memorije. Adresa date stranice na disku dobija se iz polja *disk* deskriptora stranice. Po završenom dovlačenju date stranice sa diska u polje *block* deskriptora stranice upisuje se broj bloka, a u polje *V* jedinica.

—Korisnik za koji je dovučena stranica prevodi se u red radnosposobnih korisnika.

U nekom trenutku dati korisnik ponovo dobija procesor. Dati korisnik ponovo generiše adresu za koju je bilo utvrđeno da je iz stranice koja nije bila u memoriji. Pošto je sada data stranica u memoriji ponavljaju se koraci iz tačke c).

### Zadatak 1.2.2

Posmatra se računar sa virtuelnom memorijom segmentne organizacije. Virtuelni adresni prostor korisnika je 16 Mbajta, i podeljen je na 256 segmenata maksimalne veličine 64 Kbajta. Realni adresni prostor je 1 Mbajt. Virtuelnom i realnom adresom se adresiraju reči dužine 1 bajt.

Uraditi sledeće:

a) Označiti dužine u bitovima svih delova virtuelne i realne adrese.

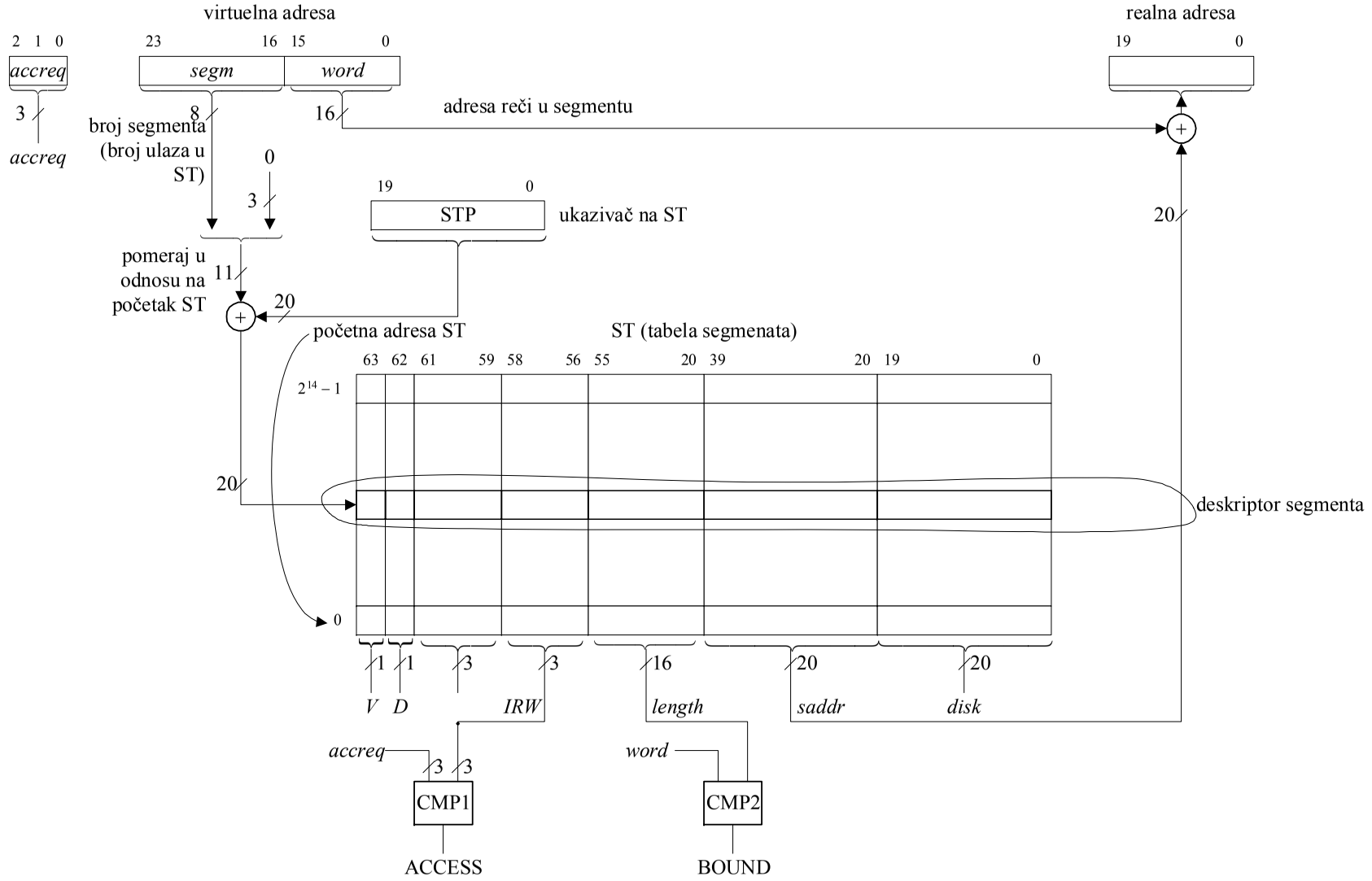
b) Nacrtati tabelu segmenata, objasniti kako se pristupa tabelama segmenata različitih korisnika i dati funkciju pojedinih polja deskriptora segmenta.

c) Objasniti kako se vrši preslikavanje virtuelne u realnu adresu za slučaj da je segment u memoriji i navesti šta se od toga radi hardverski a šta softverski. Objasniti pod kojim uslovima se dobijena realna adresa koristi za pristup memorijskoj lokaciji.

d) Objasniti šta se radi za slučaj da segment nije u memoriji i navesti šta se od toga radi hardverski a šta softverski.

**Rešenje**:

a) Kod segmentne organizacije virtuelne memorije virtuelna adresa ima dva polja: broj segmenta (*segm*) i adresa reči u segmentu (*word*). Kako je veličina virtuelnog adresnog prostora 16 Mbajta ($2^{24}$ bajta) i sastoji se od 256 ($2^8$) segmenata maksimalne veličine 64 Kbajta ($2^{16}$ bajta), to su veličine polja *segm* i *word* 8 i 16 bita, respektivno. Kako je veličina realnog adresnog prostora 1 Mbajta ($2^{20}$ bajta) to je veličina realne adrese 20 bita. Struktura virtuelne i realne adrese i dužine u bitovima delova virtuelne i realne adrese su prikazani na slici 1.2.2.


virtuelna adresa
realna adresa
2 1 0
23 16 15 0
19 0
accreq
segm
word
3
8
16
accreq
broj segmenta (broj ulaza u ST)
adresa reči u segmentu
0
3
19 0
STP
ukazivač na ST
20
pomeraj u odnosu na početak ST
11
20
početna adresa ST
ST (tabela segmenata)
63 62 61 59 58 56 55 20 39 20 19 0
$2^{14}-1$
20
deskriptor segmenta
0
1 1 3 3 16 20 20
V D IRW length saddr disk
accreq
word
3 3
CMP1
CMP2
ACCESS
BOUND

**Slika 1.2.2.** Virtuelna adresa, realna adresa i tabela segmenata za segmentnu organizaciju virtuelne memorije

b) Tabela segmenata (ST) jednog korisnika je data na slici 1.2.2. Tabela segmenata ima poseban ulaz za svaki segment korisnika. U njima se nalaze informacije neophodne za preslikavanje segmenata virtuelnog adresnog prostora korisnika u delove fizičke memorije veličine segmenata. Te informacije se nazivaju deskriptori segmenata. S obzirom da u virtuelnom adresnom prostoru korisnika ima $2^8$ segmenata, to i tabela segmenata ima $2^8$ ulaza. Početna adresa tabele segmenata sadržana je u posebnom registru procesora koji se naziva ukazivač na ST (STP). Na osnovu sadržaja ukazivača na ST i broja segmenta, jedinica za preslikavanje virtuelnih adresa u realne adrese dolazi do deskriptora segmenta, a time i do informacija neophodnih za preslikavanje.

U fizičkoj memoriji postoji posebna tabela segmenata svakog korisnika. Početne adrese tabela segmenata svih korisnika čuva operativni sistem. Prilikom prebacivanja procesora sa korisnika na korisnika operativni sistem, najpre, čuva kontekst korisnika kome se oduzima procesor, a potom, restaurira kontekst korisnika kome se dodeljuje procesor. Tom prilikom se u ukazivač na ST upisuje početna adresa tabele segmenata korisnika kome se dodeljuje procesor. Time se obezbeđuje da jedinica za preslikavanje pristupa tabeli segmenata korisnika kome je dodeljen procesor.

Polja deskriptora segmenta su:

- *V* (1 bit)—segment u memoriji,
- *D* (1 bit)—segment modifikovan,
- *IRW* (3 bita)—bitovi zaštite,
- *length* (16 bita)—veličina segmenta,
- *saddr* (20 bita)—početna adresa segmenta u fizičkoj memoriji i
- *disk* (20 bita)—adresa segmenta na disku.

Polje *V* označava da li je dati segment u operativnoj memoriji. Postavlja ga operativni sistem prilikom dovlačenja datog segmenta sa diska u operativnu memoriju. Koristi ga jedinica za preslikavanje i to ako

- je postavljen, da formira realnu adresu i
- nije postavljen, da generiše prekid.

Polje *D* označava da li je dati segment modifikovan. Postavlja ga jedinica za preslikavanje ako je bilo operacija upisa u dati segment. Koristi ga operativni sistem onda kada odabere dati segment za zamenu i to ako

- je postavljen, da dati segment vrati na disk
- nije postavljen, da dati segment ne vraća na disk.

Polje *IRW* označava dozvoljen pristup datom segmentu. Uzeto je da

- postavljen bit *I* označava da iz njega smeju da se čitaju samo instrukcije,
- postavljen bit *R* označava da iz njega smeju da se čitaju podaci i
- postavljen bit *W* označava da iz njega smeju da se upisuju podaci.

Polje *IRW* formira operativni sistem prilikom formiranja tabele segmenata datog korisnika. Koristi ga jedinica za preslikavanje da, upoređivanjem sa sadržajem registra *accreg*, utvrdi da li zahtevani pristup segmentu odgovara dozvoljenom pristupu segmentu.

Polje *length* označava veličinu segmenta. Polje *length* postavlja operativni sistem prilikom formiranja tabele segmenata datog korisnika. Koristi ga jedinica za preslikavanje da, upoređivanjem sa sadržajem polja *word* virtuelne adrese, utvrdi da li je adresa reči u segmentu unutar specificirane veličine segmenta.

Polje *saddr* označava početnu adresu dela operativne memorije u kome se nalazi dati segment. Vrednost u ovom polju ima smisla jedino ako je polje *V* postavljeno. Polje *saddr* postavlja operativni sistem prilikom dovlačenja segmenta sa diska u operativnu memoriju. Koristi ga jedinica za preslikavanje, i to ako je polje *V* postavljeno, da formira realnu adresu.

Polje *disk* označava adresu datog segmenta na disku. Polje *disk* postavlja operativni sistem prilikom formiranja tabele segmenata datog korisnika. Koristi ga samo operativni sistem i to radi lociranja datog segmenta na disku prilikom

- dovlačenja segmenta sa diska u odabrani deo operativne memorije
- vraćanja modifikovanog segmenta iz dela operativne memorije odabranog za zamenu na disk.

Uzeto je proizvoljno, jer to nije predmet razmatranja u ovom zadatku, pa stoga nije ni definisano, da je veličina adrese segmenta na disku 20 bitova.

c) Preslikavanje virtuelne u realnu adresu za slučaj da je segment u memoriji realizuje se kompletno hardverski i to pomoću jedinice za preslikavanje. Ovo preslikavanje se realizuje u sledećim koracima:

1. Polje *segm* virtuelne adrese predstavlja broj ulaza u tabelu segmenata u kome se nalazi deskriptor datog segmenta. S obzirom na to da je adresiranje fizičke memorije bajtovsko i da deskriptor segmenta zauzima osam bajta potrebno je broj ulaza u tabelu segmenata pretvoriti u pomeraj u odnosu na početak tabele segmenata. To se realizuje pomeranjem ulevo za tri mesta sadržaja polja *segm* vitruelne adrese.

2. Sadržaj registra ukazivač na ST predstavlja početnu adresu tabele segmenata. Pomeraj u odnosu na početak tabele segmenata formiran u prethodnom koraku se sabira sa sadržajem registra ukazivač na ST i time dobija adresa deskriptora datog segmenta. Počev od formirane adrese treba očitati šest bajtova, jer se u njima nalaze polja *V*, *IRW*, *length* i *saddr* koja koristi jedinica za preslikavanje.

3. Polje *V* deskriptora ukazuje na to da li je dati segment u memoriji. Stoga se, najpre, sa adrese formirane u prethodnom koraku čita prvi bajt u kome se nalazi polje *V* deskriptora i vrši provera da li je ovo polje postavljeno. Za slučaj kada je segment u memoriji utvrđuje se da je polje *V* postavljeno, pa se produžava sa sledećim koracima.

4. Polja *IRW*, *length* i *saddr* deskriptora sadrže bitove zaštite, veličinu segmenta i početnu adresu segmenta u fizičkoj memoriji, respektivno. Polje *IRW* se nalazi u prvom, polje *length* u trećem, a polje *saddr* u četvrtom, petom, i delimično šestom bajtu deskriptora. Stoga se počev od prve sledeće adrese čita još pet bajtova deskriptora. Sada se istovremeno:

- sabiranjem polja *saddr* iz deskriptora segmenta i polja *word* iz virtuelne adrese formira se realna adresa,
- upoređivanjem polja *IRW* iz deskriptora segmenta i sadržaja registra *accreg* utvrđuje da li se zahteva pristup koji je dozvoljen za dati segment i
- upoređivanjem polja *length* iz deskriptora segmenta i polja *word* iz virtuelne adrese utvrđuje da li je adresa unutar segmenta unutar specificirane veličine segmenta.

Ukoliko su signali ACCESS i BOUND neaktivni, zahtevani pristup segmentu i virtuelna adresa su korektni, pa se formirana realna adresa koristi za pristup operativnoj memoriji. U

suprotnom nema obraćanja operativnoj memoriji već se generiše prekid. Sve ostalo, što za ovaj slučaj potom treba uraditi, radi operativni sistem i to u sledećim koracima:

1. Oduzima procesor datom korisniku, njegov kontekst čuva i terminira korisnika.

2. Šalje poruku na konzolu da je dati korisnik terminiran.

3. Dodeljuje procesor nekom od radnosposobnih korisnika. Tada se restaurira njegov kontekst. U okviru toga se u programski brojač upisuje nova vrednost i time kontrola predaje tom novom korisniku.

d) U slučaju da segment nije u memoriji realizuju se samo prva tri od četiri koraka za slučaj kada je segment u memoriji. U koraku 3. se u ovom slučaju utvrđuje da polje *V* nije postavljeno, pa se generiše prekid. Svi koraci do generisanja signala prekida, uključujući i njegovo generisanje, realizuju se hardverski pomoću jedinice za preslikavanje. Sve ostalo, što za ovaj slučaj potom treba uraditi, radi operativni sistem i to u sledećim koracima:

1. Oduzima se procesor datom korisniku, njegov kontekst čuva i korisnik stavlja u red blokiranih korisnika.

2. Organizuje se dovlačenje datog segmenta sa diska u neki od delova operativne memorije veličine segmenta.

3. Dodeljuje se procesor nekom od radnosposobnih korisnika. Tada se restaurira njegov kontekst. U okviru toga se u programski brojač upisuje nova vrednost i time kontrola predaje tom novom korisniku.

Dovlačenje segmenta sa diska u neki od delova operativne memorije se realizuje u sledećim koracima:

—Traži se deo operativne memorije u koji će se smestiti dati segment. Pri tome se izvršava jedan od sledeća dva koraka:

- Ukoliko ima slobodnih delova operativne memorije po nekom algoritmu se odlučuje koji deo treba dodeliti datom segmentu.
- Ukoliko nema slobodnih delova operativne memorije po nekom algoritmu se odlučuje koji segment treba izbaciti iz operativne memorije da bi se deo operativne memorije u kome se on nalazi oslobodio i dodelio datom segmentu. Ukoliko je segment odabran za izbacivanje modifikovan mora se vratiti na disk pre nego što se deo operativne memorije u kome se on nalazi koristi da se u njega dovuče novi segment. Adresa na disku segmenta odabranog za izbacivanje dobija se iz polja *disk* deskriptora segmenta. Po završenom vraćanju datog segmenta na disk, u polje *V* deskriptora segmenta koji je vraćen na disk upisuje se nula.

—Dovlači se dati segment sa diska u odabrani deo operativne memorije. Adresa datog segmenta na disku dobija se iz polja *disk* deskriptora segmenta. Po završenom dovlačenju datog segmenta sa diska u polje *saddr* deskriptora segmenta upisuje se početna adresa dela segmenta u operativnoj memoriji, a u polje *V* jedinica.

—Korisnik za koga je dovučen segment prevodi se u red radnosposobnih korisnika.

—U nekom trenutku dati korisnik ponovo dobija procesor. Dati korisnik ponovo generiše adresu za koju je bilo utvrđeno da je iz segmenta koji nije bio u operativnoj memoriji. Pošto je sada dati segment u operativnoj memoriji ponoviće se koraci iz tačke c).

Zadatak 1.2.3

Posmatra se računar sa virtuelnom memorijom segmentno stranične organizacije. Virtuelni adresni prostor korisnika je 16 Mbajta i podeljen je na 256 segmenata maksimalne veličine 64 Kbajta. Svaki segment je podeljen na stranice veličine 1 Kbajt. Realni adresni prostor je 1 Mbajt i podeljen je na blokove veličine 1 Kbajt. Virtuelnom i realnom adresom se adresiraju reči dužine 1 bajt.

Uraditi sledeće:

a) Označiti dužine u bitovima svih delova virtuelne i realne adrese.

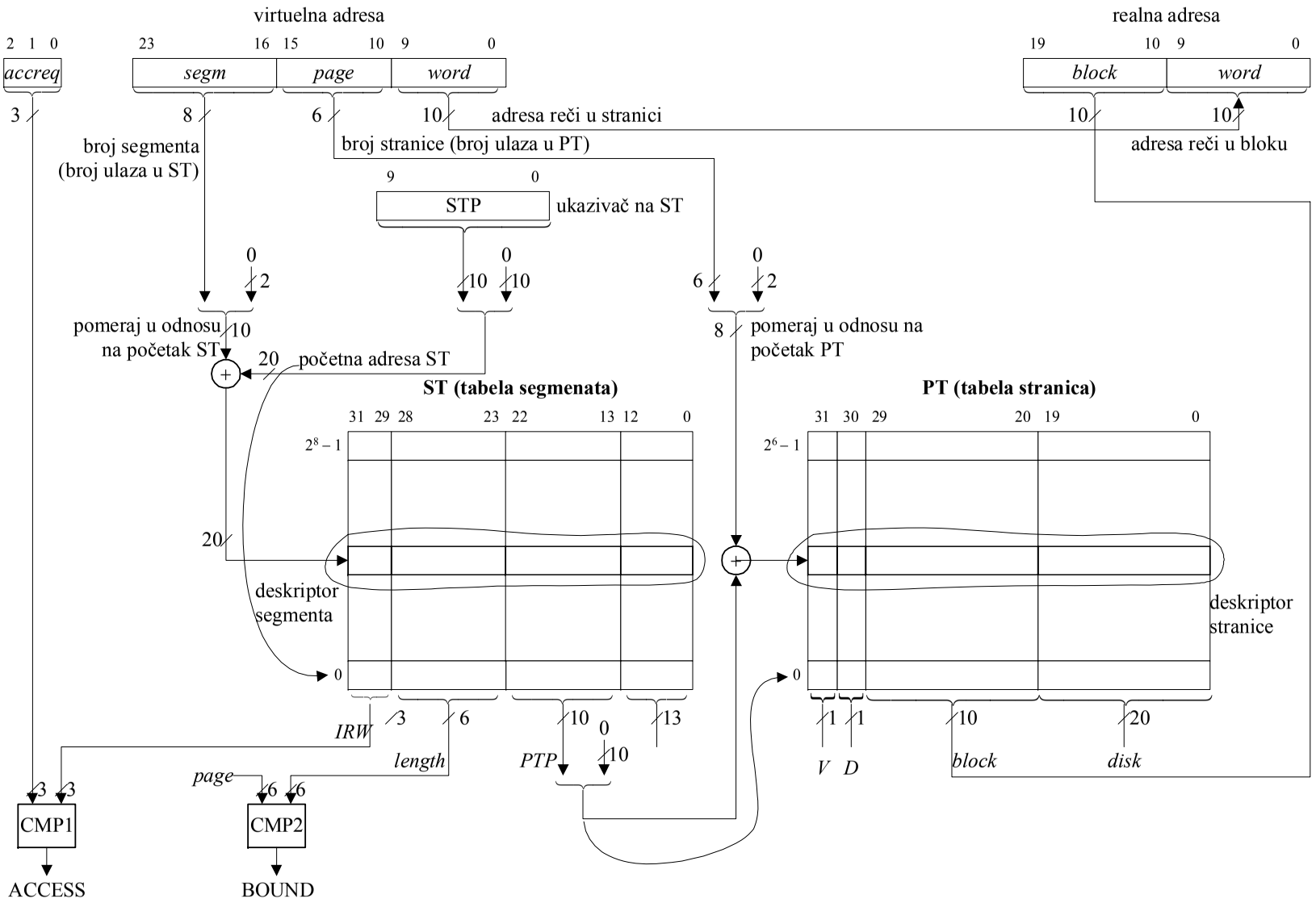
b) Nacrtati najpre tabelu segmenata, objasniti kako se pristupa tabelama segmenata različitih korisnika i dati funkciju pojedinih polja deskriptora segmenta. Nacrtati zatim jednu od tabela stranica, objasniti kako se pristupa tabelama stranica i dati funkciju pojedinih polja deskriptora stranice.

c) Objasniti kako se vrši preslikavanje virtuelne u realnu adresu za slučaj da je stranica u memoriji i navesti šta se od toga radi hardverski a šta softverski.

d) Objasniti šta se radi za slučaj da stranica nije u memoriji i navesti šta se od toga radi hardverski a šta softverski.

**Rešenje**:

a) Kod stranične organizacije virtuelne memorije virtuelna adresa ima tri polja: broj segmenta (*segm*), broj stranice (*page*) i adresa reči u stranici (*word*). Kako je veličina virtuelnog adresnog prostora 16 Mbajta ($2^{24}$ bajta) i sastoji se iz 256 segmenata ($2^8$ segmenata) maksimalne veličine 64 Kbajta, a oni su dalje podeljeni na 64 stranice ($2^6$ stranica) veličine 1 Kbajta ($2^{10}$ bajta), to su veličine polja *segm*, *page* i *word* 8, 6 i 10 bita, respektivno. Kod segmentno-stranične organizacije virtuelne memorije realna adresa ima dva polja: broj bloka (*block*) i adresa reči u bloku (*word*). Kako je veličina realnog adresnog prostora 1 Mbajta ($2^{20}$ bajta), a bloka 1 Kbajt ($2^{10}$ bajta ), to su veličine polja *block* i *word* 10 i 10 bita, respektivno. Struktura virtuelne i realne adrese i dužine u bitovima delova virtuelne i realne adrese su prikazani na slici **Error! Reference source not found.**.


virtuelna adresa
realna adresa
accreq
segm
page
word
block
word
adresa reči u stranici
adresa reči u bloku
broj segmenta
(broj ulaza u ST)
broj stranice (broj ulaza u PT)
STP
ukazivač na ST
pomeraj u odnosu
na početak ST
početna adresa ST
pomeraj u odnosu na
početak PT
ST (tabela segmenata)
PT (tabela stranica)
deskriptor
segmenta
deskriptor
stranice
IRW
length
PTP
V
D
block
disk
page
CMP1
CMP2
ACCESS
BOUND

**Slika 1.2.3.** Virtuelna adresa, realna adresa i tabela stranica za segmentno-straničnu organizaciju virtuelne memorije

b) Kod segmentno-stranične organizacije virtuelne memorije svaki korisnik ima jednu tabelu segmenata i onoliko tabela stranica koliko ima segmenata u virtuelnom adresnom prostoru korisnika. Za dati procesor postoji jedna tabela segmenata i maksimalno 256 tabela stranica.

Tabela segmenata (ST) jednog korisnika je data na slici **Error! Reference source not found.**. Tabela segmenata ima poseban ulaz za svaki segment korisnika. U njima se nalaze informacije neophodne za preslikavanje segmenata virtuelnog adresnog prostora korisnika u blokove fizičke memorije. Te informacije se nazivaju deskriptori segmenata. S obzirom da u virtuelnom adresnom prostoru korisnika ima $2^8$ segmenata, to i tabela segmenata ima $2^8$ ulaza. Početna adresa tabele segmenata sadržana je u posebnom registru procesora koji se naziva ukazivač na ST (STP). Na osnovu sadržaja ukazivača na ST i broja segmenta, jedinica za preslikavanje virtuelnih adresa u realne adrese dolazi do deskriptora segmenta, a time i do informacija neophodnih za preslikavanje.

U fizičkoj memoriji postoji posebna tabela segmenata svakog korisnika. Početne adrese tabela segmenata svih korisnika čuva operativni sistem. Prilikom prebacivanja procesora sa korisnika na korisnika operativni sistem, najpre, čuva kontekst korisnika kome se oduzima procesor, a potom, restaurira kontekst korisnika kome se dodeljuje procesor. Tom prilikom se u ukazivač na ST upisuje početna adresa tabele segmenata korisnika kome se dodeljuje procesor. Time se obezbeđuje da jedinica za preslikavanje pristupa tabeli segmenata korisnika kome je dodeljen procesor.

Polja deskriptora segmenta su:

- *IRW* (3 bita)—bitovi zaštite,
- *length* (16 bita)—veličina segmenta i
- *PTP* (20 bita)—početna adresa tabele stranica u fizičkoj memoriji.

Polje *IRW* označava dozvoljen pristup datom segmentu. Uzeto je da

- postavljen bit *I* označava da iz njega smeju da se čitaju samo instrukcije,
- postavljen bit *R* označava da iz njega smeju da se čitaju podaci i
- postavljen bit *W* označava da iz njega smeju da se upisuju podaci.

Polje *IRW* formira operativni sistem prilikom formiranja tabele segmenata datog korisnika. Koristi ga jedinica za preslikavanje da, upoređivanjem sa sadržajem registra *accreg*, utvrdi da li zahtevani pristup segmentu odgovara dozvoljenom pristupu segmentu.

Polje *length* označava veličinu segmenta. Polje *length* postavlja operativni sistem prilikom formiranja tabele segmenata datog korisnika. Koristi ga jedinica za preslikavanje da, upoređivanjem sa sadržajem polja *page* virtuelne adrese, utvrdi da li je broj stranice u segmentu unutar specificirane veličine segmenta.

Polje *PTP* označava početnu adresu operativne memorije u kome se nalazi tabela stranica datog segmenta. Polje *PTP* postavlja operativni sistem prilikom formiranja tabele segmenata i tabela stranica. Koristi ga jedinica za preslikavanje da formira realnu adresu.

Uzeto je da se *STP* i *PTP* predstavljaju u brojevima blokova, a *length* u broju stranica.

Tabela stranica (PT) jednog segmenta je data na slici **Error! Reference source not found.**. Tabela stranica ima poseban ulaz za svaku stranicu datog segmenta. U njima se nalaze informacije neophodne za preslikavanje stranica segmenta u blokove fizičke memorije.

Te informacije se nazivaju deskriptori stranica. S obzirom da segment ima najviše $2^6$ stranica, to i tabela stranica može da ima najviše $2^6$ ulaza. Početna adresa tabele stranica sadržana je u polju *PTP* deskriptora datog segmenta. Na osnovu sadržaja polja *PTP* deskriptora datog segmenta i broja stranice, jedinica za preslikavanje virtuelnih adresa u realne adrese dolazi do deskriptora stranice, a time i do informacija neophodnih za preslikavanje.

Polja deskriptora stranice su:

- *V* (1 bit)—stranica u memoriji,
- *D* (1 bit)—stranica modifikovana,
- *block* (10 bita)—broj bloka i
- *disk* (20 bita)—adresa na disku.

Polje *V* označava da li je data stranica u operativnoj memoriji. Postavlja ga operativni sistem prilikom dovlačenja date stranice sa diska u operativnu memoriju. Koristi ga jedinica za preslikavanje i to ako

- je postavljen, da formira realnu adresu i
- nije postavljen, da generiše prekid.

Polje *D* označava da li je data stranica modifikovana. Postavlja ga jedinica za preslikavanje ako je bilo operacija upisa u datu stranicu. Koristi ga operativni sistem onda kada odabere datu stranicu za zamenu i to ako

- je postavljen, da datu stranicu vrati na disk
- nije postavljen, da datu stranicu ne vraća na disk.

Polje *block* označava broj bloka operativne memorije u kome se nalazi data stranica. Vrednost u ovom polju ima smisla jedino ako je polje *V* postavljeno. Polje *block* postavlja operativni sistem prilikom dovlačenja date stranice sa diska u operativnu memoriju. Koristi ga jedinica za preslikavanje, i to ako je polje V postavljeno, da formira realnu adresu.

Polje *disk* označava adresu date stranice na disku. Polje *disk* postavlja operativni sistem prilikom formiranja tabele stranice datog korisnika. Koristi ga samo operativni sistem i to radi lociranja date stranice na disku prilikom

- dovlačenja stranice sa diska u odabrani blok operativne memorije
- vraćanja modifikovane stranice iz bloka operativne memorije odabranog za zamenu na disk.

Uzeto je proizvoljno, jer to nije predmet razmatranja u ovom zadatku, pa stoga nije ni definisano, da je veličina adrese stranice na disku 20 bitova.

c) Preslikavanje virtuelne u realnu adresu za slučaj da je stranica u memoriji realizuje se kompletno hardverski i to pomoću jedinice za preslikavanje. Ovo preslikavanje se realizuje u sledećim koracima:

1. Polje *segm* virtuelne adrese predstavlja broj ulaza u tabelu segmenata u kome se nalazi deskriptor datog segmenta. S obzirom na to da je adresiranje fizičke memorije bajtovsko i da deskriptor segmenta zauzima četiri bajta potrebno je broj ulaza u tabelu segmenata pretvoriti u pomeraj u odnosu na početak tabele segmenata. To se realizuje pomeranjem ulevo za dva mesta sadržaja polja *segm* vitruelne adrese.

2. Sadržaj registra ukazivač na ST predstavlja početnu adresu tabele segmenata. Pomeraj u odnosu na početak tabele segmenata formiran u prethodnom koraku se sabira sa sadržajem registra ukazivač na ST i time dobija adresa deskriptora datog segmenta. Počev od formirane

adrese treba očitati tri bajta, jer se u njima nalaze polja *IRW*, *length* i *PTP* koja koristi jedinica za preslikavanje.

3. Polja *IRW*, *length* i *PTP* deskriptora sadrže bitove zaštite, veličinu segmenta i početnu adresu tabele stranica u fizičkoj memoriji, respektivno. Polja *IRW* i *length* se nalaze u prvom, a polje *PTP* u drugom i delimično trećem bajtu deskriptora. Stoga se počev od formirane adrese deskriptora datog segmenta čitaju tri bajta deskriptora. Sada se istovremeno:

- upoređivanjem polja *IRW* iz deskriptora segmenta i sadržaja registra *accreg* utvrđuje da li se zahteva pristup koji je dozvoljen za dati segment i
- upoređivanjem polja *length* iz deskriptora segmenta i polja *page* iz virtuelne adrese utvrđuje da li je broj stranice u segmentu unutar specificirane veličine segmenta.

Ukoliko su signali ACCESS i BOUND neaktivni, zahtevani pristup segmentu je korektan, pa se pristupa deskriptoru stranice u tabeli stranica na način prikazan počev od koraka 4. U suprotnom nema obraćanja tabeli stranica već se generiše prekid i prelazi na aktivnosti prikazane počev od koraka 8.

4. Polje *page* virtuelne adrese predstavlja broj ulaza u tabelu stranica u kome se nalazi deskriptor date stranice. S obzirom na to da je adresiranje fizičke memorije bajtovsko i da deskriptor stranice zauzima četiri bajta potrebno je broj ulaza u tabelu stranica pretvoriti u pomeraj u odnosu na početak tabele stranica. To se realizuje pomeranjem ulevo za dva mesta sadržaja polja *page* vitruelne adrese.

5. Sadržaj polja *PTP* predstavlja početnu adresu tabele stranica izraženu kao broj bloka. S obzirom da je adresiranje fizičke memorije bajtovsko i da blok zauzima 1 Kbajt, početna adresa tabele stranica dobija se pomeranjem sadržaja polja *PTP* 10 mesta ulevo. Ovako formirana početna adresa tabele stranica sabira se sa pomerajem u odnosu na početak tabele stranica koji je formiran u prethodnom koraku i time se dobija adresa deskriptora date stranice. Počev od formirane adrese treba očitati dva bajta, jer se u njima nalaze polja *V* i *block* koja koristi jedinica za preslikavanje.

6. Polje *V* deskriptora ukazuje na to da li je data stranica u memoriji. Stoga se, najpre, sa adrese formirane u prethodnom koraku čita prvi bajt u kome se nalazi polje *V* deskriptora i vrši provera da li je ovo polje postavljeno. Za slučaj kada je stranica u memoriji utvrđuje se da je polje *V* postavljeno, pa se produžava sa sledećim koracima.

7. Polje *block* deskriptora sadrži broj bloka u kome se nalazi data stranica. Ovo polje sa nalazi delom u prvom a delom u drugom bajtu deskriptora. Stoga se sa prve sledeće adrese čita i drugi bajt deskriptora. Konkatenacijom polja *block* iz deskriptora stranice i polja *word* iz virtuelne adrese formira se realna adresa.

Ovo je kraj preslikavanja virtuelne u realnu adresu za slučaj da je stranica u memoriji.

8. U slučaju generisanog prekida zbog pojave aktivne vrednosti jednog ili oba signala **ACCESS** ili **BOUND** sve što za ovaj slučaj potom treba uraditi, radi se softverski u delu operativnog sistema i to u sledećim koracima:

- Oduzima procesor datom korisniku, njegov kontekst čuva i terminira korisnika.
- Šalje poruku na konzolu da je dati korisnik termininan.
- Dodeljuje procesor nekom od radnosposobnih korisnika. Tada se restaurira njegov kontekst. U okviru toga se u programski brojač upisuje nova vrednost i time kontrola predaje tom novom korisniku.

d) U slučaju da stranica nije u memoriji realizuju se svi koraci sem koraka 7 kao i za slučaj kada je stranica u memoriji. Moguće su dve situacije. U prvoj situaciji se ide od koraka 1 do

koraka 3 u kome se utvrđuje da je jedan od ili oba signala **ACCESS** ili **BOUND** aktivni, pa se prelazi na korak 8. U drugoj situaciji se ide od koraka 1 do koraka 6. U koraku 6 se u ovom slučaju utvrđuje da polje *V* nije postavljeno, pa se generiše prekid. Svi koraci do generisanja signala prekida, uključujući i njegovo generisanje, realizuju se hardverski pomoću jedinice za preslikavanje. Sve ostalo, što za ovaj slučaj potom treba uraditi, radi operativni sistem i to u sledećim koracima:

1. Oduzima se procesor datom korisniku, njegov kontekst čuva i korisnik stavlja u red blokiranih korisnika..

2. Organizuje se dovlačenje date stranice sa diska u neki od blokova operativne memorije.

3. Dodeljuje se procesor nekom od radnosposobnih korisnika. Tada se restaurira njegov kontekst. U okviru toga se u programski brojač upisuje nova vrednost i time kontrola predaje tom novom korisniku.

Dovlačenje stranice sa diska u neki od blokova operativne memorije se realizuje u sledećim koracima:

—Traži se blok u koji će se smestiti data stranica. Pri tome se izvršava jedan od sledeća dva koraka:

- Ukoliko ima slobodnih blokova po nekom algoritmu se odlučuje koji blok treba dodeliti datoj stranici.
- Ukoliko nema slobodnih blokova po nekom algoritmu se odlučuje koju stranicu treba izbaciti iz operativne memorije da bi se blok u kome se ona nalazi oslobodio i dodelio datoj stranici. Ukoliko je stranica odabrana za izbacivanje modifikovana mora se vratiti na disk pre nego što se blok u kome se ona nalazi koristi da se u njega dovuče nova stranica. Adresa na disku stranice odabrane za izbacivanje dobija se iz polja *disk* deskriptora stranice. Po završenom vraćanju date stranice na disk, u polje *V* deskriptora stranice koja je vraćena na disk upisuje se nula.

—Dovlači se data stranica sa diska u odabrani blok operativne memorije. Adresa date stranice na disku dobija se iz polja *disk* deskriptora stranice. Po završenom dovlačenju date stranice sa diska u polje *block* deskriptora stranice upisuje se broj bloka, a u polje *V* jedinica.

—Korisnik za koji je dovučena stranica prevodi se u red radnosposobnih korisnika.

—U nekom trenutku dati korisnik ponovo dobija procesor. Dati korisnik će ponovo generisari adresu za koju je bilo utvrđeno da je iz stranice koja nije bila u memoriji. Pošto je sada data stranica u memoriji ponoviće se koraci iz c).

### Zadatak 1.2.4

U računaru sa virtuelnom memorijom stranične organizacije definisanom u zadatku 1.2.1. postoji jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese.

a) Uzeti da računar poseduje jedinicu za ubrzavanje preslikavanja virtuelnih u realne adrese, realizovanu u tehnici asocijativnog preslikavanja u kojoj se istovremeno čuvaju relevantni delovi deskriptora 512 najčešće korišćenih stranica do 16 korisnika. Uraditi sledeće:

a1) Nacrtati, označiti dužinu u bitovima i objasniti funkciju svih relevantnih delova jedinice za ubrzavanje preslikavanja.

a2) Označiti dužine u bitovima relevantnih delova virtuelne i realne adrese i objasniti kako se:

- proverava da li se relevantni deskriptor nalazi u jedinici za ubrzavanje preslikavanja;
- dolazi do deskriptora ukoliko se utvrdi da se nalazi u jedinici za ubrzavanje preslikavanja;
- formira realna adresa;
- u jedinici za ubrzavanje preslikavanja obezbeđuje da se stranice različitih korisnika koje imaju isti broj ne preslikaju u isti, već svaka u svoj blok.

a3) U slučaju da se za generisanu virtuelnu adresu utvrdi da se relevantni deskriptor ne nalazi u jedinici za ubrzavanje preslikavanja, objasniti šta se radi u slučaju:

- kada je relevantna stranica u memoriji;
- kada relevantna stranica nije u memoriji i

navesti šta se od toga radi hardverski, a šta softverski.

b) Uzeti da računar poseduje jedinicu za ubrzavanje preslikavanja virtuelnih u realne adrese, realizovanu u tehnici direktnog preslikavanja u kojoj se istovremeno čuvaju relevantni delovi deskriptora 2048 najčešće korišćenih stranica do 16 korisnika. Uraditi isto kao pod a) ovog zadatka.

c) Uzeti da računar poseduje jedinicu za ubrzavanje preslikavanja virtuelnih u realne adrese, realizovanu u tehnici set-asocijativnog preslikavanja sa dva ulaza po setu u kojoj se istovremeno čuvaju relevantni delovi deskriptora 2048 najčešće korišćenih stranica do 16 korisnika. Uraditi isto kao pod a) ovog zadatka.

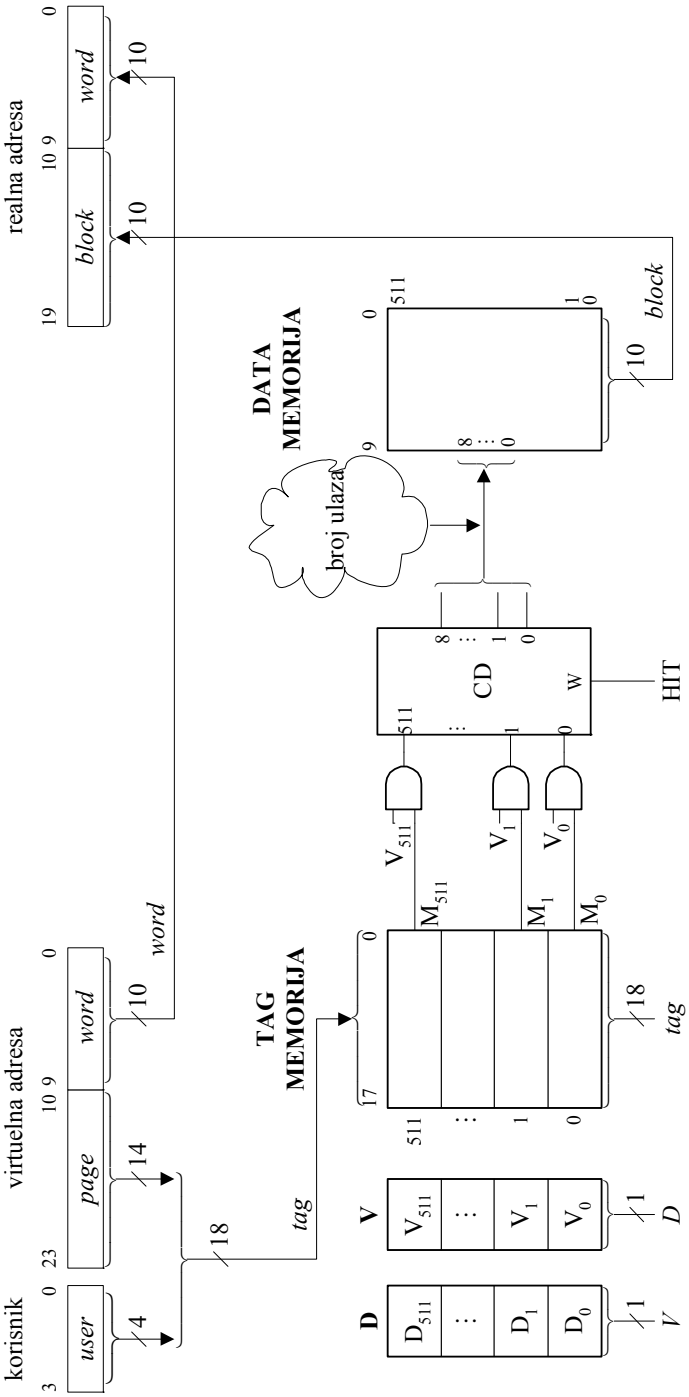
**Rešenje:**

a) S obzirom:

- da u adresnom prostoru korisnika ima 16 K stranica ($2^{14}$ stranica),
- da je broj korisnika 16 ($2^4$ korisnika) i
- da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati deskriptori stranica različitih korisnika,

zaključuje se da je, kao što je prikazano na slici 1.2.4, *tag* dužine 18 bita i da je formiran tako da:

- 4 najstarija bita predstavljaju broj korisnika (*user*) i
- 14 najmlađih bitova predstavljaju broj stranice (*page*) iz virtuelne adrese.




**Slika 1.2.4.** Jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese za virtuelnu memoriju stranične organizacije realizovana u tehnici asocijativnog preslikavanja

a1) Na osnovu zadatih karakteristika virtuelne memorije i jedinice za ubrzavanje preslikavanja dolazi se do strukture jedinice za ubrzavanje preslikavanja prikazane na istoj slici.

Jedinica za ubrzavanje preslikavanja se sastoji iz sledećih delova:

- $D_{0...511}$ (dirty bitovi)—512 flip-flopova,
- $V_{0...511}$ (valid bitovi)—512 flip-flopova,
- TAG MEMORIJA—asocijativna memorija kapaciteta 512 reči širine 18 bita,
- CD—koder 512/9 i
- DATA MEMORIJA—RAM memorija kapaciteta 512 reči širine 10 bita.

Dirty bitovi označavaju za svaki od 512 ulaza jedinice za ubrzavanje preslikavanja da li je bilo upisa u neku od lokacija date stranice.

Valid bitovi označavaju za svaki od 512 ulaza jedinice za ubrzavanje preslikavanja da li je odgovarajući ulaz TAG MEMORIJE važeći.

TAG MEMORIJA služi za čuvanje 512 TAG polja stranica čiji se delovi deskriptora nalaze u odgovarajućim ulazima DATA MEMORIJE i generisanje aktivnih vrednosti signala saglasnosti $\mathbf{M_{0...511}}$ ukoliko postoji saglasnost TAG polja generisane adrese i sadržaja odgovarajućeg ulaza TAG MEMORIJE.

CD služi za generisanje aktivne vrednosti signala saglasnosti **HIT** i broja ulaza u jedinicu za ubrzavanje preslikavanja za koji je otkrivena saglasnost.

DATA MEMORIJA služi za čuvanje 512 deskriptora stranica.

a2) TAG bitovi (18) iz generisane adrese se porede sa sadržajima svih 512 ulaza u TAG MEMORIJI. Ako se sadržaj bilo kojeg ulaza slaže sa TAG bitovima generisane adrese i odgovarajući V bit je postavljen, signal saglasnosti HIT postaje aktivan.

Bitovi (9) koji označavaju broj ulaza u jedinicu za ubrzavanje preslikavanja gde je otkrivena saglasnost dobijeni sa izlaza kodera se koriste kao adresa u DATA MEMORIJI i deskriptor se čita.

Očitano polje *block* daje 10 najstarijih bitova a polje *word* iz virtuelne adrese 10 najmlađih bitova realne adrese.

Sve ove aktivnosti se realizuju hardverski.

Da bi se stranice različitih korisnika koje imaju isti broj preslikale svaka u svoj blok, u formiranju *tag* polja učestvuje ne samo polje *page* generisane adrese, već i vrednost registra *user* procesora. Kod prebacivanja procesora sa korisnika na korianika u ovaj registar procesora se upisuje broj korisnika kome se dodeljuje procesor.

a3) Ukoliko se utvrdi da se relevantni deskriptor ne nalazi u hardveru, ide se hardverski u tabelu stranica i čita deskriptor na način prikazan u zadatku 1.2.1. Ukoliko se utvrdi da se stranica nalazi u operativnoj memoriji, polje *block* deskriptora se dovlači u hardver. Korišćenjem FIFO ili LRU algoritma zamene bira se ulaz jedinice za ubrzavanje preslikavanja u koji se smešta deskriptor. U dati ulaz DATA MEMORIJE se upisuje polje *block* deskriptora. U isti ulaz TAG MEMORIJE se upisuje polje *tag* generisane adrese. U isti ulaz *V* flip-flopova se upisuje 1, a u isti ulaz *D* flip-flopova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se stranica ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači datu stranicu sa diska na način prikazan u zadatku 1.2.1.

b) S obzirom:

- da u adresnom prostoru korisnika ima 16 K stranica ($2^{14}$ stranica),
- da je broj korisnika 16 ($2^4$ korisnika),
- da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati deskriptori stranica različitih korisnika,

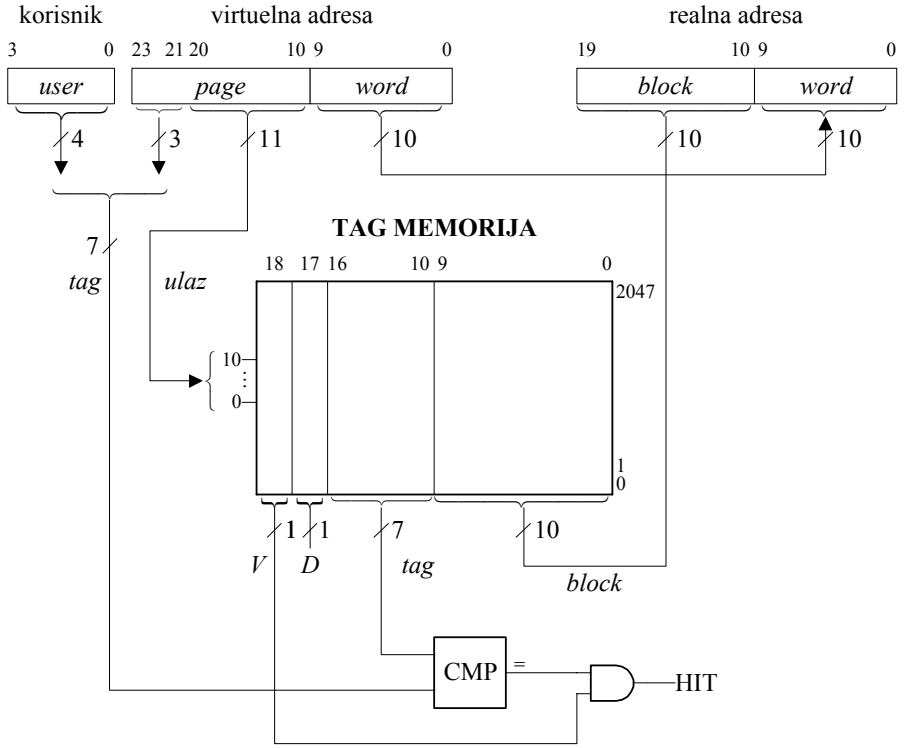
- da je jedinica za ubrzavanje preslikavanja realizovana u tehnici direktnog preslikavanja i
- da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati relevantni delovi deskriptora 2048 ($2^{11}$) najčešće korišćenih stranica do 16 korisnika ($2^4$ korisnika),

onda je po $2^{14}$ stranica svih $2^4$ korisnika grupisano u $2^7$ grupa sa po $2^{11}$ stranica u grupi. Zaključuje se, kao što je prikazano na slici 1.2.5, da su:

- polje *tag* dužine 7 bita i
- polje *ulaz* dužine 11 bita.

Polje *tag* je formirano tako da:

- 4 najstarija bita predstavljaju broj korisnika (*user*),
- 3 najmlađa bita predstavljaju 3 najstarija bita broja stranice (*page*) iz virtelne adrese.




**Slika 1.2.5.** Jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese za virtuelnu memoriju stranične organizacije realizovana u tehnici direktnog preslikavanja

b1) Na osnovu zadatih karakteristika virtuelne memorije i jedinice za ubrzavanje preslikavanja dolazi se do strukture jedinice prikazane na istoj slici.

Jedinica za ubrzavanje preslikavanja se sastoji iz sledećih delova:

- TAG MEMORIJA—RAM memorija kapaciteta 2048 reči širine 19 bita, koja se sastoji iz polja:
  - *V* (valid bitovi) dužine 1 bit,
  - *D* (dirty bitovi) dužine 1 bit,
  - *tag* dužine 7 bita,
  - *block* dužine 10 bita i
- CMP—komparator.

TAG MEMORIJA služi za čuvanje 2048 deskriptora stranica zajedno sa odgovarajućim *V*, *D* i *tag* bitovima.

*V* bitovi označavaju za svaki od 2048 ulaza jedinice za ubrzavanje preslikavanja da li je odgovarajući ulaz TAG MEMORIJE važeći.

*D* bitovi označavaju za svaki od 2048 ulaza jedinice za ubrzavanje preslikavanja da li je bilo upisa u neku od lokacija date stranice.

*tag* bitovi služe za čuvanje 2048 *tag* polja stranica čiji se deskriptori nalaze u odgovarajućim *block* poljima TAG MEMORIJE.

CMP služi za generisanje aktivne vrednosti signala saglasnosti **HIT** ukoliko:

- *tag* polje generisane adrese je isto kao sadržaj *tag* polja TAG MEMORIJE adresiran poljem *ulaz* generisane adrese i
- *V* bit adresiran poljem *ulaz* generisane adrese je postavljen.

b2) *ulaz* bitovi (11) iz generisane adrese se koriste kao adresa za TAG MEMORIJU. *tag* bitovi očitani iz TAG MEMORIJE se porede sa *tag* bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i da je *V* bit, adresiran *ulaz* bitovima generisane adrese, postavljen, signal saglasnosti **HIT** postaje aktivan.

Očitano polje *block* daje 10 najstarijih bitova a polje *word* iz virtuelne adrese 10 najmlađih bitova realne adrese.

Sve ove aktivnosti se realizuju hardverski.

Da bi se stranice različitih korisnika koje imaju isti broj preslikale svaka u svoj blok, u formiranju *tag* polja učestvuje ne samo polje *page* generisane adrese, već i vrednost registra *user* procesora. Kod prebacivanja procesora sa korisnika na korianika u ovaj registar procesora se upisuje broj korisnika kome se dodeljuje procesor.

b3) Ukoliko se utvrdi da se relevantni deskriptor ne nalazi u hardveru, ide se hardverski u tabelu stranica i čita deskriptor na način prikazan u zadatku 1.2.1. Ukoliko se utvrdi da se stranica nalazi u operativnoj memoriji, polje *block* deskriptora se dovlači u hardver. Korišćenjem FIFO ili LRU algoritma zamene bira se ulaz jedinice za ubrzavanje preslikavanja u koji se smešta deskriptor. U dati ulaz DATA MEMORIJE se upisuje polje *block* deskriptora. U isti ulaz TAG MEMORIJE se upisuje polje *tag* generisane adrese. U isti ulaz *V* flip-flopova se upisuje 1, a u isti ulaz *D* flip-flopova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se stranica ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači datu stranicu sa diska na način prikazan u zadatku 1.2.1.

c) S obzirom:

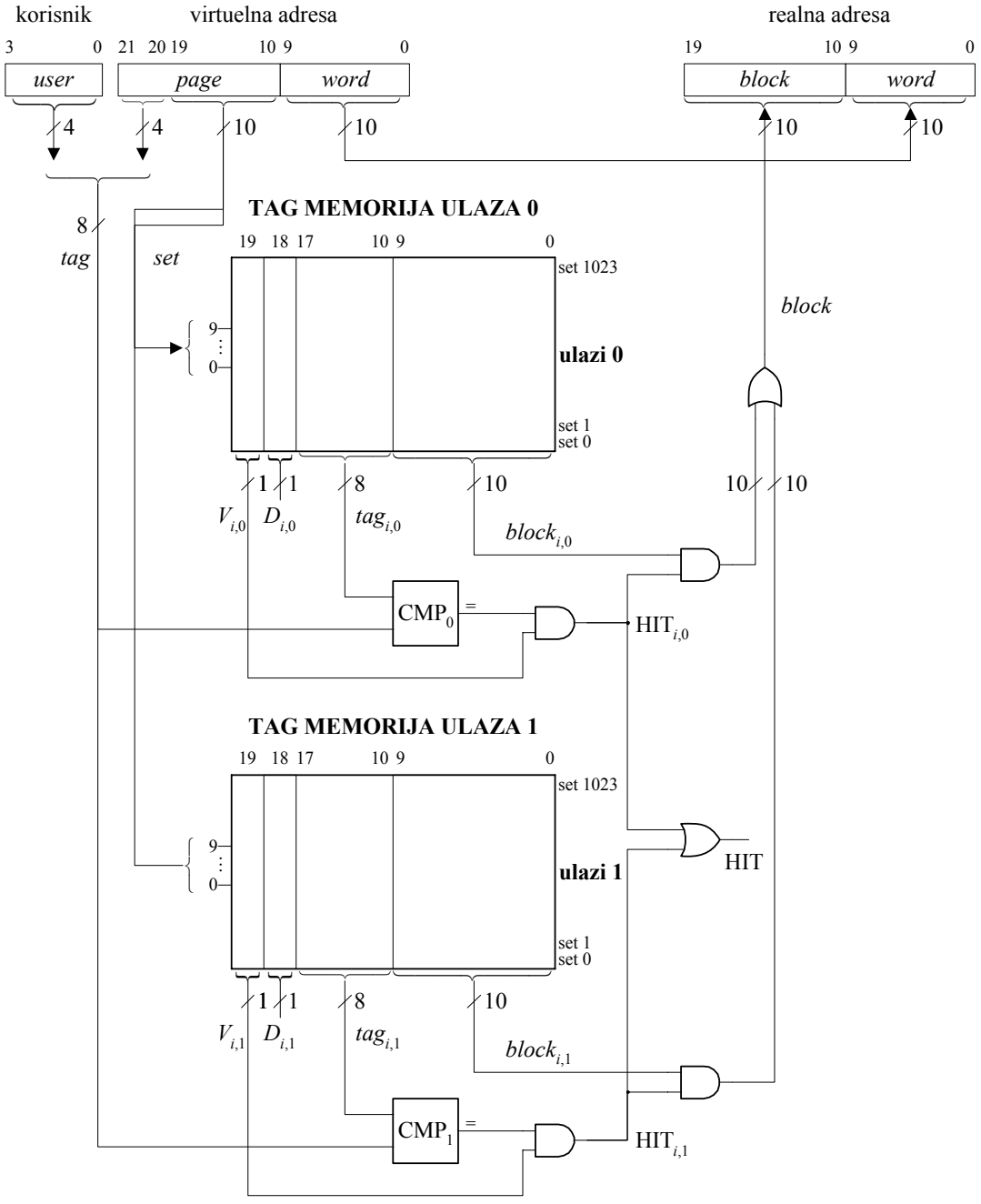
- da u adresnom prostoru korisnika ima 16 K stranica ($2^{14}$ stranica),
- da je broj korisnika 16 ($2^4$ korisnika),
- da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati deskriptori stranica različitih korisnika,
- da je broj setova u jedinici za ubrzavanje preslikavanja 1 K ($2^{10}$ setova) i
- da postoje dva ulaza po setu ($2^1$ ulaza),

onda je $2^{14}$ stranica svih $2^4$ korisnika grupisano u $2^8$ grupa sa po $2^{10}$ stranica u grupi. Zaključuje se, kao što je prikazano na slici 1.2.6, da su:

- polje *tag* dužine 8 bita i
- polje *set* dužine 10 bita.

Polje *tag* je formirano tako da:

- 4 najstarija bita predstavljaju broj korisnika (*user*),
- 4 najmlađa bita predstavljaju 4 najstarija bita broja stranice (*page*) iz virtelne adrese.




**Slika 1.2.6.** Jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese za virtuelnu memoriju stranične organizacijerelaizovaja u tehnici set-asocijativnog preslikavanja

c1) Na osnovu zadatih karakteristika virtuelne memorije i jedinice za ubrzavanje preslikavanja dolazi se do strukture jedinice prikazane na istoj slici.

Jedinica za ubrzavanje preslikavanja se sastoji iz sledećih delova:

- TAG MEMORIJA ULAZA 0—RAM memorija kapaciteta 1024 reči širine 20 bita, koja se sastoji iz polja:
  - *V* (valid bitovi) dužine 1 bit,
  - *D* (dirty bitovi) dužine 1 bit,
  - *tag* dužine 8 bita,
  - *block* dužine 10 bita i
- TAG MEMORIJA ULAZA 1—RAM memorija kapaciteta 1024 reči širine 20 bita, koja se sastoji iz istih polja kao i TAG MEMORIJA ULAZA 0.
- $CMP_0$—komparator TAG MEMORIJE ULAZA 0.
- $CMP_1$—komparator TAG MEMORIJE ULAZA 1.

TAG MEMORIJA ULAZA 0 služi za čuvanje 1024 *block* polja deskriptora stranica zajedno sa odgovarajućim *V*, *D* i *tag* bitovima setova $SET_0$ do $SET_{1023}$ smeštenih u ulaze 0 jedinice za ubrzavanje preslikavanja.

*V* bitovi označavaju za svaki od 1024 seta ulaza 0 jedinice za ubrzavanje preslikavanja da li je odgovarajući ulaz TAG MEMORIJE ULAZA 0 važeći.

*D* bitovi označavaju za svaki od 1024 seta ulaza 0 jedinice za ubrzavanje preslikavanja da li je bilo upisa u neku od lokacija date stranice.

*tag* bitovi služe za čuvanje 1024 *tag* polja stranica čija se *block* polja deskriptora nalaze u odgovarajućim *block* poljima TAG MEMORIJE ULAZA 0.

*block* bitovi služe za čuvanje 1024 *block* polja deskriptora stranica koja se nalaze u ulazima 0 jedinice za ubrzavanje preslikavanja.

$CMP_0$ služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT}_{i,0}$ za ulaze 0 jedinice za ubrzavanje preslikavanja ukoliko:

- *tag* polje generisane adrese je isto kao sadržaj $tag_{i,0}$ polja TAG MEMORIJE ULAZA 0 adresiran poljem *set* generisane adrese i
- $V_{i,0}$ bit adresiran poljem *set* generisane adrese je postavljen.

TAG MEMORIJA ULAZA 1 služi za čuvanje 1024 *block* polja deskriptora stranica zajedno sa odgovarajućim *V*, *D* i *tag* bitovima setova $SET_0$ do $SET_{1023}$ smeštenih u ulaze 1 jedinice za ubrzavanje preslikavanja. Funkcije polja *V*, *D*, *tag* i *block* su iste kao u slučaju TAG MEMORIJE ULAZA 0.

$CMP_1$ služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT}_{i,1}$ za ulaze 1 jedinice za ubrzavanje preslikavanja na isti način kao signal saglasnosti $\mathbf{HIT}_{i,0}$.

c2) *set* bitovi (10) iz generisane adrese se koriste kao adresa za TAG MEMORIJU ULAZA 0 i TAG MEMORIJU ULAZA 1. $tag_{i,0}$ bitovi očitani iz TAG MEMORIJE ULAZA 0 se porede sa *tag* bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i da je $V_{i,0}$ bit, adresiran *set* bitovima generisane adrese, postavljen, signal saglasnosti $\mathbf{HIT}_{i,0}$ postaje aktivan. $tag_{i,1}$ bitovi očitani iz TAG MEMORIJE ULAZA 1 se porede sa *tag* bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i da je $V_{i,1}$ bit, adresiran *set* bitovima generisane adrese, postavljen, signal saglasnosti $\mathbf{HIT}_{i,1}$ postaje aktivan. Signal saglasnosti **HIT** je aktivan ukoliko je aktivan jedan od signala $\mathbf{HIT}_{i,0}$ i $\mathbf{HIT}_{i,1}$.

U formiranju 10 najstarijih bitova realne adrese učestvuju $block_{i,0}$ bitovi iz TAG MEMORIJE ULAZA 0 ukoliko je signal $\mathbf{HIT}_{i,0}$ aktivan, odnosno $block_{i,1}$ bitovi iz TAG

MEMORIJE ULAZA 1 ukoliko je signal $\mathbf{HIT}_{i,1}$ aktivan. Polje *word* iz virtuelne adrese daje 10 najmlađih bitova realne adrese.

Sve ove aktivnosti se realizuju hardverski.

Da bi se stranice različitih korisnika koje imaju isti broj preslikale svaka u svoj blok, u formiranju *tag* polja učestvuje ne samo bitovi 23 do 20 polja *page* generisane adrese, već i vrednost registra *user* procesora. Kod prebacivanja procesora sa korisnika na korianika u ovaj registar procesora se upisuje broj korisnika kome se dodeljuje procesor.

c3) Ukoliko se utvrdi da se relevantni deskriptor ne nalazi u hardveru, ide se hardverski u tabelu stranica i čita deskriptor na način prikazan u zadatku 1.2.1. Ukoliko se utvrdi da se stranica nalazi u operativnoj memoriji, polje *block* deskriptora se dovlači u hardver. Polje *set* generisane adrese određuje u koji od 1024 seta se smešta polje *block* deskriptora. Pošto za svaki set postoje ulazi 0 i 1, korišćenjem FIFO ili LRU algoritma zamene u okviru datog seta bira se ulaz jedinice za ubrzavanje preslikavanja u koji se smešta deskriptor. U polje *block* datog ulaza jedinice za ubrzavanje preslikavanja se upisuje polje *block* deskriptora. U isti ulaz se upisuje polje *tag* generisane adrese. U isti ulaz *V* bitova se upisuje 1, a u isti ulaz *D* bitova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se stranica ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači datu stranicu sa diska na način prikazan u zadatku 1.2.1.

### Zadatak 1.2.5

U računaru sa virtuelnom memorijom segmentne organizacije definisanom u zadatku 1.2.2. postoji jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese.

a) Uzeti da računar poseduje jedinicu za ubrzavanje preslikavanja virtuelnih u realne adrese, realizovanu u tehnici asocijativnog preslikavanja u kojoj se istovremeno čuvaju relevantni delovi deskriptora 32 najčešće korišćena segmenta do 16 korisnika. Uraditi sledeće:

a1) Nacrtati, označiti dužinu u bitovima i objasniti funkciju svih relevantnih delova jedinice za ubrzavanje preslikavanja.

a2) Označiti dužine u bitovima relevantnih delova virtuelne i realne adrese i objasniti kako se:

- proverava da li se relevantni deskriptor nalazi u jedinici za ubrzavanje preslikavanja,
- dolazi do deskriptora ukoliko se utvrdi da se nalazi u jedinici za ubrzavanje preslikavanja,
- formira realna adresa,
- vrši provera da li je korektan pristup segmentu zahtevan i da li je generisana adresa reči u segmentu unutar zadate veličine segmenta i
- u jedinici za ubrzavanje preslikavanja obezbeđuje da se segmenti različitih korisnika koje imaju isti broj ne preslikaju u isti, već svaki u svoj deo fizičke memorije.

a3) U slučaju da se za generisanu virtuelnu adresu utvrdi da se relevantni deskriptor ne nalazi u jedinici za ubrzavanje preslikavanja, objasniti šta se radi u slučaju:

- kada je relevantan segment u memoriji,
- kada relevantan segment nije u memoriji i

navesti šta se od toga radi hardverski, a šta softverski.

b) Uzeti da računar poseduje jedinicu za ubrzavanje preslikavanja virtuelnih u realne adrese, realizovanu u tehnici direktnog preslikavanja u kojoj se istovremeno čuvaju relevantni delovi deskriptora 64 najčešće korišćenih segmenata do 16 korisnika. Uraditi isto kao pod a) ovog zadatka.

c) Uzeti da računar poseduje jedinicu za ubrzavanje preslikavanja virtuelnih u realne adrese, realizovanu u tehnici set-asocijativnog preslikavanja sa dva ulaza po setu u kojoj se istovremeno čuvaju relevantni delovi deskriptora 64 najčešće korišćenih segmenata do 16 korisnika. Uraditi isto kao pod a) ovog zadatka.

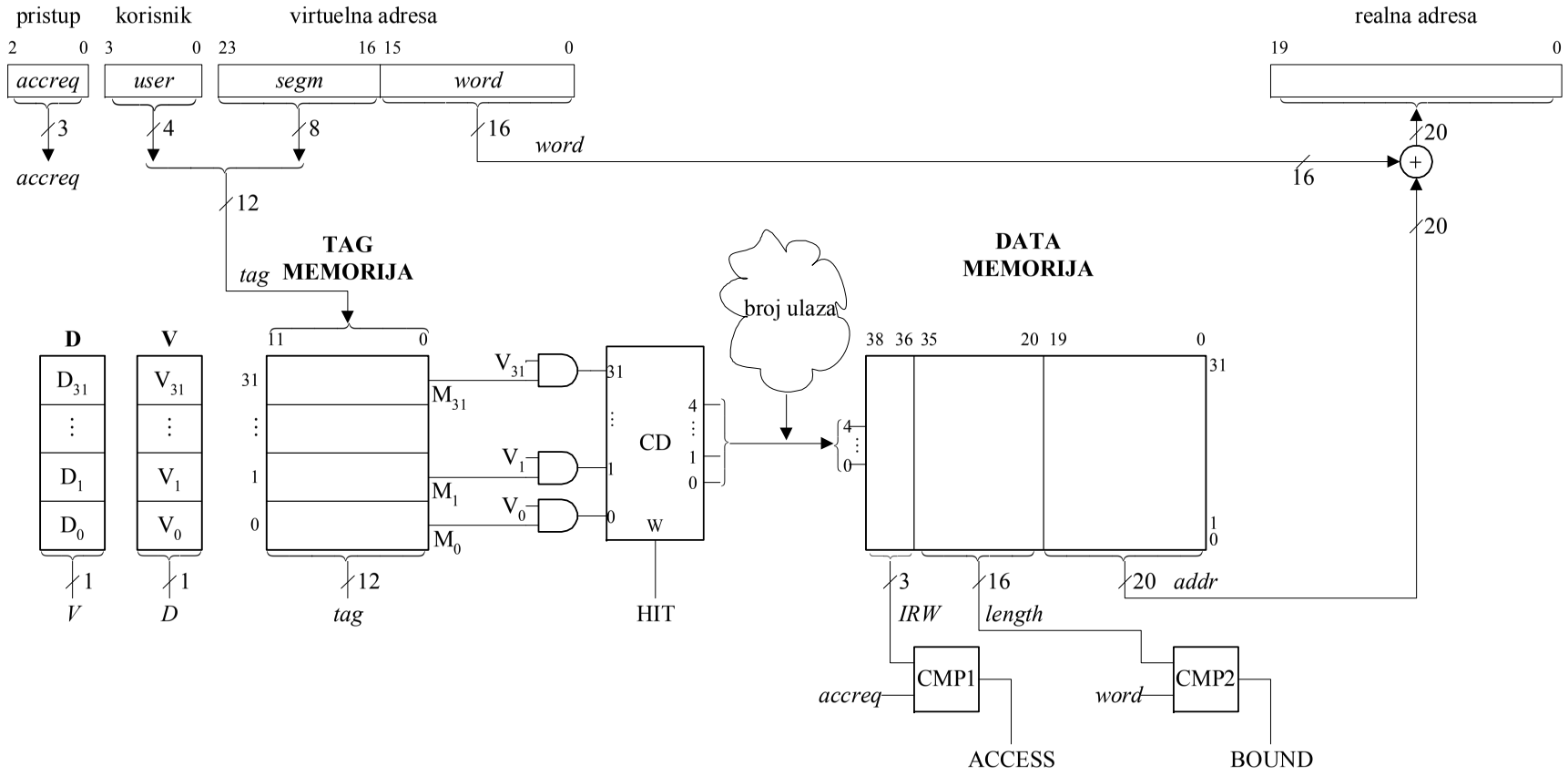
**Rešenje:**

a) S obzirom:

- da u virtuelnom adresnom prostoru korisnika ima 256 segmenata ($2^8$ segmenata),
- da je broj korisnika 16 ($2^4$ korisnika),
- da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati deskriptori stranica različitih korisnika i
- da je jedinica za ubrzavanje preslikavanja realizovana u tehnici asocijativnog preslikavanja,

zaključuje se, kao što je prikazano na slici 1.2.7, da je *tag* dužine 12 bita i da je formiran tako da:

- 4 najstarija bita predstavljaju broj korisnika (*user*) i
- 8 najmlađih bitova predstavljaju broj segmenta (*segm*) iz virtuelne adrese.


pristup
korisnik
virtuelna adresa
realna adresa
accreq
user
segm
word
word
accreq
TAG MEMORIJA
tag
DATA MEMORIJA
broj ulaza
D
V
CD
HIT
IRW
length
addr
CMP1
CMP2
accreq
word
ACCESS
BOUND

**Slika 1.2.7.** Jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese za virtuelnu memoriju segmentne organizacije realizovana u tehnici asocijativnog preslikavanja

a1) Na osnovu zadatih karakteristika virtuelne memorije i jedinice za ubrzavanje preslikavanja dolazi se do strukture jedinice za ubrzavanje preslikavanja na istoj slici.

Jedinica za ubrzavanje preslikavanja se sastoji iz sledećih delova:

- $D_{0...31}$ (dirty bitovi)—32 flip-flopa,
- $V_{0...31}$ (valid bitovi)—32 flip-flopa,
- TAG MEMORIJA—asocijativna memorija kapaciteta 32 reči širine 12 bita,
- CD—koder 32/5,
- CMP1—3-bitni komparator,
- CMP2—16-bitni komparator i
- DATA MEMORIJA—RAM memorija kapaciteta 32 reči širine 39 bita.

Dirty bitovi označavaju za svaki od 32 ulaza jedinice za ubrzavanje preslikavanja da li je bilo upisa u neku od lokacija datog segmenta.

Valid bitovi označavaju za svaki od 32 ulaza jedinice za ubrzavanje preslikavanja da li je odgovarajući ulaz TAG MEMORIJE važeći.

TAG MEMORIJA služi za čuvanje 32 TAG polja segmentat čiji se delovi deskriptora nalaze u odgovarajućim ulazima DATA MEMORIJE i generisanje aktivnih vrednosti signala saglasnosti $\mathbf{M_{0...31}}$ ukoliko postoji saglasnost polja *tag* generisane adrese i sadržaja odgovarajućeg ulaza TAG MEMORIJE.

CD služi za generisanje aktivne vrednosti signala saglasnosti **HIT** i broja ulaza u jedinicu za ubrzavanje preslikavanja za koji je otkrivena saglasnost.

Komparator CMP1 služi za proveru saglasnosti zahtevanog pristupa segmentu datog sadržajem registra *accreg* i dozvoljenog prava pristupa segmentu datog poljem *IRW* deskriptora i generisanje aktivne vrednosti signala **ACCESS** ukoliko se zahteva pristup segmentu koji nije dozvoljen.

Komparator CMP2 služi za proveru da li je generisana adresa reči u segmentu, datog poljem *word* virtuelne adrese, unutar zadate veličine segmenta, date poljem *length* deskriptora, i generisanje aktivne vrednosti signala **BOUND** ukoliko to nije slučaj.

DATA MEMORIJA služi za čuvanje delova 32 deskriptora segmenata.

a2) TAG bitovi (12) iz generisane adrese se porede sa sadržajima sva 32 ulaza u TAG MEMORIJI. Ako se sadržaj bilo kojeg ulaza slaže sa *tag* bitovima generisane adrese i odgovarajući V bit je postavljen, signal saglasnosti HIT postaje aktivan.

Bitovi (5) koji označavaju broj ulaza u jedinicu za ubrzavanje preslikavanja gde je otkrivena saglasnost dobijeni sa izlaza kodera se koriste kao adresa u DATA MEMORIJI i delovi deskriptora se čitaju.

Očitano polje *addr* dužine 20 bitova i polje *word* iz virtuelne adrese dužine 16 bitova se sabiraju čime se dobija realna adresa dužine 20 bitova.

Provera da li je korektan pristup segmentu zahtevan se realizuje upoređivanjem sadržaja registra *accreg* i polja *IRW* deskriptora. Provera da je generisana adresa reči u segmentu unutar zadate veličine segmenta se realizuje upoređivanjem polja *word* virtuelne adrese i polja *length* deskriptora.

Sve ove aktivnosti se realizuju hardverski.

Da bi se segmenti različitih korisnika koji imaju isti broj preslikali svaki u svoj deo fizičke memorije, u formiranju *tag* polja učestvuje ne samo polje *segm* generisane adrese, već i vrednost registra *user* procesora. Kod prebacivanja procesora sa korisnika na korisnika u registar *user* procesora se upisuje broj korisnika kome se dodeljuje procesor.

a3) Ukoliko se utvrdi da se relevantni deskriptor ne nalazi u hardveru, ide se hardverski u tabelu segmenata i čita deskriptor na način prikazan u zadatku 1.2.1. Ukoliko se utvrdi da se segment nalazi u operativnoj memoriji, polja *IRW*, *length* i *addr* deskriptora se dovlače u jedinicu za ubrzavanje preslikavanja. Korišćenjem FIFO ili LRU algoritma zamene bira se ulaz jedinice za ubrzavanje preslikavanja u koji se smeštaju ovi delovi deskriptora. U odabrani ulaz DATA MEMORIJE se upisuju polja *IRW*, *length* i *addr* deskriptora. U isti ulaz TAG MEMORIJE se upisuje polje *tag* generisane adrese. U isti ulaz *V* flip-flopova se upisuje 1, a u isti ulaz *D* flip-flopova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se segment ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači dati segment sa diska na način prikazan u zadatku 1.2.1.

b) S obzirom:

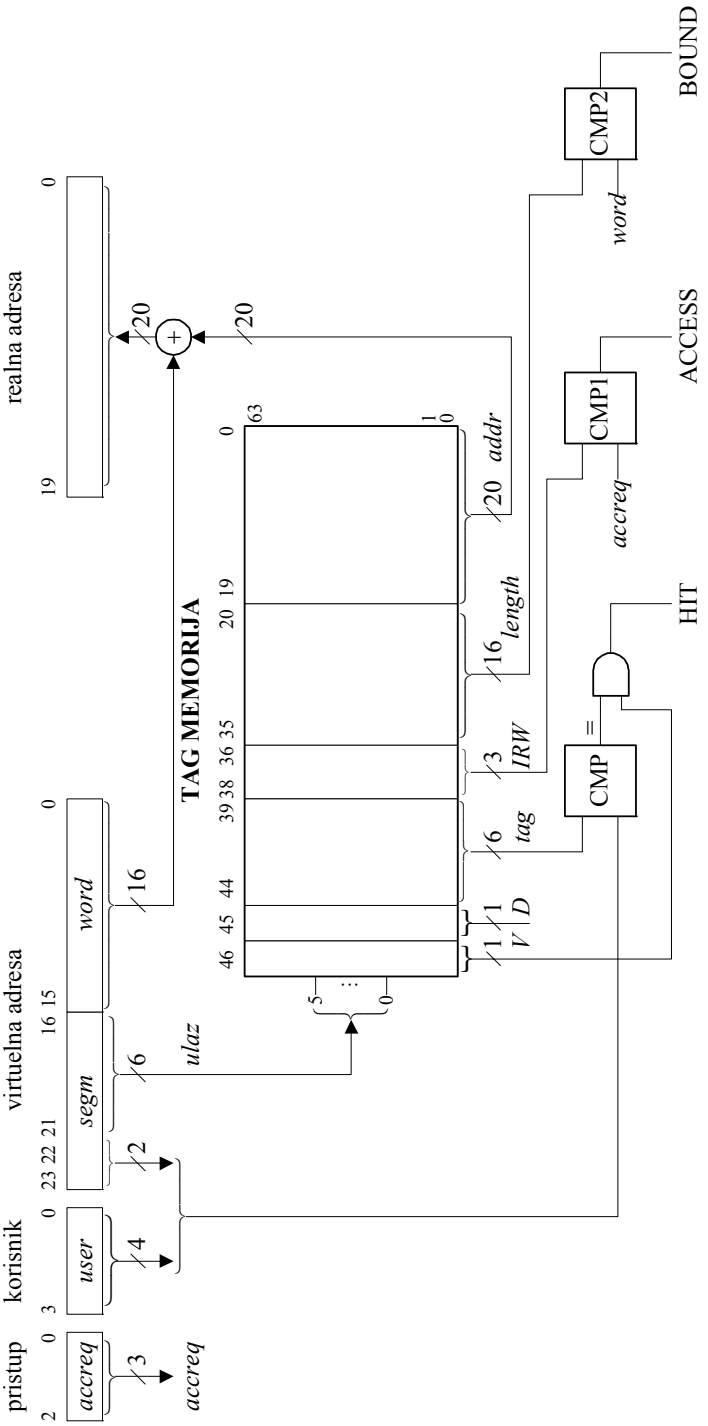
- da u virtuelnom adresnom prostoru korisnika ima 256 segmentat ($2^8$ segmenata),
- da je broj korisnika 16 ($2^4$ korisnika),
- da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati deskriptori segmenata različitih korisnika,
- da je jedinica za ubrzavanje preslikavanja realizovana u tehnici direktnog preslikavanja i
- da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati relevantni delovi deskriptora 64 ($2^6$) najčešće korišćenih segmenata do 16 korisnika ($2^4$ korisnika),

onda je $2^8$ segmenata svih $2^4$ korisnika grupisano u $2^6$ grupa sa po $2^6$ segmenata u grupi. Zaključuje se, kao što je prikazano na slici 1.2.8, da su:

- polje *tag* dužine 6 bita i
- polje *ulaz* dužine 6 bita.

Polje *tag* je formirano tako da:

- 4 najstarija bita predstavljaju broj korisnika (*user*) i
- 2 najmlađa bita predstavljaju 2 najstarija bita broja segmenta (*segm*) iz virtuelne adrese.




**Slika 1.2.8.** Jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese za virtuelnu memoriju segmentne organizacije realizovana u tehnici direktnog preslikavanja

b1) Na osnovu zadatih karakteristika virtuelne memorije i jedinice za ubrzavanje preslikavanja dolazi se do strukture jedinice prikazane na istoj slici.

Jedinica za ubrzavanje preslikavanja se sastoji iz sledećih delova:

- TAG MEMORIJA—RAM memorija kapaciteta 64 reči širine 46 bita, koja se sastoji iz polja:
  - *V* (valid bitovi) dužine 1 bit,
  - *D* (dirty bitovi) dužine 1 bit,
  - *tag* dužine 6 bita,
  - *IRW* dužine 3 bita,
  - *length* dužine 16 bita i
  - *addr* dužine 20 bita,
- CMP—6-bitni komparator,
- CMP1—3-bitni komparator i
- CMP2—16-bitni komparator.

TAG MEMORIJA služi za čuvanje delova *IRW*, *length* i *addr* 64 deskriptora segmenata zajedno sa odgovarajućim *V*, *D* i *tag* bitovima. Značenje ovih bitova je sledeće:

- *V* bitovi označavaju za svaki od 64 ulaza jedinice za ubrzavanje preslikavanja da li je odgovarajući ulaz TAG MEMORIJE važeći,
- *D* bitovi označavaju za svaki od 64 ulaza jedinice za ubrzavanje preslikavanja da li je bilo upisa u neku od lokacija datog segmenta,
- *tag* bitovi služe za čuvanje 64 *tag* polja segmenata čiji se delovi deskriptora nalaze u poljima *IRW*, *length* i *addr* TAG MEMORIJE,
- *IRW* bitovi označavaju dozvoljen pristup datom segmentu,
- *length* bitovi označavaju dužinu segmenta i
- *addr* bitovi označavaju početnu adresu dela operativne memorije u kome se nalazi dati segment.

CMP služi za generisanje aktivne vrednosti signala saglasnosti **HIT** ukoliko:

- *tag* polje generisane adrese je isto kao sadržaj *tag* polja TAG MEMORIJE adresiran poljem *ulaz* generisane adrese i
- *V* bit adresiran poljem *ulaz* generisane adrese je postavljen.

CMP1 služi za generisanje aktivne vrednosti signala **ACCESS** ukoliko je sadržaj registara *accreg* isti kao sadržaj polja *IRW* TAG MEMORIJE adresiran poljem *ulaz* generisane adrese.

CMP2 služi za generisanje aktivne vrednosti signala **BOUND** ukoliko je sadržaj polja *word* virtuelne adrese manji od sadržaja polja *length* TAG MEMORIJE adresiran poljem *ulaz* generisane adrese.

b2) *ulaz* bitovi (6) iz generisane adrese se koriste kao adresa za TAG MEMORIJU. *tag* bitovi očitani iz TAG MEMORIJE se porede sa *tag* bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i da je *V* bit, adresiran *ulaz* bitovima generisane adrese, postavljen, signal saglasnosti **HIT** postaje aktivan.

Očitano polje *addr* i polje *word* iz virtuelne adrese se sabiraju i daju 20 bitova realne adrese.

Sve ove aktivnosti se realizuju hardverski.

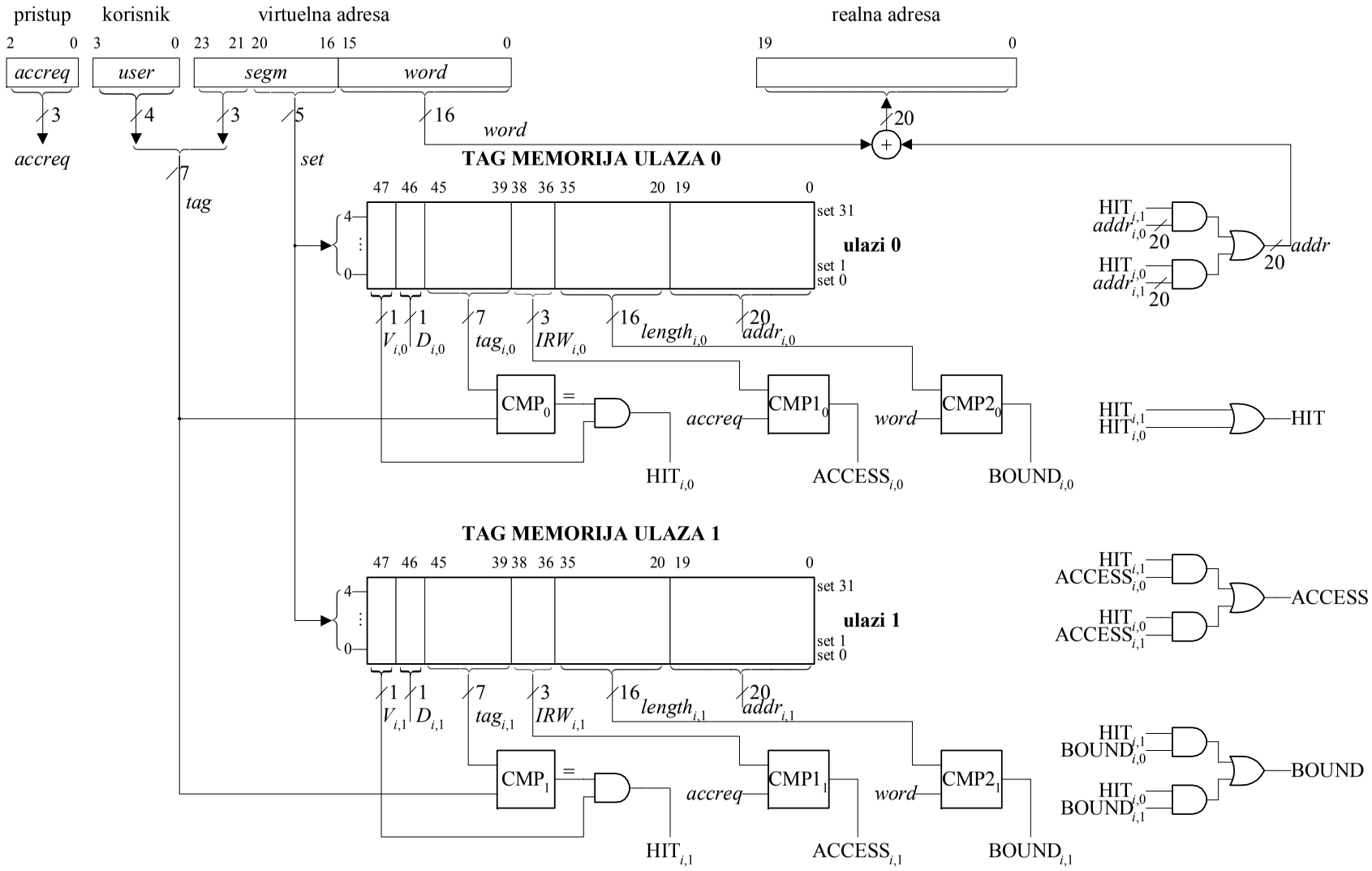
Da bi se segmenti različitih korisnika koji imaju isti broj preslikali svaki u svoj deo memorije, u formiranju *tag* polja učestvuje ne samo polje *segm* generisane adrese, već i vrednost registra *user* procesora. Kod prebacivanja procesora sa korisnika na korianika u ovaj registar procesora se upisuje broj korisnika kome se dodeljuje procesor.

b3) Ukoliko se utvrdi da se relevantni delovi deskriptora ne nalaze u jedinici za ubrzavanje preslikavanja, ide se hardverski u tabelu segmenata i čita deskriptor na način prikazan u zadatku 1.2.1. Ukoliko se utvrdi da se segment nalazi u operativnoj memoriji, polja *IRW*, *length* i *addr* deskriptora se dovlače u jedinicu za ubrzavanje preslikavanja. U ulaz DATA MEMORIJE određen *ulaz* bitovima virtuelne adrese se upisuju polja *IRW*, *length* i *addr* deskriptora. U isti ulaz TAG MEMORIJE se upisuje polje *tag* generisane adrese. U isti ulaz *V* bitova se upisuje 1, a u isti ulaz *D* bitova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se segment ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači dati segment sa diska na način prikazan u zadatku 1.2.1.

c) S obzirom

- da u virtuelnom adresnom prostoru korisnika ima 256 segmenata ($2^8$ segmenata),
- da je broj korisnika 16 ($2^4$ korisnika),
- da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati delovi deskriptora segmenata različitih korisnika,
- da je broj setova u jedinici za ubrzavanje preslikavanja 32 ($2^5$ setova) i
- da postoje dva ulaza po setu ($2^1$ ulaza),

onda je $2^8$ segmenata svih $2^4$ korisnika grupisano u $2^7$ grupa sa po $2^5$ segmenata u grupi. Zaključuje se, kao što je prikazano na slici 1.2.9, da je polje *tag* dužine 7 bita i da je polje *set* dužine 5 bita.


pristup
korisnik
virtuelna adresa
realna adresa
accreq
user
segm
word
TAG MEMORIJA ULAZA 0
TAG MEMORIJA ULAZA 1
ulazi 0
ulazi 1
set 31
set 1
set 0
tag
set
CMP0
CMP1
CMP2
HIT
ACCESS
BOUND
addr

**Slika 1.2.9.** Jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese za virtuelnu memoriju segmentne organizacije realizovana u tehnici set-asocijativnog preslikavanja

c1) Na osnovu zadatih karakteristika virtuelne memorije i jedinice za ubrzavanje preslikavanja dolazi se do strukture jedinice prikazane na istoj slici.

Jedinica za ubrzavanje preslikavanja se sastoji iz sledećih delova:

- TAG MEMORIJA ULAZA 0—RAM memorija kapaciteta 32 reči širine 48 bita, koja se sastoji iz polja:
  - *V* (valid bitovi) dužine 1 bit,
  - *D* (dirty bitovi) dužine 1 bit,
  - *tag* dužine 7 bita,
  - *IRW* dužine 3 bita,
  - *length* dužine 16 bita i
  - *addr* dužine 20 bita,
- TAG MEMORIJA ULAZA 1—RAM memorija kapaciteta 32 reči širine 48 bita, koja se sastoji iz istih polja kao i TAG MEMORIJA ULAZA 0.
- $CMP_0$, $CMP1_0$, $CMP2_0$—komparatori TAG MEMORIJE ULAZA 0.
- $CMP_1$, $CMP1_1$, $CMP2_1$—komparatori TAG MEMORIJE ULAZA 1.

TAG MEMORIJA ULAZA 0 služi za čuvanje delova *IRW*, *length* i *addr* 32 deskriptora segmenata zajedno sa odgovarajućim *V*, *D* i *tag* bitovima. Značenje ovih bitova je sledeće:

- *V* bitovi označavaju za svaki od 32 ulaza jedinice za ubrzavanje preslikavanja da li je odgovarajući ulaz TAG MEMORIJE ULAZA 0 važeći,
- *D* bitovi označavaju za svaki od 32 ulaza jedinice za ubrzavanje preslikavanja da li je bilo upisa u neku od lokacija datog segmenta,
- *tag* bitovi služe za čuvanje 32 *tag* polja segmenata čiji se delovi deskriptora nalaze u poljima *IRW*, *length* i *addr* TAG MEMORIJE ULAZA 0 ,
- *IRW* bitovi označavaju dozvoljen pristup datom segmentu,
- *length* bitovi označavaju dužinu segmenta i
- *addr* bitovi označavaju početnu adresu dela operativne memorije u kome se nalazi dati segment.

$CMP_0$ služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT}_{i,0}$ za ulaze 0 jedinice za ubrzavanje preslikavanja ukoliko:

- *tag* polje generisane adrese je isto kao sadržaj $tag_{i,0}$ polja TAG MEMORIJE ULAZA 0 adresiran poljem *set* generisane adrese i
- $V_{i,0}$ bit adresiran poljem *set* generisane adrese je postavljen.

$CMP1_0$ služi za generisanje aktivne vrednosti signala $\mathbf{ACCESS}_{i,0}$ ukoliko je sadržaj registara *accreg* isti kao sadržaj polja *IRW* TAG MEMORIJE ULAZA 0 adresiran poljem *ulaz* generisane adrese.

$CMP2_0$ služi za generisanje aktivne vrednosti signala $\mathbf{BOUND}_{i,0}$ ukoliko je sadržaj polja *word* virtuelne adrese manji od sadržaja polja *length* TAG MEMORIJE ULAZA 0 adresiran poljem *ulaz* generisane adrese.

TAG MEMORIJA ULAZA 1 služi za čuvanje delova *IRW*, *length* i *addr* 32 deskriptora segmenata zajedno sa odgovarajućim *V*, *D* i *tag* bitovima. Funkcije polja *V*, *D*, *tag*, *IRW*, *length* i *addr* su iste kao u slučaju TAG MEMORIJE ULAZA 0.

$CMP_1$ služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT_{i,0}}$ za ulaze 1 jedinice za ubrzavanje preslikavanja ukoliko:

- *tag* polje generisane adrese je isto kao sadržaj $tag_{i,1}$ polja TAG MEMORIJE ULAZA 1 adresiran poljem *set* generisane adrese i
- $V_{i,1}$ bit adresiran poljem *set* generisane adrese je postavljen.

$CMP1_1$ služi za generisanje aktivne vrednosti signala $\mathbf{ACCESS_{i,1}}$ ukoliko je sadržaj registara *accreg* isti kao sadržaj polja *IRW* TAG MEMORIJE ULAZA 1 adresiran poljem *ulaz* generisane adrese.

$CMP2_1$ služi za generisanje aktivne vrednosti signala $\mathbf{BOUND_{i,1}}$ ukoliko je sadržaj polja *word* virtuelne adrese manji od sadržaja polja *length* TAG MEMORIJE ULAZA 1 adresiran poljem *ulaz* generisane adrese.

c2) *set* bitovi (5) iz generisane adrese se koriste kao adresa za TAG MEMORIJU ULAZA 0 i TAG MEMORIJU ULAZA 1. $tag_{i,0}$ bitovi očitani iz TAG MEMORIJE ULAZA 0 se porede sa *tag* bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i da je $V_{i,0}$ bit, adresiran *set* bitovima generisane adrese, postavljen, signal saglasnosti $\mathbf{HIT_{i,0}}$ postaje aktivan. $tag_{i,1}$ bitovi očitani iz TAG MEMORIJE ULAZA 1 se porede sa *tag* bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i da je $V_{i,1}$ bit, adresiran *set* bitovima generisane adrese, postavljen, signal saglasnosti $\mathbf{HIT_{i,1}}$ postaje aktivan. Signal saglasnosti **HIT** je aktivan ukoliko je aktivan jedan od signala $\mathbf{HIT_{i,0}}$ i $\mathbf{HIT_{i,1}}$.

Realna adresa se formira tako što se $addr_{i,0}$ bitovi iz TAG MEMORIJE ULAZA 0 ukoliko je signal $\mathbf{HIT_{i,0}}$ aktivan, odnosno $addr_{i,1}$ bitovi iz TAG MEMORIJE ULAZA 1 ukoliko je signal $\mathbf{HIT_{i,1}}$ aktivan sabiraju sa sadržajem polja *word* iz virtuelne adrese.

Sve ove aktivnosti se realizuju hardverski.

Da bi se segmenti različitih korisnika koji imaju isti broj preslikali svaki u svoj deo memorije, u formiranju *tag* polja učestvuju ne samo bitovi 23 do 21 polja *segm* generisane adrese, već i vrednost registra *user* procesora. Kod prebacivanja procesora sa korisnika na korianika u ovaj registar procesora se upisuje broj korisnika kome se dodeljuje procesor.

c3) Ukoliko se utvrdi da se delovi relevantnog deskriptora ne nalaze u hardveru, ide se hardverski u tabelu segmenata i čita deskriptor na način prikazan u zadatku 1.2.1. Ukoliko se utvrdi da se segment nalazi u operativnoj memoriji, polja *IRW*, *length* i *addr* deskriptora se dovlače u jedinicu za ubrzavanje preslikavanja. Polje *set* generisane adrese određuje u koji od 32 seta se smešta polje *addr* deskriptora. Pošto za svaki set postoje ulazi 0 i 1, korišćenjem FIFO ili LRU algoritma zamene u okviru datog seta bira se ulaz jedinice za ubrzavanje preslikavanja u koji se smešta deskriptor. U polje *addr* datog ulaza jedinice za ubrzavanje preslikavanja se upisuje polje *addr* deskriptora. U isti ulaz se upisuje polje *tag* generisane adrese. U isti ulaz *V* bitova se upisuje 1, a u isti ulaz *D* bitova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se stranica ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači datu stranicu sa diska na način prikazan u zadatku 1.2.1.

### Zadatak 1.2.6

U računaru sa virtuelnom memorijom segmentno-stranične organizacije definisanom u zadatku 1.2.1. postoji jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese.

a) Uzeti da računar poseduje jedinicu za ubrzavanje preslikavanja virtuelnih u realne adrese, realizovanu u tehnici asocijativnog preslikavanja u kojoj se istovremeno čuvaju

relevantni delovi deskriptora 256 najčešće korišćenih stranica do 16 korisnika. Uraditi sledeće:

a1) Nacrtati, označiti dužinu u bitovima i objasniti funkciju svih relevantnih delova jedinice za ubrzavanje preslikavanja.

a2) Označiti dužine u bitovima relevantnih delova virtuelne i realne adrese i objasniti kako se:

- proverava da li se relevantni deskriptor nalazi u jedinici za ubrzavanje preslikavanja;
- dolazi do deskriptora ukoliko se utvrdi da se nalazi u jedinici za ubrzavanje preslikavanja;
- formira realna adresa;
- u jedinici za ubrzavanje preslikavanja obezbeđuje da se stranice različitih korisnika koje imaju isti broj ne preslikaju u isti, već svaka u svoj blok.

a3) U slučaju da se za generisanu virtuelnu adresu utvrdi da se relevantni deskriptor ne nalazi u jedinici za ubrzavanje preslikavanja, objasniti šta se radi u slučaju:

- kada je relevantna stranica u memoriji;
- kada relevantna stranica nije u memoriji i

navesti šta se od toga radi hardverski, a šta softverski.

b) Uzeti da računar poseduje jedinicu za ubrzavanje preslikavanja virtuelnih u realne adrese, realizovanu u tehnici direktnog preslikavanja u kojoj se istovremeno čuvaju relevantni delovi deskriptora 256 najčešće korišćenih stranica do 16 korisnika. Uraditi isto kao pod a) ovog zadatka.

c) Uzeti da računar poseduje jedinicu za ubrzavanje preslikavanja virtuelnih u realne adrese, realizovanu u tehnici set-asocijativnog preslikavanja sa dva ulaza po setu u kojoj se istovremeno čuvaju relevantni delovi deskriptora 256 najčešće korišćenih stranica do 16 korisnika. Uraditi isto kao pod a) ovog zadatka.

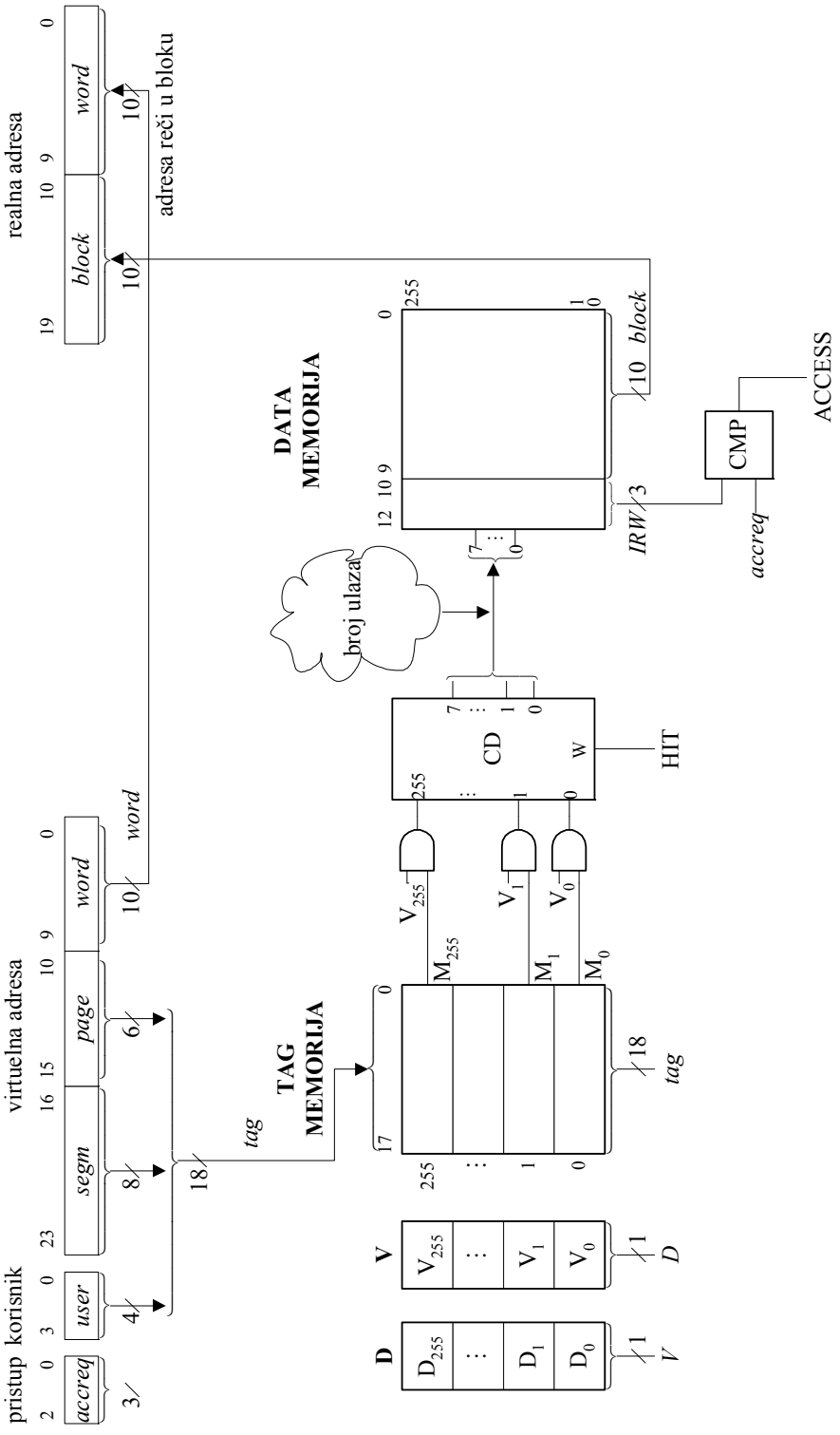
**Rešenje:**

a) S obzirom:

- da u virtuelnom adresnom prostoru korisnika ima 256 segmenata ($2^8$ segmenata),
- da je maksimalan broj stranica u segmentu 64 ($2^6$),
- da je broj korisnika 16 ($2^4$ korisnika),
- da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati delovi deskriptora stranica segmenata različitih korisnika i
- da je jedinica za ubrzavanje preslikavanja realizovana u tehnici asocijativnog preslikavanja,

zaključuje se, kao što je prikazano na slici 1.2.10, da je *tag* dužine 18 bita i da je formiran tako da:

- 4 najstarija bita predstavljaju broj korisnika (*user*),
- 8 srednjih bitova predstavljaju broj segmenta (*segm*) i
- 6 najmlađih bitova predstavljaju broj stranice (*page*) iz virtuelne adrese.




**Slika 1.2.10.** Jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese za virtuelnu memoriju segmentno-stranične organizacije realizovana u tehnici asocijativnog preslikavanja

a1) Na osnovu zadatih karakteristika virtuelne memorije i jedinice za ubrzavanje preslikavanja dolazi se do strukture jedinice prikazane na istoj slici.

Jedinica za ubrzavanje preslikavanja se sastoji iz sledećih delova:

- $D_{0...255}$ (dirty bitovi)—256 flip-flopova,
- $V_{0...255}$ (valid bitovi)—256 flip-flopova,
- TAG MEMORIJA—asocijativna memorija kapaciteta 256 reči širine 18 bita,
- CD—koder 256/8 i
- DATA MEMORIJA—RAM memorija kapaciteta 256 reči širine 13 bita.

Dirty bitovi označavaju za svaki od 256 ulaza jedinice za ubrzavanje preslikavanja da li je bilo upisa u neku od lokacija date stranice.

Valid bitovi označavaju za svaki od 256 ulaza jedinice za ubrzavanje preslikavanja da li je odgovarajući ulaz TAG MEMORIJE važeći.

TAG MEMORIJA služi za čuvanje 256 TAG polja stranica čiji se deskriptori nalaze u odgovarajućim ulazima DATA MEMORIJE i generisanje aktivnih vrednosti signala saglasnosti $\mathbf{M_{0...255}}$ ukoliko postoji saglasnost TAG polja generisane adrese i sadržaja odgovarajućeg ulaza TAG MEMORIJE.

CD služi za generisanje aktivne vrednosti signala saglasnosti **HIT** i broja ulaza u jedinicu za ubrzavanje preslikavanja za koji je otkrivena saglasnost.

DATA MEMORIJA služi za čuvanje 256 deskriptora stranica.

a2) TAG bitovi (18) iz generisane adrese se porede sa sadržajima svih 256 ulaza u TAG MEMORIJI. Ako se sadržaj bilo kojeg ulaza slaže sa TAG bitovima generisane adrese i odgovarajući V bit je postavljen, signal saglasnosti HIT postaje aktivan.

Bitovi (8) koji označavaju broj ulaza u jedinicu za ubrzavanje preslikavanja gde je otkrivena saglasnost dobijeni sa izlaza kodera se koriste kao adresa u DATA MEMORIJI i deskriptor se čita.

Očitano polje *block* daje 10 najstarijih bitova a polje *word* iz virtuelne adrese 10 najmlađih bitova realne adrese.

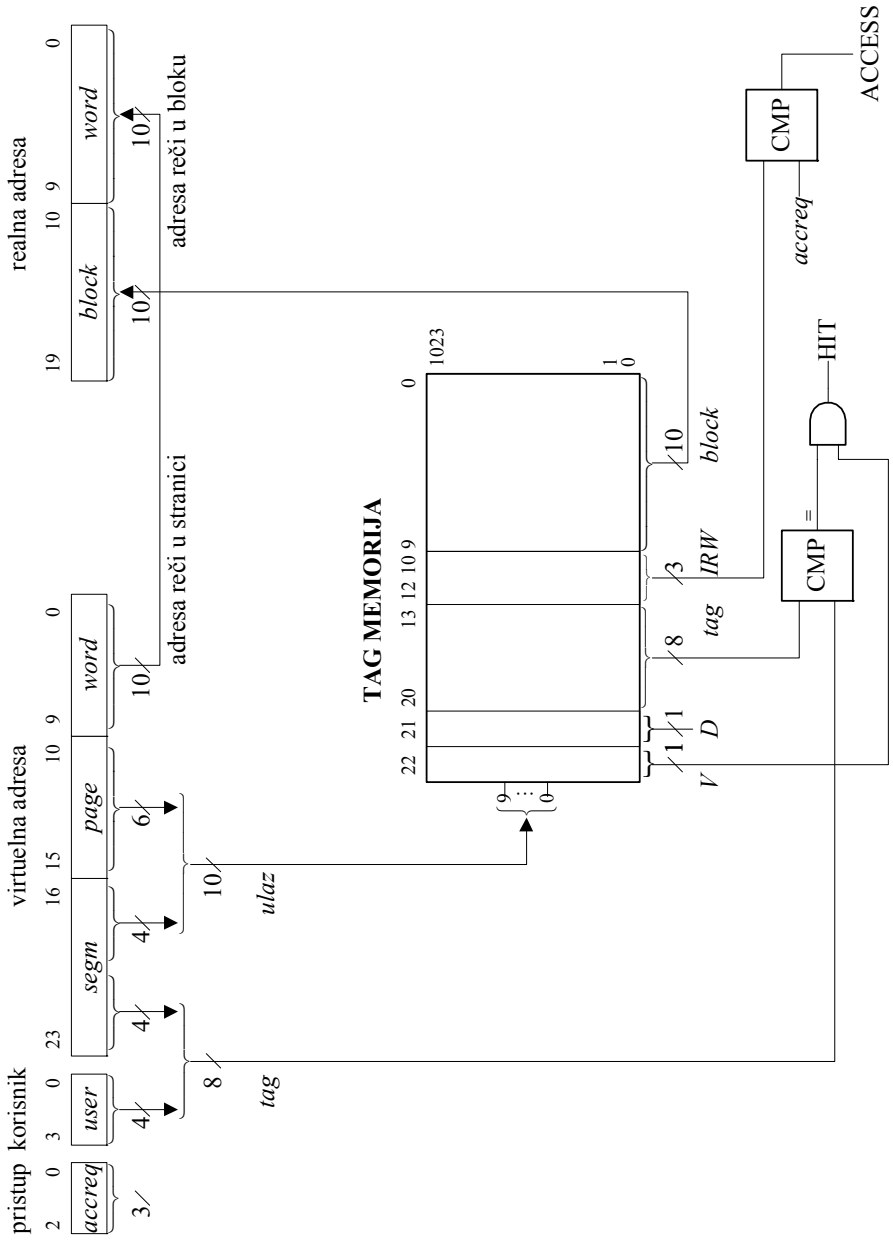
Sve ove aktivnosti se realizuju hardverski.

Da bi se stranice različitih segmenata svih korisnika koje imaju isti broj preslikale svaka u svoj blok, u formiranju *tag* polja učestvuje ne samo polje *page* generisane adrese, već i vrednost polja *segm* generisane adrese i vrednost registra *user* procesora. Kod prebacivanja procesora sa korisnika na korisnika u ovaj registar procesora se upisuje broj korisnika kome se dodeljuje procesor.

a3) Ukoliko se utvrdi da se relevantni deskriptor ne nalazi u hardveru, ide se hardverski u tabelu stranica i čita deskriptor na način prikazan u zadatku 1.2.1. Ukoliko se utvrdi da se stranica nalazi u operativnoj memoriji, polje *block* deskriptora se dovlači u hardver. Korišćenjem FIFO ili LRU algoritma zamene bira se ulaz jedinice za ubrzavanje preslikavanja u koji se smešta deskriptor. U dati ulaz DATA MEMORIJE se upisuje polje *block* deskriptora. U isti ulaz TAG MEMORIJE se upisuje polje *tag* generisane adrese. U isti ulaz *V* flip-flopova se upisuje 1, a u isti ulaz *D* flip-flopova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se stranica ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači datu stranicu sa diska na način prikazan u zadatku 1.2.1.

b) S obzirom da u adresnom prostoru korisnika ima 16 K stranica ($2^{14}$ stranica), da je broj korisnika 16 ($2^4$ korisnika), da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati deskriptori stranica različitih korisnika i da je broj ulaza u jedinici za ubrzavanje preslikavanja 2 K ulaza ($2^{11}$ ulaza), onda je $2^{14}$ stranica svih $2^4$ korisnika grupisano u $2^7$ grupa

sa po $2^{11}$ stranica u grupi. Zaključuje se da je polje TAG dužine 7 bita i da je polje *ulaz* dužine 11 bita kao što je prikazano na slici 1.2.11.




**Slika 1.2.11.** Jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese za virtuelnu memoriju segmentno-stranične organizacije realizovana u tehnici direktnog preslikavanja

b1) Na osnovu zadatih karakteristika virtuelne memorije i jedinice za ubrzavanje preslikavanja dolazi se do strukture jedinice prikazane na istoj slici.

Jedinica za ubrzavanje preslikavanja se sastoji iz sledećih delova:

- TAG MEMORIJA—RAM memorija kapaciteta 2048 reči širine 19 bita, koja se sastoji iz polja:
  - *D* (dirty bitovi) dužine 1 bit,

- *V* (valid bitovi) dužine 1 bit,
- *tag* dužine 7 bita,
- *block* dužine 10 bita i

- CMP—komparator.

TAG MEMORIJA služi za čuvanje 2048 deskriptora stranica zajedno sa odgovarajućim *V*, *D* i *tag* bitovima.

*D* bitovi označavaju za svaki od 2048 ulaza jedinice za ubrzavanje preslikavanja da li je bilo upisa u neku od lokacija date stranice.

*V* bitovi označavaju za svaki od 2048 ulaza jedinice za ubrzavanje preslikavanja da li je odgovarajući ulaz TAG MEMORIJE važeći.

*tag* bitovi služe za čuvanje 2048 *tag* polja stranica čiji se deskriptori nalaze u odgovarajućim *block* poljima TAG MEMORIJE.

CMP služi za generisanje aktivne vrednosti signala saglasnosti **HIT** ukoliko:

- *tag* polje generisane adrese je isto kao sadržaj *tag* polja TAG MEMORIJE adresiran poljem *ulaz* generisane adrese i
- *V* bit adresiran poljem *ulaz* generisane adrese je postavljen.

b2) *ulaz* bitovi (11) iz generisane adrese se koriste kao adresa za TAG MEMORIJU. *tag* bitovi očitani iz TAG MEMORIJE se porede sa *tag* bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i da je *V* bit, adresiran *ulaz* bitovima generisane adrese, postavljen, signal saglasnosti **HIT** postaje aktivan.

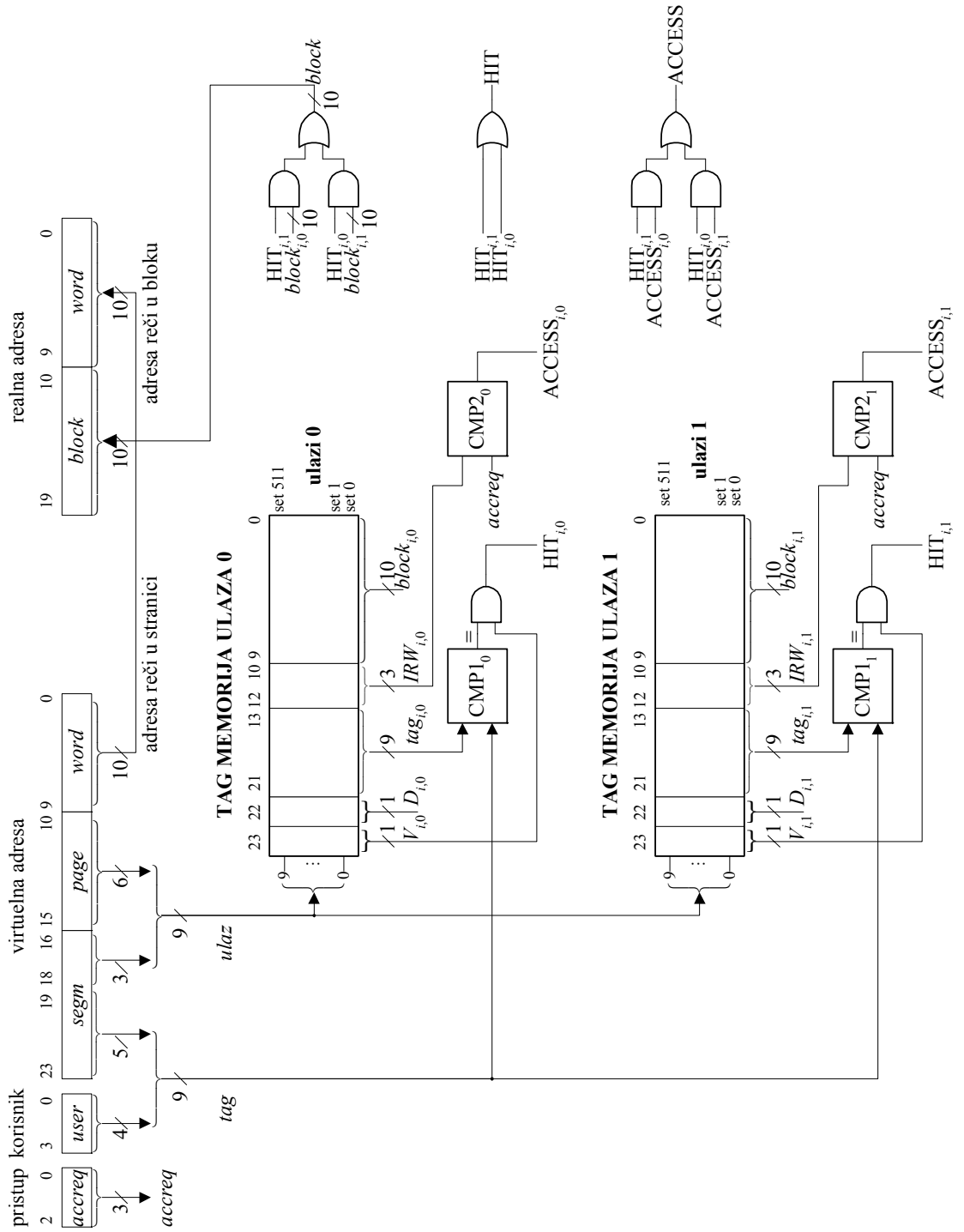
Očitano polje *block* daje 10 najstarijih bitova a polje *word* iz virtuelne adrese 10 najmlađih bitova realne adrese.

Sve ove aktivnosti se realizuju hardverski.

Da bi se stranice različitih korisnika koje imaju isti broj preslikale svaka u svoj blok, u formiranju *tag* polja učestvuje ne samo polje *page* generisane adrese, već i vrednost registra *user* procesora. Kod prebacivanja procesora sa korisnika na korianika u ovaj registar procesora se upisuje broj korisnika kome se dodeljuje procesor.

b3) Ukoliko se utvrdi da se relevantni deskriptor ne nalazi u hardveru, ide se hardverski u tabelu stranica i čita deskriptor na način prikazan u zadatku 1.2.1. Ukoliko se utvrdi da se stranica nalazi u operativnoj memoriji, polje *block* deskriptora se dovlači u hardver. Korišćenjem FIFO ili LRU algoritma zamene bira se ulaz jedinice za ubrzavanje preslikavanja u koji se smešta deskriptor. U dati ulaz DATA MEMORIJE se upisuje polje *block* deskriptora. U isti ulaz TAG MEMORIJE se upisuje polje *tag* generisane adrese. U isti ulaz *V* flip-flopova se upisuje 1, a u isti ulaz *D* flip-flopova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se stranica ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači datu stranicu sa diska na način prikazan u zadatku 1.2.1.

c) S obzirom da u adresnom prostoru korisnika ima 16 K stranica ($2^{14}$ stranica), da je broj korisnika 16 ($2^4$ korisnika), da se u jedinici za ubrzavanje preslikavanja mogu istovremeno čuvati deskriptori stranica različitih korisnika, da je broj setova u jedinici za ubrzavanje preslikavanja 1 K ($2^{10}$ setova) i da postoje dva ulaza po setu ($2^1$ ulaza), onda je $2^{14}$ stranica svih $2^4$ korisnika grupisano u $2^8$ grupa sa po $2^{10}$ stranica u grupi. Zaključuje se da je polje TAG dužine 8 bita i da je polje *set* dužine 10 bita kao što je prikazano na slici 1.2.12.




**Slika 1.2.12.** Jedinica za ubrzavanje preslikavanja virtuelnih u realne adrese za virtuelnu memoriju segmentno-stranične organizacije realizovana u tehnici set-asocijativnog preslikavanja

c1) Na osnovu zadatih karakteristika virtuelne memorije i jedinice za ubrzavanje preslikavanja dolazi se do strukture jedinice prikazane na istoj slici.

Jedinica za ubrzavanje preslikavanja se sastoji iz sledećih delova:

- TAG MEMORIJA ULAZA 0—RAM memorija kapaciteta 1024 reči širine 20 bita, koja se sastoji iz polja:
  - *D* (dirty bitovi) dužine 1 bit,
  - *V* (valid bitovi) dužine 1 bit,
  - *tag* dužine 8 bita,
  - *block* dužine 10 bita i
- TAG MEMORIJA ULAZA 1—RAM memorija kapaciteta 1024 reči širine 20 bita, koja se sastoji iz istih polja kao i TAG MEMORIJA ULAZA 0.
- $CMP_0$—komparator TAG MEMORIJE ULAZA 0.
- $CMP_1$—komparator TAG MEMORIJE ULAZA 1.

TAG MEMORIJA ULAZA 0 služi za čuvanje 1024 *block* polja deskriptora stranica zajedno sa odgovarajućim *V*, *D* i *tag* bitovima setova $SET_0$ do $SET_{1023}$ smeštenih u ulaze 0 jedinice za ubrzavanje preslikavanja.

*D* bitovi označavaju za svaki od 1024 seta ulaza 0 jedinice za ubrzavanje preslikavanja da li je bilo upisa u neku od lokacija date stranice.

*V* bitovi označavaju za svaki od 1024 seta ulaza 0 jedinice za ubrzavanje preslikavanja da li je odgovarajući ulaz TAG MEMORIJE ULAZA 0 važeći.

*tag* bitovi služe za čuvanje 1024 *tag* polja stranica čija se *block* polja deskriptora nalaze u odgovarajućim *block* poljima TAG MEMORIJE ULAZA 0.

*block* bitovi služe za čuvanje 1024 *block* polja deskriptora stranica koja se nalaze u ulazima 0 jedinice za ubrzavanje preslikavanja.

$CMP_0$ služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT_{i,0}}$ za ulaze 0 jedinice za ubrzavanje preslikavanja ukoliko:

- *tag* polje generisane adrese je isto kao sadržaj $tag_{i,0}$ polja TAG MEMORIJE ULAZA 0 adresiran poljem *set* generisane adrese i
- $V_{i,0}$ bit adresiran poljem *set* generisane adrese je postavljen.

TAG MEMORIJA ULAZA 1 služi za čuvanje 1024 *block* polja deskriptora stranica zajedno sa odgovarajućim *V*, *D* i *tag* bitovima setova $SET_0$ do $SET_{1023}$ smeštenih u ulaze 1 jedinice za ubrzavanje preslikavanja. Funkcije polja *V*, *D*, *tag* i *block* su iste kao u slučaju TAG MEMORIJE ULAZA 0.

$CMP_1$ služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT_{i,1}}$ za ulaze 1 jedinice za ubrzavanje preslikavanja na isti način kao signal saglasnosti $\mathbf{HIT_{i,0}}$.

c2) *set* bitovi (10) iz generisane adrese se koriste kao adresa za TAG MEMORIJU ULAZA 0 i TAG MEMORIJU ULAZA 1. $tag_{i,0}$ bitovi očitani iz TAG MEMORIJE ULAZA 0 se porede sa *tag* bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i da je $V_{i,0}$ bit, adresiran *set* bitovima generisane adrese, postavljen, signal saglasnosti $\mathbf{HIT_{i,0}}$ postaje aktivan. $tag_{i,1}$ bitovi očitani iz TAG MEMORIJE ULAZA 1 se porede sa *tag* bitovima iz generisane adrese. Ako se otkrije da postoji saglasnost i da je $V_{i,1}$ bit, adresiran *set* bitovima generisane adrese, postavljen, signal saglasnosti $\mathbf{HIT_{i,1}}$ postaje aktivan. Signal saglasnosti **HIT** je aktivan ukoliko je aktivan jedan od signala $\mathbf{HIT_{i,0}}$ i $\mathbf{HIT_{i,1}}$.

U formiranju 10 najstarijih bitova realne adrese učestvuju $block_{i,0}$ bitovi iz TAG MEMORIJE ULAZA 0 ukoliko je signal $\mathbf{HIT_{i,0}}$ aktivan, odnosno $block_{i,1}$ bitovi iz TAG

MEMORIJE ULAZA 1 ukoliko je signal $\mathbf{HIT}_{i,1}$ aktivan. Polje *word* iz virtuelne adrese daje 10 najmlađih bitova realne adrese.

Sve ove aktivnosti se realizuju hardverski.

Da bi se stranice različitih korisnika koje imaju isti broj preslikale svaka u svoj blok, u formiranju *tag* polja učestvuje ne samo bitovi 23 do 20 polja *page* generisane adrese, već i vrednost registra *user* procesora. Kod prebacivanja procesora sa korisnika na korianika u ovaj registar procesora se upisuje broj korisnika kome se dodeljuje procesor.

c3) Ukoliko se utvrdi da se relevantni deskriptor ne nalazi u hardveru, ide se hardverski u tabelu stranica i čita deskriptor na način prikazan u zadatku 1.2.1. Ukoliko se utvrdi da se stranica nalazi u operativnoj memoriji, polje *block* deskriptora se dovlači u hardver. Polje *set* generisane adrese određuje u koji od 1024 seta se smešta polje *block* deskriptora. Pošto za svaki set postoje ulazi 0 i 1, korišćenjem FIFO ili LRU algoritma zamene u okviru datog seta bira se ulaz jedinice za ubrzavanje preslikavanja u koji se smešta deskriptor. U polje *block* datog ulaza jedinice za ubrzavanje preslikavanja se upisuje polje *block* deskriptora. U isti ulaz se upisuje polje *tag* generisane adrese. U isti ulaz *V* bitova se upisuje 1, a u isti ulaz *D* bitova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se stranica ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači datu stranicu sa diska na način prikazan u zadatku 1.2.1.

## Zadatak 1.2.7

Virtuelne memorijske adrese su širine 16 bita, a adresiranje je na nivou 16-bitnih reči. Procesor operiše samo sa 16-bitnim celobrojnim veličinama (u daljem tekstu *reč* označava 16-bitnu veličinu). Procesor podržava postojanje više programa u memoriji istovremeno, do 16 korisnika. Fizički adresni prostor je veličine 4 GW (4 giga reči). Virtuelna memorija organizovana je na segmentnom principu, pri čemu je maksimalna veličina segmenta 4 KW (4 kilo reči). Procesor poseduje poseban hardver za ubrzavanje preslikavanja virtuelnih u realne (fizičke) adrese (u daljem tekstu TLB). TLB je realizovan sa direktnim preslikavanjem, pri čemu se svi segmenti sa istim brojem, različitih korisnika, preslikavaju u isti ulaz u TLB. Informacioni sadržaj jednog ulaza u TLB je sledeći: dužina segmenta, prava pristupa i početna adresa segmenta u fizičkoj memoriji. Prava pristupa se definišu pomoću tri bita R,W i E, pri čemu jedinica znači dozvoljeno čitanje podatka (R), upis podatka (W) i izvršavanje instrukcije (E). Procesor izvršava sledeći deo programa korisnika sa identifikatorom 3:

| **Aderesa (hex):** | **Naredba:** | | **Komentar:** |
|---|---|---|---|
| 3F00 | MOV | R0,#1234h | ; R0:=1234h |
| 3F02 | PUSH | R0 | ; R0 na stek |
| 3F03 | MOV | 2001h,R0 | ; Mem[2001h]:=R0 |

Segment programa je dužine 4 KW i smešten je počev od lokacije 200000h fizičke memorije, segment steka je dužine 1 KW i smešten je počev od lokacije 300000h fizičke memorije, a segment podataka je dozvoljen za upis i čitanje, dužine je 2 KW i smešten je od adrese 400000h fizičke memorije. SP ima vrednost 1215h pre izvršavanja datog dela programa.

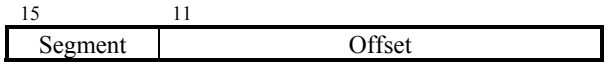
a) Prikazati logičku strukturu virtuelne adrese i označiti značenje i dužinu svakog polja.

b) Prikazati šta predstavlja ulazni podatak koji procesor daje TLB-u, šta je ključ za pretragu u TLB, i kako se dobija ulaz u TLB u kome se taj ključ traži.

c) Prikazati vrednosti ključeva i informacionog sadržaja relevantnih ulaza u TLB (označiti te ulaze), posle izvršavanja datog dela programa.

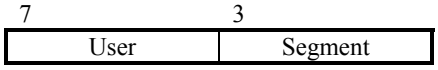
**Rešenje:**

a) Kod segmentne organizacije, virtuelna adresa ima dva polja: redni broj segmenta (engl. *segment*) i redni broj reči unutar segmenta (engl. *offset*). Kako je maksimalna veličina segmenta 4 KW, sledi da je polje redni broj reči unutar segmenta veličine 12 bita. Logička struktura virtuelne adrese je tako:




Fizička adresa se dobija sabiranjem adrese početka segmenta unutar fizičke memorije i rednog broja reči unutar segmenta.

b) TLB ima zadatak da ubrza preslikavanje rednog broja segmenta u početnu adresu segmenta u fizičkoj memoriji. Prema tome, ulazni podatak za TLB je redni broj segmenta, kao i identifikator korisnika za koga se preslikavanje vrši (čiji se program izvršava). Ulazni podatak za TLB je tako:




Kako je TLB organizovan sa direktnim preslikavanjem, ovaj ulazni podatak se preslikava uvek u tačno određeni ulaz u *Tag* memoriji TLB-a. Kako je u zadatku navedeno, ovo preslikavanje je takvo da se svi segmenti sa istim brojem, različitih korisnika, preslikavaju u isti ulaz. Zato je ključ za pretragu u TLB polje *User*, a nalazi se u ulazu sa rednim brojem jednakim polju *Segment*.

c) Redni broj segmenta određen je pomoću 4 najviša bita virtuelne adrese. Dati program koristi sledeće segmente:

| **Segment** | **Sadržaj** | **Operacije** |
|---|---|---|
| 3 | Program | E (u zavisnosti od realizacije, izvršavanje se može smatrati i čitanjem, R) |
| 1 | Stek | R, W |
| 2 | Podaci | R, W |

Sadržaj *Tag* memorije je identifikator korisnika, što je u ovom slučaju uvek 3. Prema tome, sadržaj relevantnih ulaza u TLB posle izvršavanja datog programa je sledeći:

| **Ključevi (*Tag* memorija)** | | **Informacioni sadržaj** | | |
|---|---|---|---|---|
| **Ulaz** | **Sadržaj** | **Dužina** | **RWE (bin)** | **Početna adresa (hex)** |
| 0 | | | | |
| 1 | 3 | 1K | 110 | 300000 |
| 2 | 3 | 2K | 110 | 400000 |
| 3 | 3 | 4K | ?01 | 200000 |

## Zadatak 1.2.8

Procesor podržava postojanje više programa u memoriji istovremeno, do 16 korisnika. Računar poseduje virtuelnu memoriju, kod koje 32-bitne adrese odgovaraju virtuelnom adresnom prostoru, a fizički adresni prostor je veličine 64 KW (64 kilo reči). Adresiranje je na nivou 32-bitnih reči. Virtuelna memorija organizovana je na straničnom principu, pri čemu je veličina stranice 256 W (256 reči). Procesor poseduje poseban hardver za ubrzavanje preslikavanja virtuelnih u realne (fizičke) adrese (u daljem tekstu TLB). TLB je realizovan sa direktnim preslikavanjem, pri čemu je veličina prostora za informacioni sadržaj TLB-a ukupno 64 KB (kilo bajta). Informacioni sadržaj je samo broj bloka u fizičkoj memoriji. Procesor izvršava sledeći deo programa korisnika sa identifikatorom 3:

| **Aderesa (hex):** | **Naredba:** | **Komentar:** |
|---|---|---|
| 32F00 | LOAD R0,#1234h | ; R0:=1234h |
| 32F02 | PUSH R0 | ; R0 na stek |

```
32F03                STORE  27001h,R0          ; Mem[27001h]:=R0
```

Slobodni deo operativne memorije obuhvata blokove počev od F0h zaključno sa FFh. Ovi blokovi popunjavaju se redom. Nijedna od stranica datog korisnika nije u operativnoj memoriji, a SP ima vrednost 11D15h pre izvršavanja datog dela programa.

a) Prikazati logičku strukturu virtuelne i fizičke adrese i dati značenje i dužinu svakog polja.

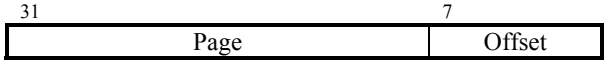
b) Prikazati šta predstavlja ulazni podatak koji procesor daje TLB-u, šta je ključ za pretragu u TLB, i kako se dobija ulaz u TLB u kome se taj ključ traži.

c) Prikazati vrednosti ključeva i informacionog sadržaja relevantnih ulaza u TLB (označiti te ulaze), posle izvršavanja datog dela programa.

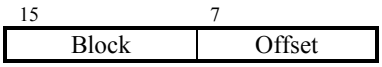
d) Koliko puta je generisan prekid tipa *Page Fault*?

**Rešenje:**

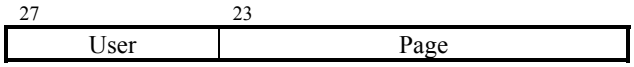
a) Kod stranične organizacije, virtuelna memorija podeljena je na stranice jednake dužine. Virtuelna adresa ima dva polja: redni broj stranice (engl. page) i redni broj reči unutar stranice (engl. offset). Kako je veličina stranice 256 W, sledi da je polje redni broj reči unutar stranice veličine 8 bita. Logička struktura virtuelne adrese je tako:




Fizička memorija podeljena je na blokove jednake veličine, pri čemu je veličina bloka jednaka veličini stranice. Fizička adresa sastoji se iz dva polja: redni broj bloka i redni broj reči unutar bloka (engl. offset). Redni broj reči unutar bloka jednak je rednom broju reči unutar stranice, pa se ova vrednost dobija prostim kopiranjem. Redni broj bloka dobija se od rednog broja stranice preslikavanjem. Kako je fizički adresni prostor kapaciteta 64 KW = 216 W, sledi da je logička struktura fizičke adrese sledeća:




b) TLB ima zadatak da ubrza preslikavanje rednog broja stranice u redni broj bloka. Prema tome, ulazni podatak za TLB je redni broj stranice, kao i identifikator korisnika za koga se preslikavanje vrši (čiji se program izvršava). Ulazni podatak za TLB je tako:




Kako je TLB organizovan sa direktnim preslikavanjem, ovaj ulazni podatak se preslikava uvek u tačno određeni ulaz u Tag memoriji TLB-a. Kapacitet TLB-a je 64 KB, a jedan ulaz u TLB sadrži kao informaciju broj bloka, što je veličine 8 bita (1 bajt). Prema tome, kapacitet TLB-a je 64 K ulaza. Tako je broj ulaza u TLB određen pomoću 16 najnižih bita ulaznog podatka (to su 16 najnižih bita polja Page), dok je ključ za pretragu viših 12 bita ulaznog podatka:

Ključ za pretragu (Tag): User(4):Page(23…16)

Ulaz u TLB: Page(15…0)

c) Dati program korisnika sa identifikatorom 3 koristi redom sledeće stranice koje se preslikavaju u date ulaze u TLB:

| **Stranica (hex):** | **Ulaz u TLB (hex):** | **Objašnjenje:** |
|---|---|---|

| | | |
|---|---|---|
| 32F | 32F | Program |
| 11D | 11D | Stek |
| 270 | 270 | Podaci |

Za ove stranice se, po datom redosledu, rezervišu slobodni blokovi F0h, F1h i F2h, redom. Stanje TLB-a je na kraju:

| **Ulaz** | ***Tag*** | ***Block*** |
|---|---|---|
| 11D | 300 | F1 |
| 270 | 300 | F2 |
| 32F | 300 | F0 |

d) Kako su pri ovom izvršavanju referisane ukupno 3 stranice, a pre izvršavanja one nisu bile u operativnoj memoriji, sve tri stranice (i samo one) morale su biti dovučene sa diska u memoriju. Prekid tipa *Page Fault* je generisan zato 3 puta.

## Zadatak 1.2.9

Procesor podržava virtuelnu memoriju, i poseduje hardver za ubrzavanje preslikavanja virtuelnih u realne adrese (u daljem tekstu TLB). Pri tom je virtuelna adresa širine 16 bita, a realna adresa 32 bita, uz podršku do 4 korisnika. Virtuelna memorija organizovana je po segmentno-straničnom principu, sa maksimalno 16 segmenata po korisniku i 64 stranice po segmentu. TLB je organizovan asocijativno, sa 4 ulaza i LRU algoritmom zamene.

a) Prikazati logičku strukturu virtuelne adrese i označiti značenje i dužinu svakog polja.

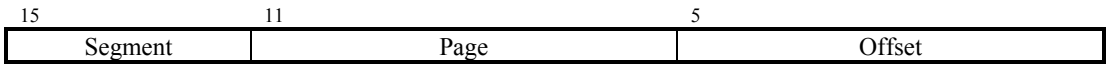
b) Prikazati logičku strukturu realne adrese i označiti značenje i dužinu svakog polja.

c) Prikazati sadržaj asocijativne memorije TLB-a posle sekvence zahteva za čitanjem sa sledećih adresa korisnika sa identifikatorom 3 (TLB je na početku bio prazan):

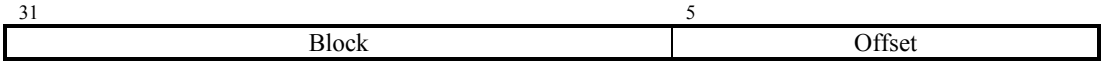
4632h, 5C5Fh, BE8Eh, 6237h, 5C5Ah, 463Fh, FCFAh.

**Rešenje:**

a) Kod segmentno-stranične organizacije, virtuelni adresni prostor podeljen je na logičke celine—segmente različite veličine, dok su segmenti podeljeni na stranice konstantne veličine. Virtuelna adresa sadrži tri polja: redni broj segmenta (engl. *segment*), redni broj stranice unutar segmenta (engl. *page*) i redni broj reči unutar stranice (engl. *offset*). Prema podacima iz zadatka, logička struktura virtuelne adrese je:




b) Fizički adresni prostor podeljen je na blokove jednake veličine, pri čemu je veličina bloka jednaka veličini stranice. Fizička adresa sadrži dva polja: redni broj bloka i redni broj reči unutar bloka. Prema podacima iz zadatka, logička struktura fizičke adrese je:




Redni broj reči unutar bloka jednak je rednom broju reči unutar stranice, pa se ova vrednost dobija prostim kopiranjem. Redni broj bloka dobija se od rednog broja segmenta i stranice unutar segmenta preslikavanjem.

c) Ulazni podatak u TLB čine identifikator korisnika (2 bita), redni broj segmenta (4 bita) i redni broj stranice unutar segmenta (6 bita). TLB daje na izlazu broj bloka u fizičkoj memoriji (26 bita). TLB je organizovan asocijativno, pa cela ulazna informacija (12) čini ključ za asocijativnu pretragu (Tag) unutar TLB-a.

Identifikator korisnika je u ovom slučaju uvek 3. Sledeća tabela prikazuje vrednosti Tag polja koja se traže unutar TLB za virtuelne adrese iz date sekvence.

| Adresa (hex) | User (bin) | Segment (bin) | Page (bin) | Tag (hex) |
|---|---|---|---|---|
| 4632 | 11 | 0100 | 0110 00 | D18 |
| 5C5F | 11 | 0101 | 1100 01 | D71 |
| BE8E | 11 | 1011 | 1110 10 | EFA |
| 6237 | 11 | 0110 | 0010 00 | D88 |
| 5C5A | 11 | 0101 | 1100 01 | D71 |
| 463F | 11 | 0100 | 0110 00 | D18 |
| FCFA | 11 | 1111 | 1100 11 | FF3 |

Prema tome, sekvenca koja se zadaje TLB-u je: D18h, D71h, EFAh, D88h, D71h, D18h, FF3h. Ove vrednosti TLB smešta u asocijativnu memoriju po LRU algoritmu zamene (videti poglavlje Keš memorije). Sadržaj asocijativnog dela je na kraju:

| | |
|---|---|
| 0 | D18h |
| 1 | D71h |
| 2 | FF3h |
| 3 | D88h |

## Zadatak 1.2.10

(Virtuelna memorija, AOR Okt 96):

Procesor podržava virtuelnu organizaciju memorije sa straničnim preslikavanjem. Stranice su dužine $2^{16}$ reči. Kapacitet fiuzičke memorije je najviše $2^{15}$ blokova. Memorijske (fizičke) adrese su širine 31 bit, širina magistrale podataka je 16 bita, a adresiranje je na nivou 16-bitnih reči. Korisničke logičke (virtuelne) adrese su 24-bitne, a procesor podržava rad 256 korisničkih programa.

Procesor poseduje i hardver za ubrzavanje preslikavanja, u daljem tekstu TLB. TLB je organizovan sa set-asocijativnim preslikavanjem, poseduje dva ulaza po setu i 64 seta. Algoritam zamene je FIFO. Korisnički program sa brojem D0h generiše sledeću sekvencu virtuelnih adresa (sve vrednosti su heksadecimalne):

278800, A700FF, 278801, E78800, B600FF, E78802, 278802, B600FF

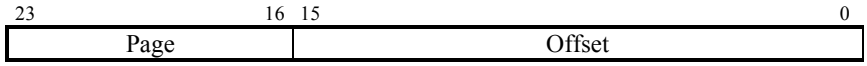
a) Prikazati logičku strukturu virtuelne adrese i označiti značenje i dužinu svakog polja.

b) Prikazati šta predstavlja ulazni podatak koji procesor daje TLB-u, šta je ključ za pretragu u TLB, i kako se dobija set u TLB u kome se taj ključ traži.

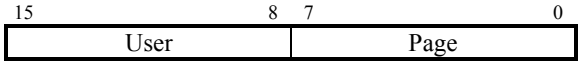
c) Prikazati vrednosti ključeva relevantnih setova u TLB (označiti te setove), posle zadavanja date sekvence adresa.

**Rešenje:**

a) Virtuelna adresa ima sledeću strukturu:




b) Ulazni podatak za TLB ima strukturu:




Ključ za pretragu je polje *User*(7:0):*Page*(7:6), a nalazi se u setu sa rednim brojem jednakim polju *Page*(5:0).

c) Sadržaj je sledeći (sve vrednosti su heksadecimalne):

| | Set | |
|---|---|---|
| **Ulaz** | 27 | 36 |
| **0** | 343 | 342 |
| **1** | 340 | |