# АРХИТЕКТУРА И ОРГАНИЗАЦИЈА РАЧУНАРА 1

**Верзија 2014 1.0**

# САДРЖАЈ

Садржај ..... 3
Кеш меморија (Cache Memory) ..... 5
Задатак 1. ..... 5
Задатак 2. ..... 6
Задатак 3. ..... 9
Задатак 4. ..... 12
Задатак 5. ..... 15
Задатак 6. ..... 19
Задатак 7. ..... 20
Задатак 8. ..... 22
Задатак 9. ..... 24
Задатак 10. ..... 27
Задатак 11. ..... 29
Задатак 12. ..... 31
Задатак 13. ..... 32
Задатак 14. ..... 35
Задатак 15. ..... 37
Задатак 16. ..... 39
Задатак 17. ..... 40
Задатак 18. ..... 41
Виртуелна меморија (Virtual Memory) ..... 43
Задатак 1. ..... 43
Задатак 2. ..... 44
Задатак 3. ..... 47
Задатак 4. ..... 48
Задатак 5. ..... 50
Задатак 6. ..... 52
Задатак 7. ..... 53
Задатак 8. ..... 55
Задатак 9. ..... 57
Преклапање приступа меморијским модулима (Memory Inteleaving) ..... 60
Задатак 1. ..... 60
Задатак 2. ..... 62
Задатак 3. ..... 63
Задатак 4. ..... 64
Задатак 5. ..... 64
Задатак 6. ..... 65
Задатак 7. ..... 65
Задатак 8. ..... 67
Задатак 9. ..... 69
Проточна обрада (Pipeline) ..... 72
Задатак 1. ..... 72
Задатак 2. ..... 73
Задатак 3. ..... 76
Задатак 4. ..... 78
Задатак 5. ..... 78
Задатак 6. ..... 79
Задатак 7. ..... 79
Задатак 8. ..... 80

Задатак 9. ..... 81
Задатак 10. ..... 81
Задатак 11. ..... 82
Задатак 12. ..... 83
Задатак 13. ..... 83
Задатак 14. ..... 84
Задатак 15. ..... 84
Задатак 16. ..... 87
Задатак 17. ..... 88
Задатак 18. ..... 90
Задатак 19. ..... 91
Задатак 20. ..... 92
Задатак 21. ..... 94
Задатак 22. ..... 96
Задатак 23. ..... 96
Задатак 24. ..... 97
Задатак 25. ..... 98
Задатак 26. ..... 99
Задатак 27. ..... 99

# КЕШ МЕМОРИЈА (CACHE MEMORY)

## Задатак 1.

Оперативна меморија рачунара је капацитета 1 MB, а ширина речи износи 1 бајт.

а) Нацртати структуру кеш меморије, означити ширину у битовима свих релевантних делова и укратко описати њихову намену за кеш меморију са 1024 блока, блоком величине четири бајта и асоцијативним пресликавањем претпостављајући да је ширина речи дела кеш меморије у коме се чува садржај један бајт.

б) Садржај релевантних локација оперативне меморије је дат на слици. Четири захтева за операцијама читања из оперативне меморије са локација 1402h, 401h, 2000h и 1003h се изводе у датом редоследу. За сваки захтев у посебној врсти табеле попунити колоне означене са: Адреса – генерисана адреса, Тип - тип приступа (Rd – читање, Wr – упис), Tag - вредност поља Tag генерисане адресе, Word - вредност поља Word генерисане адресе, Време - време потребно за обављање приступа и Адресе - адресе локација оперативне меморије којима се приступа у току извршења датог захтева. Приликом израчунавања времена потребног за обављање приступа узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SA}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности. Навести вредности свих релевантних делова кеш меморије после ове четири операције, претпостављајући да је кеш меморија на почетку била празна.

| **Адреса** | 400 | 401 | 402 | 403 | … | 1000 | 1001 | 1002 | 1003 | … | 1400 | 1401 | 1402 | 1403 | … | 2000 | 2001 | 2002 | 2003 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Садржај** | 0 | 1 | 0 | 0 | … | 0 | 0 | 0 | 2 | … | 0 | 0 | 3 | 0 | … | 4 | 0 | 0 | 0 |

Садржај релевантних локација оперативне меморије.

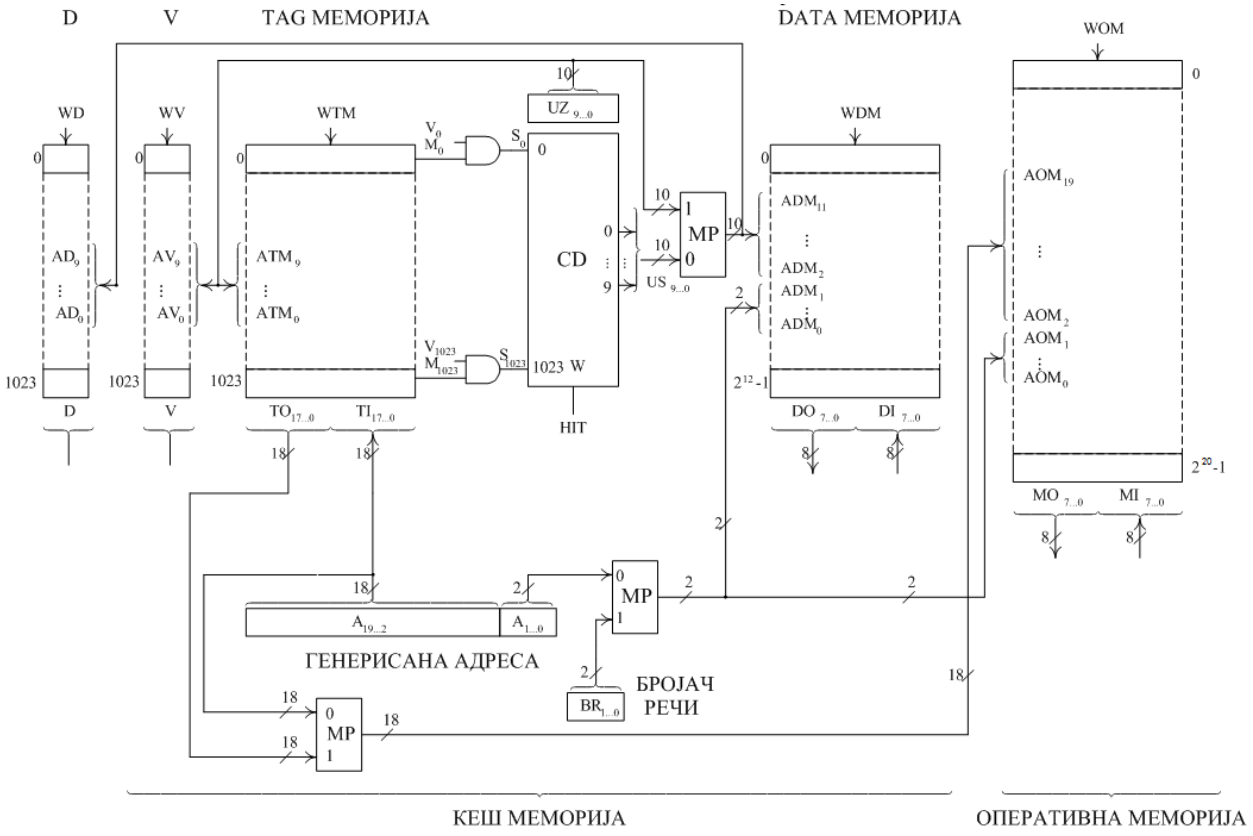
в) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморију и обрнуто, па се тек онда приступа локацији, и да се све операције раде секвенцијално. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SA}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

Решење:

Структура адреса код кеш меморије:

<table>
<tr><td>Број блока</td><td rowspan="2">Word (l бита)</td></tr>
<tr><td>Tag (m бита)</td></tr>
</table>

а) Структура кеш меморије приказана је на слици.




Структура кеш меморије са асоцијативним пресликавањем

б) Генерисане адресе су: 1402h, 401h, 2000h и 1003h. Вредности свих релевантних делова кеш меморије после четири операције читања приказани су на слици.

| Адреса | Тип | Tag | Word | Време | Адресе |
|---|---|---|---|---|---|
| 1402h | Rd | 000000010100000000 | 10 | $t_{SA}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 1400h-1403h |
| 401h | Rd | 000000000100000000 | 01 | $t_{SA}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 400h-403h |
| 2000h | Rd | 000000100000000000 | 00 | $t_{SA}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 2000h-2003h |
| 1003h | Rd | 000000010000000000 | 11 | $t_{SA}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 1000h-1003h |

| број улаза | D | V | TAG МЕМОРИЈА |
|---|---|---|---|
| 0 | 0 | 1 | 000000010100000000 |
| 1 | 0 | 1 | 000000000100000000 |
| 2 | 0 | 1 | 000000100000000000 |
| 3 | 0 | 1 | 000000010000000000 |
| 4 | 0 | 0 | |
| ⋮ | ⋮ | ⋮ | |
| 1023 | 0 | 0 | |

| адреса | DATA МЕМОРИЈА | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 3 | 0 |
| 4 | 0 | 1 | 0 | 0 |
| 8 | 4 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 2 |
| | | | | |
| ⋮ | | | | |
| 4092 | | | | |

Садржај релевантних делова кеш меморије

в) Време дохватања блока је:

$t_B=4\cdot(t_{OM}+t_{DM})$.

Укупно време је:

$t=4\cdot(t_{SA}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM})=20\cdot t_{DM}+16\cdot t_{OM}+8\cdot t_{SA}+4\cdot t_{TM}$.

## Задатак 2.

Оперативна меморија рачунара је капацитета 1 MB, а ширина речи износи 1 бајт.

а) Нацртати структуру кеш меморије, означити ширину у битовима за све релевантне делове и укратко описати њихову намену за кеш меморију са 1024 блока, блоком величине четири бајта и

директним пресликавањем претпостављајући да је ширина речи дела кеш меморије у коме се налази садржај један бајт.

б) Садржај релевантних локација оперативне меморије је дат на слици. Четири захтева за операцијама читања из оперативне меморије са локација 1402h, 401h, 2000h и 1003h се изводе у датом редоследу. За сваки захтев у посебној врсти табеле попунити колоне означене са: Адреса – генерисана адреса, Тип - тип приступа (Rd – читање, Wr – упис), Tag - вредност поља Tag генерисане адресе, Block - вредност поља Block генерисане адресе, Word - вредност поља Word генерисане адресе, Време - време потребно за обављање приступа и Адресе - адресе локација оперативне меморије којима се приступа у току извршења датог захтева. Приликом израчунавања времена потребног за обављање приступа узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SD}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности. Навести вредности свих релевантних делова кеш меморије после ове четири операције, претпостављајући да је кеш меморија на почетку била празна.

| **Адреса** | 400 | 401 | 402 | 403 | … | 1000 | 1001 | 1002 | 1003 | … | 1400 | 1401 | 1402 | 1403 | … | 2000 | 2001 | 2002 | 2003 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Садржај** | 0 | 1 | 0 | 0 | … | 0 | 0 | 0 | 2 | … | 0 | 0 | 3 | 0 | … | 4 | 0 | 0 | 0 |

Садржај релевантних локација оперативне меморије.

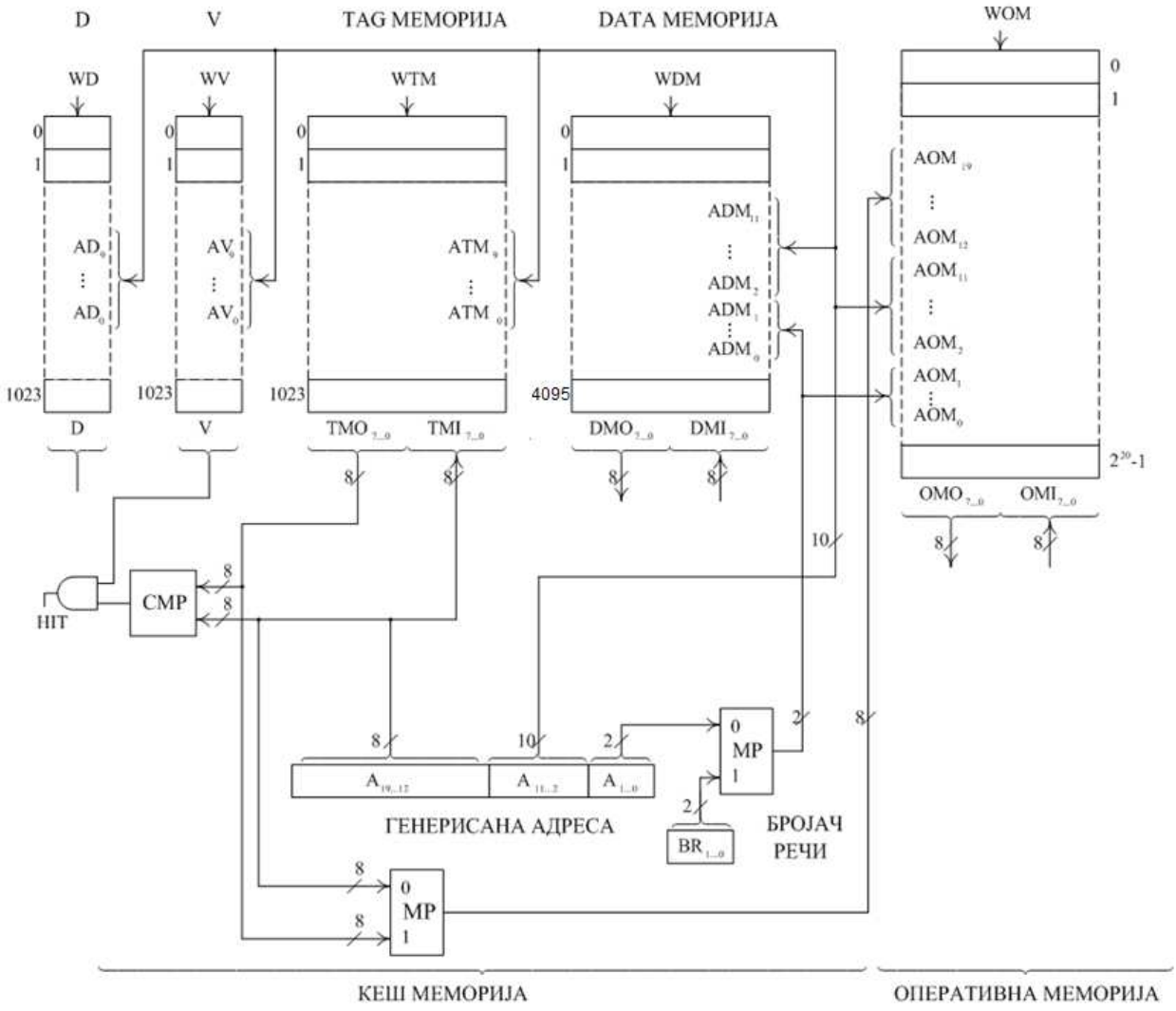
в) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморију и обрнуто, па се тек онда приступа локацији, и да се све операције раде секвенцијално. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SD}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

Решење:

Структура адреса код кеш меморије:

| Број блока | | Word (l бита) |
|---|---|---|
| Tag (m бита) | Block (k бита) | |

а) Структура кеш меморије је приказана на слици.




Структура кеш меморије са директним пресликавањем

б) Генерисане адресе су: 1402h, 401h, 2000h и 1003h. Вредности свих релевантних делова кеш меморије после четири операције читања приказани су на слици.

| Адреса | Тип | Tag | Block | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| 1402h | Rd | 00000001 | 0100000000 | 10 | $t_{SD}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SD}$ | 1400h-1403h |
| 401h | Rd | 00000000 | 0100000000 | 01 | $t_{SD}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SD}$ | 400h-403h |
| 2000h | Rd | 00000010 | 0000000000 | 00 | $t_{SD}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SD}$ | 2000h-2003h |
| 1003h | Rd | 00000001 | 0000000000 | 11 | $t_{SD}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SD}$ | 1000h-1003h |

| број улаза | D | V | TAG МЕМОРИЈА | адреса | DATA МЕМОРИЈА |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 00000001 | 0 | 40 00 00 02 |
| 1 | 0 | 0 | | 4 | |
| ⋮ | ⋮ | ⋮ | | ⋮ | |
| 256 | 0 | 1 | 00000010 | 1024 | 00 01 30 00 |
| ⋮ | ⋮ | ⋮ | | ⋮ | |
| 1023 | 0 | 0 | | 4092 | |

Садржај релевантних делова кеш меморије

в) Време дохватања блока је:

$t_B=4\cdot(t_{OM}+t_{DM})$.

Укупно време је:

$t=4\cdot(t_{SD}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SD})=16\cdot t_{DM}+16\cdot t_{OM}+8\cdot t_{SD}+4\cdot t_{TM}$.

## Задатак 3.

Оперативна меморија рачунара је капацитета 1 MB, а ширина речи износи 1 бајт.

а) Нацртати структуру кеш меморије, означити ширину у битовима за све релевантне делове и укратко описати њихову намену за:

а1) кеш меморију са 1024 блока, блоком величине четири бајта и сет-асоцијативним пресликавањем са два блока по сету

а2) кеш меморију са 1024 блока, блоком величине четири бајта и сет-асоцијативним пресликавањем са четири блока по сету,

претпостављајући да је ширина речи дела кеш меморије у коме се налази садржај један бајт

б) Садржај релевантних локација оперативне меморије је дат на слици. Четири захтева за операцијама читања из оперативне меморије са локација 1402h, 401h, 2000h и 1003h се изводе у датом редоследу. За сваки захтев у посебној врсти табеле попунити колоне означене са: Адреса – генерисана адреса, Тип - тип приступа (Rd – читање, Wr – упис), Tag - вредност поља Tag генерисане адресе, Set - вредност поља Set генерисане адресе, Word - вредност поља Word генерисане адресе, Време - време потребно да се обављање приступа и Адресе - адресе локација оперативне меморије којима је приступа у току извршења датог захтева. Навести вредности свих релевантних делова кеш меморије после ове четири операције, претпостављајући да је кеш меморија на почетку била празна.

| **Адреса** | 400 | 401 | 402 | 403 | … | 1000 | 1001 | 1002 | 1003 | … | 1400 | 1401 | 1402 | 1403 | … | 2000 | 2001 | 2002 | 2003 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Садржај** | 0 | 1 | 0 | 0 | … | 0 | 0 | 0 | 2 | … | 0 | 0 | 3 | 0 | … | 4 | 0 | 0 | 0 |

Садржај релевантних локација оперативне меморије.

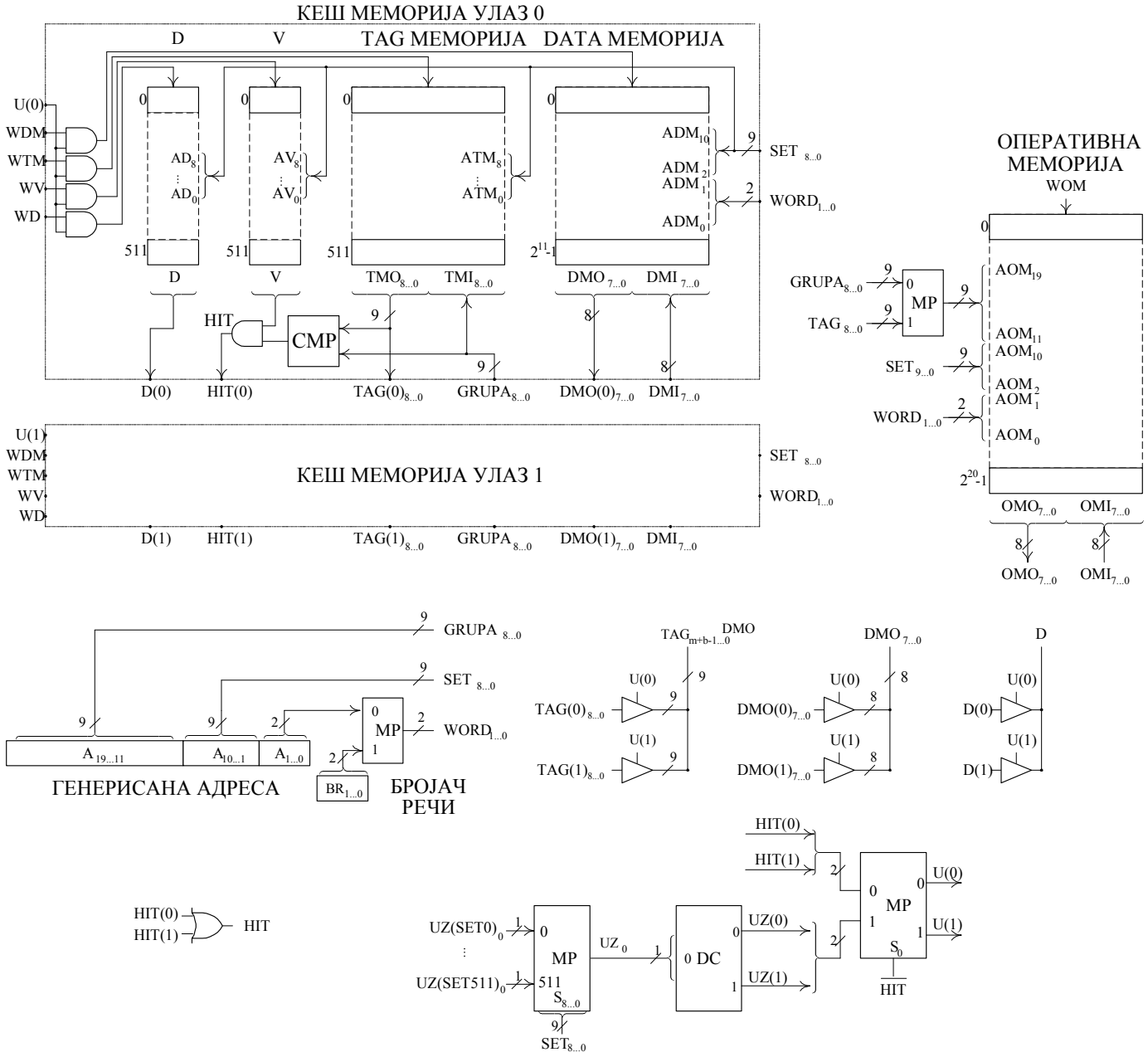
в) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији, и да се све операције раде секвенцијално. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SS}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

Решење:

Структура адреса код кеш меморије:

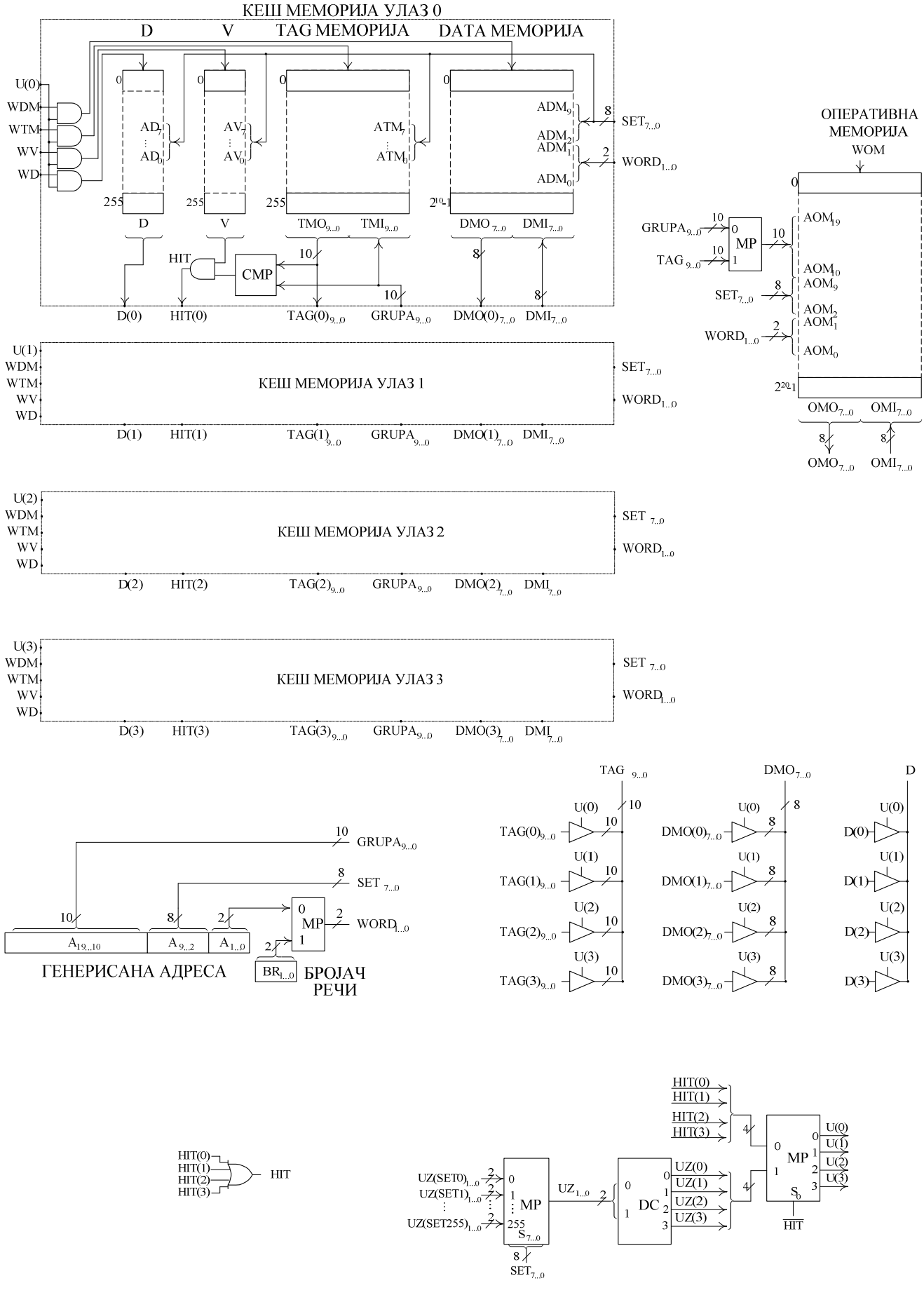
| Број блока | | Word (l бита) |
|---|---|---|
| Tag (m бита) | Set (k бита) | |

а1) Структуре кеш меморије је приказана на слици.




Структура кеш меморије са сет-асоцијативним пресликавањем са два блока по сету

а2) Структуре кеш меморије је приказана на слици.




Структура кеш меморије са сет-асоцијативним пресликавањем са четири блока по сету

б1) Сет-асоцијативно пресликавање са 2 блока по сету.

Вредности свих релевантних делова кеш меморије после четири операције читања приказани су на слици.

| Адреса | Тип | Tag | Set | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| 1402h | Rd | 000000010 | 100000000 | 10 | $t_{SS}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 1400h-1403h |
| 401h | Rd | 000000000 | 100000000 | 01 | $t_{SS}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 400h-403h |
| 2000h | Rd | 000000100 | 000000000 | 00 | $t_{SS}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 2000h-2003h |
| 1003h | Rd | 000000010 | 000000000 | 11 | $t_{SS}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 1000h-1003h |

| број сета | број улаза | D | V | TAG МЕМОРИЈА | адреса | DATA МЕМОРИЈА |
|---|---|---|---|---|---|---|
| $SET_0$ | 0 | 0 | 1 | 000000100 | 0 | 4 0 0 0 |
| | 1 | 0 | 1 | 000000010 | 4 | 0 0 0 2 |
| | ⋮ | | | ⋮ | ⋮ | |
| $SET_{256}$ | 0 | 0 | 1 | 000000010 | 2048 | 0 0 3 0 |
| | 1 | 0 | 1 | 000000000 | 2052 | 0 1 0 0 |
| | ⋮ | | | ⋮ | ⋮ | |
| $SET_{511}$ | 0 | 0 | 0 | | 4088 | |
| | 1 | 0 | 0 | | 4092 | |

Садржај релевантних делова кеш меморије.

б2) Сет-асоцијативно пресликавање са 4 блока по сету

Вредности свих релевантних делова кеш меморије после четири операције читања приказани су на слици.

| Адреса | Тип | Tag | Set | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| 1402h | Rd | 0000000101 | 00000000 | 10 | $t_{SS}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 1400h-1403h |
| 401h | Rd | 0000000001 | 00000000 | 01 | $t_{SS}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 400h-403h |
| 2000h | Rd | 0000001000 | 00000000 | 00 | $t_{SS}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 2000h-2003h |
| 1003h | Rd | 0000000100 | 00000000 | 11 | $t_{SS}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 1000h-1003h |

| број сета | број улаза | D | V | TAG МЕМОРИЈА | адреса | DATA МЕМОРИЈА |
|---|---|---|---|---|---|---|
| $SET_0$ | 0 | 0 | 1 | 0000000101 | 0 | 0 0 3 0 |
| | 1 | 0 | 1 | 0000000001 | 4 | 0 1 0 0 |
| | 2 | 0 | 1 | 0000001000 | 8 | 4 0 0 0 |
| | 3 | 0 | 1 | 0000000100 | 12 | 0 0 0 2 |
| | ⋮ | | | ⋮ | ⋮ | |
| $SET_{256}$ | 0 | 0 | 0 | | 4080 | |
| | 1 | 0 | 0 | | 4084 | |
| | 2 | 0 | 0 | | 4088 | |
| | 3 | 0 | 0 | | 4092 | |

Садржај релевантних делова кеш меморије

в1) Време дохватања блока је:

$t_B=4\cdot(t_{OM}+t_{DM})$.

Укупно време је:

$t=4\cdot(t_{SS}+4\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS})=16\cdot t_{DM}+16\cdot t_{OM}+8\cdot t_{SS}+4\cdot t_{TM.}$

в2) Исто као и в1).

## Задатак 4.

Оперативна меморија рачунара је капацитета 512 MB, а ширина речи износи 16 бита.

а) Нацртати структуру кеш меморије, означити ширину у битима свих релевантних делова и кратко објаснити функције свих делова кеш меморије за следеће случајеве:

кеш меморије са 64 K блокова, величином блока 256 речи и асоцијативним пресликавањем,

кеш меморије са 64 K блокова, величином блока 256 речи и директним пресликавањем,

кеш меморије са 64 K блокова, величином блока 256 речи и сет-асоцијативним пресликавањем, са 16 блокова по сету,

претпостављајући да је ширина речи дела кеш меморије у коме се чувају садржаји 16 бита.

б) Садржај релевантних локација оперативне меморије је дат на слици, при чему су све вредности приказане хексадецимално. Узети да се, најпре, реализују две операције читања са адреса 0A0FF00h и 0A0FFFFh, затим две операције уписа вредности 0003h и 0004h у локације на адресама 0DE1000h и 0DE10FFh, респективно и на крају две операције читања са адреса 0807500h и 08075FFh. Операције читања и уписа се искључиво реализују над садржајима кеш меморије. Уколико се у случају горњих операција читања и уписа утврди да нема сагласности довлачи се одговарајући блок садржаја из оперативне у кеш меморију. Блокови кеш меморије чији је садржај модификован враћају се у оперативну меморију.

⋮

| **адреса** | … | 0807500 | 0807501 | … | 08075FE | 08075FF | … |
|---|---|---|---|---|---|---|---|
| **садржај** | … | 0 | 0 | … | 0 | 0 | … |

⋮

| **адреса** | … | 0A0FF00 | 0A0FF01 | … | 0A0FFFE | 0A0FFFF | … |
|---|---|---|---|---|---|---|---|
| **садржај** | … | 1 | 1 | … | 1 | 1 | … |

⋮

| **адреса** | … | 0DE1000 | 0DE1001 | … | 0DE10FE | 0DE10FF | … |
|---|---|---|---|---|---|---|---|
| **садржај** | … | 2 | 2 | … | 2 | 2 | … |

⋮

Садржај релевантних локација оперативне меморије

За све случајеве кеш меморије, дати садржаје релевантних делова кеш меморије после извршења наведених операција, претпостављајући да је кеш меморија на почетку била празна.

в) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији, и да се све операције раде секвенцијално. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ код кеш меморије са асоцијатиним пресликавањем износи ($t_{SA}$), код кеш меморије са директним пресликавањем износи ($t_{SD}$), а код кеш меморије са сет-асоцијатиним пресликавањем износи ($t_{SS}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

Решење:

а1) Структура адреса код кеш меморије:

| Број блока | Word (8 бита) |
|---|---|
| Tag (20 бита) | |

а2) Структура адреса код кеш меморије:

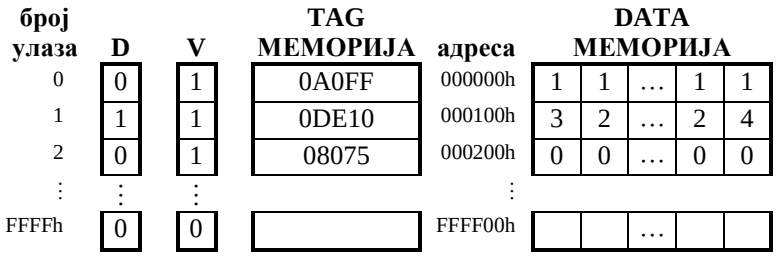
| Број блока | | Word (8 бита) |
|---|---|---|
| Tag (4 бита) | Block (16 бита) | |

а3) Структура адреса код кеш меморије:

| Број блока | | Word (8 бита) |
|---|---|---|
| Tag (8 бита) | Set (12 бита) | |

б1) Изглед релевантних делова кеш меморије је дат на слици.

| Адреса | Тип | Tag | Word | Време | Адресе |
|---|---|---|---|---|---|
| 0A0FF00h | Rd | 0000101000001111111 | 00000000 | $t_{SA}+n\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | A0FF00h-A0FFFFh |
| 0A0FFFFh | Rd | 0000101000001111111 | 11111111 | $t_{SA}+t_{DM}$ | - |
| 0DE1000h | Wr | 0000110111000010000 | 00000000 | $t_{SA}+n\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | DE1000h-DE10FFh |
| 0DE10FFh | Wr | 0000110111000010000 | 11111111 | $t_{SA}+t_{DM}$ | - |
| 0807500h | Rd | 0000100000000001110101 | 00000000 | $t_{SA}+n\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 807500h-8075FFh |
| 08075FFh | Rd | 0000100000000001110101 | 11111111 | $t_{SA}+t_{DM}$ | - |



Садржај релевантних делова кеш меморије

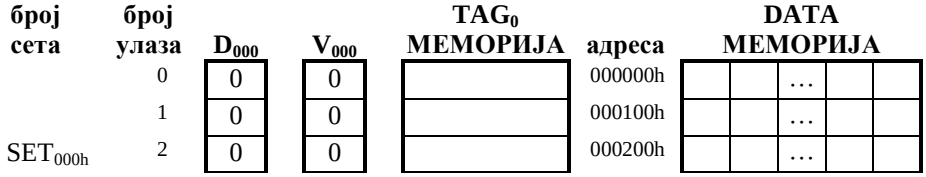
б2) Изглед релевантних делова кеш меморије је дат на слици:

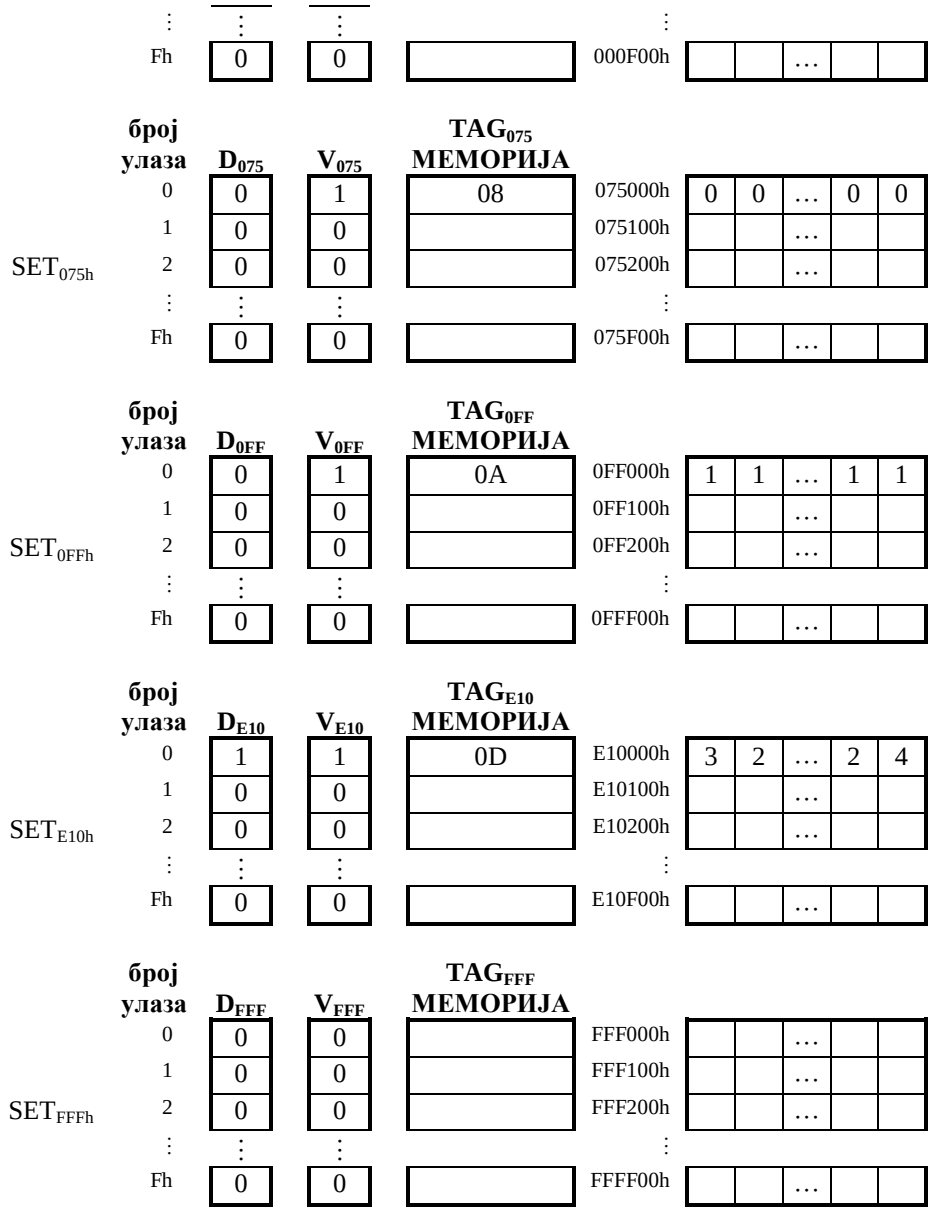
| Адреса | Тип | Tag | Block | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| 0A0FF00 | Rd | 0000 | 1010000011111111 | 00000000 | $t_{SD}+n\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SD}$ | A0FF00h-A0FFFFh |
| 0A0FFFF | Rd | 0000 | 1010000011111111 | 11111111 | $t_{SD}$ | - |
| 0DF1000 | Wr | 0000 | 1101111000010000 | 00000000 | $t_{SD}+n\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SD}+t_{DM}$ | DE1000h-DE10FFh |
| 0DE10FF | Wr | 0000 | 1101111000010000 | 11111111 | $t_{SD}+t_{DM}$ | - |
| 0807500 | Rd | 0000 | 1000000001110101 | 00000000 | $t_{SD}+n\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SD}$ | 807500h-8075FFh |
| 08075FF | Rd | 0000 | 1000000001110101 | 11111111 | $t_{SD}$ | - |

Садржај релевантних делова кеш меморије

б3) Изглед релевантних делова кеш меморије је дат на слици.

| Адреса | Тип | Tag | Block | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| 0A0FF00 | Rd | 00001010 | 000011111111 | 00000000 | $t_{SS}+n\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | A0FF00h-A0FFFFh |
| 0A0FFFF | Rd | 00001010 | 000011111111 | 11111111 | $t_{SS}$ | - |
| 0DF1000 | Wr | 00001101 | 111000010000 | 00000000 | $t_{SS}+n\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}+t_{DM}$ | DE1000h-DE10FFh |
| 0DE10FF | Wr | 00001101 | 111000010000 | 11111111 | $t_{SS}+t_{DM}$ | - |
| 0807500 | Rd | 00001000 | 000001110101 | 00000000 | $t_{SS}+n\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 807500h-8075FFh |
| 08075FF | Rd | 00001000 | 000001110101 | 11111111 | $t_{SS}$ | - |






Садржај релевантних делова кеш меморије

в) Време дохватања блока је:

$t_B$=256·($t_{OM}$+$t_{DM}$).

## Задатак 5.

Оперативна меморија рачунара има капацитет 2 MB, ширина речи износи 16 бита и време приступа меморије износи $T_m$. Процесор има раздвојене кеш меморије за инструкције и податке. Инструкцијска кеш меморија има капацитет DATA дела 4 KB, пресликавање је на нивоу блока, величина блока је 128 бајта, ширина речи је 16 бита. Претпоставити да су све инструкције ширине 16 битова, и да се адресе на које указује регистар PC односе на 16-битне величине.

а) Нацртати структуру свих релевантних делова кеш меморије са:

директним пресликавањем,

асоцијативним пресликавањем, ако је алгоритам замене FIFO и

сет-асоцијативним пресликавањем са два блока по сету, ако је алгоритам замене FIFO,

и укратко описати функцију сваког дела. Нацртати структуру адресе коју генерише процесор и објаснити како се користе групе битова у сва три случаја да се:

провери да ли је тражена инструкција у кеш меморији и

да се очита инструкција из кеш меморије.

б) Израчунати укупно време потребно за читање инструкција из кеш меморије, за сва три типа кеш меморије, када се извршава следећи програм:

10

□□□

200 ←

□□□

2555

□□□

2655

Прва очитана инструкција се налази на адреси 10, а последња на адреси 2655. Петља се изврши два пута. Кеш меморија је била празна на почетку. Када се рачуна време: треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије, па се тек онда инструкција очита. Узети у обзир само време потребно за приступ кеш меморији за проверу сагласности и приступ податку $T_k$, као и време потребно за пребацивање податка из оперативне меморије у кеш меморију или процесор и обрнуто је $T_m$, занемарити времена потребна за остале активности.

Решење:

а1) Структура адреса код кеш меморије:

| Број блока | | Word (6 бита) |
|---|---|---|
| Tag (9 бита) | Block (5 бита) | |

а2) Структура адреса код кеш меморије:

| Број блока | Word (6 бита) |
|---|---|
| Tag (14 бита) | |

а3) Структура адреса код кеш меморије:

| Број блока | | Word (6 бита) |
|---|---|---|
| Tag (10 бита) | Set (4 бита) | |

б)Адресе којима се приступа: 0000Ah, …, 009FBh, 000C8h, …, 00A5Fh;

б1) Начин смештања и садржаји сваког улаза Таг меморије приказани су на слици.

| **улаз** | **директно** | | | |
|---|---|---|---|---|
| 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 | | | |

Садржаји свих улаза Таг меморије у случају директног пресликавања

Време дохватања блока је:

$t_B=64\cdot(t_{DM}+t_{OM})$.

Укупно време дохватања свих блокова је:

$t_{Ball}=32\ t_B+8\ t_B+5\ t_B+7\ t_B=52\ t_B$.

Укупно време је:

$t=3328\cdot t_{DM}+3328\cdot t_{OM}+5054\cdot t_{SD}+52\cdot t_{TM}$

б2) Начин смештања и садржаји сваког улаза Таг меморије приказани су на слици.

| **улаз** | **асоцијативно** | | |
|---|---|---|---|
| 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 | |

Садржаји свих улаза Таг меморије у случају асоцијативног пресликавања

Време дохватања блока је:

$t_B$=64·($t_{DM}$+$t_{OM}$).

Укупно време дохватања свих блокова је:

$t_{Ball}$=32 $t_B$+8 $t_B$+24 $t_B$+15 $t_B$=79 $t_B$.

Укупно време је:

$T$=10058·$t_{DM}$+5056·$t_{OM}$+5081·$t_{SA}$+79·$t_{TM}$.

б3) Начин смештања и садржаји сваког улаза Таг меморије приказани су на слици.

| | сет | Таг | | | |
|---|---|---|---|---|---|
| улаз 0 | 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 | |
| | 7 | 0 | 2 | 1 | |
| | 8 | 0 | | | 2 |
| | 9 | 0 | | | 2 |
| | 10 | 0 | | | |
| | 11 | 0 | | | |
| | 12 | 0 | | | |
| | 13 | 0 | | | |
| | 14 | 0 | | | |
| | 15 | 0 | | | |
| улаз 1 | 0 | 1 | | | |
| | 1 | 1 | | | |
| | 2 | 1 | | | |
| | 3 | 1 | 0 | 2 | |
| | 4 | 1 | 0 | 2 | |
| | 5 | 1 | 0 | 2 | |
| | 6 | 1 | 0 | 2 | |
| | 7 | 1 | 0 | 2 | |
| | 8 | 1 | | | |
| | 9 | 1 | | | |
| | 10 | 1 | | | |
| | 11 | 1 | | | |
| | 12 | 1 | | | |
| | 13 | 1 | | | |
| | 14 | 1 | | | |
| | 15 | 1 | | | |

Садржаји свих улаза Таг меморије у случају сет-асоцијативног пресликавања

Време дохватања блока је:

$t_B$=64·($t_{DM}$+$t_{OM}$).

Укупно време дохватања свих блокова је:

$t_{Ball}$=16 $t_B$+16 $t_B$+8 $t_B$+5 $t_B$+5 $t_B$+5 $t_B$+2 $t_B$=57 $t_B$ .

Укупно време је:

$T$=3648·$t_{DM}$+3648·$t_{OM}$+5059·$t_{SS}$+57·$t_{TM}$

## Задатак 6.

Посматра се рачунар са следећим карактеристикама. Оперативна меморија је капацитета $2^{16}$ речи а ширина речи је 16 бита. Процесор генерише адресе које се односе на 16-битне адресе.

Процесор има кеш меморију са асоцијативним пресликавањем на нивоу блока са четири улаза. Величина блока је 16 речи. Користи се LRU алгоритам замене блокова и write-back алгоритам за ажурирање садржаја оперативне меморије. LRU алгоритам је реализован са четири бројача по модулу 4, сваки уз свој улаз кеш меморије. Улаз одабран за замену је онај чији бројач има вредност 0. На почетку је кеш меморија била празна а сви бројачи су имали вредност 0. Вредност бројача се ажурира на исти начин када је кеш меморија пуна и када је празна. Процесор генерише следећу секвенцу адреса са типом операције назначеним у загради после сваке адресе (Rd=read, Wr=write):

734Fh (Rd), DE38h (Rd), FF03h (Wr), 7350h (Rd), 7351h (Rd), DE48h (Wr), 7352h (Rd), 7348h (Rd).

а) Дати садржај свих улаза који су реферисани у датој секвенци, садржаје асоцијативног (TAG) дела (A0, A1, A2 и A3), вредности V (Valid) и D (Dirty) бита, као и LRU бројача, после сваког приступа кеш меморији.

б) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SA}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

в) Одговорити на питања а) и б) у случају да се уместо write-back алгоритма за ажурирање садржаја оперативне меморије користи write-through.

г) Претпоставити да је задата кеш меморија са директним пресликавањем на нивоу блока. Величина блока је 256 речи. Капацитет DATA дела кеш меморије је 1 К речи. Навести секвенцу вредности Tag поља за задату секвенцу адреса.

Решење:

**Напомена**: Када се наведе да се ради о **write-back** алгоритму за ажурирање садржаја оперативне меморије, а не наведе се о којој се политици довлачења ради приликом несагласности (Miss) приликом операције уписа онда се подразумева **write allocated**.

Када се наведе да се ради о **write-through** алгоритму за ажурирање садржаја оперативне меморије, а не наведе се о којој се политици довлачења ради приликом несагласности (Miss) приликом операције уписа онда се подразумева **no write allocated**.

а) Структура адреса код кеш меморије:

| | |
|---|---|
| Број блока | Word (4 бита) |
| Tag (12 бита) | |

| Адреса | Тип | Tag | Word | Време | Адресе |
|---|---|---|---|---|---|
| 734Fh | Rd | 011100110100 | 1111 | $t_{SA}+16\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 7340h-734Fh |
| DE38h | Rd | 110111100011 | 1000 | $t_{SA}+16\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | DE30h-DE3F |
| FF03h | Wr | 111111110000 | 0011 | $t_{SA}+16\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | FF00h-FF0Fh |
| 7350h | Rd | 011100110101 | 0000 | $t_{SA}+16\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 7350h-735Fh |
| 7351h | Rd | 011100110101 | 0001 | $t_{SA}+t_{DM}$ | - |
| DE48h | Wr | 110111100100 | 1000 | $t_{SA}+16\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | DE40h-DE4Fh |
| 7352h | Rd | 011100110101 | 0010 | $t_{SA}+t_{DM}$ | - |
| 7348h | Rd | 011100110100 | 1000 | $t_{SA}+16\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 7340h-734Fh |

Следећа табела садржи вредности свих релевантних делова кеш меморије после сваке адресе у секвенци адреса:

| адреса | 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 |

б) Време дохватања блока је:

$t_b=16\cdot(t_{OM}+t_{DM})$.

Укупно време је:

$t=104\cdot t_{DM}+96\cdot t_{OM}+14\cdot t_{SA}+6\cdot t_{TM}$.

в1)

| Адреса | Тип | Tag | Word | Време | Адресе |
|---|---|---|---|---|---|
| 734Fh | Rd | 011100110100 | 1111 | $t_{SA}+16\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 7340h-734Fh |
| DE38h | Rd | 110111100011 | 1000 | $t_{SA}+16\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | DE30h-DE3Fh |
| FF03h | Wr | 111111110000 | 0011 | $t_{SA}+t_{OM}$ | FF03h |
| 7350h | Rd | 011100110101 | 0000 | $t_{SA}+16\cdot(t_{OM}+t_{DM})+t_{TM}+t_{SA}+t_{DM}$ | 7350h-735Fh |
| 7351h | Rd | 011100110101 | 0001 | $t_{SA}+t_{DM}$ | - |
| DE48h | Wr | 110111100100 | 1000 | $t_{SA}+t_{OM}$ | DE48h |
| 7352h | Rd | 011100110101 | 0010 | $t_{SA}+t_{DM}$ | - |
| 7348h | Rd | 011100110100 | 1000 | $t_{SA}+t_{DM}$ | - |

Следећа табела садржи вредности свих релевантних делова кеш меморије после сваке адресе у секвенци адреса:

| адреса | A0 | A1 | A2 | A3 | C0 | C1 | C2 | C3 | V0 | V1 | V2 | V3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 734Fh | 734h | — | — | — | 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| DE38h | 734h | DE3h | — | — | 2 | 3 | 0 | 0 | 1 | 1 | 0 | 0 |
| FF03h | 734h | DE3h | — | — | 2 | 3 | 0 | 0 | 1 | 1 | 0 | 0 |
| 7350h | 734h | DE3h | 735h | — | 1 | 2 | 3 | 0 | 1 | 1 | 1 | 0 |
| 7351h | 734h | DE3h | 735h | — | 1 | 2 | 3 | 0 | 1 | 1 | 1 | 0 |
| DE48h | 734h | DE3h | 735h | — | 1 | 2 | 3 | 0 | 1 | 1 | 1 | 0 |
| 7352h | 734h | DE3h | 735h | — | 1 | 2 | 3 | 0 | 1 | 1 | 1 | 0 |
| 7348h | 734h | DE3h | 735h | — | 3 | 1 | 2 | 0 | 1 | 1 | 1 | 0 |

в2) Време дохватања блока је:

$T_b=16\cdot(t_{OM}+t_{DM})$.

Укупно време је:

$T=54\cdot t_{DM}+50\cdot t_{OM}+11\cdot t_{SA}+3\cdot t_{TM}$

г) Секвенца вредности Tag поља је 011100b, 110111b, 111111b, 011100b, 011100b, 110111b, 011100b, 011100b.

## Задатак 7.

Адреса је ширине 32 бита, а адресибилна јединица је 32-битна реч. Између магистрале процесора и меморије рачунара везана је кеш меморија организована сет-асоцијативно на нивоу блока, капацитета 16 MW (mega речи), са 16 сетова, два улаза по сету. На почетку је кеш био празан, а почетна вредност SP=200h, а ACC=0h, стек расте од виших ка нижим адресама, а SP указује на прву слободну локацију.

Алгоритам за ажурирање садржаја оперативне меморије је write-back. Кеш меморија је са write allocated политиком довлачења. Посматра се следећа секвенца инструкција које процесор генерише (све вредности су хексадецималне):

```
адреса:     инструкција:
   FF00     ADD #800h
   FF02     JSR  1224h      ; poziv potprograma
   FF04     SUB 8000h
   FF06     …

   1224     PUSH
   1225     LOAD #888h
   1227     RTS
```

а) Дати садржај оба улаза свих сетова који су реферисани у датој секвенци, као и вредности V (Valid) бита, после завршетка дате секвенце, ако је алгоритам замене FIFO. Означити сетове.

б) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SS}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

в) Исто као под а) и б) само је алгоритам замене LRU.

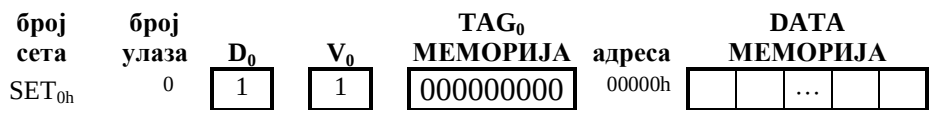
Решење:

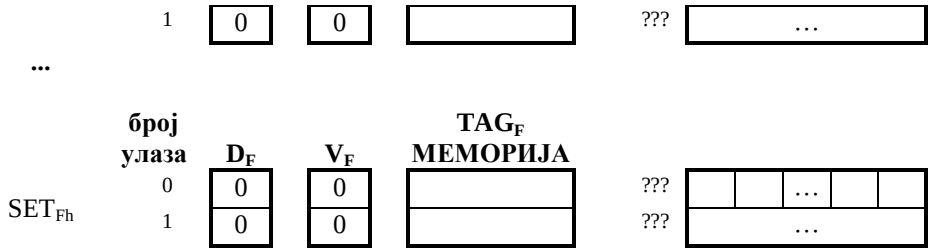
а) Структура адреса код кеш меморије:

<table>
<tr><td colspan="2">Број блока</td><td rowspan="2">Word (19 бита)</td></tr>
<tr><td>Tag (9 бита)</td><td>Set (4 бита)</td></tr>
</table>

Секвенца адреса које генерише представљени програмски сегмент: FF00h (Rd), FF01h (Rd), FF02h (Rd), FF03h (Rd), 200h (Wr), 1224h (Rd), 1FFh (Wr), 1225h (Rd), 1226h (Rd), 1227h (Rd), 1FFh (Rd), 800h (Rd). Све адресе се односе на сет 0 кеш меморије.

| Адреса | Тип | Tag | Set | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| FF00 | Rd | 000000000 | 0000 | 0001111111100000000 | $t_{SS}+524288*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 0h-7FFFFh |
| FF01 | Rd | 000000000 | 0000 | 0001111111100000001 | $t_{SS}$ | - |
| FF02 | Rd | 000000000 | 0000 | 0001111111100000010 | $t_{SS}$ | - |
| FF03 | Rd | 000000000 | 0000 | 0001111111100000011 | $t_{SS}$ | - |
| 200 | Wr | 000000000 | 0000 | 0000000001000000000 | $t_{SS}+t_{DM}$ | - |
| 1224 | Rd | 000000000 | 0000 | 0000001001000100100 | $t_{SS}$ | - |
| 1FF | Wr | 000000000 | 0000 | 0000000000111111111 | $t_{SS}+t_{DM}$ | - |
| 1225 | Rd | 000000000 | 0000 | 0000001001000100101 | $t_{SS}$ | - |
| 1226 | Rd | 000000000 | 0000 | 0000001001000100110 | $t_{SS}$ | - |
| 1227 | Rd | 000000000 | 0000 | 0000001001000100111 | $t_{SS}$ | - |
| 1FF | Rd | 000000000 | 0000 | 0000000000111111111 | $t_{SS}$ | - |
| 800 | Rd | 000000000 | 0000 | 0000000100000000000 | $t_{SS}$ | - |






Садржај релевантних делова кеш меморије

б) Време дохватања блока је: $T_b=524288\cdot(t_{OM}+t_{DM})$.

Укупно време је:

$T=524290*t_{DM}+524288*t_{OM}+13*t_{SS}+1*t_{TM}$

в) Исто како а) и б).

## Задатак 8.

Адреса је ширине 32 бита, а адресибилна јединица је 8-битна реч. Између магистрале процесора и меморије рачунара везане су две кеш меморије, једна за инструкције и друга за податке, организоване сет-асоцијативно на нивоу блока свака капацитета 16 KB, величина блока је 16 бајтова, са два улаза по сету. На почетку су обе кеш меморије биле празне, а почетна вредност SP=200h, а ACC=0h, стек расте од виших ка нижим адресама. Акумулатор је 32-битни. Алгоритам за ажурирање садржаја оперативне меморије write-back. Кеш меморија је са no write allocated политиком довлачења. Посматра се следећа секвенца инструкција које процесор генерише (све вредности су хексадецималне):

| адреса: | инструкција: |
|---|---|
| FF00 | PUSH |
| FF01 | JSR 1234h ; poziv potprograma |
| FF06 | LOAD 8000h |
| FF0B | |
| … | |
| 1234 | POP |
| 1235 | STORE 8000h |
| 123A | PUSH |
| 123B | RTS |

а) Дати садржај оба улаза свих сетова који су реферисани у датој секвенци, као и вредности V (Valid) и D (Dirty) бита, после завршетка дате секвенце, ако је алгоритам замене FIFO. Означити сетове.

б) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SS}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

в) Исто као под а) и б) само је алгоритам замене LRU.

Решење:

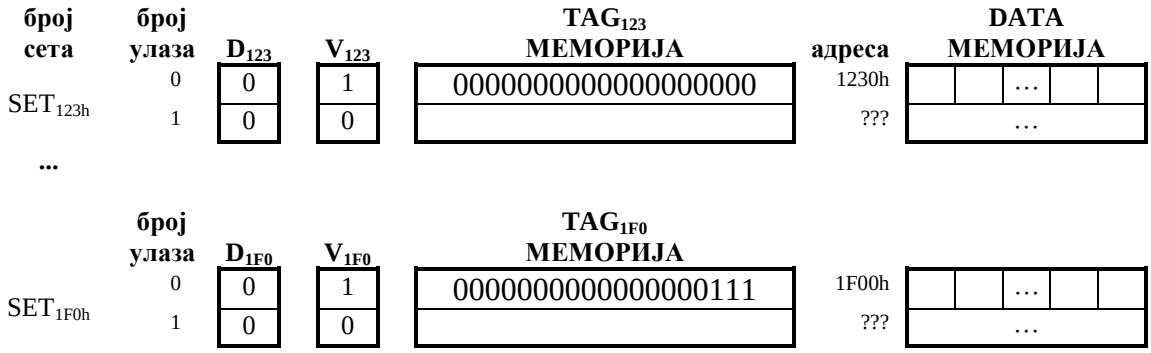
а) Структура адреса код кеш меморије:

| Број блока | | Word (4 бита) |
|---|---|---|
| Tag (19 бита) | Set (9 бита) | |

Секвенца адреса у којима се приступа: FF00, 200, 1FF, 1FE, 1FD, FF01, FF02, FF03, FF04, FF05, 1FC, 1FB, 1FA, 1F9, 1234, 1F9, 1FA, 1FB, 1FC, 1235, 1236, 1237, 1238, 1239, 8000, 8001, 8002, 8003, 123A,

1FC, 1FB, 1FA, 1F9, 123B, 1F9, 1FA, 1FB, 1FC, FF06, FF07, FF08, FF09, FF0A, 8000, 8001, 8002, 8003, FF0B.

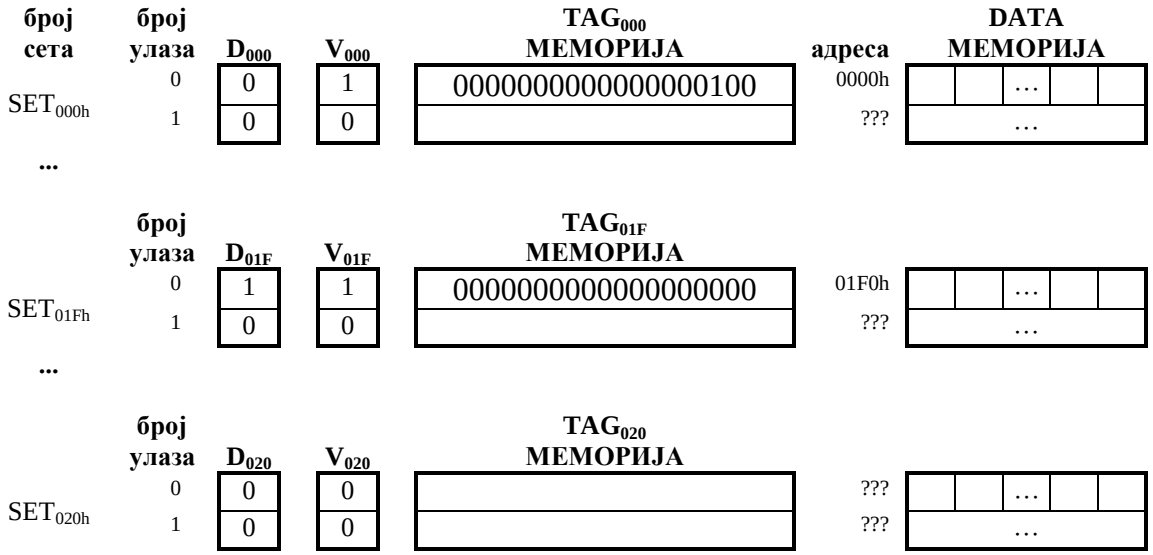
| Адреса | Тип | Tag | Set | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| FF00 | I | 00000000000000000111 | 111110000 | 0000 | $t_{SS}+16*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | FF00h-FF0Fh |
| FF01 | I | 00000000000000000111 | 111110000 | 0001 | $t_{SS}$ | - |
| FF02 | I | 00000000000000000111 | 111110000 | 0010 | $t_{SS}$ | - |
| FF03 | I | 00000000000000000111 | 111110000 | 0011 | $t_{SS}$ | - |
| FF04 | I | 00000000000000000111 | 111110000 | 0100 | $t_{SS}$ | - |
| FF05 | I | 00000000000000000111 | 111110000 | 0101 | $t_{SS}$ | - |
| 1234 | I | 00000000000000000000 | 100100011 | 0100 | $t_{SS}+16*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 1230h-123Fh |
| 1235 | I | 00000000000000000000 | 100100011 | 0101 | $t_{SS}$ | - |
| 1236 | I | 00000000000000000000 | 100100011 | 0110 | $t_{SS}$ | - |
| 1237 | I | 00000000000000000000 | 100100011 | 0111 | $t_{SS}$ | - |
| 1238 | I | 00000000000000000000 | 100100011 | 1000 | $t_{SS}$ | - |
| 1239 | I | 00000000000000000000 | 100100011 | 1001 | $t_{SS}$ | - |
| 123A | I | 00000000000000000000 | 100100011 | 1010 | $t_{SS}$ | - |
| 123B | I | 00000000000000000000 | 100110010 | 1011 | $t_{SS}$ | - |
| FF06 | I | 00000000000000000111 | 111110000 | 0110 | $t_{SS}$ | - |
| FF07 | I | 00000000000000000111 | 111110000 | 0111 | $t_{SS}$ | - |
| FF08 | I | 00000000000000000111 | 111110000 | 1000 | $t_{SS}$ | - |
| FF09 | I | 00000000000000000111 | 111110000 | 1001 | $t_{SS}$ | - |
| FF0A | I | 00000000000000000111 | 111110000 | 1010 | $t_{SS}$ | - |
| FF0B | I | 00000000000000000111 | 111110000 | 1011 | $t_{SS}$ | - |




Садржај релевантних делова кеш меморије за инструкције

| Адреса | Тип | Tag | Set | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| 200 | Wr | 00000000000000000000 | 000100000 | 0000 | $t_{SS}+t_{OM}$ | 200h |
| 1FF | Wr | 00000000000000000000 | 000011111 | 1111 | $t_{SS}+t_{OM}$ | 1FFh |
| 1FE | Wr | 00000000000000000000 | 000011111 | 1110 | $t_{SS}+t_{OM}$ | 1FEh |
| 1FD | Wr | 00000000000000000000 | 000011111 | 1101 | $t_{SS}+t_{OM}$ | 1FDh |
| 1FC | Wr | 00000000000000000000 | 000011111 | 1100 | $t_{SS}+t_{OM}$ | 1FCh |
| 1FB | Wr | 00000000000000000000 | 000011111 | 1011 | $t_{SS}+t_{OM}$ | 1FBh |
| 1FA | Wr | 00000000000000000000 | 000011111 | 1010 | $t_{SS}+t_{OM}$ | 1FAh |
| 1F9 | Wr | 00000000000000000000 | 000011111 | 1001 | $t_{SS}+t_{OM}$ | 1F9h |
| 1F9 | Rd | 00000000000000000000 | 000011111 | 1001 | $t_{SS}+16*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 1F0h-1FFh |
| 1FA | Rd | 00000000000000000000 | 000011111 | 1010 | $t_{SS}$ | - |
| 1FB | Rd | 00000000000000000000 | 000011111 | 1011 | $t_{SS}$ | - |
| 1FC | Rd | 00000000000000000000 | 000011111 | 1100 | $t_{SS}$ | - |
| 8000 | Wr | 00000000000000000100 | 000000000 | 0000 | $t_{SS}+t_{OM}$ | 8000h |
| 8001 | Wr | 00000000000000000100 | 000000000 | 0001 | $t_{SS}+t_{OM}$ | 8001h |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8002 | Wr | 0000000000000000100 | 000000000 | 0010 | $t_{SS}+t_{OM}$ | 8002h |
| 8003 | Wr | 0000000000000000100 | 000000000 | 0011 | $t_{SS}+t_{OM}$ | 8003h |
| 1FC | Wr | 0000000000000000000 | 000011111 | 1100 | $t_{SS}$ | - |
| 1FB | Wr | 0000000000000000000 | 000011111 | 1011 | $t_{SS}$ | - |
| 1FA | Wr | 0000000000000000000 | 000011111 | 1010 | $t_{SS}$ | - |
| 1F9 | Wr | 0000000000000000000 | 000011111 | 1001 | $t_{SS}$ | - |
| 1F9 | Rd | 0000000000000000000 | 000011111 | 1001 | $t_{SS}$ | - |
| 1FA | Rd | 0000000000000000000 | 000011111 | 1010 | $t_{SS}$ | - |
| 1FB | Rd | 0000000000000000000 | 000011111 | 1011 | $t_{SS}$ | - |
| 1FC | Rd | 0000000000000000000 | 000011111 | 1100 | $t_{SS}$ | - |
| 8000 | Rd | 0000000000000000100 | 000000000 | 0000 | $t_{SS}+16*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | 8000h-800Fh |
| 8001 | Rd | 0000000000000000100 | 000000000 | 0001 | $t_{SS}$ | - |
| 8002 | Rd | 0000000000000000100 | 000000000 | 0010 | $t_{SS}$ | - |
| 8003 | Rd | 0000000000000000100 | 000000000 | 0011 | $t_{SS}$ | - |




Садржај релевантних делова кеш меморије за податке

б) Време дохватања блока је: $t_B=16*(t_{OM}+t_{DM})$.

Укупно време је:

$t=t_I+t_D$

$t_I=32*t_{DM}+32*t_{OM}+22*t_{SS}+2*t_{TM}$

$t_D=32*t_{DM}+44*t_{OM}+30*t_{SS}+2*t_{TM}$

$t=64*t_{DM}+76*t_{OM}+52*t_{SS}+4*t_{TM}$

в) Исто како а) и б).

## Задатак 9.

Адреса је ширине 32 бита, а адресибилна јединица је 16-битна реч. Између магистрале процесора и меморије рачунара везана је јединствена кеш меморије за инструкције и за податке, организоване сет-асоцијативно на нивоу блока, капацитета 32 KB, величина блока је 128 бајтова, са два улаза по сету. На почетку је кеш меморија била празна, а почетна вредност SP=600Ah, а MEM[0001h]=1h, MEM[4001h]=2h, MEM[4002h]=4h стек расте од виших ка нижим адресама и указује на прву слободну локацију. Акумулатор је 32-битни. Алгоритам за ажурирање садржаја оперативне меморије write-through. Кеш меморија је са no write allocated политиком довлачења и користи LRU алгоритам замене. Посматра се следећа секвенца инструкција које процесор генерише (све вредности су хексадецималне):

адреса: инструкција:

E005 LOAD 0001h

E008 JSR A006h ; poziv potprograma

E00B STORE 201Ah

E00E

…

A006 LOAD 4001h

A009 ADD 4001h

A00C STORE 4003h

A00F RTS

а) Дати садржај свих улаза свих сетова који су реферисани у датој секвенци, као и вредности релевантних бита, после завршетка дате секвенце. Означити сетове.

б) Уместо јединствене кеш меморије капацитета 32 KB користе се две кеш меморије, једна за податке а друга за инструкције, свака капацитета 16 KB. Упоредити укупно време потребно за приступ кеш меморије у датој секвенци у оба случаја. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SS}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

Решење:

а)

1W=32b

CacheSize=16 KW

BlockSize=64 W

SetAssociativity=2

Структура адреса код кеш меморије:

| Број блока | | Word (6 бита) |
|---|---|---|
| Tag (19 бита) | Set (7 бита) | |

Секвенца адреса које генерише представљени програмски сегмент: E005h (Rd), E006h (Rd), E007h (Rd), 0001h (Rd), 0002h (Rd), E008h (Rd), E009h (Rd), E00Ah (Rd), 600Ah (Wr), 6009h (Wr), A006h (Rd), A007h (Rd), A008h (Rd), 4001h (Rd), 4002h (Rd), A009h (Rd), A00Ah (Rd), A00Bh (Rd), 4001h (Rd), 4002h (Rd), A00Ch (Rd), A00Dh (Rd), A00Eh (Rd), 4003h (Wr), 4004h (Wr), A00Fh (Rd), 6009h (Rd), 600Ah (Rd), E00Bh (Rd), E00Ch (Rd), E00Dh (Rd), 201Ah (Wr), 202Bh (Wr),

| Адреса | Тип | Tag | Set | Word | Време а) | Време б) |
|---|---|---|---|---|---|---|
| E005 | Rd | 00000000000000000111 | 0000000 | 000101 | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ |
| E006 | Rd | 00000000000000000111 | 0000000 | 000110 | $t_{SS}$ | $t_{SS}$ |
| E007 | Rd | 00000000000000000111 | 0000000 | 000111 | $t_{SS}$ | $t_{SS}$ |
| 0001 | Rd | 00000000000000000000 | 0000000 | 000001 | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ |
| 0002 | Rd | 00000000000000000000 | 0000000 | 000010 | $t_{SS}$ | $t_{SS}$ |
| E008 | Rd | 00000000000000000111 | 0000000 | 001000 | $t_{SS}$ | $t_{SS}$ |
| E009 | Rd | 00000000000000000111 | 0000000 | 001001 | $t_{SS}$ | $t_{SS}$ |
| E00A | Rd | 00000000000000000111 | 0000000 | 001010 | $t_{SS}$ | $t_{SS}$ |
| 600A | Wr | 00000000000000000011 | 0000000 | 001010 | $t_{SS}+t_{OM}$ | $t_{SS}+t_{OM}$ |
| 6009 | Wr | 00000000000000000011 | 0000000 | 001001 | $t_{SS}+t_{OM}$ | $t_{SS}+t_{OM}$ |

| | | | | | | |
|---|---|---|---|---|---|---|
| A006 | Rd | 0000000000000000101 | 0000000 | 000110 | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ |
| A007 | Rd | 0000000000000000101 | 0000000 | 000111 | $t_{SS}$ | $t_{SS}$ |
| A008 | Rd | 0000000000000000101 | 0000000 | 001000 | $t_{SS}$ | $t_{SS}$ |
| 4001 | Rd | 0000000000000000010 | 0000000 | 000001 | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ |
| 4002 | Rd | 0000000000000000010 | 0000000 | 000010 | $t_{SS}$ | $t_{SS}$ |
| A009 | Rd | 0000000000000000101 | 0000000 | 001001 | $t_{SS}$ | $t_{SS}$ |
| A00A | Rd | 0000000000000000101 | 0000000 | 001010 | $t_{SS}$ | $t_{SS}$ |
| A00B | Rd | 0000000000000000101 | 0000000 | 001011 | $t_{SS}$ | $t_{SS}$ |
| 4001 | Rd | 0000000000000000010 | 0000000 | 000010 | $t_{SS}$ | $t_{SS}$ |
| 4002 | Rd | 0000000000000000010 | 0000000 | 000101 | $t_{SS}$ | $t_{SS}$ |
| A00C | Rd | 0000000000000000101 | 0000000 | 001100 | $t_{SS}$ | $t_{SS}$ |
| A00D | Rd | 0000000000000000101 | 0000000 | 001101 | $t_{SS}$ | $t_{SS}$ |
| A00E | Rd | 0000000000000000101 | 0000000 | 001110 | $t_{SS}$ | $t_{SS}$ |
| 4003 | Wr | 0000000000000000010 | 0000000 | 000011 | $t_{SS}+t_{DM}+t_{OM}$ | $t_{SS}+t_{DM}+t_{OM}$ |
| 4004 | Wr | 0000000000000000010 | 0000000 | 000100 | $t_{SS}+t_{DM}+t_{OM}$ | $t_{SS}+t_{DM}+t_{OM}$ |
| A00F | Rd | 0000000000000000101 | 0000000 | 001111 | $t_{SS}$ | $t_{SS}$ |
| 6009 | Rd | 0000000000000000011 | 0000000 | 001001 | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ |
| 600A | Rd | 0000000000000000011 | 0000000 | 001010 | $t_{SS}$ | $t_{SS}$ |
| E00B | Rd | 0000000000000000111 | 0000000 | 001011 | $t_{SS}+64*(t_{OM}+t_{DM})+t_{TM}+t_{SS}$ | $t_{SS}$ |
| E00C | Rd | 0000000000000000111 | 0000000 | 001100 | $t_{SS}$ | $t_{SS}$ |
| E00D | Rd | 0000000000000000111 | 0000000 | 001101 | $t_{SS}$ | $t_{SS}$ |
| 201A | Wr | 0000000000000000001 | 0000000 | 011010 | $t_{SS}+t_{OM}$ | $t_{SS}+t_{OM}$ |
| 202B | Wr | 0000000000000000001 | 0000000 | 011011 | $t_{SS}+t_{OM}$ | $t_{SS}+t_{OM}$ |

Адресе се односе на сет 00h кеш меморије

| **број сета** | **број улаза** | $V_{00}$ | $TAG_{00}$ **МЕМОРИЈА** | **LRU CNT** |
|---|---|---|---|---|
| $SET_{00h}$ | 0 | 1 | 3 | 0 |
| | 1 | 1 | 7 | 1 |

Садржај релевантних делова кеш меморије

Време дохватања блока је: $t_B=64*(t_{OM}+t_{DM})$.

Укупно време је:

$t=386*t_{DM}+390*t_{OM}+39*t_{SS}+6*t_{TM}$.

б)

Структура адреса код кеш меморије:

| Број блока | | Word (6 бита) |
|---|---|---|
| Tag (20 бита) | Set (6 бита) | |

Адресе се односе на сет 00h кеш меморије

| **број сета** | **број улаза** | $V_{00}$ | $TAG_{00}$ **МЕМОРИЈА** | **LRU CNT** |
|---|---|---|---|---|
| $SET_{00h}$ | 0 | 1 | E | 1 |
| | 1 | 1 | A | 0 |

Садржај релевантних делова кеш меморије за инструкције

Укупно време приступа кеш меморији за инструкције је:

$t_I=128*t_{DM}+128*t_{OM}+21*t_{SS}+2*t_{TM}$

| број сета | број улаза | $V_{00}$ | $TAG_{00}$ МЕМОРИЈА | LRU CNT |
|---|---|---|---|---|
| $SET_{00h}$ | 0 | 1 | 6 | 1 |
| | 1 | 1 | 4 | 0 |

Садржај релевантних делова кеш меморије за податке

Укупно време приступа кеш меморији за податке је:

$t_D=194*t_{DM}+198*t_{OM}+17*t_{SS}+3*t_{TM}$

Укупно време је:

$t=322*t_{DM}+326*t_{OM}+38*t_{SS}+5*t_{TM}$

## Задатак 10.

Адреса је ширине 32 бита, а адресибилна јединица је 16-битна реч. Процесор има раздвојене кеш меморије за инструкције и податке. Кеш меморија за инструкције и кеш меморија за податке имају следеће карактеристике: организоване су сет-асоцијативно на нивоу блока, свака капацитета 8 KB, величина блока је 64 бајтова, са четири улаза по сету. Алгоритам за ажурирање садржаја оперативне меморије је write-trough. Кеш меморија је са write allocated политиком довлачења и користи LRU алгоритам замене. Посматра се следећи програмски сегмент на језику Ц:

```
for (j=0; j < 0x80; j++)
    for (i=0; i < 0x100; i++)
        x[i][j]=2 * x[i][j];
```

Променљива i и j су смештене у регистрима, а низ x (x[0][0]) почев од локације 300h. У програмском језику Ц низови се у меморију смештају тако да је адреса елемента x[i][j] дата са: x[0][0]+i*n+j. На почетку је кеш меморија била празна.

а) Нацртати структуру кеш меморије за податке, означити ширину у битовима свих релевантних делова и укратко описати њихову намену за дату кеш меморију.

б) Дати садржај свих улаза свих сетова који су реферисани у датој секвенци, садржаје асоцијативног (Tag) дела, вредности V (Valid) и D (Dirty) бита, као и LRU бројача, после завршетка дате секвенце. Означити сетове.

Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SS}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

в) Модификовати дати програмски сегмент тако да време приступа кеш меморији за податке буде краће него у тачки б). Израчунати укупно време потребно за приступ кеш меморији за податке у тако модификованом програмском сегменту.

Решење:

а)

Структура адреса код кеш меморије:

| Број блока | | Word (5 бита) |
|---|---|---|
| Tag (22 бита) | Set (5 бита) | |

б)

| | улаз 0 | | улаз 1 | | улаз 2 | | улаз 3 | | LRU COUNT | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| број сета | V | TAG МЕМОРИЈА | V | TAG МЕМОРИЈА | V | TAG МЕМОРИЈА | V | TAG МЕМОРИЈА | CNT0 | CNT1 | CNT2 | CNT3 |
| 0 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 1 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 2 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 3 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 4 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 5 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 6 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 7 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 8 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 9 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 10 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 11 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 12 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 13 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 14 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 15 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 16 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 17 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 18 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 19 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 20 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 21 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 22 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 23 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 24 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 25 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 26 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 27 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 28 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 29 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 30 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 31 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |

Садржај релевантних делова кеш меморије

$t_1=1081344*t_{DM}+1081344*t_{OM}+98304*t_{SS}+32768*t_{TM}$.

в)

```
for (i=0; i < 0x100; i++)
        for (j=0; j < 0x80; j++)
                x[i][j]=2 * x[i][j];
```

| | улаз 0 | | улаз 1 | | улаз 2 | | улаз 3 | | LRU COUNT | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| број сета | V | TAG МЕМОРИЈА | V | TAG МЕМОРИЈА | V | TAG МЕМОРИЈА | V | TAG МЕМОРИЈА | CNT0 | CNT1 | CNT2 | CNT3 |
| 0 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 2 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 3 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 4 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 5 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 6 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 7 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 8 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 9 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 10 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 11 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 12 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 13 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 14 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 15 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 16 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 17 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 18 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 19 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 20 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 21 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 22 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 23 | 1 | 00001D | 1 | 00001E | 1 | 00001F | 1 | 000020 | 0 | 1 | 2 | 3 |
| 24 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 25 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 26 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 27 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 28 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 29 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 30 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |
| 31 | 1 | 00001C | 1 | 00001D | 1 | 00001E | 1 | 00001F | 0 | 1 | 2 | 3 |

Садржај релевантних делова кеш меморије

$t_2=65536*t_{DM}+65536*t_{OM}+66560*t_{SS}+1024*t_{TM}$.

## Задатак 11.

LRU i486

Меморијски адресни простор рачунара је 4 GB, а адресибилна јединица је бајт. Кеш меморија је са сет-асоцијативним пресликавањем на нивоу блока. Блок је величине 256 B, а кеш меморија има капацитет меморије података 4 MB. Сваки сет има 4 улаза.

Код микропроцесора Intel 80486 користи се следећа апроксимација LRU алгоритма. Сваки сет поседује 4 улаза. Та четири улаза се деле на две групе (група 0 и група 1) од по два улаза (улаз 0 и улаз 1 у групи). За спровођење алгоритма користе се три бита придружена сваком сету. Један бит (бит B2) говори којој се групи улаза најскорије приступало, а друга два бита (бити B1 и B0) говоре ком се улазу из сваке групе скорије приступало.

а) Дефинисати алгоритам и хардвер којим се ажурирају описана три бита при једном приступу неком улазу.

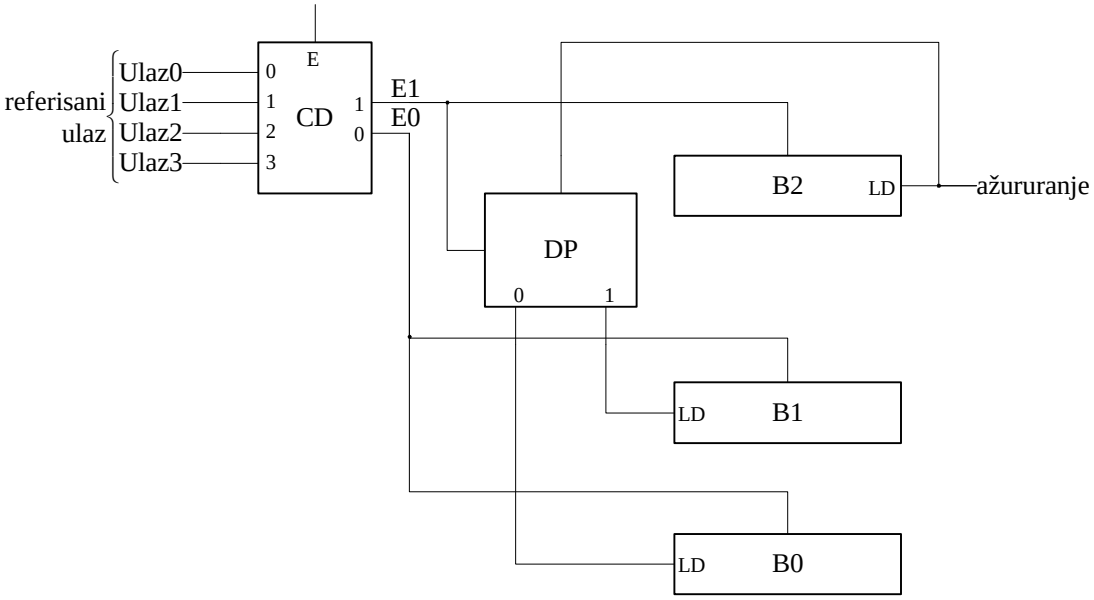
б) Дефинисати алгоритам и хардвер којим се врши избор улаза за замену.

в) За кеш меморију дати вредности описаних бита сета 2FFh после завршетка следеће секвенце адреса (све вредности су хексадецималне): 12FF0A (Rd), 22FF0B (Rd), 30C0C (Rd), 32FF00 (Wr), 42FF01 (Rd), 130C01 (Rd), 22FF00 (Wr), 52FFFF (Rd).

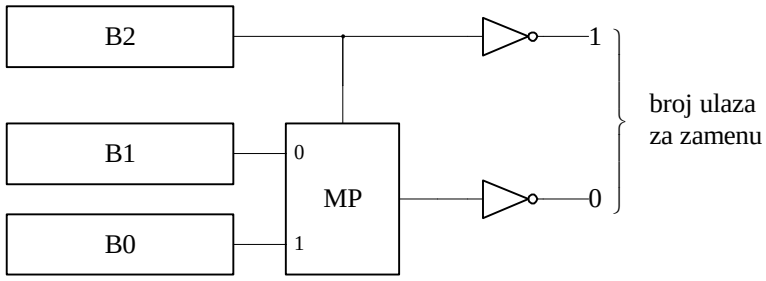
г) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SS}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

Решење:

а) Нека је број улаза (0 до 3) коме се врши обраћање кодиран са 2 бита E1 и E0. При сваком обраћању треба ажурирати описане бите на следећи начин: у бит B2 треба уписати број групе улаза којој се приступа; то је заправо вредност бита E1. Ако је то група 0, онда у бит B0 треба уписати ком се улазу из групе приступало, а то је заправо вредност E0; ако је то група 1, онда у бит B1 треба уписати ову вредност E0. Према томе, потребни хардвер изгледа овако:




б) Избор улаза у који треба сместити нови блок се обавља тако што се прво провери да ли постоји неки слободан улаз. Уколико постоји један или више слободних улаза они се попуњавају по FIFO принципу. Уколико не постоји слободан улаз онда је потребно извршити избор улаза за замену. Избор улаза за замену врши се по следећем алгоритму: бира се за замену улаз који је давније коришћен из оне групе која је давније коришћена. На пример, ако је вредност бита B2 једнака 0, бира се група 1; при том, ако је вредност бита B1 једнака 0, бира се улаз 1 из те групе. Хардвер изгледа овако:




в) Структура адреса код кеш меморије:

| Број блока | Word (8 бита) |
|---|---|

| Tag (12 бита) | Set (12 бита) | |
|---|---|---|

| Адреса | Тип | Tag | Set | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| 0012FF0A | Rd | 001 | 2FF | 0A | $t_{SS}+t_B+t_{TM}+t_{SS}$ | 12FF00h-12FFFFh |
| 0022FF0B | Rd | 002 | 2FF | 0B | $t_{SS}+t_B+t_{TM}+t_{SS}$ | 22FF00h-22FFFFh |
| 00030C0C | Rd | 000 | 30C | 0C | $t_{SS}+t_B+t_{TM}+t_{SS}$ | 30C00h-30CFFh |
| 0032FF00 | Wr | 003 | 2FF | 00 | $t_{SS}+t_B+t_{TM}+t_{SS}+t_{DM}$ | 32FF00h-32FFFF |
| 0042FF01 | Rd | 004 | 2FF | 01 | $t_{SS}+t_B+t_{TM}+t_{SS}$ | 42FF00h-42FFFFh |
| 00130C01 | Rd | 001 | 30C | 01 | $t_{SS}+t_B+t_{TM}+t_{SS}$ | 130C00h-130CFFh |
| 0022FF00 | Wr | 002 | 2FF | 00 | $t_{SS}+t_{DM}$ | - |
| 0052FFFF | Rd | 005 | 2FF | FF | $t_{SS}+t_B+t_B+t_{TM}+t_{SS}$ | 32FF00h-32FFFF, 52FF00-52FFFFh |

Дата секвенца представља тражење следећих вредности TAG у сету 2FFh: 1, 2, 3, 4, 2, 5. Стања улаза *i* бита B овог сета током извршавања ове секвенце су следећа:

| **TAG** | **Улаз0** | **Улаз1** | **Улаз2** | **Улаз3** | **битови 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 |

Упоређивањем коначног стања сета са оним из претходног задатка, може се приметити да за дати пример алгоритам из овог задатка даје лошију одлуку при замени улаза 2, који је скорије коришћен него улаз 0.

г) Време дохватања блока је: $t_B=256*(t_{DM}+t_{OM})$.

Укупно време је:

$t=2050*t_{DM}+2048*t_{OM}+15*t_{SS}+7*t_{TM}$

## Задатак 12.

Посматра се процесор са следећим карактеристикама: адреса је ширине 32 бита, а адресибилна јединица је 16-битна реч. Процесор има раздвојене кеш меморије за инструкције и податке. Кеш меморија за инструкције и кеш меморија за податке имају следеће карактеристике: организоване су сет-асоцијативно на нивоу блока, свака капацитета 16 KB, величина блока је 64 бајтова, са четири улаза по сету. Алгоритам за ажурирање садржаја оперативне меморије је write-back. Кеш меморија је са no write allocated политиком довлачења и користи псеудо LRU (i486) алгоритам замене. Процесор генерише следећу секвенцу адреса са типом операције назначеним у загради после сваке адресе (Rd=read, Wr=write):

AA59h (Rd), B251h (Rd), CA50h (Wr), D251h (Rd), EA57h (Rd), F250h (Wr), 1256h (Rd), CA54h (Rd).

На почетку је кеш меморија била празна.

а) Нацртати структуру кеш меморије за податке, означити ширину у битовима свих релевантних делова и укратко описати њихову намену за дату кеш меморију.

б) Дати садржај свих улаза свих сетова који су реферисани у датој секвенци, садржаје асоцијативног (Tag) дела, вредности V (Valid) и D (Dirty) бита, као и LRU бројача, после сваког приступа кеш меморији у датој секвенци. Означити сетове.

в) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SS}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

Решење:

а) Структура адреса код кеш меморије:

| Број блока | | Word (5 бита) |
|---|---|---|
| Tag (21 бита) | Set (6 бита) | |

б)

| Адреса | Тип | Tag | Set | Word | Време | Адресе |
|---|---|---|---|---|---|---|
| AA59h | Rd | 15h | 12h | 19h | $t_{SS}+t_B+t_{TM}+t_{SS}$ | AA40h-AA5Fh |
| B251h | Rd | 16h | 12h | 11h | $t_{SS}+t_B+t_{TM}+t_{SS}$ | B240h-B25Fh |
| CA50h | Wr | 19h | 12h | 10h | $t_{SS}+t_{OM}$ | CA50h |
| D251h | Rd | 1Ah | 12h | 11h | $t_{SS}+t_B+t_{TM}+t_{SS}$ | D240h-D25Fh |
| EA57h | Rd | 1Dh | 12h | 17h | $t_{SS}+t_B+t_{TM}+t_{SS}$ | EA40h-EA5Fh |
| F250h | Wr | 1Eh | 12h | 10h | $t_{SS}+t_{OM}$ | F250h |
| 1256h | Rd | 02h | 12h | 16h | $t_{SS}+t_B+t_{TM}+t_{SS}$ | 1240h-125Fh |
| CA54h | Rd | 19h | 12h | 14h | $t_{SS}+t_B+t_{TM}+t_{SS}$ | CA40h-CA5Fh |

| **$SET_{12h}$** | **улаз 0** | | | **улаз 1** | | | **улаз 2** | | | **улаз 3** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Адреса** | **D** | **V** | **TAG МЕМОРИЈА** | **D** | **V** | **TAG МЕМОРИЈА** | **D** | **V** | **TAG МЕМОРИЈА** | **D** | **V** | **TAG МЕМОРИЈА** | **псеудо LRU** | | |
| AA59 | 0 | 1 | 15 | 0 | 0 | | 0 | 0 | | 0 | 0 | | 0 | 0 | 0 |
| B251h | 0 | 1 | 15 | 0 | 1 | 16 | 0 | 0 | | 0 | 0 | | 0 | 0 | 1 |
| CA50h | 0 | 1 | 15 | 0 | 1 | 16 | 0 | 0 | | 0 | 0 | | 0 | 0 | 1 |
| D251h | 0 | 1 | 15 | 0 | 1 | 16 | 0 | 1 | 1A | 0 | 0 | | 1 | 0 | 1 |
| EA57h | 0 | 1 | 15 | 0 | 1 | 16 | 0 | 1 | 1A | 0 | 1 | 1D | 1 | 1 | 1 |
| F250h | 0 | 1 | 15 | 0 | 1 | 16 | 0 | 1 | 1A | 0 | 1 | 1D | 1 | 1 | 1 |
| 1256h | 0 | 1 | 2 | 0 | 1 | 16 | 0 | 1 | 1A | 0 | 1 | 1D | 0 | 1 | 0 |
| CA54h | 0 | 1 | 2 | 0 | 1 | 16 | 0 | 1 | 19 | 0 | 1 | 1D | 1 | 0 | 0 |

Садржај релевантних делова кеш меморије

в)

Време дохватања блока је: $t_B=32*(t_{OM}+t_{DM})$.

Укупно време је: $t=192*t_{DM}+194*t_{OM}+14*t_{SS}+6*t_{TM}$.

## Задатак 13.

Разматра се процесор код кога је адреса ширине 32 бита а адресибилна јединица је 8 битна реч који има L1 кеш меморију са следећим карактеристикама: Блок, односно кеш линија је величине 32 бајта; Кешу се приступа на нивоу 8 битне речи; Кеш меморија је сет асоцијативна капацитета DATA дела 64 KB, са два блока по сету; Алгоритам за ажурирање садржаја оперативне меморије је write-back. Кеш меморија је са write allocated политиком довлачења и користи FIFO алгоритам замене.

а) Нацртати структуру кеш меморије, означити ширину у битовима за све релевантне делове и укратко описати њихову намену.

б) Процесор генерише следећу секвенцу адреса са типом операције назначеним у загради после сваке адресе (R=read, W=write):

8043h (Rd), 10055h (Wr), 8044h (Rd), 1804Fh (Rd), 10056h (Rd), 005Bh (Rd), 2805Eh (Wr), 10055h (Rd), 8045h (Rd), 8046h (Wr). Дати садржај свих улаза свих сетова кеш меморије који су реферисани у датој секвенци, као и вредности V (Valid) бита и D (Dirty), после завршетка дате секвенце. Означити сетове.

в) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе цео блок из оперативне меморије у L1 кеш меморије и обрнуто, па се тек онда приступа локацији. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SS}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

г) Одговорити на питања а) – в) уколико L1 кеш меморија уместо наведених карактеристика има следеће: Алгоритам за ажурирање садржаја оперативне меморије је write-trougth. Кеш меморија је са no write allocated политиком довлачења.

д) Одговорити на питања а) – в) уколико L1 кеш меморија уместо наведених карактеристика има следеће: Алгоритам за ажурирање садржаја оперативне меморије је write-trougth. Кеш меморија је са write allocated политиком довлачења.

ђ) Одговорити на питања а) – в) уколико L1 кеш меморија уместо наведених карактеристика има следеће: Алгоритам за ажурирање садржаја оперативне меморије је write-back. Кеш меморија је са no write allocated политиком довлачења.

е) Уколико се дати процесор модификује тако да поред L1 кеш меморија има и помоћну кеш меморију (victim cache), величине блока исте као и код L1 кеш меморија, који је реализован асоцијативно са 8 блокова и FIFO алгоритмом замене. Кеш линија избачена из L1 кеш меморије се увек пребацује у помоћну кеш меморију. Уколико се у помоћној кеш меморији открије сагласност дата линија се враћа у L1 кеш. Време приступа помоћној кеш меморији за проверу сагласности износи $t_{SV}$. Уколико постоји сагласност време трансфера линије података из L1 у помоћну кеш меморију и обрнуто износи $t_{VL}$. Одговорити на питања а) - в) поново.

Решење:

а)

Структура адреса код L1 кеш меморије:

| Број блока | | Word (5 бита) |
|---|---|---|
| Tag (17 бита) | Set (10 бита) | |

б)

| Адреса | Тип | Tag | Set | Word | Време б) | Време г) | Време д) | Време ђ) |
|---|---|---|---|---|---|---|---|---|
| 8043h | Rd | 00000000000000001 | 0000000010 | 00011 | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ |
| 10055h | Wr | 00000000000000010 | 0000000010 | 10101 | $t_{SS}+t_B+t_{TM}+t_{SS}+t_{DM}$ | $t_{SS}+t_{OM}$ | $t_{SS}+t_B+t_{TM}+t_{SS}+t_{DM}+t_{OM}$ | $t_{SS}+t_{OM}$ |
| 8044h | Rd | 00000000000000001 | 0000000010 | 00100 | $t_{SS}$ | $t_{SS}$ | $t_{SS}$ | $t_{SS}$ |
| 1804Fh | Rd | 00000000000000011 | 0000000010 | 01111 | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ |
| 10056h | Rd | 00000000000000010 | 0000000010 | 10110 | $t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ |
| 005Bh | Rd | 00000000000000000 | 0000000010 | 11011 | $t_{SS}+2*t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ |
| 2805Eh | Wr | 00000000000000101 | 0000000010 | 11110 | $t_{SS}+t_B+t_{TM}+t_{SS}+t_{DM}$ | $t_{SS}+t_{OM}$ | $t_{SS}+t_B+t_{TM}+t_{SS}+t_{DM}+t_{OM}$ | $t_{SS}+t_{OM}$ |
| 10056h | Rd | 00000000000000010 | 0000000010 | 10110 | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}$ |
| 8045h | Rd | 00000000000000001 | 0000000010 | 00101 | $t_{SS}+2*t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ | $t_{SS}+t_B+t_{TM}+t_{SS}$ |
| 8046h | Wr | 00000000000000001 | 0000000010 | 00110 | $t_{SS}+t_{DM}$ | $t_{SS}+t_{DM}+t_{OM}$ | $t_{SS}+t_{DM}+t_{OM}$ | $t_{SS}+t_{DM}$ |

| **број сета** | **број улаза** | $D_2$ | $V_2$ | $TAG_2$ **МЕМОРИЈА** | **FIFO CNT** |
|---|---|---|---|---|---|
| $SET_{2h}$ | 0 | 1 | 1 | 00000000000000001 | 1 |
| | 1 | 0 | 1 | 00000000000000010 | |

Садржај релевантних делова кеш меморије

в) Време дохватања блока је: $t_B=32*(t_{OM}+t_{DM})$.

Укупно време је: $T=291*t_{DM}+288*t_{OM}+17*t_{SS}+7*t_{TM}$.

г)

| број сета | број улаза | $V_2$ | $TAG_2$ МЕМОРИЈА | FIFO CNT |
|---|---|---|---|---|
| $SET_{2h}$ | 0 | 1 | 00000000000000001 | 1 |
| | 1 | 1 | 00000000000000000 | |

Садржај релевантних делова кеш меморије

Време дохватања блока је: $t_B=32*(t_{OM}+t_{DM})$.

Укупно време је: $T=161*t_{DM}+163*t_{OM}+15*t_{SS}+5*t_{TM}$.

д)

| број сета | број улаза | $V_2$ | $TAG_2$ МЕМОРИЈА | FIFO CNT |
|---|---|---|---|---|
| $SET_{2h}$ | 0 | 1 | 00000000000000001 | 1 |
| | 1 | 1 | 00000000000000010 | |

Садржај релевантних делова кеш меморије

Време дохватања блока је: $t_B=32*(t_{OM}+t_{DM})$.

Укупно време је: $T=227*t_{DM}+227*t_{OM}+17*t_{SS}+7*t_{TM}$.

ђ)

| број сета | број улаза | $D_2$ | $V_2$ | $TAG_2$ МЕМОРИЈА | FIFO CNT |
|---|---|---|---|---|---|
| $SET_{2h}$ | 0 | 1 | 1 | 00000000000000001 | 1 |
| | 1 | 0 | 1 | 00000000000000000 | |

Садржај релевантних делова кеш меморије

Време дохватања блока је: $t_B=32*(t_{OM}+t_{DM})$.

Укупно време је: $T=161*t_{DM}+162*t_{OM}+15*t_{SS}+5*t_{TM}$.

е) Ова техника се користи да смањи **conflict misses** (директно и сет-асоцијативно). Кешу се додаје један мали потпуно асоцијативан кеш (victim кеш). Он садржи блокове који су били жртве (victims) pri **miss**-у. При **miss**-у се најпре врши провера у **victim**-кешу. Ако јесте, онда **кеш блок** и **victim блок** мењају место. Ако није у **victim** кешу тек онда се иде у оперативну меморију.

а4)

Структура адреса код L1 кеш меморије:

| Број блока | | Word (5 бита) |
|---|---|---|
| Tag (17 бита) | Set (10 бита) | |

Структура адреса код victim кеш меморије:

| Број блока | Word (5 бита) |
|---|---|
| Tag (27 бита) | |

б4)

| Адреса | Тип | Tag | Set | Word | Време |
|---|---|---|---|---|---|
| 8043h | Rd | 00000000000000001 | 0000000010 | 00011 | $t_{SS}+t_{SV}+t_B+t_{TM}+t_{SS}$ |

| | | | | | |
|---|---|---|---|---|---|
| 10055h | Wr | 0000000000000010 | 0000000010 | 10101 | $t_{SS}+t_{SV}+t_B+t_{TM}+t_{SS}$ |
| 8044h | Rd | 0000000000000001 | 0000000010 | 00100 | $t_{SS}$ |
| 1804Fh | Rd | 0000000000000011 | 0000000010 | 01111 | $t_{SS}+t_{SV}+t_B+t_{TM}+t_{SS}$ |
| 10056h | Rd | 0000000000000010 | 0000000010 | 10110 | $t_{SS}$ |
| 005Bh | Rd | 0000000000000000 | 0000000010 | 11011 | $t_{SS}+t_{SV}+t_B+t_{TM}+t_{SS}$ |
| 2805Eh | Wr | 0000000000000101 | 0000000010 | 11110 | $t_{SS}+t_{SV}+t_B+t_{TM}+t_{SS}$ |
| 10056h | Rd | 0000000000000010 | 0000000010 | 10110 | $t_{SS}+t_{SV}+2*t_{BL}+t_{TM}+t_{SS}$ |
| 8045h | Rd | 0000000000000001 | 0000000010 | 00101 | $t_{SS}+t_{SV}+2*t_{BL}+t_{TM}+t_{SS}$ |
| 8046h | Wr | 0000000000000001 | 0000000010 | 00110 | $t_{SS}+t_{DM}$ |

| број сета | број улаза | $D_2$ | $V_2$ | $TAG_2$ МЕМОРИЈА | FIFO CNT |
|---|---|---|---|---|---|
| $SET_{2h}$ | 0 | 1 | 1 | 0000000000000001 | 1 |
| | 1 | 1 | 1 | 0000000000000010 | |

Садржај релевантних делова L1 кеш меморије

| број улаза | D | V | TAG МЕМОРИЈА |
|---|---|---|---|
| 0 | 1 | 1 | 00000000000000101010000000010 |
| 1 | 0 | 1 | 00000000000000000000000000010 |
| 2 | 0 | 1 | 00000000000000011000000010 |
| 3 | 0 | 0 | |
| 4 | 0 | 0 | |
| ⋮ | ⋮ | ⋮ | |
| 8 | 0 | 0 | |

Садржај релевантних делова victim кеш меморије

в4) Време дохватања блока је: $t_B=32*(t_{OM}+t_{DM})$.

Укупно време је: $T=161*t_{DM}+160*t_{OM}+18*t_{SS}+7*t_{TM}+7*tSV+4*t_{BL}$.

## Задатак 14.

Разматра се процесор код кога је адреса ширине 32 бита а адресибилна јединица је 8 битна реч који има кеш меморију са следећим карактеристикама: Блок, односно кеш линија је величине 16 бајта; Кешу се приступа на нивоу 8 битне речи; Кеш меморија је сет асоцијативна са 32 сета, са четири блока по сету; Кеш подржава само write-back политику уписа; Провера валидности и модификованости се посматра на нивоу 4 бајта.

а) Нацртати структуру кеш меморије, означити ширину у битовима за све релевантне делове и укратко описати њихову намену.

б) Процесор генерише следећу секвенцу адреса са типом операције назначеним у загради после сваке адресе (R=read, W=write):

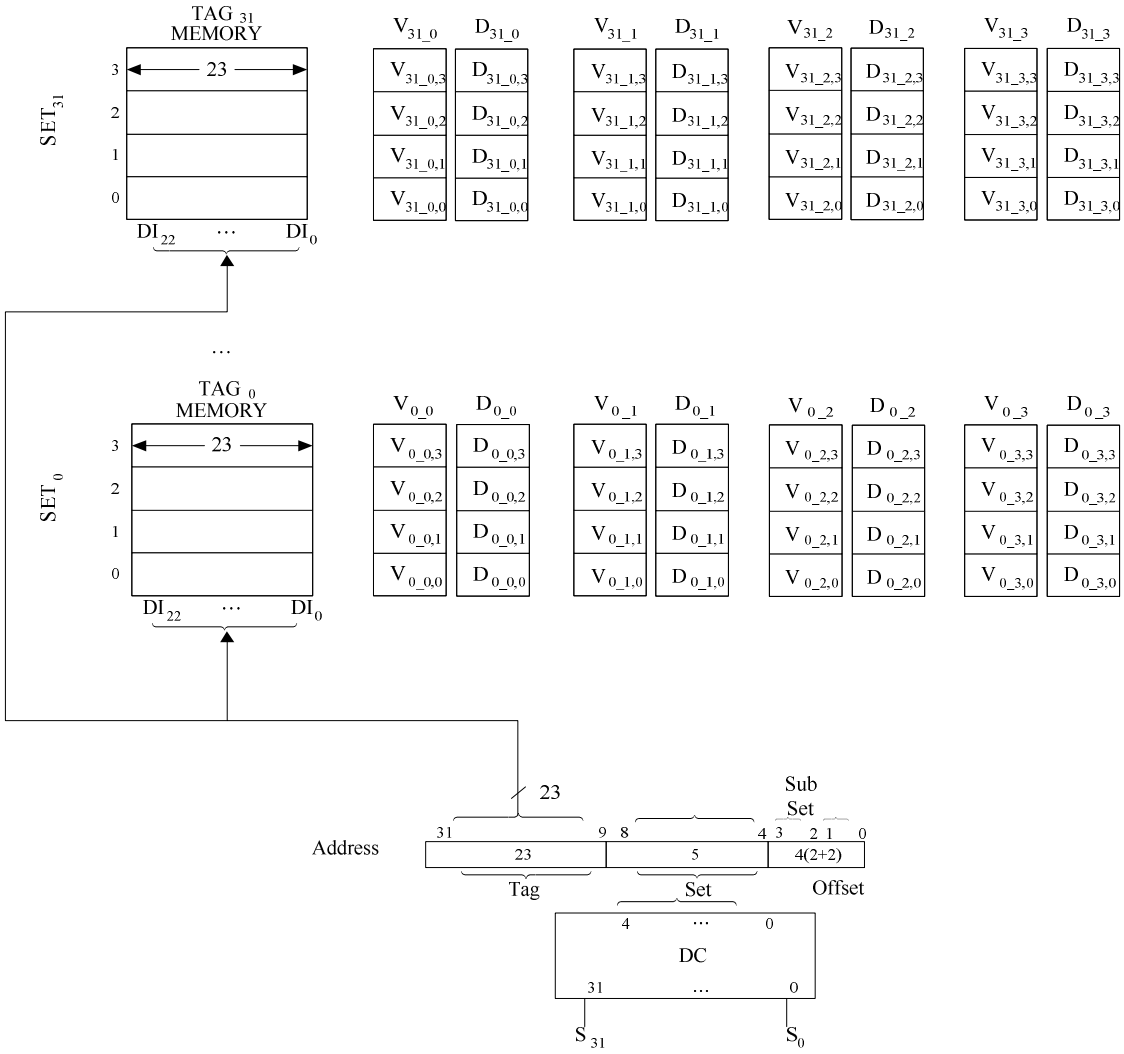
20Ah (R), A0Fh (W), 200h (W), FE00h (R), FE0Ch (R), C00Ah (W), A05h (R), B80Dh (R). Дати садржај свих улаза свих сетова који су реферисани у датој секвенци, као и вредности V (Valid) бита и D (Dirty), после завршетка дате секвенце, ако је алгоритам замене FIFO. Означити сетове.

в) Израчунати укупно време потребно за приступ кеш меморији за податке у датој секвенци. Треба претпоставити да се прво пренесе део блок из оперативне меморије у кеш меморије и обрнуто, па се тек онда приступа локацији. Приликом израчунавања времена потребног да се добије садржај узети у обзир само време утврђивања сагласности у TAG МЕМОРИЈИ ($t_{SA}$), време приступа оперативној меморији ($t_{OM}$), време приступа DATA МЕМОРИЈИ ($t_{DM}$) и време приступа TAG МЕМОРИЈИ ($t_{TM}$), занемарити времена потребна за остале активности.

г) Како би гласили одговори на питања а) – в) у случају да се валидност посматра на нивоу блока а модификованост на нивоу 4 бајта.

Решење:

а)




б)

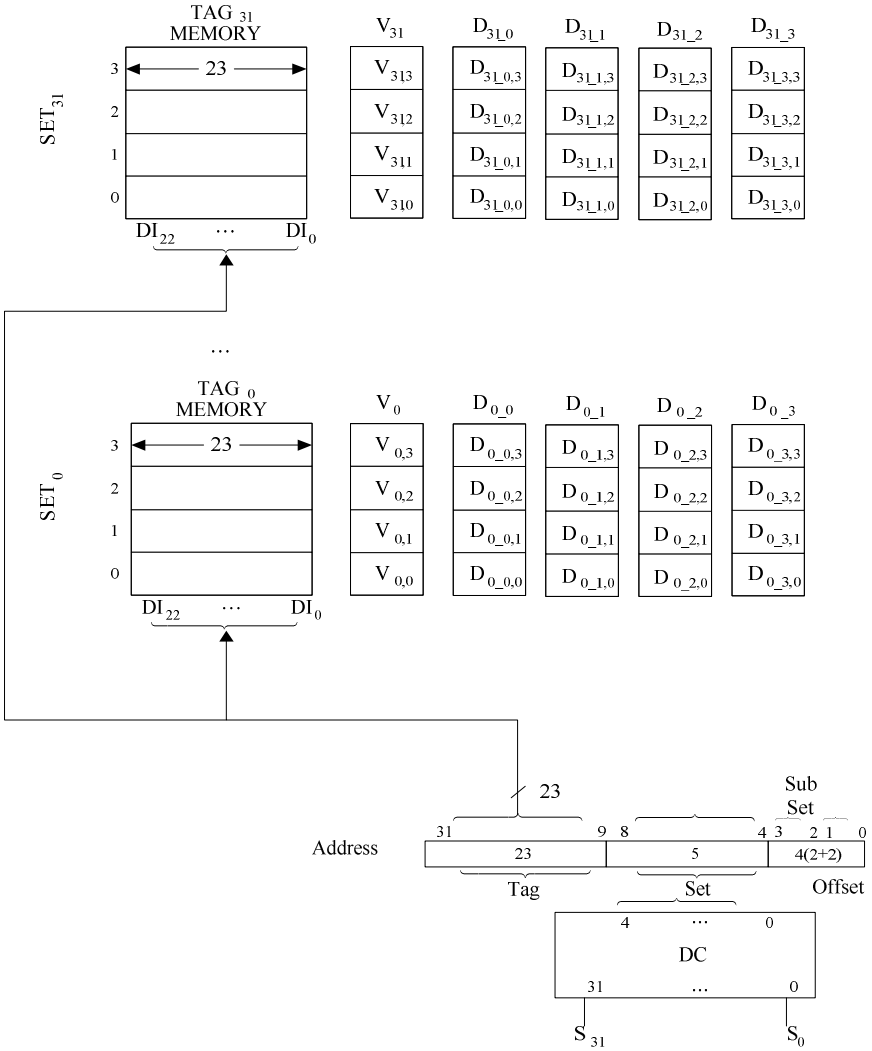
| Адреса | Тип | Tag | Set | Под сет | Word | Време |
|---|---|---|---|---|---|---|
| 20A | Rd | 00000000000000000001 | 00000 | 10 | 10 | $t_{SS}+t_{PB}+t_{TM}+t_{SS}$ |
| A0F | Wr | 00000000000000000101 | 00000 | 11 | 11 | $t_{SS}+t_{PB}+t_{TM}+t_{SS}+t_{DM}$ |
| 200 | Wr | 00000000000000000001 | 00000 | 00 | 00 | $t_{SS}+t_{PB}+t_{TM}+t_{SS}+t_{DM}$ |
| FE00 | Rd | 00000000000000001111111 | 00000 | 00 | 00 | $t_{SS}+t_{PB}+t_{TM}+t_{SS}$ |
| FE0C | Rd | 00000000000000001111111 | 00000 | 11 | 00 | $t_{SS}+t_{PB}+t_{TM}+t_{SS}$ |
| C00A | Wr | 00000000000000001100000 | 00000 | 10 | 10 | $t_{SS}+t_{PB}+t_{TM}+t_{SS}+t_{DM}$ |
| A05 | Rd | 00000000000000000101 | 00000 | 01 | 01 | $t_{SS}+t_{PB}+t_{TM}+t_{SS}$ |
| B80D | Rd | 00000000000000001011100 | 00000 | 11 | 01 | $t_{SS}+2*t_{PB}+t_{TM}+t_{SS}$ |

| **број сета** | **број улаза** | **D0** | **V0** | **D1** | **V1** | **D2** | **V2** | **D3** | **V3** | **TAG МЕМОРИЈА** |
|---|---|---|---|---|---|---|---|---|---|---|
| $SET_{0h}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 00000000000000001011100 |
| | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 00000000000000000101 |
| | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 00000000000000001111111 |
| | 3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 00000000000000001100000 |

Садржај релевантних делова кеш меморије

в) Време дохватања под блока је: $t_{PB}=4\cdot(t_{OM}+t_{DM})$, док је време дохватања блока $t_B=4\cdot(t_{OM}+t_{DM})$.

а2)




б2)

| Адреса | Тип | Tag | Set | Под сет | Word | Време |
|---|---|---|---|---|---|---|
| 20A | Rd | 00000000000000000001 | 00000 | 10 | 10 | $t_{SS}+t_B+t_{TM}+t_{SS}$ |
| A0F | Wr | 00000000000000000101 | 00000 | 11 | 11 | $t_{SS}+t_B+t_{TM}+t_{SS}+t_{DM}$ |
| 200 | Wr | 00000000000000000001 | 00000 | 00 | 00 | $t_{SS}+t_{DM}$ |
| FE00 | Rd | 00000000000000001111111 | 00000 | 00 | 00 | $t_{SS}+t_B+t_{TM}+t_{SS}$ |
| FE0C | Rd | 00000000000000001111111 | 00000 | 11 | 00 | $t_{SS}$ |
| C00A | Wr | 00000000000000001100000 | 00000 | 10 | 10 | $t_{SS}+t_B+t_{TM}+t_{SS}+t_{DM}$ |
| A05 | Rd | 00000000000000000101 | 00000 | 01 | 01 | $t_{SS}$ |
| B80D | Rd | 00000000000000001011100 | 00000 | 11 | 01 | $t_{SS}+t_B+t_{PB}+t_{TM}+t_{SS}$ |

| број сета | број улаза | D0 | D1 | D2 | D3 | V | TAG МЕМОРИЈА |
|---|---|---|---|---|---|---|---|
| $SET_{0h}$ | 0 | 0 | 0 | 0 | 0 | 1 | 00000000000000001011100 |
| | 1 | 0 | 0 | 0 | 1 | 1 | 00000000000000000000101 |
| | 2 | 0 | 0 | 0 | 0 | 1 | 00000000000000001111111 |
| | 3 | 0 | 0 | 1 | 0 | 1 | 00000000000000001100000 |

Садржај релевантних делова кеш меморије

в2) Време дохватања под блока је: $t_{PB}=4\cdot(t_{OM}+t_{DM})$, док је време дохватања блока $t_B=4\cdot(t_{OM}+t_{DM})$.

## Задатак 15.

Посматра се кеш меморија са look through приступом приликом читања. Време приступа кеш меморији је 2 ns док је Miss penalty 8 ns.

а) Уколико је вероватноћа да се податак налази у кеш меморији 0.9 колико износи просечно време приступа меморији?

б) Уколико би се величина блока дуплирала вероватноћа поготка би се повећала на 0.95. За колико процената би се смањило просечно време приступа меморији?

в) Како би се променило време приступа меморији уколико би се користила техника look aside приликом читања?

г) Који су недостатци look aside технике приступа меморији?

Решење:
Просечно време приступа кеш меморији укључује време утврђивања сагласности и приступа подацима у кеш емморији.

CacheCycleTime=2 ns

MissPenalty=8 ns

а) look-trough read

HitRate=0.9

AverageMemoryAccessTime=HitTime+MissPenalty * MissRate

MissRate=1 - HitRate

AverageMemoryAccessTime=CacheCycleTime+MissPenalty * (1 - HitRate)

AverageMemoryAccessTime=2+8 * 0.1=2.8 ns

б) look-trough read

HitRate=0.95

AverageMemoryAccessTime=HitTime+MissPenalty * MissRate

AverageMemoryAccessTime=CacheCycleTime+MissPenalty * MissRate

MissRate=1 - HitRate

AverageMemoryAccessTimeDoubled=2+8 * 0.05=2.4 ns

$$\frac{\text{AverageMemoryAccessTime - AverageMemoryAccessTimeDoubled}}{\text{AverageMemoryAccessTimeDoubled}} \cdot 100 = 16.67\%$$

$$\frac{\text{AverageMemoryAccessTime - AverageMemoryAccessTimeDoubled}}{\text{AverageMemoryAccessTime}} \cdot 100 = 14.29\%$$

в) look-aside read

HitRate=0.9

AverageMemoryAccessTime=CacheCycleTime * HitRate+MissPenalty * MissRate

AverageMemoryAccessTime=CacheCycleTime * HitRate+MissPenalty * (1 - HitRate)

AverageMemoryAccessTime=2 * 0.9+8 * (1 - 0.9)=2.6 ns

г) Приликом коришћења технике look-aside податак се тражи и у кеш меморији и у главној меморији у паралели. Уколико се податак открије у кеш меморији циклус према главној меморији се отказује. Главни недостатак овог приступа је повећан саобраћај на магистрали.

## Задатак 16.

Посматра се кеш меморија капацитета 10KB чије је време приступа речи 0.2 ns. Она се користи за убрзавање приступа главној RAM меморији капацитета 4 GB, са временом приступа 10 ns. Величина једне кеш линије је 16 речи, а вероватноћа да се податак нађе у кеш меморији (hit rate) је 0.95. Уколико се реч не налази у меморији прво се довлачи тражени податак па тек онда преостале речи из кеш линије. У свим случајевима претпоставити write-trough политику уписа. Колико је просечно време приступа кешу за случај:

а) операције читања код look-through приступом приликом читања?

б) операције читања код look-aside приступом приликом читања?

в) операције уписа код write-trough приступом приликом уписа?

Решење:

CacheSize=10 KB

CacheCycleTime=0.2 ns

MemorySize=4 GB

MemoryCycleTime=10 ns

CacheLineSize=16 W

HitRate=0.95

а) look-trough

AverageMemoryAccessTime=HitTime+MissPenalty * MissRate

MissRate=1 – HitRate

MissPenalty=MemoryCycleTime

HitTime=CacheCycleTime

AverageMemoryAccessTime=CacheCycleTime+MemoryCycleTime * (1 - HitRate)

AverageMemoryAccessTime=0.2+10 * (1 - 0.95)=0.7 ns

б) look-aside

AverageMemoryAccessTime=CacheCycleTime * HitRate+MemoryCycleTime * (1 - HitRate)

AverageMemoryAccessTime=0.2 * 0.95+10 * (1 - 0.95)=0.69 ns

в) write-trough write са write allocated политиком довлачења

AverageMemoryAccessTime=MemoryCycleTime * HitRate+

+(MemoryCycleTime+MemoryCycleTime) * (1 - HitRate)

AverageMemoryAccessTime=MemoryCycleTime+MemoryCycleTime * (1 - HitRate)

AverageMemoryAccessTime=10.5 ns

c2) write-trough write са non-write allocated политиком довлачења

AverageMemoryAccessTime=MemoryCycleTime * HitRate+

+MemoryCycleTime * (1 - HitRate)=MemoryCycleTime

## Задатак 17.

Посматра се кеш меморија капацитета 2MB чије је време приступа речи 0.2 ns. Она се користи за убрзавање приступа главној RAM меморији капацитета 1 GB, са временом приступа једној речи је 2 ns. Из меморије се чита и у меморију уписује реч по реч. Величина једне кеш линије је 8 речи и цела линија се довлачи/одвлачи у случају било ког промашаја, а вероватноћа да се податак нађе у кеш меморији (hit rate) је 0.95. Вероватноћа операције уписа је 30%. Вероватноћа да је кеш линија модификована износи 40%. Приликом операције уписа се користи write allocate политика. Колико је просечан број приступа меморији за случај:

а) кеш меморије са look-through приступом?

б) кеш меморије са look-aside приступом?

Размотрити решења за случај write through и write back начина уписа са write allocated политиком.

Решење:

CacheSize=2 MB

CacheCycleTime=0.2 ns

MemorySize=1 GB

MemoryCycleTime=2 ns

CacheLineSize=8 W

HitRate=0.95

Write=30%

Modified=40%

а) look-trough са write throug

Read+Write=1

| Hit? | Операција | Фреквенција | Број приступа меморији |
|---|---|---|---|
| + | Read | Read * HitRate=0.665 | 0 |
| - | Read | Read * (1 – HitRate)=0.035 | CacheLineSize/W=8 |
| + | Write | Write * HitRate=0.285 | 1 |
| - | Wtite | Write * (1 – HitRate)=0.015 | CacheLineSize/W+1=9 |

AverageAccessesNumber=0.665 * 0+0.035 * 8+0.285 * 1+0.015 * 9=0.7

look-trough са write back

| Hit? | Операција | Фреквенција | Број приступа меморији |
|---|---|---|---|
| + | Read | Read * HitRate=0.665 | 0 |
| - | Read | Read * (1 – HitRate)=0.035 | CacheLineSize/W * (1+Modifier)=11.2 |
| + | Write | Write * HitRate=0.285 | 0 |
| - | Wtite | Write * (1 – HitRate)=0.015 | CacheLineSize/W * (1+Modifier)=11.2 |

AverageAccessesNumber=0.665 * 0+0.035 * 11.2+0.285 * 0+0.015 * 11.2=0.56

б) look-aside са write throug

| Hit? | Операција | Фреквенција | Број приступа меморији |
|---|---|---|---|
| + | Read | Read * HitRate=0.665 | 1 |
| - | Read | Read * (1 – HitRate)=0.035 | CacheLineSize/W=8 |
| + | Write | Write * HitRate=0.285 | 1 |

| | | | |
|---|---|---|---|
| - | Wtite | Write * (1 – HitRate)=0.015 | CacheLineSize/W+1=9 |

AverageAccessesNumber=0.665 * 1+0.035 * 8+0.285 * 1+0.015 * 9=1.365

look-aside ca write back

| Hit? | Операција | Фреквенција | Број приступа меморији |
|---|---|---|---|
| + | Read | Read * HitRate=0.665 | 1 |
| - | Read | Read * (1 – HitRate)=0.035 | CacheLineSize/W * (1+Modifier)=11.2 |
| + | Write | Write * HitRate=0.285 | 0 |
| - | Wtite | Write * (1 – HitRate)=0.015 | CacheLineSize/W * (1+Modifier)=11.2 |

AverageAccessesNumber=0.665 * 1+0.035 * 11.2+0.285 * 0+0.015 * 11.2=1.225

## Задатак 18.

Посматра се кеш меморија капацитета 128KB чије је време приступа речи 1 такт. Фреквенција система је 2 GHz. Она се користи за убрзавање приступа главној RAM меморији капацитета 1 GB. Време одзива меморије 50 тактова, брзина трансфера 4 бајта по такту. Величина једне кеш линије је 16 B и цела линија се довлачи/одвлачи у случају било ког промашаја, а вероватноћа да се податак нађе у кеш меморији (hit rate) је 0.90. Вероватноћа операције уписа је 35%. Вероватноћа да је кеш линија модификована износи 30%. Приликом операције уписа се користи non-write allocate политика и користи се write back техником уписа. Колико је просечан број приступа меморији за случај:

а) кеш меморије са look-through приступом?

б) кеш меморије са look-aside приступом?

Решење:

f=2 GHz

Write=0.3

CacheSize=128 KB

HitRate=0.9

MemoryDelayCycles=50

MemoryTransferRatePerCycle=4 B

Modified=0.3

CachLIneSize=16 B

a)

AverageMemoryAccessTimeR=HitTime+MissPenalty * MissRate

MissRate=1 – HitRate=0.1

MissPenaltyCycles=(MemoryDelayCycles+CacheLineSize/W) * (1+Modifier)

MissPenalty=MissPenaltyCycles / f

MissPenalty=(50+16/4) * (1+0.3) * 0.5=35.1 ns

HitTime=CacheCycleTime=0.5 ns

AverageMemoryAccessTimeR=0.5+35.1 * 0.1=4.01 ns

AverageMemoryAccessTimeW=HitTime+MissPenalty * MissRate

MissRate=1 – HitRate=0.1

MissPenaltyCycles=MemoryDelayCycles+1

MissPenalty=MissPenaltyCycles / f

MissPenalty=((50+16/4) * 0.3)+50+1) * 0.5=25.5 ns

HitTime=CacheCycleTime=0.5 ns

AverageMemoryAccessTimeW=0.5+25.5 * 0.1=3.05 ns

AverageMemoryAccessTime=Read*AverageMemoryAccessTimeR+Write*AverageMemoryAccessTimeW

AverageMemoryAccessTime=0.7 * 4.01+0.3 * 3.05=3.722 ns

б)

AverageMemoryAccessTimeR=HitTime * HitRate+MissPenalty * MissRate

MissRate=1 – HitRate=0.1

MissPenaltyCycles=(MemoryDelayCycles+CacheLineSize/W) * (1+Modifier)

MissPenalty=MissPenaltyCycles / f

MissPenalty=(50+16/4) * (1+0.3) * 0.5=35.1 ns

HitTime=CacheCycleTime=0.5 ns

AverageMemoryAccessTimeR=0.5 * 0.9+35.1 * 0.1=3.96 ns

AverageMemoryAccessTimeW=3.05 ns

AverageMemoryAccessTime=Read*AverageMemoryAccessTimeR+Write*AverageMemoryAccessTimeW

AverageMemoryAccessTime=0.7 * 4.01+0.3 * 3.05=3.687 ns

# ВИРТУЕЛНА МЕМОРИЈА (VIRTUAL MEMORY)

## Задатак 1.

Посматра се рачунар са страничном организацијом виртуелне меморије. Виртуелни адресни простор корисника је 16 MB и подељен је на странице величине 1 KB. Реални адресни простор је 1 MB и подељен је на блокове величине 1 KB. Виртуелном и реалном адресом се адресирају речи дужине 1 бајт.

Урадити следеће:

а) Означити дужине у битовима свих делова виртуелне и реалне адресе.

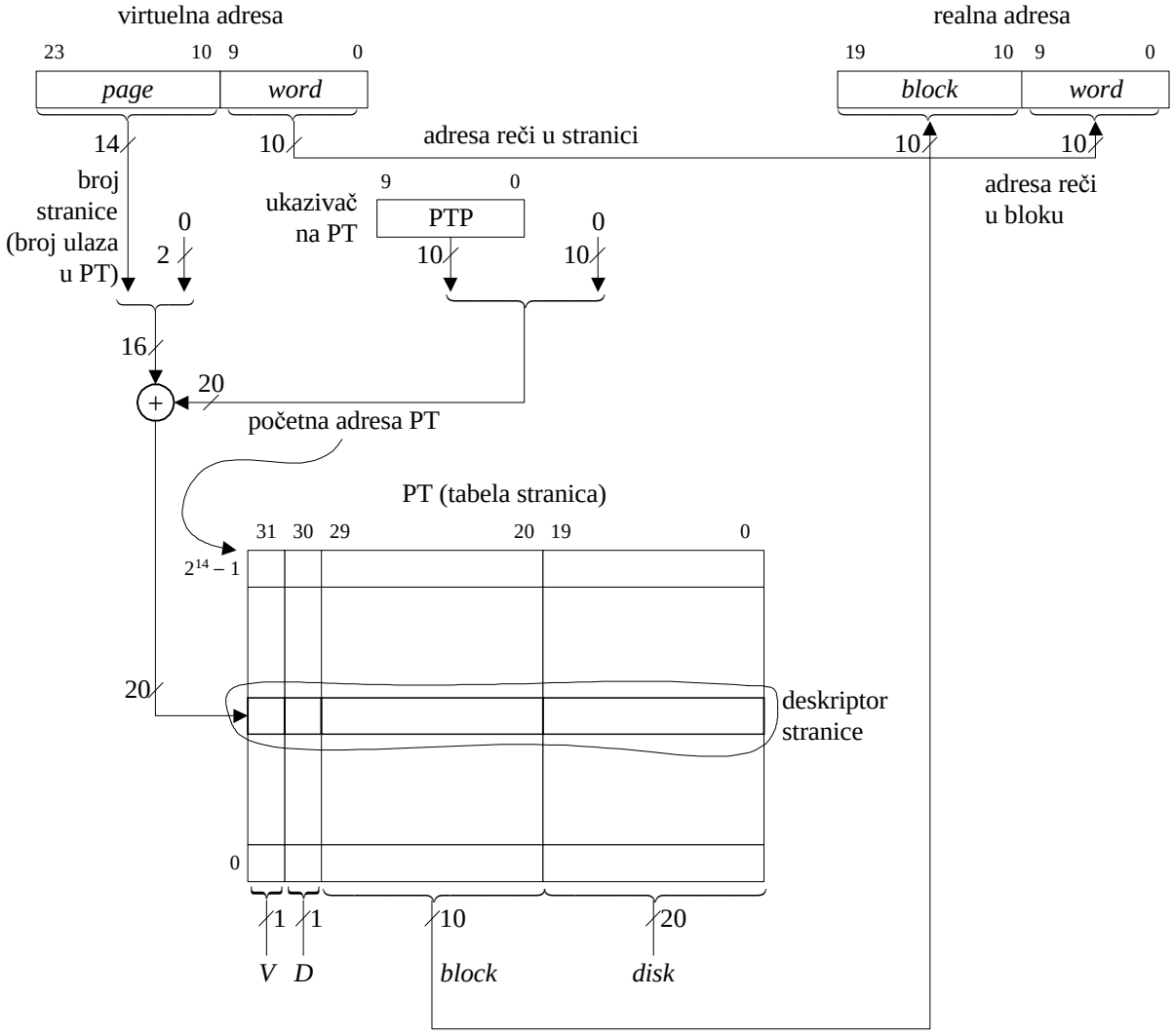
б) Нацртати табелу страница, објаснити како се приступа табелама страница различитих корисника и дати функцију појединих поља дескриптора странице.

в) Објаснити како се врши пресликавање виртуелне у реалну адресу за случај да је страница у меморији и навести шта се од тога ради хардверски а шта софтверски.

г) Објаснити шта се ради за случај да страница није у меморији и навести шта се од тога ради хардверски а шта софтверски.

Решење:

а) Структура виртуелне и реалне адресе и дужине у битовима делова виртуелне и реалне адресе су приказани на слици.



## Задатак 2.

У рачунару са виртуелном меморијом страничне организације дефинисаном у претходном задатку постоји јединица за убрзавање пресликавања виртуелних у реалне адресе.

а) Узети да рачунар поседује јединицу за убрзавање пресликавања виртуелних у реалне адресе, реализовану у техници асоцијативног пресликавања у којој се истовремено чувају релевантни делови дескриптора 512 најчешће коришћених страница до 16 корисника. Урадити следеће:

а1) Нацртати, означити дужину у битовима и објаснити функцију свих релевантних делова јединице за убрзавање пресликавања.

а2) Означити дужине у битовима релевантних делова виртуелне и реалне адресе и објаснити како се:

проверава да ли се релевантни дескриптор налази у јединици за убрзавање пресликавања;

долази до дескриптора уколико се утврди да се налази у јединици за убрзавање пресликавања;

формира реална адреса;

у јединици за убрзавање пресликавања обезбеђује да се странице различитих корисника које имају исти број не пресликавају у исти, већ свака у свој блок.

а3) У случају да се за генерисану виртуелну адресу утврди да се релевантни дескриптор не налази у јединици за убрзавање пресликавања, објаснити шта се ради у случају:

када је релевантна страница у меморији;

када релевантна страница није у меморији и

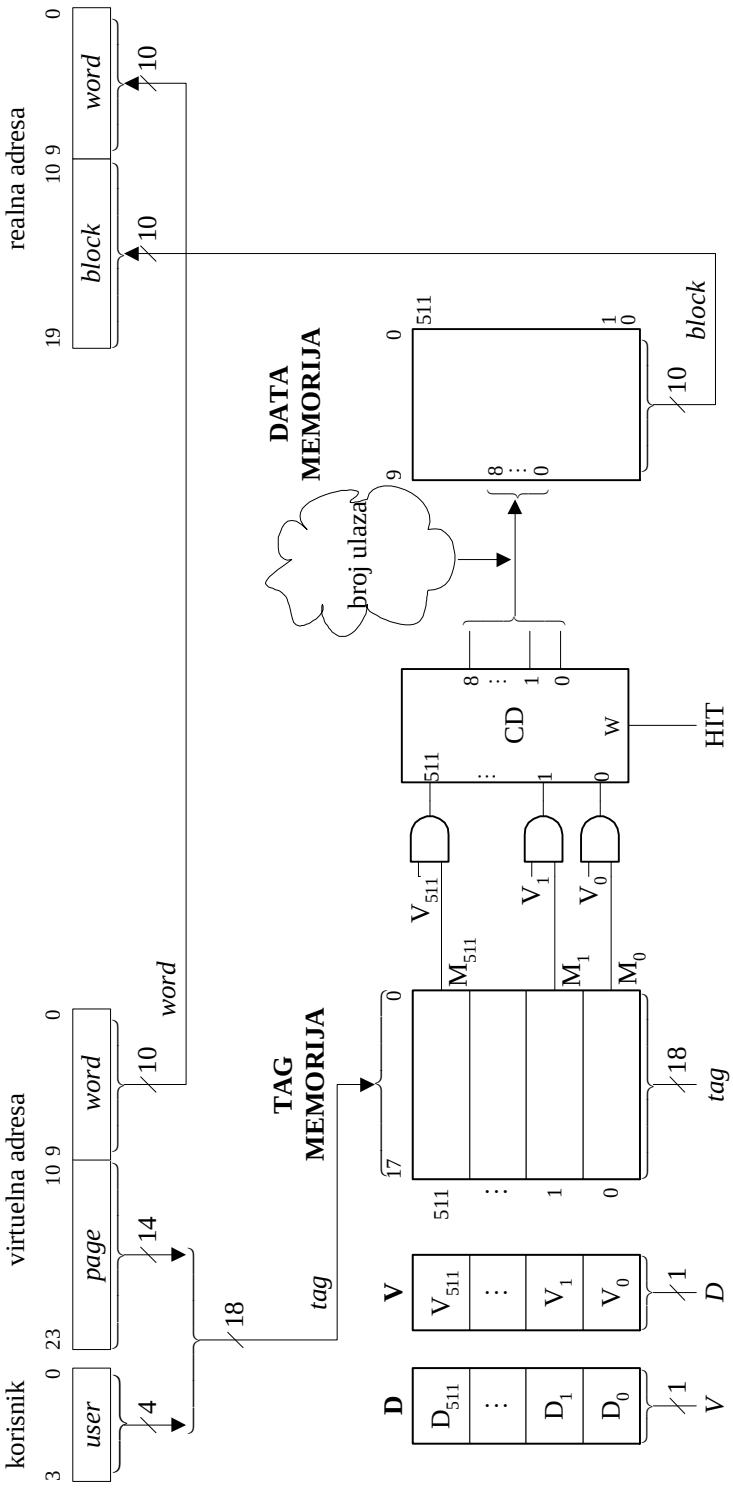
навести шта се од тога ради хардверски, а шта софтверски.

б) Узети да рачунар поседује јединицу за убрзавање пресликавања виртуелних у реалне адресе, реализовану у техници директног пресликавања у којој се истовремено чувају релевантни делови дескриптора 2048 најчешће коришћених страница до 16 корисника. Урадити исто као под а) овог задатка.

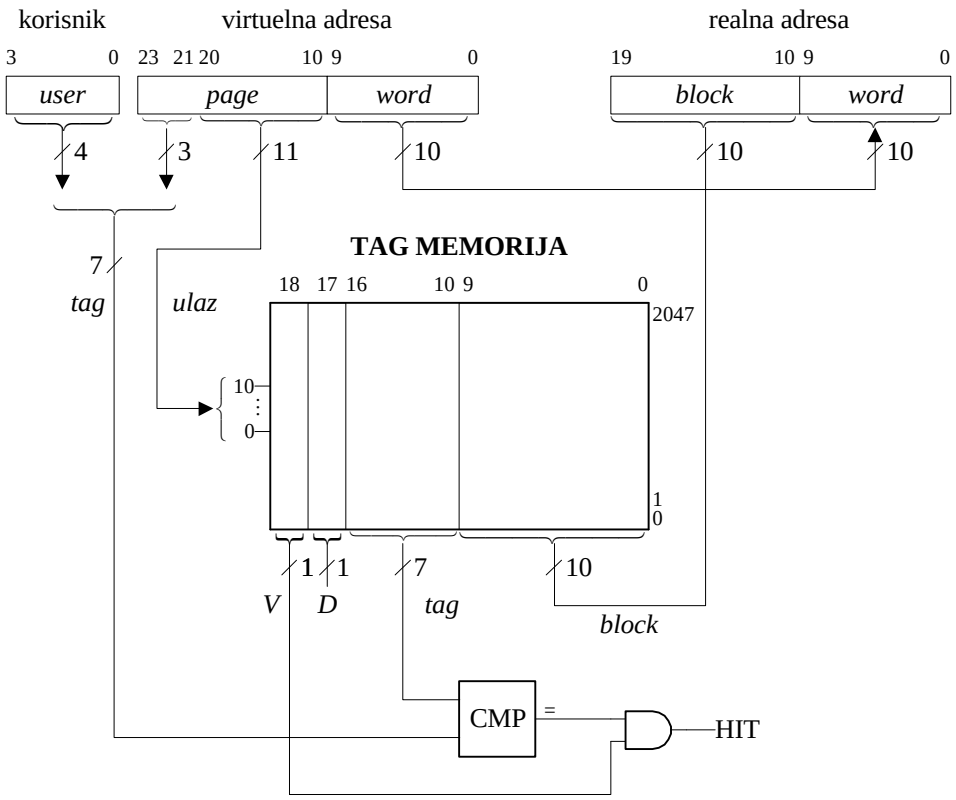
в) Узети да рачунар поседује јединицу за убрзавање пресликавања виртуелних у реалне адресе, реализовану у техници сет-асоцијативног пресликавања са два улаза по сету у којој се истовремено чувају релевантни делови дескриптора 2048 најчешће коришћених страница до 16 корисника. Урадити исто као под а) овог задатка.

Решење:

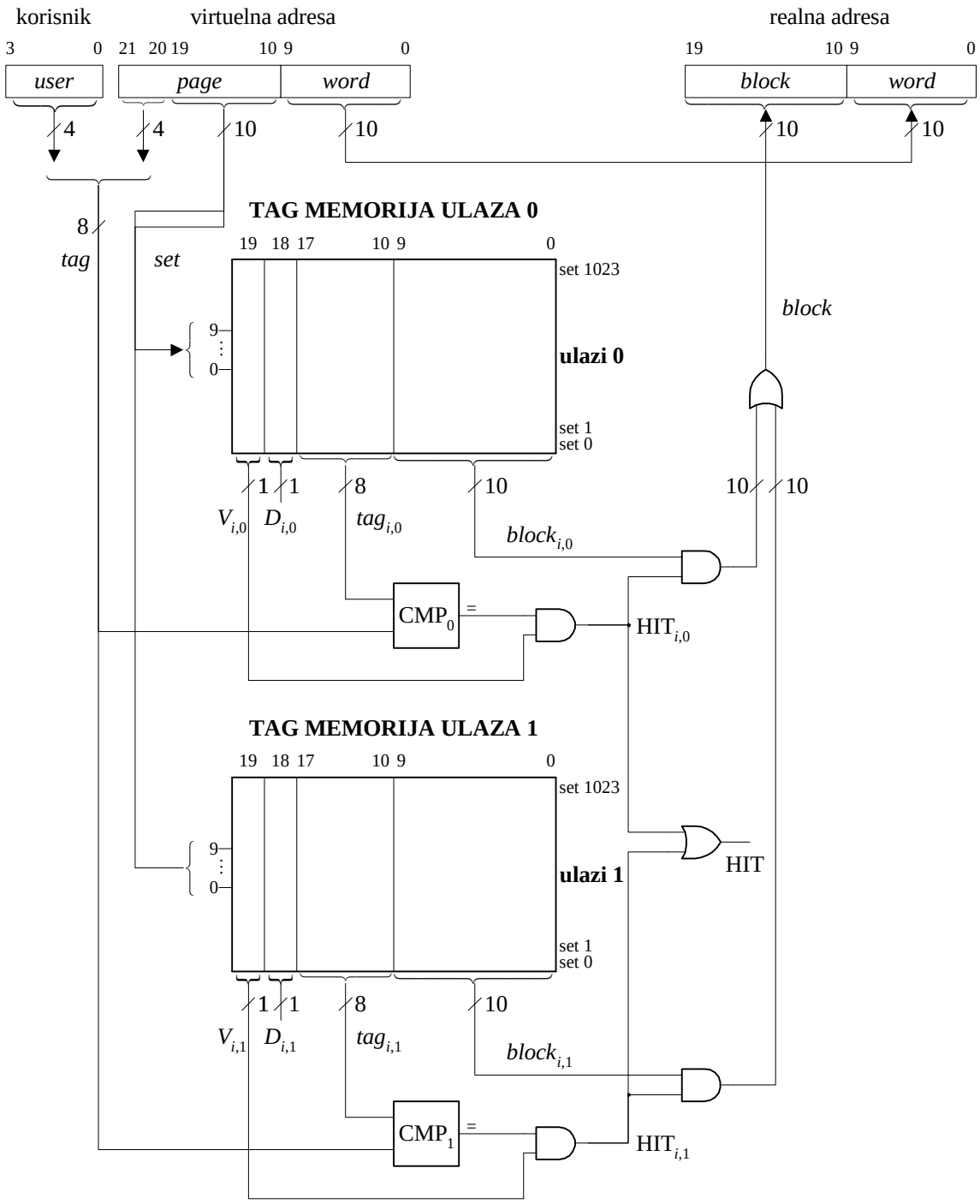
а)


realna adresa
block
word
DATA MEMORIJA
broj ulaza
CD
HIT
TAG MEMORIJA
tag
virtuelna adresa
page
korisnik
user
D
V

б)


korisnik
virtuelna adresa
realna adresa
3 0
23 21 20 10 9 0
19 10 9 0
user
page
word
block
word
4
3
11
10
10
10
TAG MEMORIJA
7
tag
ulaz
18 17 16 10 9 0
2047
10
0
1
0
1
1
7
10
V
D
tag
block
CMP
=
HIT

в)




## Задатак 3.

Виртуелна меморија капацитета 8K речи, а адресирање је на нивоу 16-битних речи. Процесор оперише само са 16-битним целобројним величинама (у даљем тексту *реч* означава 16-битну величину). Физичка адресни простор је величине 4 KW (4 кило речи). Виртуелна меморија организована је на страничном принципу, при чему је величина странице односно блока 1 KW. Посматра се програмски сегмент који приступа следећим виртуалним страницама: 4, 2, 0, 1, 2, 6, 1, 4, 0, 1, 0, 2, 3, 5, 7.

а) Приказати логичку структуру виртуелне адресе и означити значење и дужину сваког поља.

б) Приказати логичку структуру реалне адресе и означити значење и дужину сваког поља.

в) Утврдити које 4 странице се налазе у оперативној меморији у после сваког приступа страници уколико се користи FIFO алгоритам замене.

Решење:

VirtualAddressSpace=8K

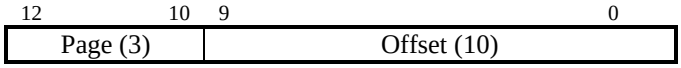
MainMemory=4K //Physical Address Space

PageSize=1K

BlockSize=1K

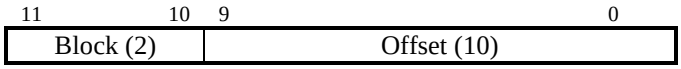
а)

Виртуална адреса:




б)

Реална адреса:




Постоји 4 блока у физичкој меморији и 8 у виртуелној меморији.

в)

4 2 0 1 2 6 1 4 0 1 0 2 3 5 7

Корак 1 тражи се страница 4. У меморији ће се налазити странице: 4.

Корак 2 тражи се страница 2. У меморији ће се налазити странице: 2, 4.

Корак 3 тражи се страница 0. У меморији ће се налазити странице: 0, 2, 4.

Корак 4 тражи се страница 1. У меморији ће се налазити странице: 1, 0, 2, 4.

Корак 5 тражи се страница 2. У меморији ће се налазити странице: 1, 0, 2, 4.

Корак 6 тражи се страница 6. У меморији ће се налазити странице: 6, 1, 0, 2.

Корак 7 тражи се страница 1. У меморији ће се налазити странице: 6, 1, 0, 2.

Корак 8 тражи се страница 4. У меморији ће се налазити странице: 4, 6, 1, 0.

Корак 9 тражи се страница 0. У меморији ће се налазити странице: 4, 6, 1, 0.

Корак 10 тражи се страница 1. У меморији ће се налазити странице: 4, 6, 1, 0.

Корак 11 тражи се страница 0. У меморији ће се налазити странице: 4, 6, 1, 0.

Корак 12 тражи се страница 2. У меморији ће се налазити странице: 2, 4, 6, 1.

Корак 13 тражи се страница 3. У меморији ће се налазити странице: 3, 2, 4, 6.

Корак 14 тражи се страница 5. У меморији ће се налазити странице: 5, 3, 2, 4.

Корак 15 тражи се страница 7. У меморији ће се налазити странице: 7, 5, 3, 2.

## Задатак 4.

Процесор подржава постојање више програма у меморији истовремено, до 16 корисника. Рачунар поседује виртуелну меморију, код које 32-битне адресе одговарају виртуелном адресном простору, а физички адресни простор је величине 64 KW (64 кило речи). Адресирање је на нивоу 32-битних речи. Виртуелна меморија организована је на страничном принципу, при чему је величина странице 256 W (256 речи). Процесор поседује посебан хардвер за убрзавање пресликавања виртуелних у реалне (физичке) адресе (у даљем тексту TLB). TLB је реализован са директним пресликавањем, при чему је величина простора за информациони садржај TLB-а укупно 16 B (бајта). Информациони садржај је само број блока у физичкој меморији. Процесор извршава следећи део програма корисника са идентификатором 3:

| Адреса (hex): | Наредба: | Коментар: |
|---|---|---|
| 32F00 | LOAD R0, #1234h | ; R0:=1234h |
| 32F02 | PUSH R0 | ; Mem[SP--]:=R0 |
| 32F03 | STORE 27001h, R0 | ; Mem[27001h]:=R0 |

Слободни део оперативне меморије обухвата блокове почев од F0h закључно са FFh. Ови блокови попуњавају се редом. Ниједна од страница датог корисника није у оперативној меморији, а SP има вредност 11D15h пре извршавања датог дела програма. Претпоставити да се приликом промене контекста садржај TLB јединице неће променити.

а) Приказати логичку структуру виртуелне и физичке адресе и дати значење и дужину сваког поља.

б) Приказати шта представља улазни податак који процесор даје TLB-у, шта је кључ за претрагу у TLB, и како се добија улаз у TLB у коме се тај кључ тражи.

в) За сваки приступ меморији означити: виртуелну адресу којој се приступа, тип операције (Rd, Wr, Ex), идентификацију корисника, вредност поља Page и Word, вредност поља Tag и Entry у TLB јединици, коментар да ли је било сагласности у TLB јединици, блок оперативне меморије Block, физичку адресу којој се приступило или коментар уколико је дошло до прекида. Одговор дати табеларно.

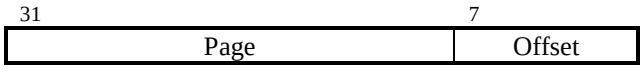
| Виртуелна адреса | Тип | User | Page | Word | Tag | Entry | Коментар | Block | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

г) Приказати вредности кључева и информационог садржаја релевантних улаза у TLB (означити те улазе), после извршавања датог дела програма.

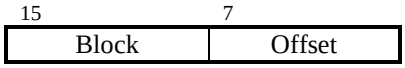
д) Приказати изглед табеле страница након извршавања датог програмског сегмента.

Решење:

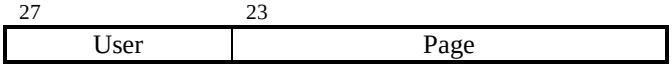
а) Логичка структура виртуелне адресе је:




Логичка структура физичке адресе је:




б) Улазни податак за TLB је тако:




Број улаза у TLB је одређен помоћу 4 најнижих бита улазног податка (то су 4 најнижих бита поља Page), док је кључ за претрагу виших 24 бита улазног податка:

Кључ за претрагу (Tag): *User*(4):*Page*(23…4)

Улаз у TLB: *Page*(3…0)

в)

| Виртуелна адреса | Тип | User | Page | Word | Tag | Entry | Коментар | Block | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|
| 32F00 | Ex | 3 | 00032F | 00 | 300032 | F | Miss | - | Page Fault |
| 32F00 | Ex | 3 | 00032F | 00 | 300032 | F | Miss | F0 | F000 |
| 32F01 | Ex | 3 | 00032F | 01 | 300032 | F | Hit | F0 | F001 |
| 32F02 | Ex | 3 | 00032F | 02 | 300032 | F | Hit | F0 | F002 |
| 11D15 | Wr | 3 | 00011D | 15 | 300011 | D | Miss | - | Page Fault |
| 32F02 | Ex | 3 | 00032F | 02 | 300032 | F | Hit | F0 | F002 |
| 11D15 | Wr | 3 | 00011D | 15 | 300011 | D | Miss | F1 | F115 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 32F03 | Ex | 3 | 00032F | 03 | 300032 | F | Hit | F0 | F003 |
| 32F04 | Ex | 3 | 00032F | 03 | 300032 | F | Hit | F0 | F004 |
| 27001 | Wr | 3 | 000270 | 01 | 300027 | 0 | Miss | - | Page Fault |
| 32F03 | Ex | 3 | 00032F | 03 | 300032 | F | Hit | F0 | F003 |
| 32F04 | Ex | 3 | 00032F | 03 | 300032 | F | Hit | F0 | F004 |
| 27001 | Wr | 3 | 000270 | 01 | 300027 | 0 | Miss | F2 | F201 |

г)

| **Кључеви (*Tag* меморија)** | | | | **Информациони садржај** |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Tag** | **Блок** |
| 0 | 1 | 1 | 300027 | F2 |
| D | 1 | 1 | 300011 | F1 |
| F | 1 | 0 | 300032 | F0 |

д)

| **Табела страница корисника 3** | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| ... | | | | - |
| 11D | 1 | 1 | F1 | - |
| ... | | | | - |
| 270 | 1 | 1 | F2 | - |
| ... | | | | - |
| 32F | 1 | 0 | F0 | - |
| ... | | | | - |

## Задатак 5.

Процесор подржава виртуелну организацију меморије са страничним пресликавањем. Странице су дужине $2^{16}$ речи. Капацитет физичке меморије је највише $2^{15}$ блокова. Меморијске (физичке) адресе су ширине 31 бит, ширина магистрале података је 16 бита, а адресирање је на нивоу 16-битних речи. Корисничке логичке (виртуелне) адресе су 24-битне, а процесор подржава рад 256 корисничких програма.

Процесор поседује и хардвер за убрзавање пресликавања, у даљем тексту TLB. TLB је организован са сет-асоцијативним пресликавањем, поседује два улаза по сету и 64 сета. Алгоритам замене је FIFO. Кориснички програм са бројем D0h генерише следећу секвенцу виртуелних адреса (све вредности су хексадецималне):

278800 (Ex), A700FF(Rd), 278801(Ex), E78800(Rd), B600FA(Wr), 278802(Ex), B600FF(Rd), E78802(Rd)

Слободни део оперативне меморије обухвата блокове почев од 00F0h закључно са 00FFh. Ови блокови попуњавају се редом. Изглед табеле страница датог корисника пре почетка дате секвенце:

| **Табела страница корисника D0h** | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| ... | | | | - |
| 27 | 1 | 0 | 0037h | - |
| ... | | | | - |
| E7 | 1 | 1 | 004Eh | - |
| ... | | | | - |

Остале странице корисника се не налазе у оперативној меморији, пре дате секвенце TLB је била празна. Претпоставити да се приликом промене контекста садржај TLB јединице неће променити.

а) Приказати логичку структуру виртуелне адресе и означити значење и дужину сваког поља.

б) Приказати шта представља улазни податак који процесор даје TLB-у, шта је кључ за претрагу у TLB, и како се добија сет у TLB у коме се тај кључ тражи.

в) За сваки приступ меморији означити: виртуелну адресу којој се приступа, тип операције (Rd, Wr), идентификацију корисника, вредност поља Page и Word, вредност поља Tag и Set у TLB јединици, коментар да ли је било сагласности у TLB јединици, блок оперативне меморије Block, физичку адресу којој се приступило или коментар уколико је дошло до прекида. Одговор дати табеларно.

| Виртуелна адреса | Тип | User | Page | Word | Tag | Set | Коментар | Block | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

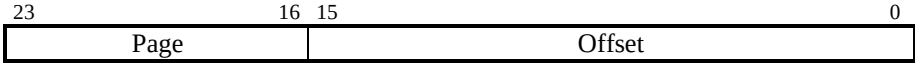
г) Приказати вредности кључева и информационог садржаја релевантних улаза у TLB (означити те улазе), после задавања дате секвенце адреса.

д) Приказати изглед табеле страница након извршавања датог програмског сегмента.

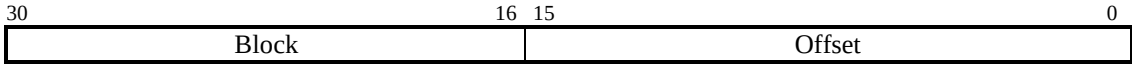
Решење:

а)

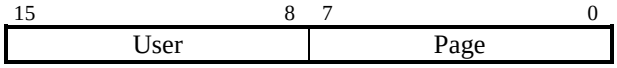
Логичка структура виртуелне адресе је:




Логичка структура физичке адресе је:




б) Улазни податак за TLB има структуру:




Кључ за претрагу је поље *User*(7:0):*Page*(7:6), а налази се у сету са редним бројем једнаким пољу *Page*(5:0).

в)

| Виртуелна адреса | Тип | User | Page | Word | Tag | Set | Коментар | Block | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|
| 278800 | Ex | D0 | 27 | 8800 | 1101000000 | 100111 | Miss | 0037 | 00378800 |
| A700FF | Rd | D0 | A7 | 00FF | 1101000010 | 100111 | Miss | - | Page Fault |
| 278800 | Ex | D0 | 27 | 8800 | 1101000000 | 100111 | Hit | 0037 | 00378800 |
| A700FF | Rd | D0 | A7 | 00FF | 1101000010 | 100111 | Miss | 00F0 | 00F000FF |
| 278801 | Ex | D0 | 27 | 8801 | 1101000000 | 100111 | Hit | 0037 | 00378801 |
| E78800 | Rd | D0 | E7 | 8800 | 1101000011 | 100111 | Miss | 004E | 004E8800 |
| B600FA | Wr | D0 | B6 | 00FA | 1101000010 | 110111 | Miss | - | Page Fault |
| 278801 | Ex | D0 | 27 | 8801 | 1101000000 | 100111 | Miss | 0037 | 00378801 |
| E78800 | Rd | D0 | E7 | 8800 | 1101000010 | 100111 | Hit | 004E | 004E8800 |
| B600FA | Wr | D0 | B6 | 00FA | 1101000010 | 110111 | Miss | 00F1 | 00F100FA |
| 278802 | Ex | D0 | 27 | 8802 | 1101000000 | 100111 | Miss | 0037 | 00378802 |
| B600FF | Rd | D0 | B6 | 00FF | 1101000000 | 100111 | Hit | 00F1 | 00F100FF |
| E78802 | Rd | D0 | E7 | 8802 | 1101000011 | 100111 | Hit | 004E | 004E8802 |

г)

Садржај је следећи (све вредности су хексадецималне):

Сет 27h

| **Кључеви (*Tag* меморија)** | | | | **Информациони садржај** |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Tag** | **Блок** |
| 0 | 1 | 0 | 343h | 004E |
| 1 | 1 | 0 | 340h | 0037 |

Сет 36h

| **Кључеви (*Tag* меморија)** | | | | **Информациони садржај** |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Tag** | **Блок** |
| 0 | 1 | 1 | 342h | 00F1 |
| 1 | 0 | | | |

д)

| Табела страница корисника D0h | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| ... | | | | - |
| 27 | 1 | 0 | 0037h | - |
| ... | | | | - |
| A7 | 1 | 0 | 00F0h | - |
| ... | | | | - |
| B6 | 1 | 1 | 00F1h | - |
| ... | | | | - |
| E7 | 1 | 1 | 004Eh | - |
| ... | | | | - |

## Задатак 6.

Процесор подржава постојање више програма у меморији истовремено, до 8 корисника. Рачунар поседује виртуелну меморију, код које 32-битне адресе одговарају виртуелном адресном простору, а физички адресни простор је величине 256 MW (256 мега речи). Адресирање је на нивоу 32-битних речи. Виртуелна меморија организована је на страничном принципу, при чему је величина странице 1 KW (1 кило речи). Процесор поседује посебан хардвер за убрзавање пресликавања виртуелних у реалне (физичке) адресе (у даљем тексту TLB). TLB је организован са сет-асоцијативним пресликавањем, поседује два улаза по сету и 64 сета. Алгоритам замене TLB-а је FIFO. Процесор извршава следећи део програма корисника са идентификатором 7:

| Адреса (hex): | Наредба: | Коментар: |
|---|---|---|
| 10000 | LOAD R0, #40004h | ; R0 :=40004h |
| 10002 | PUSH R0 | ; Mem[SP--]:=R0 |
| 10003 | STORE 20005h, R0 | ; Mem[20005h] :=R0 |
| 10005 | | |

Кориснику су додељена два блока оперативне меморије почев од F0h закључно са F1h. Ови блокови попуњавају се редом. Алгоритам замене страница у оперативној меморији је FIFO. Ниједна од страница датог корисника није у оперативној меморији, а стек расте од виших ка нижим адресама и SP указује не прву слободну локацију, и има вредност 30009h пре извршавања датог дела програма. Регистар R0 је 32-битни. Претпоставити да се приликом промене контекста садржај TLB јединице неће променити.

а) Приказати логичку структуру виртуелне и физичке адресе и дати значење и дужину сваког поља.

б) Приказати шта представља улазни податак који процесор даје TLB-у, шта је кључ за претрагу у TLB, и како се добија улаз у TLB у коме се тај кључ тражи.

в) За сваки приступ меморији означити: виртуелну адресу којој се приступа, тип операције (Rd, Wr, Ex), идентификацију корисника, вредност поља Page и Word, вредност поља Tag и Set у TLB јединици, коментар да ли је било сагласности у TLB јединици, блок оперативне меморије Block, физичку адресу којој се приступило или коментар уколико је дошло до прекида. Одговор дати табеларно.

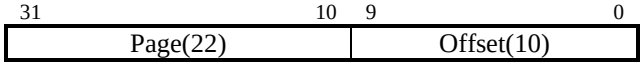
| Виртуелна адреса | Тип | User | Page | Word | Tag | Set | Коментар | Block | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

г) Приказати вредности кључева и информационог садржаја релевантних улаза у TLB (означити те улазе), после задавања дате секвенце адреса.

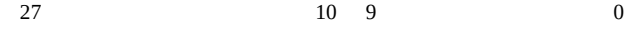
д) Приказати изглед табеле страница након извршавања датог програмског сегмента.

Решење:

а) Логичка структура виртуелне адресе је:




Логичка структура физичке адресе је:



| Block(18) | Offset(10) |
|---|---|

б)

| Tag(19) | Set(6) | Offset(0) |
|---|---|---|

в)

| Виртуелна адреса | Тип | User | Page | Word | Tag | Set | Коментар | Block | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|
| 10000 | Ex | 7 | 40 | 0 | 70001 | 0 | Miss | - | Page Fault |
| 10000 | Ex | 7 | 40 | 0 | 70001 | 0 | Miss | F0 | 3C000 |
| 10001 | Ex | 7 | 40 | 1 | 70001 | 0 | Hit | F0 | 3C001 |
| 10002 | Ex | 7 | 40 | 2 | 70001 | 0 | Hit | F0 | 3C002 |
| 30009 | Wr | 7 | C0 | 9 | 70003 | 0 | Miss | - | Page Fault |
| 10002 | Ex | 7 | 40 | 2 | 70001 | 0 | Hit | F0 | 3C002 |
| 30009 | Wr | 7 | C0 | 9 | 70003 | 0 | Miss | F1 | 3C409 |
| 10003 | Ex | 7 | 40 | 3 | 70001 | 0 | Hit | F0 | 3C003 |
| 10004 | Ex | 7 | 40 | 4 | 70001 | 0 | Hit | F0 | 3C004 |
| 20005 | Ex | 7 | 80 | 5 | 70002 | 0 | Miss | - | Page Fault |
| 10003 | Ex | 7 | 40 | 3 | 70001 | 0 | Miss | - | Page Fault |
| 10003 | Ex | 7 | 40 | 3 | 70001 | 0 | Miss | F1 | 3C403 |
| 10004 | Ex | 7 | 40 | 4 | 70001 | 0 | Hit | F1 | 3C404 |
| 20005 | Ex | 7 | 80 | 5 | 70002 | 0 | Miss | F0 | 3C005 |

г)

Сет 0h

| **Кључеви (*Tag* меморија)** | | | | **Информациони садржај** |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Tag** | **Блок** |
| 0 | 1 | 0 | 70001h | F1 |
| 1 | 1 | 1 | 70002h | F0 |

д)

| **Табела страница** | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| ... | | | | - |
| 40h | 1 | 0 | F1h | - |
| ... | | | | - |
| 80h | 1 | 1 | F0h | - |
| ... | | | | - |
| C0h | 0 | 1 | F1h | - |
| ... | | | | - |

## Задатак 7.

Виртуелне меморијске адресе су ширине 16 бита, а адресирање је на нивоу 16-битних речи. Процесор оперише само са 16-битним целобројним величинама (у даљем тексту *реч* означава 16-битну величину). Процесор подржава постојање више програма у меморији истовремено, до 16 корисника. Физички адресни простор је величине 4 GW (4 гига речи). Виртуелна меморија организована је на сегментом принципу, при чему је максимална величина сегмента 4 KW (4 кило речи). Процесор поседује посебан хардвер за убрзавање пресликавања виртуелних у реалне (физичке) адресе (у даљем тексту TLB). TLB је реализован са директним пресликавањем, при чему се сви сегменти са истим бројем, различитих корисника, пресликавају у исти улаз у TLB. Информациони садржај једног улаза у TLB је следећи: дужина сегмента, права приступа и почетна адреса сегмента у физичкој меморији. Права приступа се дефинишу помоћу три бита R, W и E, при чему јединица значи дозвољено читање податка (R), упис податка (W) и извршавање инструкције (E). Процесор извршава следећи део програма корисника са идентификатором 3:

| Адреса (hex): | Наредба: | Коментар: |
|---|---|---|
| 3F00 | MOV R0, #1234h | ; R0:=1234h |
| 3F02 | PUSH R0 | ; Mem[SP--]:=R0 |

```
3F03        MOV  2001h, R0        ; Mem[2001h]:=R0
3F05             ...
```

Пре почетка извршавања датог програмског сегмента у оперативном меморији су били присутни сегменти програма, стека и података датог корисника. Сегмент програма је дужине 4 KW и смештен је почев од локације 200000h физичке меморије, сегмент стека је дужине 1 KW и смештен је почев од локације 300000h физичке меморије, а сегмент података је дозвољен за упис и читање, дужине је 2 KW и смештен је од адресе 400000h физичке меморије. Стек расте од виших ка нижим адресама и указује на прву слободну локацију, а SP има почетну вредност 1215h пре извршавања датог дела програма. Претпоставити да се приликом промене контекста садржај TLB јединице неће променити.

а) Приказати логичку структуру виртуелне адресе и означити значење и дужину сваког поља.

б) Приказати шта представља улазни податак који процесор даје TLB-у, шта је кључ за претрагу у TLB, и како се добија улаз у TLB у коме се тај кључ тражи.

в) За сваки приступ меморији означити: виртуелну адресу којој се приступа, тип операције (Rd, Wr, Ex), идентификацију корисника, вредност поља Segment и Word, вредност поља Tag и Entry у TLB јединици, коментар да ли је било сагласности у TLB јединици, почетну адресу сегмента Saddr, физичку адресу којој се приступило или коментар уколико је дошло до прекида. Одговор дати табеларно.

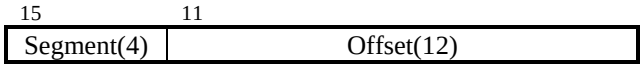
| Виртуелна адреса | Тип | User | Segment | Word | Tag | Entry | Коментар | Saddr | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

г) Приказати вредности кључева и информационог садржаја релевантних улаза у TLB (означити те улазе), после извршавања датог дела програма.

д) Приказати изглед табеле сегмената након извршавања датог програмског сегмента.

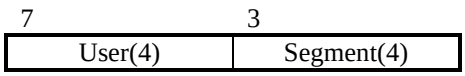
Решење:

а) Логичка структура виртуелне адресе је тако:




Физичка адреса се добија сабирањем адресе почетка сегмента унутар физичке меморије и редног броја речи унутар сегмента.

б) Улазни податак за TLB је :




Кључ за претрагу у TLB поље *User*, а налази се у улазу са редним бројем једнаким пољу *Segment*.

в)

| Виртуелна адреса | Тип | User | Segment | Word | Tag | Entry | Коментар | Saddr | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|
| 3F00 | Ex | 3 | 3 | F00 | 3 | 3 | Miss | 200000 | 200F00 |
| 3F01 | Ex | 3 | 3 | F01 | 3 | 3 | Hit | 200000 | 200F01 |
| 3F02 | Ex | 3 | 3 | F02 | 3 | 3 | Hit | 200000 | 200F02 |
| 1215 | Wr | 3 | 1 | 215 | 3 | 1 | Miss | 300000 | 300215 |
| 3F03 | Ex | 3 | 3 | F03 | 3 | 3 | Hit | 200000 | 200F03 |
| 3F04 | Ex | 3 | 3 | F04 | 3 | 3 | Hit | 200000 | 200F04 |
| 2001 | Wr | 3 | 2 | 001 | 3 | 2 | Miss | 400000 | 400001 |

г)

| **Кључеви (*Tag* меморија)** | | | | **Информациони садржај** | | |
|---|---|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Tag** | **RWE (bin)** | **Дужина** | **Почетна адреса(hex)** |
| 0 | 0 | 0 | | | | |
| 1 | 1 | 1 | 3 | 110 | 1K | 300000 |
| 2 | 1 | 1 | 3 | 110 | 2K | 400000 |
| 3 | 1 | 0 | 3 | 001 | 4K | 200000 |

д)

| Табела сегмената | | | | | | |
|---|---|---|---|---|---|---|
| **Улаз** | **V** | **D** | **RWE (bin)** | **Дужина** | **Почетна адреса(hex)** | **Диск** |
| 0 | 0 | 0 | | | | - |
| 1 | 1 | 1 | 110 | 1K | 300000 | - |
| 2 | 1 | 1 | 110 | 2K | 400000 | - |
| 3 | 1 | 0 | 001 | 4K | 200000 | - |

## Задатак 8.

Процесор подржава виртуелну меморију, и поседује хардвер за убрзавање пресликавања виртуелних у реалне адресе (у даљем тексту TLB). При том је виртуелна адреса ширине 16 бита, а реална адреса 32 бита, уз подршку до 4 корисника. Виртуелна меморија организована је по сегментно-страничном принципу, са максимално 4 сегмената по кориснику и 16 странице по сегменту. TLB је организован асоцијативно, са 4 улаза и LRU алгоритмом замене. Кориснички програм са бројем 3h генерише следећу секвенцу виртуелних адреса (све вредности су хексадецималне):

4632h (Ex), 5C5Fh (Ex), BE8Eh (Rd), 2237h (Wr), 5C60h (Ex), 463Fh (Ex), FCFAh (Wr).

Пре почетка извршавања овог програмског сегмента TLB јединица је била празна, а у оперативној меморији се није налазила ни једна страница датог корисника. Кориснику се додељују странице у оперативне меморији почев од блока C1h физичке меморије. Ове странице се додељују редом. Алгоритам замене страница у оперативној меморији је FIFO. Одредити минималну величину сегмената тако да не дође до прекида услед прекорачења опсега. Претпоставити да се приликом промене контекста садржај TLB јединице неће променити.

а) Приказати логичку структуру виртуелне и физичке адресе и дати значење и дужину сваког поља.

б) Приказати шта представља улазни податак који процесор даје TLB-у, шта је кључ за претрагу у TLB, и како се добија улаз у TLB у коме се тај кључ тражи.

в) За сваки приступ меморији означити: виртуелну адресу којој се приступа, тип операције (Rd, Wr, Ex), идентификацију корисника, вредност поља Segment, Page и Word, вредност поља Tag TLB јединици, коментар да ли је било сагласности у TLB јединици, блок оперативне меморије Block, физичку адресу. Одговор дати табеларно.

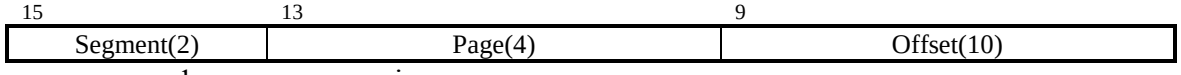
| Виртуелна адреса | Тип | User | Segment | Page | Word | Tag | Коментар | Block | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

г) Приказати вредности кључева и информационог садржаја релевантних улаза у TLB (означити те улазе), после извршавања датог дела програма.

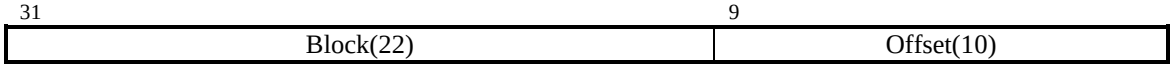
д) Приказати изглед табеле сегмената и свих табела страница након извршавања датог програмског сегмента.

Решење:

а) Логичка структура виртуелне адресе је:

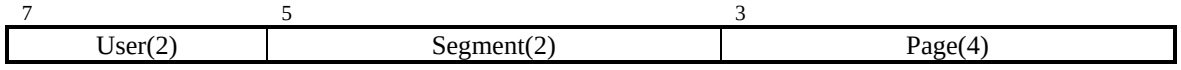



Логичка структура физичке адресе је:




б)

Улазни податак за TLB има структуру:




Кључ за претрагу је поље *User*(1:0):*Segment*(1:0):*Page*(3:0).

в)

| Виртуелна | Тип | User | Segment | Page | Word | Tag | Коментар | Block | Физичка |
|---|---|---|---|---|---|---|---|---|---|

| адреса | | | | | | | | | адреса |
|---|---|---|---|---|---|---|---|---|---|
| 4632 | Ex | 11 | 01 | 0001 | 1000110010 | 11010001 | Miss | - | Page Fault |
| 4632 | Ex | 11 | 01 | 0001 | 1000110010 | 11010001 | Miss | C1 | 00030632 |
| 5C5F | Ex | 11 | 01 | 0111 | 0001011111 | 11010111 | Miss | - | PageFault |
| 5C5F | Ex | 11 | 01 | 0111 | 0001011111 | 11010111 | Miss | C2 | 0003085F |
| BE8E | Rd | 11 | 10 | 1111 | 1010001110 | 11101111 | Miss | - | Page Fault |
| 5C5F | Ex | 11 | 01 | 0111 | 0001011111 | 11010111 | Hit | C2 | 0003085F |
| BE8E | Rd | 11 | 10 | 1111 | 1010001110 | 11101111 | Miss | C3 | 00030E8E |
| 2237 | Wr | 11 | 00 | 1000 | 1000110111 | 11001000 | Miss | - | Page Fault |
| 5C5F | Ex | 11 | 01 | 0111 | 0001011111 | 11010111 | Hit | C2 | 0003085F |
| BE8E | Rd | 11 | 10 | 1111 | 1010001110 | 11101111 | Hit | C3 | 00030E8E |
| 2237 | Wr | 11 | 00 | 1000 | 1000110111 | 11001000 | Miss | C4 | 00031237 |
| 5C60 | Ex | 11 | 01 | 0111 | 0001100000 | 11010111 | Hit | C2 | 00030860 |
| 463F | Ex | 11 | 01 | 0001 | 1000111111 | 11010001 | Hit | C1 | 0003063F |
| FCFA | Wr | 11 | 11 | 1111 | 0011111010 | 11111111 | Miss | - | Page Fault |
| 463F | Ex | 11 | 01 | 0001 | 1000111111 | 11010001 | Hit | C1 | 0003063F |
| FCFA | Wr | 11 | 11 | 1111 | 0011111010 | 11111111 | Miss | C5 | 000314FA |

г)

| **Кључеви (*Tag* меморија)** | | | | **Информациони садржај** | | |
|---|---|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Tag** | **RWE (bin)** | **Блок** | **LRU–CNT** |
| 0 | 1 | 0 | 11010001 | 001 | C1 | 2 |
| 1 | 1 | 0 | 11010111 | 001 | C2 | 1 |
| 2 | 1 | 1 | 11111111 | 110 | C5 | 3 |
| 3 | 1 | 1 | 11001000 | 110 | C4 | 0 |

д)

| **Табела сегмената** | | | |
|---|---|---|---|
| **Улаз** | **RWE (bin)** | **Дужина** | **PTP** |
| 0 | 110 | 9K | 0 |
| 1 | 001 | 8K | 1 |
| 2 | 110 | 16K | 2 |
| 3 | 110 | 16K | 3 |

| **Табела страница 0** | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| ... | | | | - |
| 8 | 1 | 1 | C4 | - |
| ... | | | | - |

| **Табела страница 1** | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| ... | | | | - |
| 1 | 1 | 0 | C1 | - |
| 7 | 1 | 0 | C2 | - |

| **Табела страница 2** | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| ... | | | | - |
| … | | | | - |
| F | 1 | 0 | C3 | - |

| **Табела страница 3** | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| ... | | | | - |
| … | | | | - |

| | | | | |
|---|---|---|---|---|
| F | 1 | 1 | C5 | - |

## Задатак 9.

Процесор подржава постојање више програма у меморији истовремено, до 8 корисника. Рачунар поседује виртуелну меморију, код које 32-битне адресе одговарају виртуелном адресном простору, а физички адресни простор је величине 256 MW (256 мега речи). Адресирање је на нивоу 32-битних речи. Виртуелна меморија организована је на сегментно страничном принципу, при чему је величина странице 1 KW (1 кило речи). Постоје 4 сегмента, то су сегмент програма, сегмент података, сегмент стека и хип сегмент. Процесор поседује посебан хардвер за убрзавање пресликавања виртуелних у реалне (физичке) адресе (у даљем тексту TLB). TLB је организован са сет-асоцијативним пресликавањем, поседује четири улаза по сету и 32 сета. Алгоритам замене TLB-а је LRU. Процесор извршава следећи део програма корисника са идентификатором 7:

| Адреса (hex): | Наредба: | Коментар: |
|---|---|---|
| 00000D76 | MOV R0, #1234h | ; R0:=1234h |
| 00000D78 | PUSH R0 | ; Mem[SP--]:=R0 |
| 00000D79 | MOV 40000F56h, R0 | ; Mem[40000F56h]:=R0 |
| 00000D7B | MOV (40000FABh), R0 | ; Mem[Mem[40000FABh]]:=R0 |
| 00000D7D | … | |

За дати програм сегмент програма је величине 16 KW, сегмент података 6 KW, сегмент стека 8 KW, док је хип сегмент 10 KW. Стек расте од виших ка нижим адресама и SP указује не прву слободну локацију има вредност 80000C13h пре извршавања датог дела програма. Садржај меморијске локације 40000FABh је C0000E24h.

Пре почетка извршавања овог програмског сегмента TLB јединица је била празна, а у оперативној меморији се није налазила ни једна страница датог корисника. Кориснику се додељују странице у оперативне меморији почев од блока ABh физичке меморије. Ове странице се додељују редом. Алгоритам замене страница у оперативној меморији је FIFO. Регистар R0 је 32-битни. Приликом промене контекста брише се комплетан садржај TLB јединице.

а) Приказати логичку структуру виртуелне и физичке адресе и дати значење и дужину сваког поља.

б) Приказати шта представља улазни податак који процесор даје TLB-у, шта је кључ за претрагу у TLB, и како се добија улаз у TLB у коме се тај кључ тражи.

в) За сваки приступ меморији означити: виртуелну адресу којој се приступа, тип операције (Rd, Wr, Ex), идентификацију корисника, вредност поља Segment, Page и Word, вредност поља Tag и Set у TLB јединици, коментар да ли је било сагласности у TLB јединици, блок оперативне меморије Block, физичку адресу којој се приступило или коментар уколико је дошло до прекида. Одговор дати табеларно.

| Виртуелна адреса | Тип | User | Segment | Page | Word | Tag | Set | Коментар | Block | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

г) Приказати вредности кључева и информационог садржаја релевантних улаза у TLB (означити те улазе), после извршавања датог дела програма.

д) Приказати изглед табеле сегмената и свих табела страница након извршавања датог програмског сегмента.

Решење:

а) Логичка структура виртуелне адресе је:

| 31 | 29 | 9 |
|---|---|---|
| Segment(2) | Page(20) | Offset(10) |

Логичка структура физичке адресе је:

27 9

| Block(18) | Offset(10) |
|---|---|

б) Улазни податак за TLB има структуру:

| 24 | 21 | 19 |
|---|---|---|
| User(3) | Segment(2) | Page(20) |

Кључ за претрагу је поље *User*(7:0):*Segment*(1:0):*Page*(19:5), а налази се у сету са редним бројем једнаким пољу *Page*(4:0).

в)

| Виртуелна адреса | Тип | User | Segment | Page | Word | Tag | Set | Коментар | Block | Физичка адреса |
|---|---|---|---|---|---|---|---|---|---|---|
| 00000D76 | Ex | 7 | 0 | 3 | 176 | E0000 | 3 | Miss | - | Page Fault |
| 00000D76 | Ex | 7 | 0 | 3 | 176 | E0000 | 3 | Miss | AB | 002AD76 |
| 00000D77 | Ex | 7 | 0 | 3 | 177 | E0000 | 3 | Hit | AB | 002AD77 |
| 00000D78 | Ex | 7 | 0 | 3 | 178 | E0000 | 3 | Hit | AB | 002AD78 |
| 80000C13 | Wr | 7 | 2 | 3 | 013 | F0000 | 3 | Miss | - | Page Fault |
| 00000D78 | Ex | 7 | 0 | 3 | 178 | E0000 | 3 | Miss | AB | 002AD78 |
| 80000C13 | Wr | 7 | 2 | 3 | 013 | F0000 | 3 | Miss | AC | 002B013 |
| 00000D79 | Ex | 7 | 0 | 3 | 179 | E0000 | 3 | Hit | AB | 002AD79 |
| 00000D7A | Ex | 7 | 0 | 3 | 17A | E0000 | 3 | Hit | AB | 002AD7A |
| 40000F56 | Wr | 7 | 1 | 3 | 356 | E8000 | 3 | Miss | - | Page Fault |
| 00000D79 | Ex | 7 | 0 | 3 | 179 | E0000 | 3 | Miss | AB | 002AD79 |
| 00000D7A | Ex | 7 | 0 | 3 | 17A | E0000 | 3 | Hit | AB | 002AD7A |
| 40000F56 | Wr | 7 | 1 | 3 | 356 | E8000 | 3 | Miss | AD | 002B756 |
| 00000D7B | Ex | 7 | 0 | 3 | 17B | E0000 | 3 | Hit | AB | 002AD7B |
| 00000D7C | Ex | 7 | 0 | 3 | 17C | E0000 | 3 | Hit | AB | 002AD7C |
| 40000FAB | Rd | 7 | 1 | 3 | 3AB | E8000 | 3 | Hit | AD | 002B7AB |
| C0000E24 | Wr | 7 | 3 | 3 | 224 | F8000 | 3 | Miss | - | Page Fault |
| 00000D7B | Ex | 7 | 0 | 3 | 17B | E0000 | 3 | Miss | AB | 002AD7B |
| 00000D7C | Ex | 7 | 0 | 3 | 17C | E0000 | 3 | Hit | AB | 002AD7C |
| 40000FAB | Rd | 7 | 1 | 3 | 3AB | E8000 | 3 | Miss | AD | 002B7AB |
| C0000E24 | Wr | 7 | 3 | 3 | 224 | F8000 | 3 | Miss | AE | 002BA24 |

г)

Сет 3h

| **Кључеви (*Tag* меморија)** | | | | **Информациони садржај** | | |
|---|---|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Tag** | **RWE (bin)** | **Блок** | **LRU–CNT** |
| 0 | 1 | 0 | E0000 | 001 | AB | 1 |
| 1 | 1 | 1 | E8000 | 110 | AD | 2 |
| 2 | 1 | 1 | F8000 | 110 | AE | 3 |
| 3 | 0 | 0 | | | | |

д)

| **Табела сегмената** | | | |
|---|---|---|---|
| **Улаз** | **RWE (bin)** | **Дужина** | **PTP** |
| 0 | 001 | 16K | Табела страница програма |
| 1 | 110 | 6K | Табела страница података |
| 2 | 110 | 8K | Табела страница стека |
| 3 | 110 | 10K | Табела странца хипа |

| **Табела страница програма** | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| 0 | | | | - |
| ... | | | | - |
| 3 | 1 | 0 | AB | - |
| ... | | | | - |

| Табела страница података | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| 0 | | | | - |
| ... | | | | - |
| 3 | 1 | 1 | AD | - |
| ... | | | | - |

| Табела страница стека | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| 0 | | | | - |
| ... | | | | - |
| 3 | 1 | 1 | AC | - |
| ... | | | | - |

| Табела странца хипа | | | | |
|---|---|---|---|---|
| **Улаз** | **V** | **D** | **Блок** | **Диск** |
| 0 | | | | - |
| ... | | | | - |
| 3 | 1 | 1 | AE | - |
| ... | | | | - |

# ПРЕКЛАПАЊЕ ПРИСТУПА МЕМОРИЈСКИМ МОДУЛИМА (MEMORY INTELEAVING)

## Задатак 1.

Оперативна меморија се састоји од осам меморијских модула, а капацитет модула је 128 KB, што чини тотални капацитет меморије од 1 MB. Ширина речи оперативне меморије је 1 бајт. Адреса оперативне меморије је ширине 20 бита, а њени најстарији и најмлађи битови су означени са $A_{19}$ и $A_0$, респективно.

а) Нацртати и описати који опсег адреса меморијског адресног простора је додељен ком од осам модула за четири случаја преклапања приступа модулима ако је број модула спецификован следећим битовима адресе:

а1) $A_{19}$ $A_{18}$ $A_{17}$

а2) $A_{19}$ $A_{18}$ $A_0$

а3) $A_{19}$ $A_1$ $A_0$

а4) $A_2$ $A_1$ $A_0$

и показати за сваки од њих који адресни битови формирају адресу бајта унутар модула.

б) Објаснити да ли преплитање меморијских модула убрзава рад рачунара који се састоји од процесора, оперативне меморије, неколико периферија и магистрале, али не садржи DMA контролере или друге процесоре. У случају да

убрзава, објаснити на који начин се то постиже,

не убрзава, објаснити зашто.

в) У случају рачунара који се састоји од четири процесора, четири DMA контролера, оперативне меморије са преклапањем приступа меморијским модулима као што је описано у задатку и асинхроне магистрале, постоји потреба за арбитрацијом магистрале где master прво учествује у арбитрацији, па тек онда када прими дозволу од арбитратора магистрале сме да изврши циклус на магистрали. За описани рачунар објаснити:

в1) Које врсте циклуса на магистрали постоје.

в2) Колико дуго је магистрала заузета за време сваког од наведених врста циклуса.

в3) Колико група линија постоје на магистрали и чему свака од њих служи за сваки тип циклуса на магистрали.

в4) Нацртати и објаснити временске дијаграме свих сигнала на магистрали за све типове циклуса на магистрали.

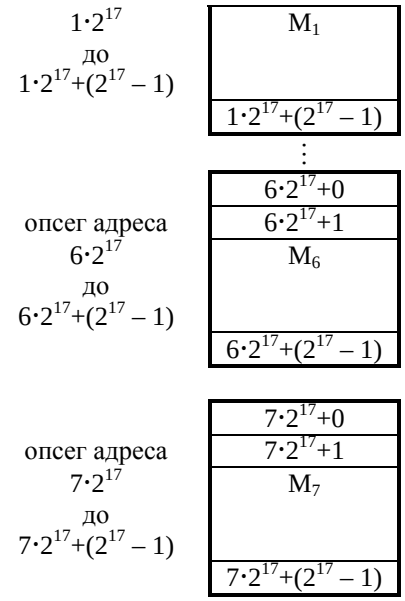
Решење:

а)

а1) Адресе су распоређене по меморијским модулима на следећи начин:

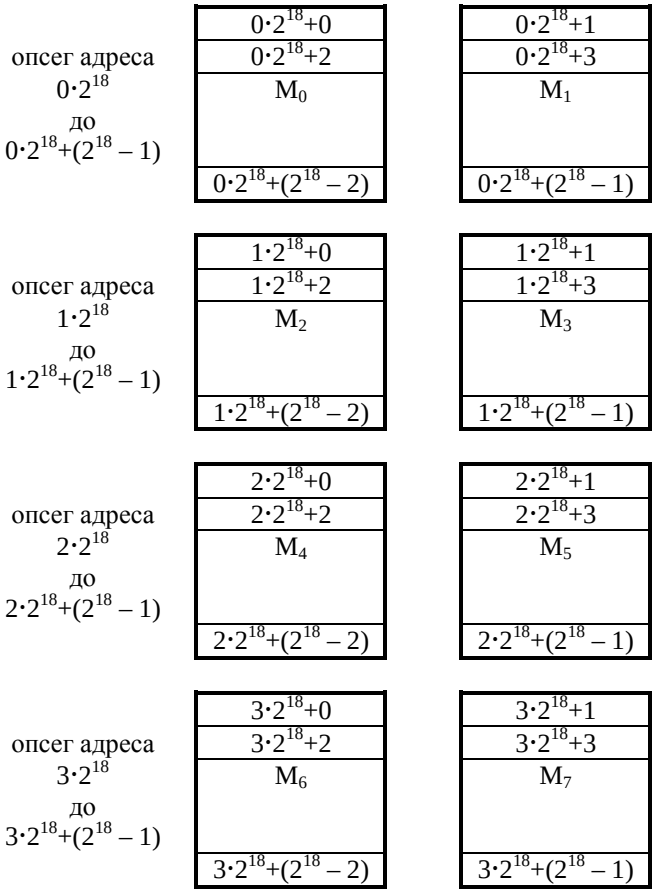
| | |
|---|---|
| | $0 \cdot 2^{17}+0$ |
| опсег адреса | $0 \cdot 2^{17}+1$ |
| $0 \cdot 2^{17}$ | $M_0$ |
| до | |
| $0 \cdot 2^{17}+(2^{17}-1)$ | |
| | $0 \cdot 2^{17}+(2^{17}-1)$ |

| | |
|---|---|
| | $1 \cdot 2^{17}+0$ |
| опсег адреса | $1 \cdot 2^{17}+1$ |



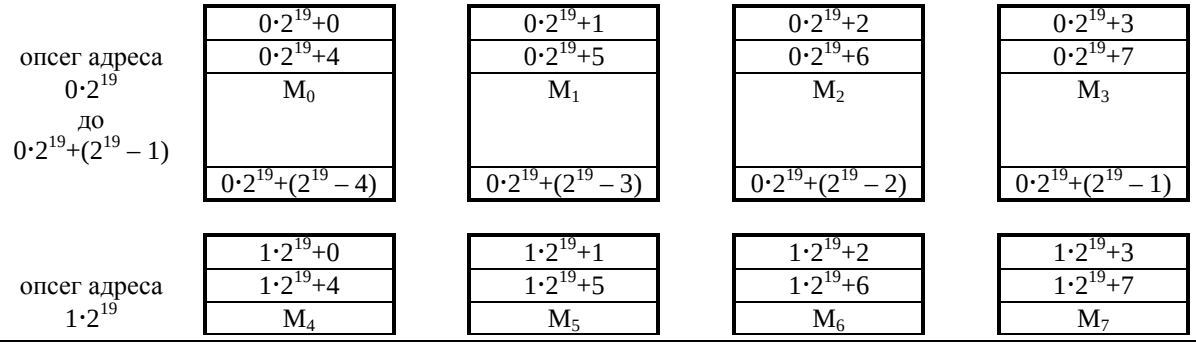

Битови $A_{16}$ до $A_0$ спецификују адресу унутар модула.

а2) Адресе су распоређене по меморијским модулима на следећи начин:




Битови $A_{17}$ до $A_1$ спецификују адресу унутар модула.

а3) Адресе су распоређене по меморијским модулима на следећи начин:



до $1\cdot2^{19}+(2^{19}-1)$

| | | | |
|---|---|---|---|
| | | | |
| $1\cdot2^{19}+(2^{19}-4)$ | $1\cdot2^{19}+(2^{19}-3)$ | $1\cdot2^{19}+(2^{19}-2)$ | $1\cdot2^{19}+(2^{19}-1)$ |

Битови $A_{18}$ до $A_2$ спецификују адресу унутар модула.

а4) Адресе су распоређене по меморијским модулима на следећи начин:

| опсег адреса $0\cdot2^{20}$ до $0\cdot2^{20}+(2^{20}-1)$ | | | | | | |
|---|---|---|---|---|---|---|
| | $0\cdot2^{20}+0$ | $0\cdot2^{20}+1$ | | $0\cdot2^{20}+6$ | $0\cdot2^{20}+7$ |
| | $0\cdot2^{20}+8$ | $0\cdot2^{20}+9$ | | $0\cdot2^{20}+14$ | $0\cdot2^{20}+15$ |
| | $M_0$ | $M_1$ | … | $M_6$ | $M_7$ |
| | $0\cdot2^{20}+(2^{20}-8)$ | $0\cdot2^{20}+(2^{20}-7)$ | | $0\cdot2^{20}+(2^{20}-2)$ | $0\cdot2^{20}+(2^{20}-1)$ |

Битови $A_{19}$ до $A_3$ спецификују адресу унутар модула.

б) Меморија са преклапањем приступа неће убрзати рад рачунара.

в)

в1) Постоје три типа циклуса на магистрали:

циклус иницијализације операције читања податка,

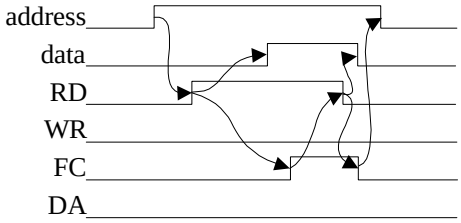
циклус уписа податка и

циклус враћања податка.

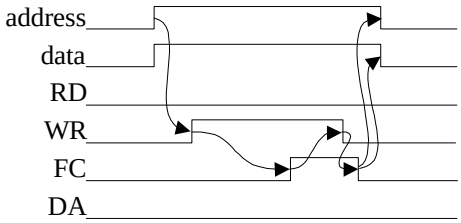
в2) За сва три типа циклуса магистрала је заузета само док траје пренос релевантних информација за тај циклус из регистара master-а у регистре slave-а.

в3)

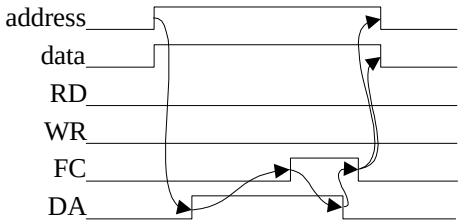
**циклус иницијализације операције читања податка**




**циклус уписа податка**




**циклус враћања податка**




## Задатак 2.

Меморијске адресе су ширине 16 бита, ширина магистрале података је 8 бита, адресирање је бајтовско, а двобајтни подаци се у меморију смештају тако да је на нижој адреси нижи бајт. Процесор

оперише само са 16-битним целобројним величинама (у даљем тексту *реч* означава 16-битну величину), и све инструкције су 16-битне. Улазно/излазни адресни простор је меморијски пресликан. Време одзива меморије је неодређено, магистрала је асинхрона.

а) Предложити организацију оперативне меморије са преклопљеним приступом меморијским модулима, при чему се рачунар састоји само од датог (једног) процесора, оперативне меморије, периферија и једне асинхроне магистрале. Прецизно дати следеће: број меморијских модула, начин адресирања модула, начин адресирања локација унутар модула и распоред адреса по модулима. Време одзива меморије је довољно пута веће од трајања циклуса на магистрали.

б) Да ли је, за организацију из претходне тачке, неопходно да процесор, при иницирању читања из меморије, по линијама података шаље неку идентификацију?

Ако јесте, коју идентификацију треба да шаље, а ако није, како процесор може да зна на који му захтев меморија одговара, у случају да модули нису потпуно једнаки по времену одзива?

в) Одговорити на иста питања из тачке а), ако се између процесора и меморије постави кеш меморија организована сет-асоцијативно на нивоу блока, са 128 сетова, 4 блока у сету и капацитетом 8 KB података.

Решење:

а) Организација модула је следећа: два модула M0 и M1, модули се адресирају битом најмање тежине у адреси (A0), адресирање унутар модула је помоћу бита A15 до A1, а распоред адреса је такав да су у M0 све парне, а у M1 све непарне адресе.

б) Није потребно.

в) Адресе су расподељене на следећи начин (све вредности су хексадецималне):

```
M0:0000, 0010, 0020, 0030, …. , FFF0
M1:0001, 0011, 0021, 0031, …. , FFF1
….
Mi:000i, 001i, 002i, 003i, …. , FFFi
….
M15:    000F, 001F, 002F, 003F, …. , FFFF
```

## Задатак 3.

Процесор је двоадресни. Адресе су ширине 16 бита, а адресирање је на нивоу 16-битних речи. Процесор поседује регистре за податке D3…0 и адресне регистре A3…0. Једини начин адресирања је регистарско индиректно, преко адресних регистара.

Пројектује се меморијски систем са преклапањем приступа модулима. Време приступа модулу (трајање једног читања или уписа) је 100 ns. Када процесор и модул комуницирају, трајање циклуса на магистрали је 15 ns. Модули се организују тако да се приступ оптимизује за секвенцијално обраћање меморији.

а) Колико (највише) модула предлажете? Образложити.

б) Приказати начин адресирања модула и распоред адреса по модулима.

в) Колико циклуса на магистрали се обави при извршавању следеће секвенце инструкција? Образложити.

```
LOAD D0, A1        ; D0:=Mem[A1]
ADD D0, A2         ; D0:=D0+Mem[A2]
STORE D0, A2       ; Mem[A2]:=D0
```

Решење:

а) Највећи број модула који (теоријски) има смисла је 7, што значи да треба инсталирати 8 модула, као број који је први наредни степен 2.

б) За секвенцијални приступ суседне адресе треба да буду у различитим модулима.

в) Укупан број циклуса на магистрали је 11.

## Задатак 4.

Меморијске адресе су ширине 32 бита, ширина магистрале података је 16 бита, а адресирање је на нивоу 16-битних речи. Између процесора и магистрале везана је кеш меморија организована блоковски, са величином блока од $2^{16}$ речи. Ради убрзања приступа меморији, меморија је организована са преклапањем модула (*memory inteleaving).* Време одзива меморијског модула је толико да се може обавити иницијализација операције у још 14 меморијских модула после модула X од стране кеш меморије, док операција у модулу X не буде завршена.

а) Предложити број меморијских модула до којих би се преклапао приступ.

б) Предложити распоред адреса по овим модулима.

в) Како се врши препознавање модула и како се добија интерна адреса унутар модула?

Решење:

а) Највећи број модула који (теоријски) има смисла је 15, што значи да треба инсталирати 16 модула, као број који је први наредни степен 2.

б) Суседне адресе су у суседним модулима, због секвенцијалног (блоковског) приступа.

в) Најнижа 4 бита адресе одређују број модула, а остали бити интерну адресу унутар модула.

## Задатак 5.

Меморија је организована у 16 модула до којих се преклапа приступ, тако да је оптимизација извршена за секвенцијални приступ процесора меморији на нивоу бајта.

а) Који бити адресе одређују модул, а који локацију унутар модула? Како су адресе распоређене по модулима?

б) Ако осим описаног процесора не постоје други иницијатори операција са меморијом, да ли је потребна арбитрација приступа магистрали? Кратко образложити зашто.

в) У наставку су дате вредности релевантних сигнала на магистрали у току неколико узастопних циклуса на магистрали (све вредности су хексадецималне и сви приступи су на нивоу бајта). Пронаћи које је операције са меморијом вршио процесор. За сваку операцију дати вредност адресе меморијске локације којој је процесор приступао, вредност податка који је прочитан/уписан и тип операције (читање/упис).

| Такт | Addres Bus | Data Bus | rd | wr | ack |
|---|---|---|---|---|---|
| 1 | A00E | FE | 1 | 0 | 1 |
| 2 | B144 | F4 | 1 | 0 | 1 |
| 3 | D000 | F2 | 0 | 1 | 1 |
| 4 | C222 | F2 | 1 | 0 | 1 |
| 5 | FFFE | D0 | 1 | 1 | 1 |
| 6 | FFF4 | D1 | 1 | 1 | 1 |
| 7 | FFF2 | 20 | 1 | 1 | 1 |

**Табела садржаја на магистрали**

Решење:

а) Бити 3..0 одређују модул, а бити 15..4 локацију унутар модула. У модулу и су све адресе које дају остатак и при дељењу са 16 (*i*=0, ..., 15).

б) Потребна је. Без обзира што постоји само један *иницијатор операција* (процесор), постоји више *master-а* који воде *циклусе на магистрали* (процесор и меморијски модули). Ови master-и конкуришу за приступ магистрали, па је потребна арбитрација.

в)

| Адреса | Податак | Операција |
|---|---|---|
| A00E | 00D0 | Read |
| B144 | 00D1 | Read |
| D000 | FFF2 (odnosno F2) | Write |
| C222 | 0020 | Read |

## Задатак 6.

Разматра се рачунарски систем који се састоји од једног процесора, меморије и магистрале. Меморијске адресе су ширине 32 бита, ширина магистрале података је 16 бита, а адресирање је на нивоу 16-битних речи. За описани рачунар који је једини иницијатор операција са меморијом у систему, пројектује се меморијски систем са преклопљеним приступом меморијским модулима, тако да се оптимизује секвенцијални приступ до векторских података у меморији. Сваки циклус на магистрали траје тачно 4 такта (укључујући и арбитрацију). Време одзива меморије, од завршетка циклуса иницирања операције читања, до почетка циклуса враћања очитаног податка је 16 тактова. Број модула је 4. Виши приоритет имају циклуси враћања податка из меморије.

а) Како су распоређене адресе по модулима, који бити адресне речи одређују модул, а који адресу унутар модула?

б) Који типови циклуса постоје на магистрали? Дати временске облике сигнала на магистрали за све типове и навести ко је *master*, а ко *slave*.

в) Колико периода такта траје операција читања вектора дужине 64 из меморије, под претпоставком да процесор може да прихвати све очитане податке и иницира узастопно операције читања, без празног хода? Колико периода такта би трајало исто читање, али ако је меморија организована без преклапања, а циклус читања једне речи траје 16 тактова (приближно исте перформансе меморије)?

Решење:

а) Суседне адресе у различитим модулима. A1..0 одређују модул, а остали адресу унутар модула.

б) Видети предавања или вежбе.

в) Читање 4 суседне речи из 4 модула траје 4+16+4+4+4+4=36 тактова. После тога може да почне поновно иницирање читања 4 наредне речи. Дакле, укупно читање 64 речи траје (64/4)*36=576 тактова.

Без преклапања: 64*16=1024 такта.

## Задатак 7.

Разматра се рачунарски систем који се састоји од четири процесора, меморије и магистрале. Меморијске адресе су ширине 32 бита, ширина магистрале података је 16 бита, а адресирање је на нивоу 16-битних речи. За описани рачунар пројектује се меморијски систем са преклопљеним приступом меморијским модулима, са 256 модула који покривају цео адресни простор. Циклус повратка прочитаног податка из меморије идентификује се тако што су обе контролне линије *rd* и *wr* активне. Сви захтеви за приступ магистрали постављају се синхроно, на исти сигнал такта. Арбитрација се обавља комбинационо, тако да се у истом такту арбитрирају сви захтеви постављени на тај сигнал такта (или раније), даје дозвола једном *master*-у и тај *master* обавља циклус на магистрали. Међу процесорима, виши приоритет има процесор са мањим идентификационим бројем (ID). Међу модулима, виши приоритет има модул са мањим бројем. Циклус враћања очитаног податка из меморије има

највиши приоритет. Уколико процесор постави захтев модулу који је заузет обрадом раније задате операције, модул му одговара негативно у истом такту (*ack*=0), процесор одустаје од захтева, паузира 2 такта, а онда понавља исти захтев. Након иницијализације операције уписа у меморију меморијски модул је заузет наредне две периоде сигнала такта. Након иницијализације операције читања из меморије модул је заузет две периоде сигнала такта и одговор даје тек у трећој. Четири процесора постављају захтеве за операције са меморијом на следећи начин (формат је: такт:IDprocesora-адреса(операција)):

1:0-00000000h(Rd), 1:1-0000FFFFh(Rd), 4:3-FF000000h(Wr), 5:2-FF0000FFh(Rd)

Приказати како теку циклуси на магистрали за ову секвенцу. Приказ дати табеларно, тако да се по редовима табеле наводе циклуси на магистрали, прва колона даје редни број такта, друга садржај на адресној магистрали, трећа садржај на магистрали података, а четврта, пета и шеста вредности сигнала *rd*, *wr* и *ack*, редом. За вредност на магистрали података током циклуса уписа податка или враћања прочитаног податка из меморије ставити X.

а) За случај да најнижи бити адресе одређују модул.

б) За случај да највиши бити адресе одређују модул.

Решење:

| а) Приспео | Уређај | Адреса | Податак | Операција | Обрађено | Ack | Нови захтеви |
|---|---|---|---|---|---|---|---|
| 1 | P0 | 00000000 | 0 | Rd | 1 | 1 | M0 одговара у T=4 |
| 1 | P1 | 0000FFFF | 1 | Rd | 2 | 1 | M255 одговара у T=5 |
| 4 | P3 | FF000000 | X | Wr | 7 | 1 | M0 заузето до T9 |
| 5 | P2 | FF0000FF | 2 | Rd | 6 | 1 | M255 одговара у T=9 |
| 4 | M0 | 0 | X | Da | 4 | 1 | - |
| 5 | M255 | 1 | X | Da | 5 | 1 | - |
| 9 | M255 | 2 | X | Da | 9 | 1 | - |

**Табела захтева арбитратору**

| Такт | Addres Bus | Data Bus | rd | wr | ack |
|---|---|---|---|---|---|
| 1 | 00000000 | 0 | 1 | 0 | 1 |
| 2 | 0000FFFF | 1 | 1 | 0 | 1 |
| 3 | - | | | | |
| 4 | 0 | X | 1 | 1 | 1 |
| 5 | 1 | X | 1 | 1 | 1 |
| 6 | FF0000FF | 2 | 1 | 0 | 1 |
| 7 | FF000000 | X | 0 | 1 | 1 |
| 8 | - | | | | |
| 9 | 2 | X | 1 | 1 | 1 |

**Табела садржаја на магистрали**

б)

| Приспео | Уређај | Адреса | Податак | Операција | Обрађено | Ack | Нови захтеви |
|---|---|---|---|---|---|---|---|
| 1 | P0 | 00000000 | 0 | Rd | 1 | 1 | M0 одговара у T=4 |
| 1 | P1 | 0000FFFF | 1 | Rd | 2 | 0 | Поново у T5 |
| 4 | P3 | FF000000 | X | Wr | 7 | 0 | Поново у T10 |
| 5 | P2 | FF0000FF | 2 | Rd | 6 | 1 | M255 одговара у T=9 |
| 4 | M0 | 0 | X | Da | 4 | 1 | - |
| 5 | P1 | 0000FFFF | 1 | Rd | 5 | 1 | M0 одговара у T=8 |
| 8 | M0 | 1 | X | Da | 8 | 1 | - |
| 9 | M255 | 2 | X | Da | 9 | 1 | - |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10 | P3 | FF000000 | X | Wr | 10 | 1 | M255 заузето до T12 |

**Табела захтева арбитратору**

| Такт | Addres Bus | Data Bus | rd | wr | ack |
|---|---|---|---|---|---|
| 1 | 00000000 | 00 | 1 | 0 | 1 |
| 2 | 0000FFFF | 01 | 1 | 0 | 0 |
| 3 | - | | | | |
| 4 | 00 | X | 1 | 1 | 1 |
| 5 | 0000FFFF | 01 | 1 | 0 | 1 |
| 6 | FF0000FF | 02 | 1 | 0 | 1 |
| 7 | FF000000 | X | 0 | 1 | 0 |
| 8 | 01 | X | 1 | 1 | 1 |
| 9 | 02 | X | 1 | 1 | 1 |
| 10 | FF000000 | X | 0 | 1 | 1 |

**Табела садржаја на магистрали**

## Задатак 8.

Посматра се рачунарски систем са меморијским системом са преклопљеним приступом меморијским модулима, тако да је оптимизација извршена за секвенцијални приступ процесора меморији. Величина једног меморијског модула је 1MB и постоји 16 модула. Адресибилна јединица и ширина акумулатора су по 1 бајт. Циклус повратка прочитаног податка из меморије идентификује се тако што су обе контролне линије *rd* и *wr* активне. Претпоставити да након иницијализације операције уписа у меморију меморијски модул је заузет наредне три периода сигнала такта и да је након иницијализације операције читања из меморије модул заузет две периоде сигнала такта и одговор даје тек у трећој.

Сви захтеви за приступ магистрали постављају се синхроно, на исти сигнал такта. Арбитрација се обавља комбинационо, тако да се у истом такту арбитрирају сви захтеви постављени на тај сигнал такта (или раније), даје дозвола једном *master*-у и тај *master* обавља циклус на магистрали. Међу процесорима, виши приоритет има процесор са вишим идентификационим бројем (ID). Међу модулима, виши приоритет има модул са мањим бројем. Циклус враћања очитаног податка из меморије има највиши приоритет. Уколико процесор постави захтев модулу који је заузет обрадом раније задате операције, модул му одговара негативно у истом такту (*ack*=0), процесор одустаје од захтева, паузира 2 такта, а онда понавља исти захтев. Три процесора извршавају следеће програмске сегменте (формат заглавља је: такт:IDprocesora, све вредности су хексадецималне):

| такт 0:1 (IDprocesora) | 0:2 | 0:3 |
|---|---|---|
| 1000 LOAD A001h | F000 PUSH | 4800 STORE 18h |
| 1004 .... | F001 ... | 4802 ... |

Претпоставити да су вредност за процесор 1 ACC=8h, за процесор 2 ACC=8h и SP=300h и за процесор 3 ACC=88h. Стек расте од виших на нижим локацијама, а SP показује на прву слободну локацију. Инструкције се дохватају тако што се одмах након пријема једног бајта инструкције у следећем такту упућује захтев за довлачењем следећег бајта. Све инструкције које током извршавања врше приступ подацима у меморији, сам приступ започињу у трећем такту након фазе читања инструкције (прочитају инструкцију, чекају два такт, па крећу са приступом у меморију).

Приказати како теку циклуси на магистрали за ову секвенцу. Приказ дати табеларно, тако да се по редовима табеле наводе циклуси на магистрали, прва колона даје редни број такта, друга садржај на адресној магистрали, трећа садржај на магистрали података, а четврта, пета и шеста вредности сигнала *rd*, *wr* и *ack*, редом. За вредност на магистрали података током циклуса уписа податка или враћања прочитаног податка из меморије ставити X.

Решење:

а)

| Приспео | Уређај | Адреса | Податак | Операција | Обрађено | Ack | Нови захтеви |
|---|---|---|---|---|---|---|---|
| 0 | P1 | 001000 | 1 | Rd | 2 | 0 | Поново у T=5 |
| 0 | P2 | 00F000 | 2 | Rd | 1 | 0 | Поново у T=4 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | P3 | 004800 | 3 | Rd | 0 | 1 | M0 одговара у T=3 |
| 3 | M0 | 3 | X | Da | 3 | 1 | Нови захтев у T=4 |
| 4 | P2 | 00F000 | 2 | Rd | 5 | 1 | M0 одговара у T=8 |
| 5 | P1 | 001000 | 1 | Rd | 6 | 0 | Поново у T=9 |
| 4 | P3 | 004801 | 3 | Rd | 4 | 1 | M1 одговара у T=7 |
| 7 | M1 | 3 | X | Da | 7 | 1 | Нови захтев у T=10 |
| 8 | M0 | 2 | X | Da | 8 | 1 | Нови захтев у T=11 |
| 9 | P1 | 001000 | 1 | Rd | 9 | 1 | M0 одговара у T=12 |
| 10 | P3 | 000018 | 88 | Wr | 10 | 1 | M8 заузет до T=13 |
| 11 | P2 | 000300 | 8 | Wr | 11 | 0 | Поново у T=14 |
| 12 | M0 | 1 | X | Da | 12 | 1 | Нови захтев у T=13 |
| 14 | P2 | 000300 | 8 | Wr | 14 | 1 | M0 заузет до T=17 |
| 13 | P1 | 001001 | 1 | Rd | 13 | 1 | M1 одговара у T=16 |
| 16 | M0 | 1 | X | Da | 16 | 1 | Нови захтев у T=17 |
| 17 | P1 | 001002 | 1 | Rd | 17 | 1 | M2 одговара у T=20 |
| 20 | M2 | 1 | X | Da | 20 | 1 | Нови захтев у T=21 |
| 21 | P1 | 001003 | 1 | Rd | 21 | 1 | M3 одговара у T=24 |
| 24 | M3 | 1 | X | Da | 24 | 1 | Нови захтев у T=27 |
| 27 | P1 | 00A001 | 1 | Rd | 27 | 1 | M1 одговара у T=30 |
| 30 | M1 | 1 | X | Da | 30 | 1 | - |

**Табела захтева арбитратору**

| **Такт** | **Addres Bus** | **Data Bus** | **rd** | **wr** | **ack** |
|---|---|---|---|---|---|
| 0 | 004800 | 3 | 1 | 0 | 1 |
| 1 | 00F000 | 2 | 1 | 0 | 0 |
| 2 | 001000 | 1 | 1 | 0 | 0 |
| 3 | 3 | X | 1 | 1 | 1 |
| 4 | 004801 | 3 | 1 | 0 | 1 |
| 5 | 00F000 | 2 | 1 | 0 | 1 |
| 6 | 001000 | 1 | 1 | 0 | 0 |
| 7 | 3 | X | 1 | 1 | 1 |
| 8 | 2 | X | 1 | 1 | 1 |
| 9 | 001000 | 1 | 1 | 0 | 1 |
| 10 | 000018 | 88 | 0 | 1 | 1 |
| 11 | 000300 | 8 | 0 | 1 | 0 |
| 12 | 1 | X | 1 | 1 | 1 |
| 13 | 001001 | 1 | 1 | 0 | 1 |
| 14 | 000300 | 8 | 0 | 1 | 1 |
| 15 | - | | | | |
| 16 | 1 | X | 1 | 1 | 1 |
| 17 | 001002 | 1 | 1 | 0 | 1 |
| 18 | - | | | | |
| 19 | - | | | | |
| 20 | 1 | X | 1 | 1 | 1 |
| 21 | 001003 | 1 | 1 | 0 | 1 |
| 22 | - | | | | |
| 23 | - | | | | |
| 24 | 1 | X | 1 | 1 | 1 |
| 25 | - | | | | |
| 26 | - | | | | |
| 27 | 00A001 | 1 | 1 | 0 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| 28 | - | | | | |
| 29 | - | | | | |
| 30 | 1 | X | 1 | 1 | 1 |

**Табела садржаја на магистрали**

## Задатак 9.

Рачунарски систем се састоји из три процесора, DMA контролера и оперативне меморије. Меморијске адресе су ширине 32 бита, ширина магистрале података је 32 бита, адресирање је нивоу 32 битне речи. Улазно/излазни адресни простор је меморијски пресликан. Претпоставити да након иницијализације операције уписа у меморију меморијски модул је заузет наредне три периоде сигнала такта и да је након иницијализације операције читања из меморије модул заузет две периоде сигнала такта и одговор даје тек у трећој. Када процесор или DMA контролер и модул комуницирају, трајање циклуса на магистрали је једна периода сигнала такта. Адреса оперативне меморије је ширине 32 бита, а њени најстарији и најмлађи битови су означени са $A_{31}$ и $A_0$, респективно.

а) Предложити организацију оперативне меморије са преклопљеним приступом меморијским модулима, тако да се обезбеди максимална конкурентност и приступ сукцесивним локацијама. Прецизно дати следеће: број меморијских модула, начин адресирања модула, начин адресирања локација унутар модула и распоред адреса по модулима.

б) Циклус повратка прочитаног податка из меморије идентификује се тако што су обе контролне линије *rd* и *wr* активне. Сви захтеви за приступ магистрали постављају се синхроно, на исти сигнал такта. Арбитрација се обавља комбинационо, тако да се у истом такту арбитрирају сви захтеви постављени на тај сигнал такта (или раније), даје дозвола једном *master*-у и тај *master* обавља циклус на магистрали. Међу процесорима, виши приоритет има процесор са мањим идентификационим бројем (ID). DMA контролер има највиши приоритет, редни број му је 3. Међу модулима, виши приоритет има модул са већим бројем. Циклус враћања очитаног податка из меморије има највиши приоритет. Уколико се постави захтев модулу који је заузет обрадом раније задате операције, модул му одговара негативно у истом такту (*ack*=0), одустаје од захтева, паузира 3 такта, а онда понавља исти захтев. Уређаји постављају захтеве за операције са меморијом на следећи начин (формат је: такт:IDUredjaja-адреса(операција)):

1:0-00000000h(Rd), 1:1-00002468h(Rd), 2:2-00076543h(Wr), 0:3-12345678h(Rd)

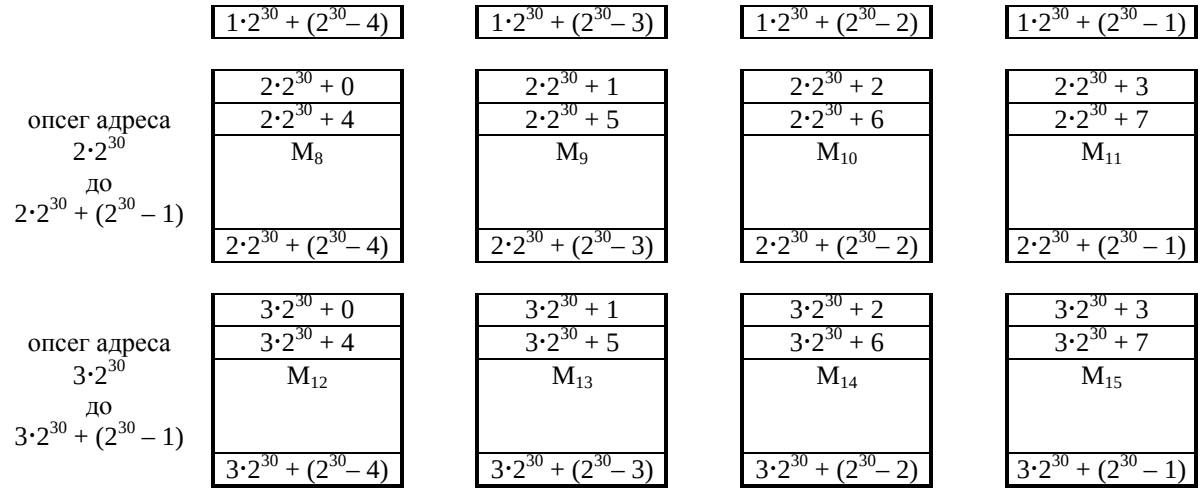
DMA контролер је иницијализован да пребацује 4 речи из меморије на периферију почев од адресе 12345678h. DMA контролер ради у режиму циклус по циклус. Након дохватања једне речи из меморије DMA контролер следећи захтев за приступ меморији генерише након три периоде сигнала такта.

Приказати како теку циклуси на магистрали за ову секвенцу. Приказ дати табеларно, тако да се по редовима табеле наводе циклуси на магистрали, прва колона даје редни број такта, друга садржај на адресној магистрали, трећа садржај на магистрали података, а четврта, пета и шеста вредности сигнала *rd, wr* и *ack*, редом. За вредност на магистрали података током циклуса уписа податка или враћања прочитаног податка из меморије ставити X.

Решење:

а)

| | | | | |
|---|---|---|---|---|
| опсег адреса $0\cdot2^{30}$ до $0\cdot2^{30}+(2^{30}-1)$ | $0\cdot2^{30}+0$ | $0\cdot2^{30}+1$ | $0\cdot2^{30}+2$ | $0\cdot2^{30}+3$ |
| | $0\cdot2^{30}+4$ | $0\cdot2^{30}+5$ | $0\cdot2^{30}+6$ | $0\cdot2^{30}+7$ |
| | $M_0$ | $M_1$ | $M_2$ | $M_3$ |
| | $0\cdot2^{30}+(2^{30}-4)$ | $0\cdot2^{30}+(2^{30}-3)$ | $0\cdot2^{30}+(2^{30}-2)$ | $0\cdot2^{30}+(2^{30}-1)$ |
| опсег адреса $1\cdot2^{30}$ до $1\cdot2^{30}+(2^{30}-1)$ | $1\cdot2^{30}+0$ | $1\cdot2^{30}+1$ | $1\cdot2^{30}+2$ | $1\cdot2^{30}+3$ |
| | $1\cdot2^{30}+4$ | $1\cdot2^{30}+5$ | $1\cdot2^{30}+6$ | $1\cdot2^{30}+7$ |
| | $M_4$ | $M_5$ | $M_6$ | $M_7$ |




б)

| **Приспео** | **Уређај** | **Адреса** | **Податак** | **Операција** | **Обрађено** | **Ack** | **Нови захтеви** |
|---|---|---|---|---|---|---|---|
| 1 | P0 | 00000000 | 0 | Rd | 1 | 0 | Поново у T=5 |
| 1 | P1 | 00002468 | 1 | Rd | 2 | 0 | Поново у T=6 |
| 2 | P2 | 00076543 | X | Wr | 4 | 1 | M3 заузет до T=7 |
| 0 | DMA | 12345678 | 3 | Rd | 0 | 1 | M0 одговара у T=3 |
| 3 | M0 | 03 | X | Da | 3 | 1 | Нови захтев у T=7 |
| 5 | P0 | 00000000 | 0 | Rd | 5 | 1 | M0 одговара у T=8 |
| 6 | P1 | 00002468 | 1 | Rd | 6 | 0 | Поново у T=10 |
| 7 | DMA | 12345679 | 3 | Rd | 7 | 1 | M1 одговара у T=10 |
| 8 | M0 | 00 | X | Da | 8 | 1 | - |
| 10 | P1 | 00002468 | 1 | Rd | 11 | 1 | M0 одговара у T=14 |
| 10 | M1 | 03 | X | Da | 10 | 1 | Нови захтев у T=14 |
| 14 | DMA | 1234567A | 3 | Rd | 15 | 1 | M2 одговара у T=18 |
| 14 | M0 | 01 | X | Da | 14 | 1 | - |
| 18 | M2 | 03 | X | Da | 18 | 1 | Нови захтев у T=22 |
| 22 | DMA | 1234567B | 3 | Rd | 22 | 1 | M3 одговара у T=25 |
| 25 | M3 | 03 | X | Da | 25 | 1 | - |

**Табела захтева арбитратору**

| **Такт** | **Addres Bus** | **Data Bus** | **rd** | **wr** | **ack** |
|---|---|---|---|---|---|
| 0 | 12345678 | 3 | 1 | 0 | 1 |
| 1 | 00000000 | 0 | 1 | 0 | 0 |
| 2 | 00002468 | 1 | 1 | 0 | 0 |
| 3 | 3 | X | 1 | 1 | 1 |
| 4 | 00076543 | 2 | 0 | 1 | 1 |
| 5 | 00000000 | 0 | 1 | 0 | 1 |
| 6 | 00002468 | 1 | 1 | 0 | 0 |
| 7 | 12346579 | 3 | 1 | 0 | 1 |
| 8 | 0 | X | 1 | 1 | 1 |
| 9 | - | | | | |
| 10 | 3 | X | 1 | 1 | 1 |
| 11 | 00002468 | 1 | 1 | 0 | 1 |
| 12 | - | | | | |
| 13 | - | | | | |
| 14 | 1 | X | 1 | 1 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| 15 | 1234567A | 3 | 1 | 0 | 1 |
| 16 | - | | | | |
| 17 | - | | | | |
| 18 | 3 | X | 1 | 1 | 1 |
| 19 | - | | | | |
| 20 | - | | | | |
| 21 | - | | | | |
| 22 | 1324657B | 3 | 1 | 0 | 1 |
| 23 | - | | | | |
| 24 | - | | | | |
| 25 | 3 | X | 1 | 1 | 1 |

**Табела садржаја на магистрали**

# ПРОТОЧНА ОБРАДА (PIPELINE)

## Задатак 1.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном обрадом. За наведене програмске сегменте навести да ли постоји хазард података или не, да ли постоји stall циклуси, ако постоји хазард података да ли се може решити прослеђивањем података и ако постоји како се прослеђују подаци:

а) ADD R4, R6, R7
ADD R8, R2, R7
ADD R7, R6, R4

б) SW R4, (R6)4
LW R3, (R4)8
ADD R4, R6, R8

в) ADD R7, R9, R10
SW R10, (R7)4
LW R7, (R9)4

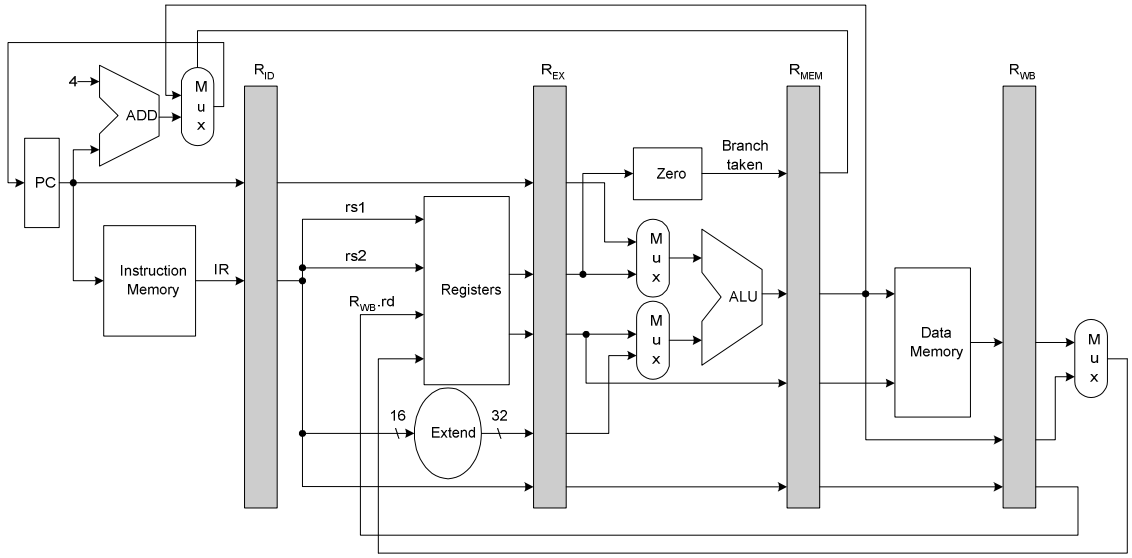
г) LW R8, (R4)8
ADD R4, R6, R10
ADD R2, R8, R10

д) ADD R4, R5, R10
ADD R10, R2, R4
SW R11, (R5)8

ђ) LW R4, (R5)8
ADD R7, R5, R4
ADD R5, R8, R5

е) LW R4, (R5)8
SW R4, (R3)4
ADD R5, R3, R2

Решење:




Процесор са стандардном проточном обрадом

а) Хазард података, прослеђивање

б) Без хазарда података

в) Хазард података, прослеђивање

г) Хазард података, прослеђивање

д) Хазард података, прослеђивање

ђ) Хазард података, stalla циклус између LOAD и ADD, прослеђивање

е) Хазард података, прослеђивање

## Задатак 2.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном обрадом.

а) Набројати микрооперације потребне за извршавање инструкције LOAD са регистарско индиректним адресирања са померајем.

б) Набројати микрооперације потребне за извршавање ALU инструкције.

в) Посматра се програмски сегмент који се састоји од 3 инструкције LOAD-ALU-ALU. Приказати табеларно шта се дешава у којој фази за сваку инструкцију. Размотрити организације без прослеђивања и са прослеђивањем.

г) Колико је број инструкција у јединици времена за овај процесор? Колико времена је потребно за извршење наведеног програмског сегмента?

Решење:

а)

“register-relative” LOAD

IF

IF/ID.IR <- Mem[PC];

IF/ID.NPC,PC <- (if ((EX/MEM.opcode==branch) & EX/MEM.conr){EX/MEM.ALUOutput}

else {PC+4});

ID

ID/EX.A <- Regs[IF/ID.IR6..10];

ID/EX.B <- Regs[IF/ID.IR11..15];
ID/EX.NPC <- IF/ID.NPC;
ID/EX.IR <- IF/ID.IR;
ID/EX.Imm <- $(IF/ID.IR16)^{16}$##IF/ID.IR16..31;

EX
EX/MEM.IR<- ID/EX.IR
EX/MEM.ALUOutput <- ID/EX.A+ID/EX.Imm;
EX/MEM.cond <- 0;
EX/MEM.B<-ID/EX.B;

MEM
MEM/WB.IR <- EX/MEM.IR;
MEM/WB.LMD <- Mem[EX/MEM.ALUOutput];

WB
Regs[MEM/WB.IR11..15] <- MEM/WB.LMD;

б) “ALU-type instruction.”
IF
IF/ID.IR <- Mem[PC];
IF/ID.NPC,PC <- (if ((EX/MEM.opcode==branch) & EX/MEM.conr){EX/MEM.ALUOutput}
else {PC+4});
ID
ID/EX.A <- Regs[IF/ID.IR6..10]; ID/EX.B <- Regs[IF/ID.IR11..15];
ID/EX.NPC <- IF/ID.NPC; ID/EX.IR <- IF/ID.IR;
ID/EX.Imm <- $(IF/ID.IR16)^{16}$##IF/ID.IR16..31;

EX
EX/MEM.IR <- ID/EX.IR;
EX/MEM.ALUOutput <- ID/EX.A func ID/EX.B;
or
EX/MEM.ALUOutput <- ID/EX.A op ID/EX.Imm;
EX/MEM.cond <- 0;

MEM
MEM/WB.IR <- EX/MEM.IR;
MEM/WB.ALUOutput <- EX/MEM.ALUOutput;

WB

Regs[MEM/WB.IR16..20] <- MEM/WB.ALUOutput;

or

Regs[MEM/WB.IR11..15] <- MEM/WB.ALUOutput;

в)

У случају да нама хазарда

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R4, R5 | | IF | ID | EX | MEM | WB | | | | |
| OR R6, R7, R8 | | | IF | ID | EX | MEM | WB | | | |

RW хазард између LOAD и прве ALU инструкције (R1). Без прослеђивања. Упис у регистар и читање уписаног податка није могуће у устом такту.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R1, R5 | | IF | stall | stall | stall | ID | EX | MEM | WB | |
| OR R6, R7, R8 | | | stall | stall | stall | IF | ID | EX | MEM | WB |

RW хазард између LOAD и прве ALU инструкције (R1). Без прослеђивања. Упис у регистар и читање уписаног податка је могуће у устом такту.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R1, R5 | | IF | stall | stall | ID | EX | MEM | WB | | |
| OR R6, R7, R8 | | | stall | stall | IF | ID | EX | MEM | WB | |

RW хазард између LOAD и прве ALU инструкције (R1). Са прослеђивањем.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R1, R5 | | IF | ID | stall | EX | MEM | WB | | | |
| OR R6, R7, R8 | | | IF | stall | ID | EX | MEM | WB | | |

RW хазард између LOAD и друге ALU инструкције (R1). Без прослеђивања. Упис у регистар и читање уписаног податка није могуће у устом такту.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R4, R5 | | IF | ID | EX | MEM | WB | | | | |
| OR R6, R1, R8 | | | IF | stall | stall | ID | EX | MEM | WB | |

RW хазард између LOAD и друге ALU инструкције (R1). Са прослеђивањен.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R4, R5 | | IF | ID | EX | MEM | WB | | | | |
| OR R6, R1, R8 | | | IF | ID | EX | MEM | WB | | | |

RW хазард између прве и друге ALU инструкције (R3). Без прослеђивања. Упис у регистар и читање уписаног податка је могуће у устом такту.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R4, R5 | | IF | ID | EX | MEM | WB | | | | |
| OR R6, R3, R8 | | | IF | stall | stall | ID | EX | MEM | WB | |

RW хазард између прве и друге ALU инструкције (R3). Са прослеђивањем.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R4, R5 | | IF | ID | EX | MEM | WB | | | | |
| OR R6, R3, R8 | | | IF | ID | EX | MEM | WB | | | |

RW хазард између LOAD и друге ALU инструкције (R1), и између прве и друге ALU инструкције (R3). Без прослеђивања. Упис у регистар и читање уписаног податка је могуће у устом такту.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R4, R5 | | IF | ID | EX | MEM | WB | | | | |
| OR R6, R1, R3 | | | IF | stall | stall | ID | EX | MEM | WB | |

RW хазард између LOAD и прве ALU инструкције (R1), и RW хазард између прве и друге ALU инструкције (R3). Без прослеђивања. Упис у регистар и читање уписаног податка није могуће у устом такту.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R1, R5 | | IF | stall | stall | stall | ID | EX | MEM | WB | |
| OR R6, R3, R8 | | | stall | stall | stall | IF | stall | stall | stall | ID |

| **Инструкција** | ... | **11** | **12** | **13** |
|---|---|---|---|---|
| LW R1, (R2)0 | | | | |
| ADD R3, R1, R5 | | | | |
| OR R6, R3, R8 | | EX | MEM | WB |

RW хазард између LOAD и прве ALU инструкције (R1), и RW хазард између прве и друге ALU инструкције (R3). Са прослеђивањем.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R1, R5 | | IF | ID | stall | EX | MEM | WB | | | |
| OR R6, R3, R8 | | | IF | stall | ID | EX | MEM | WB | | |

…

г)

CPI=1.

Уколико се не јаве хазарди потребно је 7 (4+3=7) тактова да се изврше 3 инструкције.

…

## Задатак 3.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном обрадом.

а) Навести шта чини интерну структуру сваког степена проточне обраде.

б) Навести који се све делови процесора користе у сваком кораку проточне обраде уколико се јави LOAD инструкција.

в) Навести који се све делови процесора користе у сваком кораку проточне обраде уколико се јави ALU инструкција.

г) Посматра се програмски сегмент који се састоји од 3 инструкције LOAD-ALU-STORE. Приказати табеларно шта се дешава у којој фази за сваку инструкцију. Размотрити организације без прослеђивања и са прослеђивањем.

Решење:

а)

IF: PC register, Address calculation unit, Instruction Cache memory

IF/ID: IF/ID.IR, IF/ID.NPC

ID: Register file, sign extension unit

ID/EX: ID/EX.A, ID/EX.B, ID/EX.NPC, ID/EX.IR, ID/EX.Imm

EX: Arithmetical and Logical Unit, Branch condition calculation unit

EX/MEM: EX/MEM.IR, EX/MEM.ALUOUT, EX/MEM.cond, EX/MEM.B, EX/MEM.NPC

MEM: Data Cache memory

MEM/WB: MEM/WB.IR, MEM/WB.ALUOUT, MEM/WB.LMD

WB: Multiplexer

б)

IF: PC register, Address calculation unit, Instruction Cache memory

IF/ID: IF/ID.IR, IF/ID.NPC

ID: Register file, sign extension unit

ID/EX: ID/EX.A, ID/EX.B, ID/EX.NPC, ID/EX.IR, ID/EX.Imm

EX: Arithmetical and Logical Unit

EX/MEM.IR, EX/MEM.ALUOUT, EX/MEM.cond, EX/MEM.B

MEM: Data Cache memory

MEM/WB: MEM/WB.IR, MEM/WB.LMD

WB: Multiplexer, Register file

в)

IF: PC register, Address calculation unit, Instruction Cache memory

IF/ID: IF/ID.IR, IF/ID.NPC

ID: Register file, sign extension unit

ID/EX: ID/EX.A, ID/EX.B, ID/EX.NPC, ID/EX.IR, ID/EX.Imm

EX: Arithmetical and Logical Unit

EX/MEM: EX/MEM.IR, EX/MEM.ALUOUT, EX/MEM.cond

MEM: -

MEM/WB: MEM/WB.IR, MEM/WB.ALUOUT

WB: Multiplexer

г)

RW хазард између LOAD и ALU инструкције (R1). RW хазард између ALU и STORE инструкције (R3). Без прослеђивања. Упис у регистар и читање уписаног податка није могуће у устом такту.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| ADD R3, R1, R5 | | IF | stall | stall | stall | ID | EX | MEM | WB | |
| SW R3, (R8)0 | | | stall | stall | stall | IF | stall | stall | stall | ID |

| **Инструкција** | ... | **11** | **12** | **13** |
|---|---|---|---|---|
| LW R1, (R2)0 | | | | |
| ADD R3, R1, R5 | | | | |
| SW R3, (R8)0 | | EX | MEM | WB |

RW хазард између LOAD и ALU инструкције (R1). RW хазард између ALU и STORE инструкције (R3). Са прослеђивањем. Упис у регистар и читање уписаног податка је могуће у устом такту.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)0 | IF | ID | EX | MEM* | WB | | | | | |
| ADD R3, R1, R5 | | IF | ID | stall | *EX* | MEM | WB | | | |
| SW R3, (R8)0 | | | IF | stall | ID | EX | *MEM | WB | | |

…

## Задатак 4.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном обрадом код кога постоји прослеђивање података (forwarding). Посматра се следећи програмски сегмент:

SUB R2, R1, R3

AND R12, R2, R5

OR R13, R6, R2

ADD R14, R2, R2

SW R15, (R2)100h

Приказати табеларно шта се дешава у којој фази за сваку инструкцију.

Решење:

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| SUB R2, R1, R3 | IF | ID | EX | MEM | WB | | | | | |
| AND R12, R2, R5 | | IF | ID | EX | MEM | WB | | | | |
| OR R13, R6, R2 | | | IF | ID | EX | MEM | WB | | | |
| ADD R14, R2, R2 | | | | IF | ID | EX | MEM | WB | | |
| SW R15, (R2)100h | | | | | IF | ID | EX | MEM | WB | |

## Задатак 5.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном обрадом код кога постоји прослеђивање података (forwarding). Написати асемблерски програм који избегава stalls циклусе за извршавање следећег низа израза:

a=b+c ;

d=a - f ;

e=g - h ;

Претпоставити да су све променљиве целобројне.

Решење:

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LW Rb, (R0)b | IF | ID | EX | M | WB | | | | | | | | | | |
| LW Rc, (R0)c | | IF | ID | EX | M | WB | | | | | | | | | |
| LW Rf, (R0)f | | | IF | ID | EX | M | WB | | | | | | | | |
| ADD Ra, Rb, Rc | | | | IF | ID | EX | M | WB | | | | | | | |
| SW Ra, (R0)a | | | | | IF | ID | EX | M | WB | | | | | | |
| SUB Rd, Ra, Rf | | | | | | IF | ID | EX | M | WB | | | | | |
| LW Rg, (R0)g | | | | | | | IF | ID | EX | M | WB | | | | |
| LW Rh, (R0)h | | | | | | | | IF | ID | EX | M | WB | | | |
| SW Rd, (R0)d | | | | | | | | | IF | ID | EX | M | WB | | |
| SUB Re, Rg, Rh | | | | | | | | | | IF | ID | EX | M | WB | |
| SW Re, (R0)e | | | | | | | | | | | IF | ID | EX | M | WB |

## Задатак 6.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном обрадом код кога постоји прослеђивање података (forwarding). Извршава се следећи програмски сегмент:

LW R1, (R1)0

SUB R4, R1, R5

AND R6, R1 R7

OR R8, R1, R9

Приказати табеларно шта се дешава у којој фази за сваку инструкцију.

Решење:

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R1)0 | IF | ID | EX | MEM | WB | | | | |
| SUB R4, R1, R5 | | IF | ID | stall | EX | MEM | WB | | |
| AND R6, R1 R7 | | | IF | stall | ID | EX | MEM | WB | |
| OR R8, R1, R9 | | | | | IF | ID | EX | MEM | WB |

## Задатак 7.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном код кога постоји прослеђивање података (forwarding), при чему се користити јединствена меморија и за приступ инструкцијама (Instruction Memoru) и за приступ подацима (Data Memory). Уколико се из више степена покуша приступи меморији предност има инструкција која се дуже налази у проточној обради. Извршава се следећи програмски сегмент:

LW R1, (R1)0

SW R5, (R6)100h

SUB R2, R3, R4

OR R5, R1, R9

ADD R7, R8, R11

Приказати табеларно шта се дешава у којој фази за сваку инструкцију.

Решење:

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R1)0 | IF | ID | EX | MEM* | WB | | | | | | |
| SW R5, (R6)100h | | IF | ID | EX | MEM* | WB | | | | | |
| SUB R2, R3, R4 | | | IF | ID | EX | MEM | WB | | | | |
| OR R5, R1, R9 | | | | stall | stall | IF | ID | EX | MEM | WB | |
| ADD R7, R8, R11 | | | | | | | IF | ID | EX | MEM | WB |

# Задатак 8.

Посматра се систем са процесором који садржи петостепену стандардну проточну обраду, код кога постоји прослеђивање података (forwarding), код које се утврђивање да ли је дошло до скока или не обавља у другом степену и извршава се следећи програмски сегмент

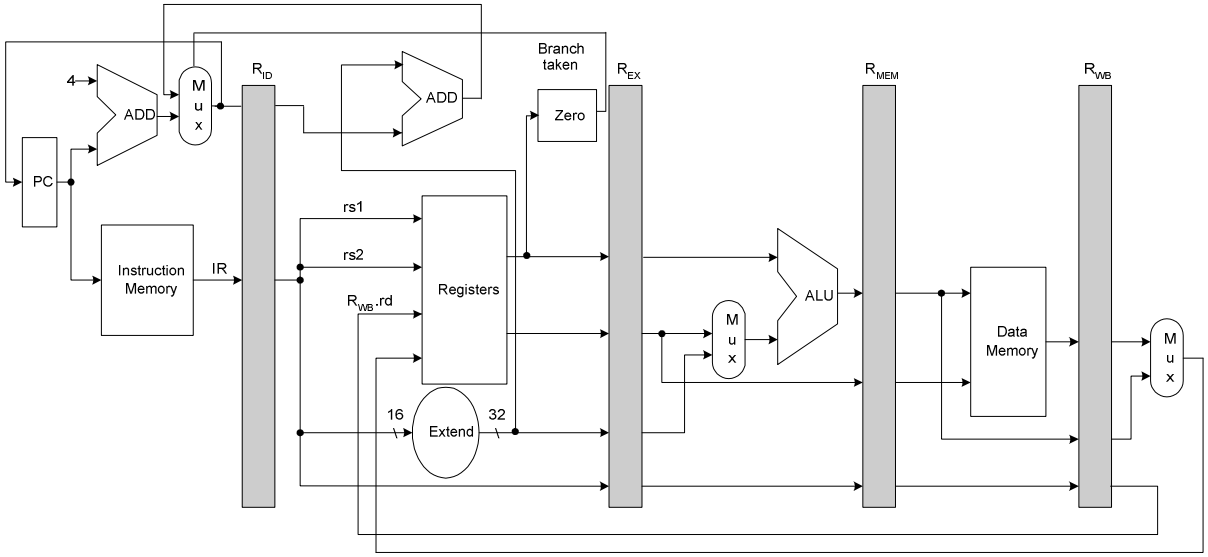
```
    LW R2, (R0)10;
    BNEZ R1, skip;
    ADD R1, R1, R2;
skip: SUB R1, R2, R1;
```

а) Шта се дешава током извршавања овог сегмента ако се инструкције скока извршавају без предикције са заустављањем и пражњењем проточне обраде?

б) Шта се дешава током извршавања овог сегмента ако се инструкције скока извршавају без предикције са заустављањем проточне обраде?

в) Шта се дешава током извршавања овог сегмента ако се инструкције скока извршавају са предикцијом да нема скока?

Решење:




Процесор са стандардном проточном обрадом код које се утврђивање да ли је дошло до скока или не обавља у другом степену

а) Уколико услов скока није испуњен.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| LW R2, (R0)10 | IF | ID | EX | MEM | WB | | | | |
| BNEZ R1, skip | | IF | ID | EX | MEM | WB | | | |
| ADD R1, R1, R2 | | | IF | idle | idle | idle | idle | | |
| ADD R1, R1, R2 | | | | IF | ID | EX | MEM | WB | |
| SUB R1, R2, R1 | | | | | IF | ID | EX | MEM | WB |

Уколико је услов скока испуњен.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| LW R2, (R0)10 | IF | ID | EX | MEM | WB | | | | |
| BNEZ R1, skip | | IF | ID | EX | MEM | WB | | | |
| ADD R1, R1, R2 | | | IF | idle | idle | idle | idle | | |
| skip: SUB R1, R2, R1 | | | | IF | ID | EX | MEM | WB | |

б) Уколико услов скока није испуњен.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| LW R2, (R0)10 | IF | ID | EX | MEM | WB | | | |
| BNEZ R1, skip | | IF | ID | EX | MEM | WB | | |
| ADD R1, R1, R2 | | | IF | ID | EX | MEM | WB | |
| SUB R1, R2, R1 | | | | IF | ID | EX | MEM | WB |

Уколико је услов скока испуњен.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| LW R2, (R0)10 | IF | ID | EX | MEM | WB | | | |
| BNEZ R1, skip | | IF | ID | EX | MEM | WB | | |
| ADD R1, R1, R2 | | | IF | idle | idle | idle | idle | |
| skip: SUB R1, R2, R1 | | | | IF | ID | EX | MEM | WB |

в) Уколико услов скока није испуњен.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| LW R2, (R0)10 | IF | ID | EX | MEM | WB | | | |
| BNEZ R1, skip | | IF | ID | EX | MEM | WB | | |
| ADD R1, R1, R2 | | | IF | ID | EX | MEM | WB | |
| SUB R1, R2, R1 | | | | IF | ID | EX | MEM | WB |

Уколико је услов скока испуњен.

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| LW R2, (R0)10 | IF | ID | EX | MEM | WB | | | |
| BNEZ R1, skip | | IF | ID | EX | MEM | WB | | |
| ADD R1, R1, R2 | | | IF | idle | idle | idle | idle | |
| skip: SUB R1, R2, R1 | | | | IF | ID | EX | MEM | WB |

## Задатак 9.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном обрадом код кога постоји прослеђивање података (forwarding) и статичка предикција да скок није прихваћен (branch non-taken prediction), код које се утврђивање да ли је дошло до скока или не обавља у другом степену. Која ће инструкција бити учитана из меморије у 5 такту, уколико је почетни садржај регистра R0 је 0h?

```
        ADDI    R1, R0, #10
        SUBI    R1, R1, #10
        BEQZ    R1, exit
        ADD     R2, R3, R4
Exit    SUB     R2, R3, R4
```

Решење:

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| ADDI R1, R0, #10 | IF | ID | EX | MEM | WB | | | | | |
| SUBI R1, R1, #10 | | IF | ID | EX | MEM | WB | | | | |
| BEQZ R1, exit | | | IF | ID | stall | EX | MEM | WB | | |
| ADD R2, R3, R4 | | | | IF | stall | idle | idle | idle | idle | |
| SUB R2, R3, R4 | | | | | | IF | ID | EX | MEM | WB |

У такту број 5 из меморије ће бити учитана инструкција: ADD R2, R3, R4.

## Задатак 10.

Посматра се систем са процесором који садржи петостепену стандардну проточну обраду, код које се утврђивање да ли је дошло до скока или не обавља у другом степену, код кога постоји прослеђивање и извршава се следећи програмски сегмент

LW R1, (R0)10;

BNEZ R1, skip;

ADD R1, R1, R2;

skip: SUB R1, R2, R1;

Шта се дешава током извршавања овог сегмента ако се инструкције скока извршавају са предикцијом да нема скока?

Решење:

Погодак

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R0)10; | IF | ID | EX | MEM | WB | | | | | |
| BNEZ R1, skip; | | IF | ID | stall | stall | EX | MEM | WB | | |
| ADD R1, R1, R2; | | | IF | stall | stall | ID | EX | MEM | WB | |
| SUB R1, R2, R1; | | | | | | IF | ID | EX | MEM | WB |

Промашај:

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R0)10; | IF | ID | EX | MEM | WB | | | | | |
| BNEZ R1, skip; | | IF | ID | stall | stall | EX | MEM | WB | | |
| ADD R1, R1, R2; | | | IF | stall | stall | idle | idle | idle | idle | |
| SUB R1, R2, R1; | | | | | | IF | ID | EX | MEM | WB |

## Задатак 11.

Посматра се систем са процесором који садржи петостепену стандардну проточну обраду, код које се утврђивање да ли је дошло до скока или не обавља у другом степену, код кога постоји прослеђивање и извршава се следећи програмски сегмент

ADDI R1, R0, #10;

BNEZ R1, skip;

ADD R1, R1, R2;

skip: SUB R1, R2, R1;

Шта се дешава током извршавања овог сегмента ако се инструкције скока извршавају са предикцијом да нема скока?

Решење:

Погодак

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| ADDI R1, R0, #10; | IF | ID | EX | MEM | WB | | | | |
| BNEZ R1, skip; | | IF | ID | stall | EX | MEM | WB | | |
| ADD R1, R1, R2; | | | IF | stall | ID | EX | MEM | WB | |
| SUB R1, R2, R1; | | | | | IF | ID | EX | MEM | WB |

Промашај:

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| ADDI R1, R0, #10; | IF | ID | EX | MEM | WB | | | | |
| BNEZ R1, skip; | | IF | ID | stall | EX | MEM | WB | | |
| ADD R1, R1, R2; | | | IF | stall | idle | idle | idle | idle | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| SUB R1, R2, R1; | | | | | IF | ID | EX | MEM | WB |

## Задатак 12.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном обрадом код кога постоји прослеђивање података (forwarding) и статичка предикција да скок није прихваћен (branch non-taken prediction), код које се утврђивање да ли је дошло до скока или не обавља у другом степену. Која ће се вредност налазити у регистру R2 на крају сегмента уколико се на почетку у њему налазила променљива X?

```
        SUBI    R1, R2, #10
        BEQZ    R1, exit
exit    ADDI    R2, R2, #1
```

Решење:

if (X <> 10)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| SUBI R1, R2, #10 | IF | ID | EX | MEM | WB | | | | | |
| BEQZ R1, exit | | IF | ID | stall | EX | MEM | WB | | | |
| ADDI R2, R2, #1 | | | IF | stall | ID | EX | MEM | WB | | |

if (X=10)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| SUBI R1, R2, #10 | IF | ID | EX | MEM | WB | | | | | |
| BEQZ R1, exit | | IF | ID | stall | EX | MEM | WB | | | |
| ADDI R2, R2, #1 | | | IF | stall | idle | idle | idle | idle | | |
| ADDI R2, R2, #1 | | | | | IF | ID | EX | MEM | WB | |

У регистру R2 ће се на крају сегмента налазити X=X+1.

## Задатак 13.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза помоћу процесора са стандардном проточном обрадом. Извршава се следећи програмски сегмент:

```
loop:   LW      R1, (R2)0
        ADDI    R1, R1, #1
        SW      R1, (R2)0
        ADDI    R2, R2, #4
        SUB     R4, R3, R2
        BNEZ    R4, loop
        ADD     R1, R1, R2
```

Претпоставити да је почетна вредност R3 регистра R2+396. Приказати табеларно шта се дешава у којој фази за сваку инструкцију:

а) за случај да нема прослеђивања, али се читање и упис у регистар у истом такту прослеђују кроз регистре. Утврђивање да ли је дошло до скока или не обавља у EX/MEM степену. Када се утврди да се ради о инструкцији скока pipeline се празни.

б) Претпоставити да постоји прослеђивање података (forwarding) и статичка предикција да скок није прихваћен (branch non-taken prediction), код које се утврђивање да ли је дошло до скока или не обавља у другом степену.

Решење:

а)

| **Инструкција** | | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| loop: | LW | R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | |
| | ADDI | R1, R1, #1 | | IF | ID | stall | stall | EX | MEM | WB | | |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | SW | R1, (R2)0 | | | IF | stall | stall | ID | stall | stall | EX | MEM |
| | ADDI | R2, R2, #4 | | | | | | IF | stall | stall | ID | EX |
| | SUB | R4, R3, R2 | | | | | | | | | IF | ID |
| | BNEZ | R4, loop | | | | | | | | | | IF |
| | ADD | R1, R1, R2 | | | | | | | | | | |
| | LW | R1, (R2)0 | | | | | | | | | | |

| **Инструкција** | | | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** |
|---|---|---|---|---|---|---|---|---|---|---|
| loop: | LW | R1, (R2)0 | | | | | | | | |
| | ADDI | R1, R1, #1 | | | | | | | | |
| | SW | R1, (R2)0 | WB | | | | | | | |
| | ADDI | R2, R2, #4 | MEM | WB | | | | | | |
| | SUB | R4, R3, R2 | stall | stall | EX | MEM | WB | | | |
| | BNEZ | R4, loop | stall | stall | ID | stall | stall | EX | MEM | WB |
| | ADD | R1, R1, R2 | | | IF | idle | idle | idle | idle | idle |
| | LW | R1, (R2)0 | | | | | | | | IF |

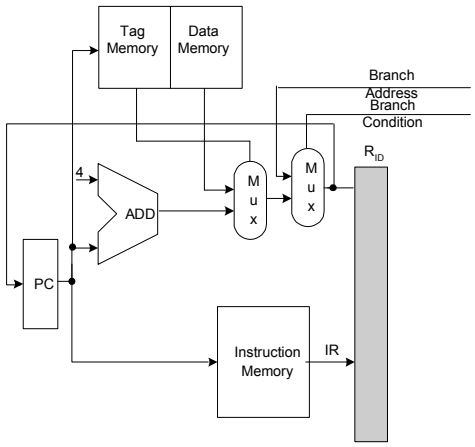
б)

| **Инструкција** | | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| loop: | LW | R1, (R2)0 | IF | ID | EX | MEM | WB | | | | | | | |
| | ADDI | R1, R1, #1 | | IF | ID | stall | EX | MEM | WB | | | | | |
| | SW | R1, (R2) | | | IF | stall | ID | EX | MEM | WB | | | | |
| | ADDI | R2, R2, #4 | | | | | IF | ID | EX | MEM | WB | | | |
| | SUB | R4, R3, R2 | | | | | | IF | ID | EX | MEM | WB | | |
| | BNEZ | R4, loop | | | | | | | IF | ID | stall | EX | MEM | WB |
| | ADD | R1, R1, R2 | | | | | | | | IF | stall | idle | idle | idle |
| | LW | R1, (R2)0 | | | | | | | | | | IF | ID | EX |

## Задатак 14.

Нацртати структурну шему јединице за учитавање инструкција (IF) процесор са стандардном проточном обрадом у случају да се користи кеш за предикцију скока.

Решење:




## Задатак 15.

Посматра се систем са стандардном проточном обрадом. Извршава се следећи програмски сегмент

```
        LW R1, (R7)0
        ADDI R7, R7, #1
LOOP:   LW R2, (R7)0
        ADDI R7, R7, #1
```

```
        SUB R3, R1, R2
        BEQZ  R3, NOTE
        ADDI R1, R2, #0
NOTE:   SUB R4, R7, R5
        BNEZ  R4, LOOP
        ADDI  R7, R7, #1
```

Почетна вредност регистра R7 је 100h, R5 је 105h, а изглед дела меморије почев од адресе 100 је приказан на следећој слици. Адресирање је на нивоу 32-битне речи. Из меморије се чита и у њу уписује 32-битне реч по реч. Почетна адреса програма је 0, а свака инструкција заузима тачно једну адресу. Приказати изглед релевантних делова система након завршетка програмског сегмента уколико се:

а) шема за предвиђање са једним битом за предвиђање скока.

б) шема за предвиђање са једним битом за предвиђање скока и јединица са бафером предвиђања (branch prediction buffer) величине 8.

в) корелисана шема за предвиђање (1, 1) и јединица са бафером предвиђања величине 8.

г) шема за предвиђање са два бита за предвиђање скока.

д) шема за предвиђање са два бита за предвиђање скока и јединица са бафером предвиђања величине 8.

ђ) шема за предвиђање са једним битом за предвиђање скока и јединица са кешом предвиђања (branch target cache) са 8 асоцијативних улаза.

е) шема за предвиђање са два бита за предвиђање скока и јединица са кешом предвиђања (branch target cache) са 8 асоцијативних улаза.

| **Адреса** | 100 | 101 | 102 | 103 | 104 | 105 | 106 |
|---|---|---|---|---|---|---|---|
| **Садржај** | 1 | 1 | 2 | 2 | 2 | 2 | 5 |

Изглед дела меморије

Решење:

Редослед извршавања инструкција: 0, 1, 2, 3, 4, 5+, 7, 8+, 2, 3, 4, 5-, 6, 7, 8+, 2, 3, 4, 5+, 7, 8+, 2, 3, 4, 5+, 7, 8+, 2, 3, 4, 5+, 7, 8-. Ознака+(-) поред адресе инструкције означава да се ради о инструкцији скока и да је услов испуњен (није испуњен).

а)

0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0=> 0

б)

| **улаз** | **предикција** |
|---|---|
| 0 | 0, 1, 1, 1, 1, 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0, 1, 0, 1, 1, 1 |
| 6 | 0 |
| 7 | 0 |

в)

| **улаз** | **предикција** |
|---|---|

| **улаз** | **предикција** |
|---|---|

| | |
|---|---|
| 0 | 0, 1 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0, 1 |
| 6 | 0 |
| 7 | 0 |

| | |
|---|---|
| 0 | 0, 1, 1, 1, 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0, 0, 1, 1, 1 |
| 6 | 0 |
| 7 | 0 |

0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0=> 0

г) Предикција 11 означава јаку предикцију да ће бити скока, 10 слабу предикцију да ће бити скока, 00 означава јаку предикцију да неће бити скока, а 01 слабу предикцију да неће бити скока.

01, 11, 11, 10, 11, 11, 11, 11, 11, 11, 10=> 10

д)

| **Улаз** | **предикција** |
|---|---|
| 0 | 01, 11, 11, 11, 11, 10 |
| 1 | 01 |
| 2 | 01 |
| 3 | 01 |
| 4 | 01 |
| 5 | 01, 11, 10, 11, 11, 11 |
| 6 | 01 |
| 7 | 01 |

ђ) Кеш за предикцију скока дозвољава и читање и упис у истом такту. Уколико се адреса не налази у кешу за предикцију користи се статичка предикција да неће бити скока.

| **улаз** | **PC** | **Next PC** | **Предикција** |
|---|---|---|---|
| 0 | 5 | 7, 6, 7, 7, 7 | 1, 0, 1, 1, 1 |
| 1 | 8 | 2, 2, 2, 2, 9 | 1, 1, 1, 1, 0 |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |

е) Кеш за предикцију скока дозвољава и читање и упис у истом такту. Уколико се адреса не налази у кешу за предикцију користи се слаба предикција да неће бити скока. Оно шта се уписује у кеш за предикцију скока у случају да се одређена адресе на налази и кеш меморији је ствар пројектантске одлуке. У овом примеру је узето да приликом уметања неке адресе у кеш за предикцију у случају да услов скока није испуњен уписује се јака предикција да неће бити скока, а у случају да је услов скока испуњен уписује се јака предикција да ће бити скока.

| **улаз** | **PC** | **Next PC** | **Предикција** |
|---|---|---|---|
| 0 | 5 | 7, 7, 7, 7, 7 | 11, 10, 11, 11, 11 |
| 1 | 8 | 2, 2, 2, 2, 2 | 11, 11, 11, 11, 10 |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |

## Задатак 16.

Посматра се систем са стандардном проточном обрадом код кога постоји прослеђивање података (forwarding), код које се утврђивање да ли је дошло до скока или не обавља у EX/MEM степену. Извршава се следећи програмски сегмент

```
LOOP: LW R1, (R7)0
      BEQZ  R1, EXIT
      ADDI R7, R7, #1
      SUBI R3, R3, #1
      ADD R2, R2, R1
      BNEZ  R3, LOOP
EXIT: SUB R4, R4, R5
      ADD R6, R7, R8
      ADD R9, R10, R11
```

Почетна вредност регистра R2 је 0h, R3 је 0h, R7 је 100h, а изглед дела меморије почев од адресе 100 је приказан на следећој слици. Из меморије се чита и у њу уписује 32-битне реч по реч. Почетна адреса програма је 1000h, а свака инструкција заузима тачно једну адресибилну јединицу, користи кеш за предикцију скока са 8 асоцијативних улаза и два бита за предикцију скока по улазу чији је садржај приказан на следећој слици. Предикција 11 означава јаку предикцију да ће бити скока, 10 слабу предикцију да ће бити скока, 00 означава јаку предикцију да неће бити скока, а 01 слабу предикцију да неће бити скока. Кеш за предикцију скока дозвољава и читање и упис у истом такту. Приказати табеларно шта се дешава у којој фази за првих 9 инструкција датог програма у случају да постоји прослеђивање. Дати табеларно и приказ кеш меморије за предикцију.

| **Адреса** | 100 | 101 | 102 | 103 | 104 | 105 | 106 |
|---|---|---|---|---|---|---|---|
| **Садржај** | 1 | 2 | 0 | 4 | 5 | 6 | 7 |

Изглед дела меморије

| **улаз** | **PC** | **Next PC** | **Предикција** |
|---|---|---|---|
| 0 | 1001 | 1006 | 10 |
| 1 | 1005 | 1000 | 11 |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |

Кеш за предикцију скока

Решење:

Редослед извршавања инструкција: 1000, 1001-, 1002, 1003, 1004, 1005+, 1000, 1001-, 1002.... Ознака+(-) поред адресе инструкције означава да се ради о инструкцији скока и да је услов испуњен (није испуњен).

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R1)0 | IF | ID | EX | MEM | WB | | | | |
| BEQZ  R1, EXIT | | IF | ID | stall | **EX** | **MEM** | **WB** | | |
| SUB R4, R4, R5 | | | IF | stall | ID | EX | idle | idle | |
| ADD R6, R7, R8 | | | | | IF | ID | idle | idle | idle |
| ADD R9, R10, R11 | | | | | | IF | idle | idle | idle |
| ADDI R7, R7, #1 | | | | | | | IF | ID | EX |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| SUBI R3, R3, #1 | | | | | | | | IF | ID |
| ADD R2, R2, R1 | | | | | | | | | IF |
| BNEZ R3, LOOP | | | | | | | | | |
| LW R1, (R7)0 | | | | | | | | | |
| BEQZ R1, EXIT | | | | | | | | | |
| ADDI R7, R7, #1 | | | | | | | | | |

| **Инструкција** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
|---|---|---|---|---|---|---|---|---|
| LW R1, (R1)0 | | | | | | | | |
| BNEZ R1, EXIT | | | | | | | | |
| SUB R4, R4, R5 | | | | | | | | |
| ADD R6, R7, R8 | | | | | | | | |
| ADD R9, R10, R11 | idle | | | | | | | |
| ADDI R7, R7, #1 | MEM | WB | | | | | | |
| SUBI R3, R3, #1 | EX | MEM | WB | | | | | |
| ADD R2, R2, R1 | ID | EX | MEM | WB | | | | |
| BNEZ R3, LOOP | IF | ID | EX | MEM | WB | | | |
| LW R1, (R7)0 | | IF | ID | EX | MEM | WB | | |
| BEQZ R1, EXIT | | | IF | ID | stall | EX | MEM | WB |
| ADDI R7, R7, #1 | | | | IF | stall | ID | EX | MEM |

| **улаз** | **PC** | **Next PC** | **Предикција** |
|---|---|---|---|
| 0 | 1001 | 1006, 1002, 1002 | 10, 00, 00 |
| 1 | 1005 | 1000, 1000 | 11, 11 |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |

## Задатак 17.

Посматра се систем са стандардном проточном обрадом код кога не постоји прослеђивање података (forwarding), код које се утврђивање да ли је дошло до скока или не обавља у EX/MEM степену. Свака фаза извршавања инструкције траје једну периоду сигнала такта укључујући и фазу 2 у којој се чита из регистарског фајла (Registers) и фазу 5 у којој се уписује у регистарски фајл (Registers).

Извршава се следећи програмски сегмент

```
LOOP: LW R2, (R1)0
      ADDI R1, R1, #1
      BEQZ R2, END
      ADD R3, R3, R2
      SUB R4, R1, R5
      BNEZ R4, LOOP
END:  ADDI R1, R1, #1
      SW R3, (R1)0
      ADDI R7, R7, #1
```

Почетна вредност регистра R1 је 100h, R3 је 0h, R5 је 104h, а изглед дела меморије почев од адресе 100 је приказан на следећој слици. Адресирање је на нивоу 32-битне речи. Из меморије се чита и у њу уписује 32-битне реч по реч. Почетна адреса програма је 0, а свака инструкција заузима тачно једну

адресу. Потребно је модификовати јединице за учитавање инструкција (IF) процесор тако да се користи кеш за предикцију скока са 8 асоцијативних улаза и два бита за предикцију скока по улазу који дозвољава и читање и упис у истом такту.

| **Адреса** | 100h | 101h | 102h | 103h | 104h | 105h | 106h |
|---|---|---|---|---|---|---|---|
| **Садржај** | 1 | 1 | 0 | 2 | 2 | 2 | 5 |

Приказати табеларно шта се дешава у којој фази за сваку. Кеш меморија за предикцију скока је пре почетка извршавања сегмента била празна. Користи се стандардна шема за предвиђање са два бита. Предикција 11 означава јаку предикцију да ће бити скока, 10 слабу предикцију да ће бити скока, 00 означава јаку предикцију да неће бити скока, а 01 слабу предикцију да неће бити скока. Кеш за предикцију скока дозвољава и читање и упис у истом такту. Уколико се адреса не налази у кешу за предикцију користи се слаба предикција да неће бити скока. Приликом уметања неке адресе у кеш за предикцију у случају да услов скока није испуњен уписује се слаба предикција да неће бити скока, а у случају да је услов скока испуњен уписује се слаба предикција да ће бити скока.

Решење:

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LOOP: LW R2, (R1)0 | IF | ID | EX | MEM | WB | | | | | |
| ADDI R1, R1, #1 | | IF | ID | EX | MEM | WB | | | | |
| BEQZ R2, END | | | IF | ID | stall | stall | EX | MEM | WB | |
| ADD R3, R3, R2 | | | | IF | stall | stall | ID | EX | MEM | WB |
| SUB R4, R1, R5 | | | | | | | IF | ID | EX | MEM |
| BNEZ R4, LOOP | | | | | | | | IF | ID | stall |
| ADDI R1, R1, #1 | | | | | | | | | IF | stall |

| **Инструкција** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
|---|---|---|---|---|---|---|---|---|---|---|
| SUB R4, R1, R5 | WB | | | | | | | | | |
| BNEZ R4, LOOP | stall | stall | EX | MEM | WB | | | | | |
| ADDI R1, R1, #1 | stall | stall | ID | EX | idle | idle | | | | |
| SW R3, (R1)0 | | | IF | ID | idle | idle | idle | | | |
| ADDI R7, R7, #1 | | | | IF | idle | idle | idle | idle | | |
| LW R2, (R1)0 | | | | | IF | ID | EX | MEM | WB | |
| ADDI R1, R1, #1 | | | | | | IF | ID | EX | MEM | WB |
| BEQZ R2, END | | | | | | | IF | ID | stall | stall |
| ADD R3, R3, R2 | | | | | | | | IF | stall | stall |

| **Инструкција** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** |
|---|---|---|---|---|---|---|---|---|---|---|
| BEQZ R2, END | EX | MEM | WB | | | | | | | |
| ADD R3, R3, R2 | ID | EX | MEM | WB | | | | | | |
| SUB R4, R1, R5 | IF | ID | EX | MEM | WB | | | | | |
| BNEZ R4, LOOP | | IF | ID | stall | stall | stall | EX | MEM | WB | |
| LW R2, (R1)0 | | | IF | stall | stall | stall | ID | EX | MEM | WB |
| ADDI R1, R1, #1 | | | | | | | IF | ID | EX | MEM |
| BEQZ R2, END | | | | | | | | IF | ID | stall |
| ADD R3, R3, R2 | | | | | | | | | IF | stall |

| **Инструкција** | **31** | **32** | **33** | **34** | **35** | **36** | **37** | **38** | **39** | **40** |
|---|---|---|---|---|---|---|---|---|---|---|
| ADDI R1, R1, #1 | WB | | | | | | | | | |
| BEQZ R2, END | stall | EX | MEM | WB | | | | | | |
| ADD R3, R3, R2 | stall | ID | EX | idle | idle | | | | | |
| SUB R4, R1, R5 | | IF | ID | idle | idle | idle | | | | |
| BNEZ R4, LOOP | | | IF | idle | idle | idle | idle | | | |
| ADDI R1, R1, #1 | | | | IF | ID | EX | MEM | WB | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| SW R3, (R1)0 | | | | | IF | ID | stall | stall | stall | EX |
| ADDI R7, R7, #1 | | | | | | IF | stall | stall | stall | ID |

| **Инструкција** | **41** | **42** | **43** | **44** | **45** | **46** | **47** | **48** | **49** | **50** |
|---|---|---|---|---|---|---|---|---|---|---|
| SW R3, (R1)0 | MEM | WB | | | | | | | | |
| ADDI R7, R7, #1 | EX | MEM | WB | | | | | | | |

## Задатак 18.

Посматра се систем са стандардном проточном обрадом код кога постоји прослеђивање података (forwarding), код које се утврђивање да ли је дошло до скока или не обавља у другом степену. Свака фаза извршавања инструкције траје једну периоду сигнала такта укључујући и фазу 2 у којој се чита из регистарског фајла (Registers) и фазу 5 у којој се уписује у регистарски фајл (Registers).

Извршава се следећи програмски сегмент

```
LOOP: LW R2, (R1)0
      ADDI R1, R1, #1
      ADD R3, R3, R2
      BEQZ  R2, END
      SUB R4, R1, R5
      BNEZ  R4, LOOP
END:  ADDI  R1, R1, #1
      SW R3, (R1)0
      ADDI R7, R7, #1
```

Почетна вредност регистра R1 је 100h, R3 је 0h, R5 је 104h, а изглед дела меморије почев од адресе 100 је приказан на следећој слици. Адресирање је на нивоу 32-битне речи. Из меморије се чита и у њу уписује 32-битне реч по реч. Почетна адреса програма је 0, а свака инструкција заузима тачно једну адресу.

| Адреса | 100h | 101h | 102h | 103h | 104h | 105h | 106h |
|---|---|---|---|---|---|---|---|
| Садржај | 1 | 0 | 2 | 0 | 3 | 0 | 4 |

а) Приказати табеларно шта се дешава у којој фази за сваку инструкцију ако се инструкције скока извршавају са предикцијом да нема скока?

б) Дати процесор, који поседује могућност прослеђивања, се модификује тако да уместо заустављања проточне обраде користе технике закашњеног пуњења (*delayed load*) и закашњеног скока (*delayed branch*). Модификовати дати програм тако да одговаран новом процесору, приказати табеларно шта се дешава у којој фази за сваку инструкцију тако модификованог програма.

Решење:

а)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LOOP: LW R2, (R1)0 | IF | ID | EX | MEM | WB | | | | | |
| ADDI R1, R1, #1 | | IF | ID | EX | MEM | WB | | | | |
| ADD R3, R3, R2 | | | IF | ID | EX | MEM | WB | | | |
| BEQZ R2, END | | | | IF | ID | EX | MEM | WB | | |
| SUB R4, R1, R5 | | | | | IF | ID | EX | MEM | WB | |
| BNEZ R4, LOOP | | | | | | IF | ID | stall | EX | MEM |
| ADDI R1, R1, #1 | | | | | | | IF | stall | idle | idle |
| LW R2, (R1)0 | | | | | | | | | IF | ID |
| ADDI R1, R1, #1 | | | | | | | | | | IF |

| **Инструкција** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
|---|---|---|---|---|---|---|---|---|---|---|
| BNEZ R4, LOOP | WB | | | | | | | | | |
| ADDI R1, R1, #1 | idle | idle | | | | | | | | |
| LW R2, (R1)0 | EX | MEM | WB | | | | | | | |
| ADDI R1, R1, #1 | ID | EX | MEM | WB | | | | | | |
| ADD R3, R3, R2 | IF | ID | EX | MEM | WB | | | | | |
| BEQZ R2, END | | IF | ID | EX | MEM | WB | | | | |
| SUB R4, R1, R5 | | | IF | idle | idle | idle | idle | | | |
| ADDI R1, R1, #1 | | | | IF | ID | EX | MEM | WB | | |
| SW R3, (R1)0 | | | | | IF | ID | EX | MEM | WB | |
| ADDI R7, R7, #1 | | | | | | IF | ID | EX | MEM | WB |

б)

```
LOOP: LW R2, (R1)0
      ADDI R1, R1, #1
      NOP
      BEQZ  R2, END
      ADD R3, R3, R2
      SUB R4, R1, R5
      NOP
      BNEZ  R4, LOOP
      NOP
END:  ADDI  R1, R1, #1
      SW R3, (R1)0
      ADDI R7, R7, #1
```

## Задатак 19.

Посматра се систем са процесором који садржи петостепену стандардну проточну обраду. У степен EX се поред постојеће ALU јединице додаје нова EALU јединица која служи за рад са целим бројевима. Операције MUL, DIV трају 3, 10 тактова, респективно. Извршава се следећи програмски сегмент

```
MUL R1, R2, R3;
ADD R4, R5, R3;
OR R5, R6, R7;
AND R3, R1, R4;
```

Приказати табеларно шта се дешава у којој фази за сваку инструкцију ако се инструкције:

а) завршавају у редоследу у коме су започете (In Order), без прослеђивања, уколико се упис и читање податка у регистарски фајл може реализовати у истом такту.

б) завршавају у редоследу у коме су започете (In Order), са прослеђивањем.

в) завршавају независно од редоследа у коме су започете (Out of Order).

г) извршавају на процесору који има могућност да извршава две инструкције у паралели (Super Scalar), које се завршавају независно од редоследа у коме су започете (Out of Order).

Решење:

а)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MUL R1, R2, R3 | IF | ID | EX1 | EX2 | EX3 | MEM | WB | | | | |
| ADD R4, R5, R3 | | IF | ID | EX | stall | stall | MEM | WB | | | |
| OR R5, R6, R7 | | | IF | ID | stall | stall | EX | MEM | WB | | |
| AND R3, R1, R4 | | | | IF | stall | stall | ID | stall | EX | MEM | WB |

б)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| MUL R1, R2, R3 | IF | ID | EX1 | EX2 | EX3 | MEM | WB | | | |
| ADD R4, R5, R3 | | IF | ID | EX* | stall | stall | MEM | WB | | |
| OR R5, R6, R7 | | | IF | ID | stall | stall | EX | MEM | WB | |
| AND R3, R1, R4 | | | | IF | stall | stall | *ID | EX | MEM | WB |

в)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| MUL R1, R2, R3 | IF | ID | EX1 | EX2 | EX3* | MEM | WB | | | |
| ADD R4, R5, R3 | | IF | ID | EX* | MEM | WB | | | | |
| OR R5, R6, R7 | | | IF | ID | EX | stall | MEM | WB | | |
| AND R3, R1, R4 | | | | IF | *ID | stall | *EX | MEM | WB | |

г)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| MUL R1, R2, R3 | IF | ID | EX1 | EX2 | EX3* | MEM | WB | | | |
| ADD R4, R5, R3 | IF | ID | EX | MEM | WB | | | | | |
| OR R5, R6, R7 | | IF | ID | EX | MEM | WB | | | | |
| AND R3, R1, R4 | | IF | ID | stall | stall | *EX | MEM | WB | | |

## Задатак 20.

Посматра се процесором са проточном обрадом који има следећу организацију степене у оквиру проточне обраде: IF, ID, ALU1, MEM1, MEM2, ALU2 и WB. Степен ALU1 се користи за израчунавање адресе код инструкција скока, load и store. Степен ALU2 се користи за сва остала израчунавања и за разрешавање скока. Једине инструкције за приступ меморији су load и store. Једини начин адресирања за приступ меморији је регистарско индиректно адресирање са померајем. Како је како је кеш меморија за податке спора приступ меморији се извршава у два степена MEM1 и MEM2, није дозвољено да две инструкције приступају истовремено кеш меморији за податке. Посматра се следећи програмски сегмент:

LW R1, (R2)50

ADD R3, R1, R4

LW R5, (R3)100

XOR R6, R5, R7

SW R6, (R2)50

ADDI R1, R1, #100

SUBI R4, R4, #8

а) Одредити све зависности у оквиру програмског сегмента.

б) Претпоставити да се pipeline извршава без прослеђивања, уколико се упис и читање податка у регистарски фајл може реализовати у истом такту, колико циклуса је потребно да се изврши посматрани програмски сегмент?

в) Претпоставити да се pipeline извршава са прослеђивањем, колико циклуса је потребно да се изврши посматрани програмски сегмент?

г) Да ли је могуће променити редослед операција да се добије мањи број циклуса?

Решење:

а)

1. LW R1, (R2)50
2. ADD R3, R1, R4
3. LW R5, (R3)100
4. XOR R6, R5, R7
5. SW R6, (R2)50
6. ADDI R1, R1, #100
7. SUBI R4, R4, #8

Зависности:

2 од 1 за R1

6 од 1 за R1

3 од 2 за R3

4 од 3 за R5

5 од 4 за R6

б)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)50 | IF | ID | ALU1 | MEM1 | MEM2 | ALU2 | WB | | | |
| ADD R3, R1, R4 | | IF | ID | stall | stall | stall | stall | ALU1 | MEM1 | MEM2 |
| LW R5, (R3)100 | | | IF | stall | stall | stall | stall | ID | stall | stall |
| XOR R6, R5, R7 | | | | | | | | IF | stall | stall |
| SW R6, (R2)50 | | | | | | | | | | |
| ADDI R1, R1, #100 | | | | | | | | | | |
| SUBI R4, R4, #8 | | | | | | | | | | |

| **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| ALU2 | WB | | | | | | | | |
| stall | stall | ALU1 | MEM1 | MEM2 | ALU2 | WB | | | |
| stall | stall | ID | stall | stall | stall | stall | ALU1 | MEM1 | MEM2 |
| | | IF | stall | stall | stall | stall | ID | stall | stall |
| | | | | | | | IF | stall | stall |
| | | | | | | | | | |

| **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| ALU2 | WB | | | | | | | |
| stall | stall | ALU1 | MEM1 | MEM2 | ALU2 | WB | | |
| stall | stall | ID | ALU1 | MEM1 | MEM2 | ALU2 | WB | |
| | | IF | ID | ALU1 | MEM1 | MEM2 | ALU2 | WB |

в)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|

| LW R1, (R2)50 | IF | ID | ALU1 | MEM1 | MEM2* | ALU2 | WB | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ADD R3, R1, R4 | | IF | ID | ALU1 | MEM1 | MEM2 | *ALU2* | WB | | |
| LW R5, (R3)100 | | | IF | ID | stall | stall | stall | *ALU1 | MEM1 | MEM2* |
| XOR R6, R5, R7 | | | | IF | stall | stall | stall | ID | ALU1 | MEM1 |
| SW R6, (R2)50 | | | | | | | | IF | ID | ALU1 |
| ADDI R1, R1, #100 | | | | | | | | | IF | ID |
| SUBI R4, R4, #8 | | | | | | | | | | IF |

| **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| ALU2 | WB | | | | | | |
| MEM2 | *ALU2* | WB | | | | | |
| stall | stall | *MEM1 | MEM2 | ALU2 | WB | | |
| stall | stall | ALU1 | MEM1 | MEM2 | ALU2 | WB | |
| stall | stall | ID | ALU1 | MEM1 | MEM2 | ALU2 | WB |

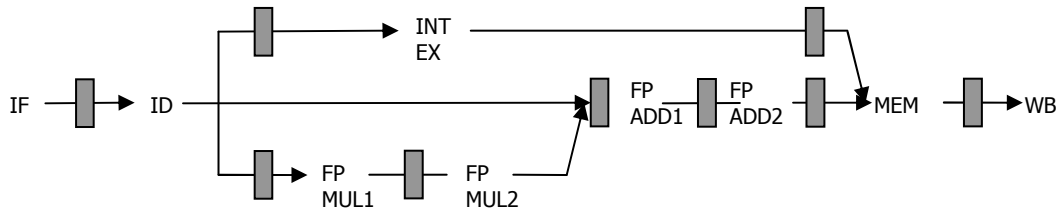
г)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| LW R1, (R2)50 | IF | ID | ALU1 | MEM1 | MEM2* | ALU2 | WB | | | |
| ADD R3, R1, R4 | | IF | ID | ALU1 | MEM1 | MEM2 | *ALU2* | WB | | |
| LW R5, (R3)100 | | | IF | ID | stall | stall | stall | *ALU1 | MEM1 | MEM2* |
| XOR R6, R5, R7 | | | | IF | stall | stall | stall | ID | ALU1 | MEM1 |
| ADDI R1, R1, #100 | | | | | | | | IF | ID | ALU1 |
| SUBI R4, R4, #8 | | | | | | | | | IF | ID |
| SW R6, (R2)50 | | | | | | | | | | IF |

| **11** | **12** | **13** | **14** | **15** | **16** |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| ALU2 | WB | | | | |
| MEM2 | *ALU2* | WB | | | |
| MEM1 | MEM2 | ALU2 | WB | | |
| ALU1 | MEM1 | MEM2 | ALU2 | WB | |
| ID | ALU1 | *MEM1 | MEM2 | ALU2 | WB |

## Задатак 21.

Посматра се систем са проточном обрадом приказаном на следећој слици.




У степен EX се поред постојеће ALU јединице додаје нова FALU јединица која служи за рад са бројевиа у покретном зарезу. Ови бројеви су смештену у регистре F0-F15. Операције FADD, FSUB и FMUL трају 2, 2 и 4 тактова, респективно. Претпоставити да се проточна обрада извршава са прослеђивањем и да се адреса операнда у меморији израчунава у кораку EX. Уколико две инструкције треба да приступе истој јединици предност има она која је прва учитана. Извршава се следећи програмски сегмент

```
LOOP: FLW   F1, (R1)0       /* F1=MEM[R1]       */
```

```
FMUL F0, F0, F1      /* F0=F0*F1   */
FSW  F0, (R1)0       /* MEM[R1]=F0 */
FADD F0, F2, F2      /* F0=F2+F2   */
FADDI F2, F2, #1     /* F2=F2+1    */
SUBI  R1, R1, #2     /* R1=R1-2    */
BNEZ  R1, LOOP       /* Loop if R1≠0 */
FADDI F3, F4, #1     /* F3=F4+1    */
```

Почетна вредност регистра R1 је 100

а) Приказати табеларно шта се дешава у којој фази за сваку инструкцију.

б) Које врсте хазарда постоје у датом програмском сегменту?

в) Колико је тактова потребно да се изврши представљени програмски сегмент?

Решење:

а)

| **Инструкција** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|
| FLW F1, (R1)0 | IF | ID | EX | MEM | WB | | | | |
| FMUL F0, F0, F1 | | IF | ID | stall | MUL1 | MUL2 | ADD1 | ADD2 | MEM |
| FSW F0, (R1)0 | | | IF | stall | ID | EX | stall | stall | stall |
| FADD F0, F2, F2 | | | | | IF | ID | stall | ADD1 | ADD2 |
| FADDI F2, F2, #1 | | | | | | IF | stall | ID | ADD1 |
| SUBI R1, R1, #2 | | | | | | | | IF | ID |
| BNEZ R1, LOOP | | | | | | | | | IF |
| FADDI F3, F4, #1 | | | | | | | | | |
| FLW F1, (R1)0 | | | | | | | | | |
| FMUL F0, F0, F1 | | | | | | | | | |
| FSW F0, (R1)0 | | | | | | | | | |
| FADD F0, F2, F2 | | | | | | | | | |

| **Инструкција** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
|---|---|---|---|---|---|---|---|
| FLW F1, (R1)0 | | | | | | | |
| FMUL F0, F0, F1 | WB | | | | | | |
| FSW F0, (R1)0 | MEM | WB | | | | | |
| FADD F0, F2, F2 | stall | MEM | WB | | | | |
| FADDI F2, F2, #1 | ADD2 | stall | MEM | WB | | | |
| SUBI R1, R1, #2 | EX | stall | stall | MEM | WB | | |
| BNEZ R1, LOOP | stall | ID | idle | idle | idle | | |
| FADDI F3, F4, #1 | | IF | idle | idle | idle | idle | |
| FLW F1, (R1)0 | | | IF | ID | EX | MEM | WB |
| FMUL F0, F0, F1 | | | | IF | ID | stall | MUL1 |
| FSW F0, (R1)0 | | | | | IF | stall | ID |
| FADD F0, F2, F2 | | | | | | | IF |

б)

1 RAW хазард

2 Структурни хазард

3 RAW хазард+Структурни хазард

4 Структурни хазард

5 RAW хазард

6 Структурни хазард

7 Контролни хазард

в)

AverageTimePipelined=Start+N * AverageInstructionTimePipelined

Start=4

## Задатак 22.

Разматра се рачунарски систем у коме се извршавање одређене инструкције одвија у 5 фаза:

| Фаза | IF | ID | EX | MEM | WB |
|---|---|---|---|---|---|
| Време (ns) | 3 | 1 | 2 | 3 | 1 |

а) Колико траје извршавање инструкције помоћу процесора који нема могућност паралелног извршавања инструкција?

б) Колико траје извршавање инструкције помоћу процесора који има стандардну проточну обраду?

в) Ако се извршава програм који се састоји од N поменутих инструкција, колико је трајање програма за системе описане у тачкама а и б?

г) Ако се у оквиру проточне обраде извршавање IF степена смањи на 2 ns, да ли ће се то одразити на укупне перформансе система? А ако се повећа на 4ns?

Решење:

а) AverageInstructionTimeUnpipelined=3+1+2+3+1=10 ns,

б) Time to perform single instruction from stage 1 to stage 5=5 * max{3, 1, 2, 3, 1}=15,

AverageInstructionTimePipelined=max{3, 1, 2, 3, 1}=3 ns

в) AverageTime=NumOfInstructions * AverageInstructionTime

AverageTimeUnpipelined=N * AverageInstructionTimeUnpipelined=10N,

AverageTimePipelined=Start+N * AverageInstructionTimePipelined=15 - 3+N*3 ~ 3N

г) $AverageInstructionTimePipelined_{2ns}$=max{2, 1, 2, 3, 1}=3 ns

$AverageInstructionTimePipelined_{4ns}$=max{4, 1, 2, 3, 1}=4 ns

## Задатак 23.

Разматра се процесор са 6-то степеном проточном обрадом на који ради на фреквенцији од 5 GHz. Претпоставити да је за сваки корак проточне обраде потребан један такт. На овом процесору се извршава 500 инструкција од којих су 40 условни скокови од којих је 50% испуњено.

а) Колико је просечно тактова по инструкцији (CPI) потребно приликом извршавања овог програма?

б) Колико траје извршавање овог програма?

в) Да ли ће се ишта променити уколико се pipeline продужи на 7 степени и уколико се радна фреквенција повећа на 8GHz?

Решење:

Приликом овог решења узети су у разматрање само управљачки хазарди.

n=6

f=5 GHz

IdealCPI=1

а)

NumOfInstructions=500

NumOfConditionalBranch=40

ConditionalBranchTakenFrequency=0.5

BranchPenalty

ClockCycleTime=1/f=0.2 ns

CPI=IdealCPI+PipelineStallClockCyclesPerInstruction

PipelineStallClockCyclesPerInstruction=StructuralStalls+DataStalls+ControlStalls

DataStalls=RAWStalls+WARStalls+WAWStalls

ControlStalls=BranchFrequency * BranchPenalty

BranchFrequency=ConditionalBranchFrequency * ConditionalBranchTakenFrequency

ConditionalBranchFrequency=NumOfConditionalBranch / NumOfInstructions

| **BranchPenalty** | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| CPI | 1.04 | 1.08 | 1.12 | 1.16 | 1.20 | 1.24 |

б)

AverageTime=NumOfInstructions * ClockCycleTime * CPI

| **BranchPenalty** | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| AverageTime[ns] | 104 | 108 | 112 | 116 | 120 | 124 |

в)

n=7

f=8 GHz

ClockCycleTime=1 / f=0.125 ns

AverageInstructionTime=ClockCycleTime * CPI

| **BranchPenalty** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|
| AverageInstructionTimeA[ns] | 0.208 | 0.216 | 0.224 | 0.232 | 0.24 | 0.248 | - |
| AverageInstructionTimeC[ns] | 0.13 | 0.135 | 0.14 | 0.145 | 0.15 | 0.155 | 0.16 |

## Задатак 24.

Разматра се процесор са 6-то степеном проточном обрадом. Времена извршавања појединих фаза инструкције су: 0.3, 0.5, 0.4, 0.4, 0.4 и 0.6 ns, респективно.

а) Колико траје извршавање инструкције помоћу процесора који нема могућност паралелног извршавања инструкција?

б) Колико траје извршавање инструкције помоћу процесора који има проточну обраду?

в) Колико траје извршавање 200 инструкција међу којима нема инструкција скока?

г) Колико траје извршавање 200 инструкција међу којима је 4% безусловних скокова, 16% условних скокова од којих је 40% испуњено уколико се користе следећи приступи: заустављање проточне обраде, претпоставка да ће доћи до скока, претпоставка да неће доћи до скока? У следећој табели је дато колико циклуса се губи у случају појединих претпоставки:

| Branch scheme | Penalty unconditional | Penalty untaken | Penalty taken |
|---|---|---|---|
| Заустављање | 2 | 3 | 3 |
| Прихватање скока | 2 | 3 | 2 |

| | | | |
|---|---|---|---|
| Неприхватање скока | 2 | 0 | 3 |

Решење:

а) AverageInstructionTimeUnpipelined=0.3+0.5+0.4+0.4+0.4+0.6=2.6 ns

б) Time to perform single instruction from stage 1 to stage 6=6 * max{0.3, 0.5, 0.4, 0.4, 0.4, 0.6}=3.6 ns,

AverageInstructionTimePipelined=max{0.3, 0.5, 0.4, 0.4, 0.4, 0.6}=0.6 ns

в) AverageTime=NumOfInstructions * AverageInstructionTime

AverageTimeUnpipelined=N * AverageInstructionTimeUnpipelined=200 * 2.6=520 ns,

AverageTimePipelined=N * AverageInstructionTimePipelined=200 * 0.6=120 ns

г) CPI=IdealCPI+PipelineStallClockCyclesPerInstruction

ControlStalls=UnconditionalBranchFrequency * UnconditionalBranchPenalty+BranchTakenFrequency * BranchTakenPenalty+BranchUnTakenFrequency * BranchUnTakenPenalty

Успорење проточне обраде:

| | Безусловни скокови | Условни испуњени | Условни неиспуњени | Укупно |
|---|---|---|---|---|
| Фреквенција скокова | 4% | 16%*40%=6.4% | 16%*60%=9.6% | 20% |
| Заустављање | 0.04 * 2=0.08 | 0.064 * 3=0.192 | 0.096 * 3=0.288 | 0.08+0.192+0.288=0.56 |
| Прихватање скока | 0.04 * 2=0.08 | 0.064 * 3=0.192 | 0.096 * 2=0.192 | 0.08+0.192+0.192=0.464 |
| Неприхватање скока | 0.04 * 2=0.08 | 0.064 * 0=0 | 0.096 * 3=0.288 | 0.08+0+0.288=0.368 |

## Задатак 25.

Разматра се процесор са 5-то степеном стандардном проточном обрадом на који ради на фреквенцији од 2GHz. Претпоставити да је за сваки корак проточне обраде потребан један такт, при чему се користити јединствена меморија и за приступ инструкцијама (Instruction Memory) и за приступ подацима (Data Memory). Уколико се из више степена покуша приступи меморији предност има инструкција која се дуже налази у проточној обради. У неком програму вероватноћа појављивања инструкција које приступају меморији (LOAD и STORE) је 30%. Да ли ће се ишта променити уколико се pipeline продужи на 6 степени додавањем екстра корака за инструкције које приступају меморији (MEM1 и MEM2) и уколико се радна фреквенција повећа на 3GHz? Напомена: узети у разматрање само структурне хазарде.

Решење:

CPI=IdealCPI+PipelineStallClockCyclesPerInstruction

StructuralStalls=MemoryAccessFrequency * MemoryAccessPenalty

AverageTime=NumOfInstructions * ClockCycleTime * CPI

$MemoryAccessPenalty_{2GHz}=1$

$MemoryAccessPenalty_{3GHz}=2$

$$\frac{AverageTime_{3GHz}}{AverageTime_{2GHz}}=\frac{NumOfInstructions\cdot ClockCycleTime_{3GHz}\cdot CPI_{3GHz}}{NumOfInstructions\cdot ClockCycleTime_{2GHz}\cdot CPI_{2GHz}}=\frac{ClockCycleTime_{3GHz}\cdot CPI_{3GHz}}{ClockCycleTime_{2GHz}\cdot CPI_{2GHz}}$$

$$=\frac{(1+0.3\cdot 2)\cdot 2GHz}{(1+0.3\cdot 1)\cdot 3GHz}=0.82$$

Побољшање перформанси је приближно 20%, а не 50% за колико се подигла радна фреквенција.

## Задатак 26.

Разматра се процесор са 5-то степеном стандардном проточном обрадом на који ради на фреквенцији од 2GHz. Претпоставити да је за сваки корак проточне обраде потребан један такт. У неком програму вероватноћа појављивања LOAD инструкција је 30% и вероватноћа да је податак који довлачи та инструкција потребан следећој 50%. Одредити средње време извршавања инструкција у овом случају. Напомена: узети у разматрање само хазарде података.

Решење:

CPI=IdealCPI+PipelineStallClockCyclesPerInstruction

DataStalls=DataDependencyFrequency * DataDependencyPenalty

AverageInstructionsTime=ClockCycleTime * CPI

AverageInstructionsTime=0.5 * (1+0.3 * 0.5)=0.575 ns

## Задатак 27.

Разматра се процесор са 5-то степеном стандардном проточном обрадом на који ради на фреквенцији од 2GHz. У случају идеалног меморијског систем CPI износи 1.5. Користити се јединствена кеш меморија и за приступ инструкцијама и за приступ подацима. У неком програму вероватноћа појављивања инструкција које приступају меморији (LOAD и STORE) је 30%. У систему постоји 128 KB јединствена кеш меморија са директним пресликавањем и write back техником уписа. Вероватноћа да се податак налази у кеш меморији 0.95. Разматра се меморијски систем са преклопљеним приступима код кога је време одзива меморије 40 тактова, брзина трансфера 4 бајта по такту. Вероватноћа да је кеш линија (блок) модификован износи 40%. Свака линија се састоји из 32 бајта. Не постоји write buffer. Процесор поседује посебан хардвер за убрзавање пресликавања виртуелних у реалне (физичке) адресе (у даљем тексту TLB). TLB потребно 20 тактова уколико се догоди TLB Miss. TLB не успорава кеш меморију уколико дође до проналаска податка у кеш меморији (cache hit). Када се ради о TLB, важе следеће претпоставке: TLB не садржи 0.2% адреси које се пресликавају, без обзира да ли адресе проистичу директно од процесора или их је генерисла кеш меморија као последицу несагласности у кеш меморији (cache Miss).

а) Израчунати CPI уз претпоставку да је TLB идеалан.

б) Израчунати CPI уз претпоставку да је TLB реалан.

в) Какав утиче на перформансе TLB-а уколико су кеш адресе виртуалне или физичке?

Решење:

f=2 GHz

n=5

IdealMemoryCPI=1.5

MemoryAccessFrequency=0.3

CacheSize=128 KB

HitRate=0.95

MemoryDelayCycles=40

MemoryTransferRatePerCycle=W=4 B

Modified=0.4

CachLIneSize=32 B

TLBMissPenalty=20

TLBMissRate=0.002

а)

CPI=IdealMemoryCPI+PipelineStallClockCyclesPerInstruction

PipelineStallClockCyclesPerInstruction=CacheStallClock+StructuralStalls

CacheStallClock=MemoryAccessPerInstruction * MissPenalty * MissRate

MemoryAccessPerInstruction=1+MemoryAccessFrequency=1.3

MissPenalty=(MemoryDelayCycles+CacheLineSize/W) * (1+Modifier)

MissPenalty=(40+32/4) * (1+0.4)=67.2

MissRate=1 - HitRate

CacheStallClock=1.3 * 67.2 * 0.05=4.368

StructuralStalls=MemoryAccessFrequency * MemoryAccessPenalty

MemoryAccessPenalty=1

StructuralStalls=0.3 * 1=0.3

CPI=1.5+4.368+0.3=6.168

б)

$CPI=CPI_a+TLBStalls$

TLBStalls=MemoryAccessPerInstruction*TLBAccessPerMemoryAccess*TLBMissRate*TLBMissPenalty

TLBAccessPerMemoryAccess=MissRate * TLBAccessesPerCacheMiss

TLBAccessPerMemoryAccess=MissRate * (1+Modifier)

TLBAccessPerMemoryAccess=0.05 * (1+0.4)=0.07

TLBStalls=1.3 * 0.07 * 0.002 * 20=0.00364

CPI=6.168+0.00364=6.17164

в)

У случају реалног кеша пре приступа кешу мора, најпре, да се изврши превођење виртуелних у реалне адресе. То би повећало “hit time” и успорило рад процесора.

Што се тиче виртуалне кеш меморије TLB-у се може приступати у паралели са кеш меморијом, постоји опасност од успоравања које је последица несагласности у TLB али постоји сагласност у кеш меморији. Први проблем је последица чињенице да различити корисници имају исте виртуелне адресе које се пресликавају на различите физичке адресе. Други проблем са виртуелним кешом се састоји у томе да различити корисници могу да користе различите виртуелне адресе за исту физичку адресу.