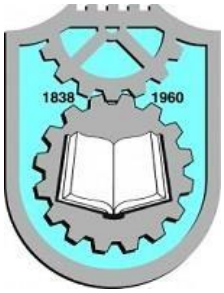
Факултет инжењерских наука
Универзитета у Крагујевцу



Назив студијског програма: Машинско инжењерство
Ниво студија: Основне академске студије
Модул: Информатика у инжењерству
Предмет: Архитектура рачунарских система
Број индекса: 109/2016

Богдан Д. Јоксић

Кеш меморија

Завршни рад

Комисија за преглед и одбрану:

1. Проф. др Јасна Радуловић – ментор
2. _____
3. _____

Датум одбране: _____

Оцена: _____

У оквиру дипломског рада потребно је да кандидат изврши систематизацију података о меморији, кеш меморији, њеним поделама и карактеристикама.Након тога су набројани и објашњени алгоритми замене кеш меморије, технике пресликавања и начин уписа.

Препоручена литература:

[1] Andrew S. Tanenbaum, 2006,*Архитктура и организација рачунара* (превод петог издања)

[2] Душан Регодић, 2018, *Архитектура и организација рачунарских система,* Београд

Крагујевац, 2020.

Ментор:
Проф. др Јасна Радуловић

_____

Резиме

Овај рад је базиран на најновијим подацима из архитектуре рачунарских система на тему кеш меморија. Првенствено је писано уопштено о меморији, а затим се приступило описивању кеш меморије. Анализирана је улога кеш меморије као и њена подела. Описан је принцип рада, затим су описане и објашњене све технике пресликавања појединачно.Затим су описани начини уписа и на крају рада су набројани нивои кеш меморије.

Кључне речи: кеш, меморија, CPU (процесор), брзина протока, капацитет...

Abstract

This paper is based on the latest data of architecture computers system, in cache memory. First, it was written about memory, then cache memory. Afterwards, it's analyzed role of cache memory and it's division. The working principle is described, then it's described and explained all mapping techniques severally. Then it is described method of enrollment and in the end of the paper cache levels are listed.

Keywords: cache, memory, CPU, flow rate, capacity...

Садржај:

Увод

Рачунарски систем је дигитални електронски уређај који служи за аутоматску обраду података. Користи се приликом решавања различитих проблема када постоји потреба да се улазне информације, које представљају полазно стање проблема, обраде по одређеном поступку и генеришу излазне информације, које представљају решење тог проблема. За извођење обраде, неопходни су хардвер (опипљиве, физичке компоненте рачунара) и софтвер (програми које рачунар може да изврши на постојећем хардверу). Рачунар може да обрађује само податке који су представљени у бинарном облику (низовима нула и јединица).

За извршавање програма задужен је хардвер рачунара. Основне хардверске компоненте су:

- **меморија**, која служи за чување података,
- **процесор**, који обавља обраду података извршавањем програмсих инструкција,
- **улазно/излазни уређаји**, који омогућавају унос улазних података у рачунар, као и приказ резултата обраде и
- **магистрала**, која служи за повезивање и размену садржаја између осталих хардверских компонената.

Меморија представља скуп локација (ћелија) у којима се чувају бинарне речи које представљају инструкције и операнде. Ако у једну ћелију меморије може да се смести n битова, то значи да свака бинарна реч представља низ од n нула и јединица. Посматрањем овог низа, не може се закључити да ли он представља инструкцију или податак. Значење бинарне речи зависи од тога како је рачунар интерпретира, тј. тумачи [1].

1. Меморија

У рачунарству, меморија се односи на физичке уређаје који се користе за складиштење програма (низови инструкција) или података (нпр. информације о стању програма) привремено или трајно за употребу у рачунару или другом дигиталном електронском уређају. Термин "примарна меморија" се користи за информацију у физичким системима који раде на високим брзинама (нпр. RAM), као разлика од секундарне меморије, који су физички уређаји за складиштење програма и података који имају спорији приступ, али имају већи меморијски капацитет. Примарна меморија ускладиштена на секундарној меморији се зове „виртуелна меморија".

Термин „меморија", са значењем примарна меморија је често повезана са адресабилном полупроводничком меморијом, тачније интегрисаним колима који се састоје из транзистора на бази силикона. Они се користе као примарна меморија, али имају и друге сврхе у рачунарима и другим електронским уређајима. Постоје две главне врсте полупроводничке меморије: постојане и непостојане. Примери постојане меморије су флеш меморија (некад се користи као секундарна а некад као примарна рачунарска меморија) и ROM/PROM/EPROM/EEPROM меморија. Пример за непостојане меморије су примарне меморије (углавном DRAM), и брза кеш меморија централне процесорске јединице (углавном статички RAM односно SRAM, који је брз али користи пуно енергије и има мањи меморијски капацитет по јединици површине од DRAM меморије).

Већина полупроводничке меморије је организована у меморијске ћелије или бистабилне флип-флопове, од којих сваки садржи један бит (0 или 1). Организација флеш меморије садржи и један бит по меморијској ћелији и више битова по ћелији (ова врста се зове ћелија на више нивоа, скраћено MLC). Меморијске ћелије су груписане у речи фиксне дужине, на пример 1, 2, 4, 8, 16, 32, 64 или 128 бита. Свакој речи се може приступити бинарним адресама од N бита, што омогућава складиштење 2 на N речи у меморији. Следи да се процесорски регистри обично не сматрају за меморију, јер они складиште по једну реч и не садрже механизам адресирања. Термин „сторић" се често користи као назив за секундарну меморију типа трака, магнетних дискова и оптичких дискова (CD-ROM и DVD-ROM) [1].

1.1. Историја меморије

Током раних 1940-их, меморијска технологија је углавном имала капацитет од неколико бајтова. Први електрични програмабилни дигитални рачунар, ENIAC, је могао да изводи просте рачунице које би се састајале из 20-децималних цифара који су држани у вакуумским цевима.

Следећи битан напредак у рачунарској меморији је дошао са меморијом акустичне линије за кашњење, коју је развио J. Преспер Екерт раних 1940-их. Кроз стаклену цев испуњену живом са кварцним кристалом на сваком крају, линије за кашњење су могле да складиште битове које би се задржавале унутар цеви и пролазиле кроз живу у виду звучних таласа. Меморија акустичне линије за кашњење би морала да се ограничи на неколико стотина хиљада битова како би остала ефикасна.

Алтернативе линији са закашњењем, Вилијамсова цев и Селекциона цев, измишљене су 1946. године и обе су користиле зраке електрона у стакленим цевима као начин складиштења. Фред Вилијамс је изумео Вилијамсову цев која је прва РАМ меморија. Фредова цев је већег капацитета од Селекционе цеви (ова је била ограничена на 256 бита, док је Вилијамсова могла да ускладишти хиљаде) и била је јефтинија. Вилијамсова цев је, додуше, била веома осетљива на дешавања у окружењу.

1940-их су научници покушавали да пронађу постојану меморију. Џеј Форестер, Џен А. Рајхмен и Ен Венг су развили меморију са магнетним језгром, која је могла да се позове и након губитка струје. Меморија са магнетним језгром је био доминантан облик меморије до појаве транзисторске меморије крајем 1960-их.

Развоји у технологији и економији омогућили су појаву Рачунара са веома великом меморијом (VLM). Термин меморија се генерално односи на RAM када се говори о рачунарима [1].

Историјски преглед [2]:

- 1944. – направљен је први рачунар са унутрашњим програмом EDVAC.
- 1952. – појављују се магнетни добоши.
- 1953. – Џеј Форестер са својим тимом уграђује меморију од магнетних језгара у рачунар *Wirlwind* и двоструко убрзавају време приступа на 6 микро секунди.
- 1957. – Представљен је први хард диск као део IBM-ов RAMAC-а.
- 1967. – IBM прави први дискетни уређај.
- 1969. – *Intel* представља 1 килобајт RAM чип, који има значајно већи капацитет од претходно произведених меморијских чипова.
- 1972. – Појављује се први персонални рачунар са 256 бајтова меморије, проширive до 64 килобајта, такозвани *Altair*.
- 1972. – Појављује се компакт диск (CD).
- 1983. – Филипс и Сони развијају CD-ROM као проширење аудио CD технологије.
- 1984. – Појављује се меморија са директним приступом, односно РАМ меморија.
- 1987. – Појављују се хард дискови димензија 3,5 инча.
- 1994. – Представљени су први ZIP уређаји и ZIP дискови, заменљиве дискове величине дискете капацитета 25 и 100 мегабајта.
- 1998. – IBM представља хард диск од 25 гигабајта. Први диск који се појавио 1956. године имао је капацитет 5 мегабајта.
- 1998. – Појављује се DVD- RAM уређај који има капацитет од 5,2 гигабајта на двостраним преписивим касетама.
- 1999. – IMB представља *Microdrive*, најмањи и најлакши могући хард диск на свету који изазива револуцију у индустрији преносивих уређаја (дигитални фотоапарати, преносиви рачунари и MP3 плејери).
- 2003. – Појављује се серијски ATA хард диск и постиже успех као најбржи за персоналне рачунаре.
- 2007. – Представљен је први хард диск величине 1 терабајт.
- 2010. – Представљен је први хард диск величине 3 терабајта.

1.2. Параметри меморије

Параметри меморије се користе за опис карактеристика меморијског система:

- **Капацитет меморије** – број бајтова или битова који се могу запамтити у меморији. Главне меморије персоналних рачунара су капацитета неколико стотина MB. Капацитет хард дискова је реда неколико стотина GB, па чак и TB.
- **Време приступа (кашњење)** – временски интервал који протекне од довођења сигнала за дефинисање приступа до завршетка уписа или читања, односно време потребно да се одређени захтев комплетира.
  Код меморија са непосредним приступом ово време је константно и не зависи од места локације којој се приступа. Време приступа у овим меморијама мање је од 10 ns.
  Код дискова и магнетних трака време приступа је средње време потребно за приступ неком месту на медијуму ради читања или уписа, а састоји се од времена тражења потребног за постављање уписно/читајуће главе изнад ћелије са подацима и времена преноса података. Средње време приступа подацима на диску је реда 5-15 ms. Време приступа код меморија са секвенцијалним приступом (магнетних трака) може износити и неколико минута.
- **Циклус приступа (меморијски циклус)** – минимални дозвољени временски интервал између два узастопна приступа меморији. Не може бити краћи од времена приступа, а обично је нешто дужи. Дефинише се и наводи само за меморије са непосредним приступом.
  Број задовољених захтева у јединици времена назива се пропусни опсег. Максимална пропусна моћ меморијског система дата је бројем меморијских модула подељених са временом циклуса меморије.
- **Јединица преноса** – број битова који се истовремено чита или уписује. За оперативну меморију то је једна меморијска реч. Ако јединица преноса садржи више сукцесивних меморијских речи назива се блок. За спољне меморије јединица преноса је сектор или блок.
- **Брзина преноса података** – број битова, бајтова или меморијских речи које уређај може пренети у једној секунди после постављања уписно/читајуће главе на почетак блока или сегмента података. Назива се и пропусност меморије.

1.3. Начин приступа меморији

Према начину приступа меморији, разликују се:

1) Меморије са непосредним (произвољним) приступом – код меморија са непосредним приступом време приступа подацима не зависи од места у меморији са кога се чита или у које се врши упис. Непосредни приступ обично се реализује помоћу бистабилних кола и прекидачких мрежа. Број битова који се истовремено уписује или чита назива се меморијска реч.
2) Меморије са директним приступом – код меморија са директним приступом, као што су нпр. дискови и дискете, носилац података (диск) стално ротира тако да се приступ неком делу носиоца ради читања или уписа циклички понавља. Средње време приступа овим носиоцима је 5 -15 ms.
3) Меморије са секвенцијалним приступом – код меморија са секвенцијалним приступом, као што је нпр. магнетна трака, врши се сукцесивна (редна) провера делова носиоца података док тражени део не дође у положај да му се приступи. Време приступа овде може бити и неколико минута.

1.4. Хијерархија меморије

Класично решење за складиштење велике количине података јесте хијерархијско организовање меморије, као на слици 1. На врху се налазе регистри процесора, којима процесор може приступати пуном брзином. Испод њих је кеш меморија, чија се величина тренутно креће од 32 KB до неколико мегабајта. Следећа је главна меморија, од 16 MB за најједноставније система, па до десетина гигабајта за најсавршеније. После ње су на делу магнетни дискови, данашњи главни медијум за трајно складиштење података. На крају се налазе магнетне траке и оптички дискови за потребе архивирања података [2].

Слика 1. *Петостепена хијерархија меморије*

Укупно расположива меморија може се разматрати као хијерархијски систем компонената који се састоји од свих уређаја за чување података (меморијских уређаја) које користи рачунарски систем. Овај систем садржи све меморије, од спорих, са великим капацитетом, до релативно мале, али брзе основне меморије и још мање и врло брзе кеш меморије. Због тога се меморија рачунара организује у хијерархијску структуру меморијских уређаја, који на појединим нивоима имају различите брзине и капацитет.

Како се крећемо низ хијерархију, расту вредности три кључна параметра. Прво, повећава се време приступања меморији. Регистрима процесора може се приступити за неколико наносекунди, кеш меморији за само неколико пута више времена, док приступање главној меморији по правилу траје неколико десетина наносекунди. Сада настаје велики скок, јер је време приступању диска најмање 10 ms, а приступ траци или оптичком диску може се мерити секундама уколико медијум треба однекле преузети и ставити на кључ.

Друго, како идемо низ хијерархију, тако расте и капацитет складиштења. Регистри процесора су добри за чување до 128 бајтова, кеш меморије за неколико мегабајта, а магнетни дискови за неколико десетине гигабајта. Траке и оптички дискови обично се чувају посебно, па само корисникова финансијска ситуација ограничава њихов капацитет.

Треће, број битова меморије које можете купити за долар, расте низ хијерархију. Иако се конкретне цене брзо мењају, оне се за главну меморију мере у доларима по мегабајту, за магнетне дискове у центима по мегабајту, а за магнетне траке у доларима по гигабајту.

1.5. Примарна меморија

Примарна меморија је меморија која је директно доступна процесору рачунара преко његових главних магистрала (адресних и за податке), без потребе за алтернативним улазно-излазним и често значајно споријим начинима комуникације.

Примарна меморија се користи за смештај података и инструкција које рачунар у том тренутку извршава, односно као радна меморија рачунара (сем у случају када се у њен садржај не може мењати - разне ROM меморије) [3].

Насупрот секундарној меморији, примарна меморија има следеће особине:

- Значајно је бржа (често хиљадама пута). Као пример, почетком 21. века, било ком делу примарне је могуће приступити у просеку за свега неколико наносекунди. Насупрот томе, тврди диск, као најпознатија секундарна меморија, мора прво да механички позиционира главу за читање до цилиндра са подацима, што у просеку траје око 9 ms а у најгорем случају и до 20 ms. Затим мора да сачека да се диск окрене тако да подаци „дођу под главу“, што у случају 10.000 обртаја у минути траје у просеку додатних 3 ms. На крају, потребно је саме податке прочитати (трајање значајно зависи од густине података) и послати у примарну меморију. Како тврди дискови не могу да раде са блоковима мањим од једног сектора (типично 512 бајтова) то је потребно пребацити све њих у примарну меморију пре него

што је могуће употребити било који, што је само по себи толико пута спорије у односу да директан приступ примарној меморији. Збирно, приступ било ком податку на тврдом диску у просеку траје преко 12 ms.

- Директно је адресабилна (RAM). Пример: процесор може у било ком тренутну да зада адресу било ког јединичног податка ком жели да приступи и да изврши операцију читања или уписа. Насупрот томе, тврди дискови не омогућавају рад са мање од једног сектора података (типично 512 бајтова) - свако ишчитавање захтева читање целог сектора, а сваки упис захтева поновно снимање целог сектора, независно од количине промена.
- Губи садржај по престанку напајања (сем у случају ROM меморија). Пример: По искључивању рачунара сео садржај радне меморије се губи. Насупрот томе, подаци снимљени на тврди диск, CD-R и сл. остају и могуће им је приступити по поновном укључивању.

Виртуелна меморија и утицај на ефективну брзину рачунара

Оперативни системи са подршком за виртуелну меморију могу да симулирају повећање примарне меморије тако што тренутно ређе коришћене делове примарне меморије пребацују у део секундарне, а део који је следећи потребан „врате“ из секундарне назад у примарну. На овај начин физичка примарна меморија може бити мања од оне која је реално потребна извршаваним програмима, али по цени брзине рада рачунара - што је разлика између потребне и реалне/физичке примарне меморије већа, то ће чешће бити потребне поменуте замене делова и, тако, комуникација са хиљадама пута споријом секундарном меморијом.

Одатле долази и појава да повећање примарне меморије рачунара не утиче само на ограничење количине података и програма који се могу обрађивати у исто време него и на повећање ефективне брзине рачунара (јер се мање времена троши на спору комуникацију са секундарном меморијом).

1.6. Секундарна меморија

Без обзира на то колика је, меморије никад није довољно. Корисници увек желе да сачувају више информација него што она може да прихвати, нарочито зато што под утицајем нових технологија мисле да у меморију могу сместити све оно о чему су некада само маштали. За око 50 милиона књига, сваке са 1 MB текста и 1 MB компримованих слика, потребно је ускладити 100 терабајтова. На такав начин се размишља и о складиштењу свих досад направљених 50000 филмова. Оваква количина информација неће моћи да стане у главну меморију, барем не још неколико деценија.

Секундарна меморија је меморија која није директно доступна процесору рачунара преко његових главних магистрала (адресних и за податке). Да би се дошло до садржаја секундарне меморије он прво мора да се (привремено) пребаци у примарну (радну) меморију, а по завршетку обраде (уколико је то потребно) „врати“, односно адекватно ажурира одговрајући садржај секундарне меморије.

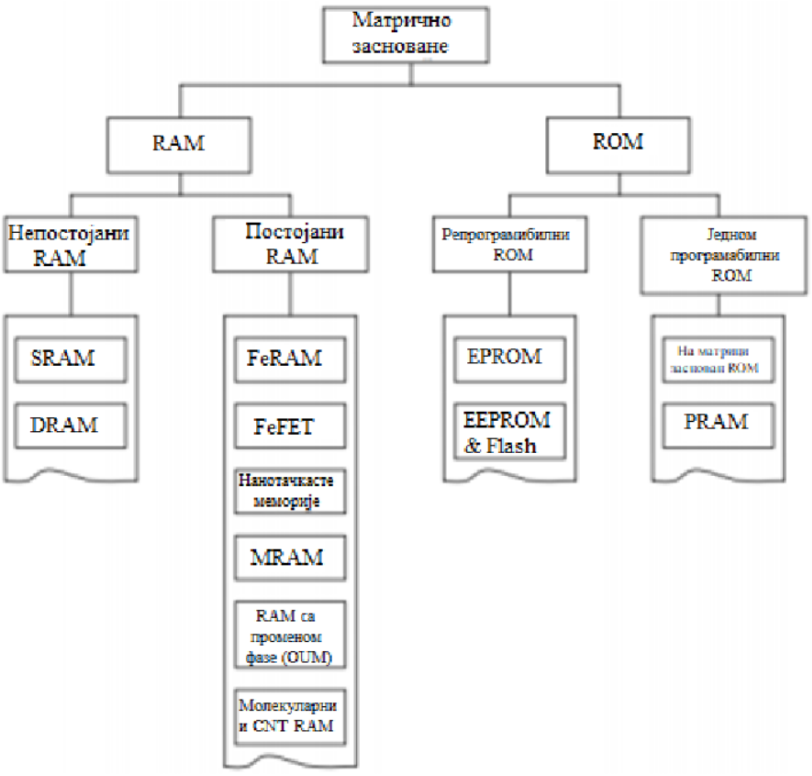
За присуп секундарној меморији потребни су додатни системи и често мноштво процесорских инструкција ради контроле тих додатних система. Због тога је приступ секундарној меморији и далеко компликованији и спорији него примарној меморији [3].

2. Подела меморије

Меморије се могу поделити по различитим критеријумима, један од њих је физичко својство материјала који се користи за чување података. Користе се следећи физички принципи чувања података: електронски, магнетни, оптички и др.

Меморије се често класификују и према начину организације приступа запамћеним подацима. Тако, постоје меморије са непосредним приступом, меморије са директним приступом и меморије са секвенцијалним приступом. Код меморија са непосредним (произвољним) приступом (енгл. RAM) на основу адресе може се приступити било којој локацији. Посебну категорију ових меморија чине читачке (фиксне) меморије (енгл. ROM). У овим меморијама подаци су константни на меморијском медијуму. Код меморија са директним приступом и меморија са секвенцијалним приступом меморијски медијум се на одређени начин креће у односу на систем за читање и упис. Овде спадају меморије са магнетним тракама, магнетним дисковима, дискетама и меморије са оптичким дисковима.

Још један типичан пример класификације меморија је принцип размештаја и тражења података. Ту се разликују адресне меморије и безадресне меморије. У адресне меморије се убрајају меморије са непосредним приступом. Од безадресних меморија најпознатије су асоцијативне меморије код којих се подаци траже на основу њиховог садржаја и стекмеморије код којих се подацима приступа преко врха стека. На слици 2 приказана је подела меморије [4].



Слика 2. *Подела меморије*

Према могућности задржавања података по престанку (искључењу) напајања, разликују се постојане меморије и непостојане меморије. Код постојаних меморија подаци се при нестанку напајања задржавају, што је случај код магнетних дискова, дискета и магнетних трака. Код непостојаних меморија по престанку напајања садржај се неповратно губи [4].

2.1. ROM

ROM (R/O - *Read Only*) се користи за чување програма и података који су потребни за покретање рачунара при укључивању. ROM је тип меморије који се може само читати, за разлику од RAM-а у који се може и уписивати садржај. Подаци уписани у ROM-у су увек тамо, без обзира да ли је прикључено напајање или не. ROM може бити извађен из рачунара неодређено дуг временски период и онда поново враћен, и подаци ће још увек бити у њему. ROM спада у класу постојаних меморија.

Чињеница да се ROM не може (лако) модификовати обезбеђује одређени ниво сигурности против случајних промена његовог садржај. ROM се најчешће користи за чување системских програма за које желимо да буду расположиви рачунару у свако доба. Пример за то је системски БИОС програм, који се чува у посебном ROM-у на матичној плочи - ово значи да је програм доступан када се укључи напајање и да PC рачунар може да га искористи да подигне систем (по укључењу рачунара системска меморија је празна, па мора да постоји нешто што би рачунар могао да користи по укључењу) [2].

Пошто ROM не може бити модификован, самим тим је погодан за складиштење података који не требају бити промењени током живота уређаја. У том циљу, ROM је коришћен да у многим рачунарима складишти *look-up* табелу за рачунање математичких и логичких функција (нпр, уређај са покретним зарезом може брже да рачуна синусну функцију). Наиме, графичке картице првих рачунара чувале су табелу битмапираних фонт карактера у ROM-у. Ово значи да фонт текста није могао бити мењан интерактивно [5].

ROM меморија је, у суштини, претварач кода са н улаза и м излаза. Сваки ROM се састоји из два дела: декодерског и кодерског. Декодер служи за одређивање адресе и назива се адресни декодер. Кодерски део мреже у ствари представља фиксну меморију. ROM меморија садржи три скупа сигнала: адресни (декодерски), управљачки и улаз за податке (кодерски). Управљачки улаз омогућава селектовање IC-а. Ово је практично улаз који омогућава или онемогућава појаву података на излазу ROM-а.

2.1.1. Начин функционисања ROM-а

Слично RAM-у, ROM чипови праве мрежу колона и редова. ROM користи диоду за повезивање линија ако је вредност 1. Ако је вредност 0, онда линије нису конектоване. Диода обично дозвољава струји да тече само у једном смеру и има одређени праг, који одређује колико струје је потребно пре него што ће диода да је пропусти. У елементима као што су процесори и меморијски чипови, напон је приближно 0,6 волти.

Ако је диода присутна у тој ћелији, пуњење ће бити спроведено до краја и, у бинарном систему, ћелија ће бити прочитана као "укључена" (вредност 1). Уређени део ROM-а је да ако је вредност ћелије 0, на тој раскрсници нема диоде како би се повезале колона и ред, тако да се сигнал на колони не преноси на ред. Као што се може видети, начин рада ROM чипа захтева програмирање савршених и комплетних података када се чип креира. Не може се репрогамирати или преправити стандардни ROM чип. Стварање првобитног шаблона за чип ROM-а често је тежак процес препун грешака. Али предности ROM чипова превазилазе недостатке. Када се шаблон заврши, стварни чипови могу коштати свега неколико центи. Они користе веома мало напајања, изузетно су поуздани и, у случају већине малих електронских уређаја, садрже све неопходне програме за управљање уређајем.

2.1.2. Врсте ROM меморије

Временом, развијањем технологија јавила се потреба за побољшаним карактеристикама меморије. Коришћење ROM меморије за складиштење тако малог броја података готово је нестала са појавом модерних рачунара опште намене. Међутим Флеш ROM је преузео нову улогу као медиј за масовно складиштење или секундарно складиштење фајлова.

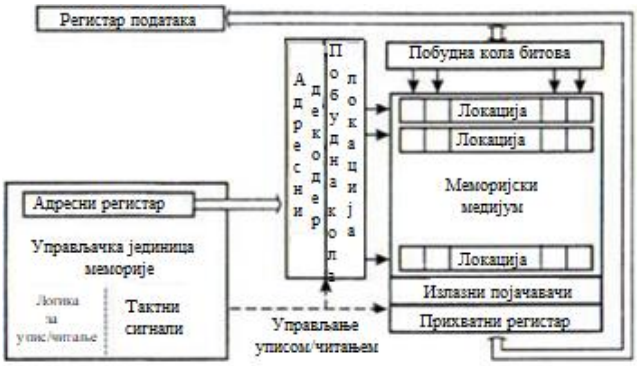
Надограђене верзије ROM-а су:

- Једномпрограмибилни ROM – MASK ROM,
- PROM,
- Репрограмибилни ROM – EPROM,
- EEPROM и Флеш.

2.2. RAM

Оперативна меморија, главна меморија или меморија са непосредним приступом (engl. RAM – меморија са произвољним приступом) јесте адресивна меморија која служи за чување података који се користе у процесу извршења машинских операција у аритметичко-логичкој јединици и управљачкој јединици процесора. То су полазни подаци за обраду, међурезултати и коначни резултати обраде и инструкције програма који се извршава. Карактеристике оперативне меморије непосредно утичу на особине самог рачунарског система, пре свега на његову брзину.

Типична структура оперативне меморије приказана је на слици 3. Оперативна меморија састоји се од меморијских локација реализованих бистабилних кола и одговарајућих прекидачких мрежа које обезбеђују упис података у меморију и читање података из меморије. Како се на основу адресе може приступити било којој локацији, користи се термин меморија са непосредним приступом или меморија са произвољним приступом.



Слика 3. *Структура оперативне меморије*

У меморији се бинарни подаци чувају у облику група битова које се називају меморијске речи. Свака реч чува се у посебној локацији а чита се или уписује као целина. Свака реч може представљати бројчани податак, алфанумерички податак, код инструкције или било који други бинарни код.

Веза са оперативном меморијом остварује се преко појединих сигнала на управљачким линијама, адресним линијама и линијама за улаз и излаз података. Управљачки сигнали одређују смер преноса (упис или читање) а сигнали на адресној линији одређују меморијску локацију којој се приступа.

Управљачки и синхронизациони сигнали за извршење захтеваних операција обезбеђују унутрашња управљачка кола меморије која су саставни део управљачке јединице оперативне меморије.

За реализацију оперативне меморије данас се готово искључиво користе као меморијски елементи полупроводничка бистабилна кола. Постоје статичке и динамичке полупроводничке меморије са непосредним приступом.

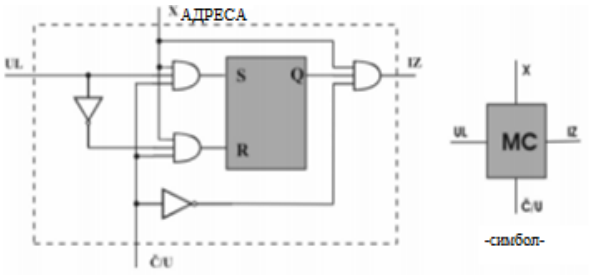
Статичка меморија као меморијске елементе користи полупроводничка бистабилна кола. Добре одлике ових кола су велика брзина уписа и читања и дозвољена одступања параметара компонената. Недостатак је мали капацитет пошто троше више енергије него динамичке, због чега је мања густина на силицијумској плочици интегрисаног кола.

Основна меморијска ћелија у статичкој RAM матрици је флип флоп, чији је садржај статичан, тј. величина сигнала (напона) у ћелији се не мења у току времена, па није потребно да се обнавља (задржава податке без спољашњег "освежавања" докле год је прикључено напајање, по чему се разликује од динамичког RAM-а који се мора "освежавати" пуно пута у току сваке секунде да се подаци не би изгубили) [4].

2.2.1. Статичка RAM меморија

На слици 4 је приказана једна ћелија биполарне статичке RAM меморије, која има једноставан улаз ради уписа, и једноставан излаз ради очитавања меморисаног податка. Овом меморијском колу може се приступити само када је побуђен његов прикључак X за адресирање. Приступ улазу и излазу је преко истог прикључка ČU, што значи, да се овим

прикључком обавља и читање и упис. Упис информације са улазу UL врши се када је X=1 и ČU=1. Читање уписаног садржаја на излазу врши се када су X=1 и Č/U=0. Инвертор у ћелији омогућава упис податка без претходног брисања садржаја SR леч кола. Овде се уместо приказаног SR леч кола може употребити и D флип флоп или било који други. На истој слици је приказан и симбол једне ћелије SRAM меморије, за линеарно адресирање са адресним улазом X.



Слика 4. *Ћелија биполарне SRAM memorije*

Статичке RAM меморије се реализују и у MOSFET технологији, или у CMOS технологији. Према томе, принцип рада ових меморија је исти као и биполарних, али су технологије израде, а тиме и карактеристике другачије. Биполарне меморије су брже али знатно мањег капацитета, а најбрже се израђују у ECL техници где време приступа може бити и мање од 10 ns. Статичке меморије су мањег капацитета по чипу, а користе се у системима где се захтева већа брзина приступа меморији и мања потрошња струје из извора за напајање. Такође је вероватноћа грешке код статичких меморија мања него код динамичких, тако да се користе у системима где се захтева висока поузданост.

У статичком RAM-у, облик флип-флопа држи сваки бит меморије. Флип-флоп за меморијску ћелију садржи четири или шест транзистора заједно са неким ожичењем, које не треба изнова обнављати. То чини статичку RAM меморију знатно бржу од динамичке RAM меморије. Међутим, пошто има више делова, статичка меморијска ћелија заузима много више простора на чипу од динамичке меморијске ћелије. Због тога се добија мања меморија по чипу, а то чини статичку RAM меморију много скупљом.

Меморијски чипови на десктоп рачунарима првобитно су користили пин конфигурацију под називом *dual inline package* (DIP). Ова конфигурација пинова може се спајати у рупице на матичној плочи рачунара или прикључити у утичницу која је лемљена на матичној плочи. Овај метод је добро функционисао када су рачунари радили са неколико мегабајта или мање RAM-а, али како је повећана потреба за меморијом, тако је бројним чиповима потребан већи простор на матичној плочи.

Решење је било постављање меморијских чипова заједно са свим компонентама подршке на посебну штампану плочу која би могла да буде прикључена на посебан конектор (меморијска банка) на матичној плочи. Већина ових чипова користи *small outlineJ-lead* (SOJ) пин конфигурацију, али већина произвођача користи конфигурацију *thin small outline package* (TSOP). Кључна разлика између ових новијих типова пинова и оригиналне DIP конфигурације јесте то што су SOJ и TSOP чипови монтирани на површину плоче. Другим речима, клинови се леме директно на површину плоче, а не убацују у рупе или утичнице.

Меморијски чипови су обично доступни само као део картице која се зове модул. Вероватно сте видели да је меморија наведена као 8x32 или 4x16. Ови бројеви представљају број чипова помножених капацитетом сваког појединачног чипа, који се мери у мегабитима (Mb) или милион битова. Узмите резултат и поделите га на осам да бисте добили број мегабајта на том модулу. На пример, 4x32 значи да модул има четири 32-мегабитна чипа. Множењем 4 са 32 добијају се 128 мегабита. Пошто бајт има 8 бита, дељењем се добија резултат од 16 мегабита.

2.2.2. Динамичка DRAM меморија

За разлику од SRAM-а, меморијски елементи код динамичког RAM-а су направљени од минијатурних кондензатора, који временом губе своје наелектрисање, па је неопходно повремено обнављање уписаних података. То обнављање се обавља сваких неколико милисекунди, а тај процес се назива освежавањем.

DRAM ради тако што шаље напон како би активирао транзисторе на сваком биту у колони. Када се пише, линије редова садрже стање које кондензатор треба да преузме. При читању, појачало одређује ниво напона у кондензатору. Ако је више од 50 процената, то чита као 1; у супротном чита као 0. Бројач прати редослед освежавања на основу којих редова је приступ у којем редоследу. Трајање времена потребно за све то је толико кратко да се изражава у наносекундама (милијарди секунди). Капацтет меморијског чипа је 70 nc што значи да је потребно 70 наносекунди да потпуно очита и напуни сваку ћелију. Саме меморијске ћелије би биле бескорисне без начина за добијање информација у њима и изван њих. Дакле, меморијске ћелије имају целокупну подршку других специјализованих кругова. Ови кругови извршавају функције као што су:

- Идентификовање сваког реда и колоне (одабир адреса редова и одабир адресе колоне);
- Одржавање праћења секвенце освежавања (бројач);
- Читање и враћање сигнала из ћелије (сензор појачала);
- Наређује ћелији да ли треба да прими напон или не (дозвола писања).

Врсте DRAM меморије су:

- FPM,
- *Extended data out* DRAM (EDO DRAM),
- *Burst* EDO DRAM (BEDO DRAM),
- *Synchronous dynamic* RAM (SDRAM),
- *Single data rate synchronous* DRAM (SDR SDRAM),
- *Double data rate synchronous* DRAM (DDR SDRAM),
- *Double data rate* SDRAM (DDR),
    - DDR2 SDRAM,
    - DDR3 SDRAM и DDR4 SDRAM,
    - DDR3L SDRAM,

- *Direct Rambus* DRAM (DRDRAM) и
- *Reduced Latency* DRAM (RLDRAM).

Данас се највише користе Double data rate DDR SDRAM.

2.2.3. Разлика између динамичке и статичке RAM меморије

DRAM је наследник SRAM-а. Дизајнери меморије смањили су број елемената по биту и елиминисали диференцијалне битне линије како би се сачувала област чипа за креирање DRAM-а. Као резултат, DRAM је јефтинији за производњу него SRAM. Али SRAM задржава неке предности у односу на DRAM. SRAM не треба освежавати јер ради на принципу пребацивања струјног тока у један од два правца, притом не држећи пуњач у оквиру ћелије за складиштење. SRAM се генерално користи за кеш меморију, која се може приступити брже од DRAM-а. SRAM је способан за читање и писање бајтова и бржи је у читању и писању од DRAM-а. DRAM пише податке на нивоу бајтова и чита на нивоу вишеструког бита. Разлике у напајању се разликују у зависности од тога да ли је систем активан или у режиму спавања. DRAM захтева мање снаге него SRAM у активном стању, али SRAM троши знатно мање снаге него што DRAM ради док је у режиму спавања. Када би поредили SRAM и DRAM, закључујемо да је израда DRAM-а једноставнија, због тога што се DRAM израђује од једног кондензатора и једног транзистора по биту, за разлику од SRAM-а за који су потребна од 4 до 6 транзистора. Самим тим је цена DRAM-а мања [2].

2.2.4. Разлика између RAM и ROM

Разлика између RAM и ROM приказана је у табели 1.

Табела 1. *Разлика између RAM и ROM*

	PAM	POM
Постојаност	Брише меморију одмах након гашења рачунара	Чува податке и након гашења рачунара
Приступ	Директно се приступа преко процесора	Не приступа се преко процеса директно
Складиштење	Привремено чува информације на одређено време	Чува информације неограничено
Структура хардвера	У облику чипа	Оптички драјвери од магнетних трака
Цена	Скупљи од POM-а	Јефтинији од PAM-а
Величина	Величина чипа већа од POM-овог	Мања од PAM-а
Брзина писања	Брзо се уписују подаци	Споро се уписују подаци
Капацитет	16GB по чипу и више	До 4MB по чипу и више

3. Кеш меморија

Микропроцесори су увек били бржи од меморија. Како се меморија унапређивала, побољшавали су се и процесори. У ствари, пошто су пројектанти процесора имали могућност да поставе све више електричних кола на чип, искористили су је за паралелну обраду података и суперскаларни рад, због чега су процесори постали још бржи.

Пројектанти меморија су нове технологије обично користили за повећавање капацитета, а не брзине, тако да је проблем брзине меморије с временом постајао све већи. Ова несразмера у брзини практично значи да ће процесор издати захтев за рад с меморијом и чекати више циклуса радног такта да би добио одговор. Што је спорија меморија, процесор ће морати дуже да чека.

Као што смо рекли, постоје два начина за решавање овог проблема. Најједноставније је да се започне читање садржаја меморије када се наиђе на одговарајућу инструкцију и да се настави извршавање програма, али да се процесор заустави уколико нека инструкција покуша да употреби меморијску реч пре него што се она учита. Што је меморија спорија, такве ситуације ће бити чешће а штете веће. На пример, ако меморија касни 10 циклуса, веома је вероватно да ће једна од следећих инструкција покушати да употреби реч која се учитава.

Друго решење је да се уместо рачунара, који ће се у таквој ситуацији зауставити, употреби преводилац који неће генерисати код док потребне речи не стигну из меморије. Само, то је много лакше рећи него учинити. После инструкције за учитавање по правилу нема шта друго да се ради, па је преводилац принуђен да умеће инструкције NOP, које само заузимају место у коду и троше време. Овакав приступ, у ствари, уместо заустављања хардвера проузрокује заустављање софтвера, али се то на исти начин одражава на перформансе.

Проблем заправно није технолошке, већ економске природе. Инжењери знају да направе меморије које су брзе као процесори, али да би радиле пуном брзином, оне се морају сместити на процесорски чип (пошто је путовање магистралом до меморије врло дуго). Када се на процесорски чип стави меморија, не постаје већи и скупљи. Чак и да цена не представља проблем, постоји ограничење величине процесорског чипа. Према томе, опције су мала количина брзе меморије или велика количина споре меморије. А ми бисмо највише желели велику количину брзе меморије по ниској цени.

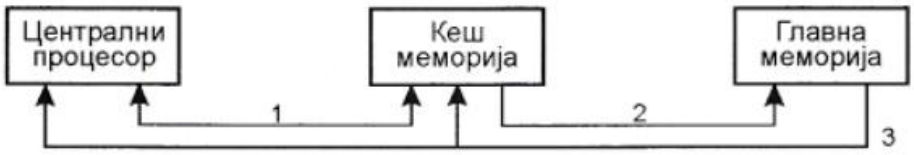
Занимљиво је да постоје технике за комбиновање мале количине брзе меморије с великом количином споре меморије да би се добила (скоро) брзина брзе меморије и капацитет велике меморије, а све по умереној цени. Мала, брза меморија зове се кеш (енгл. cache, од француског cacher, што значи сакрити). У наставку ћемо укратко објаснити како се кеш користи и како ради.

Кеш је начињен да би се у њему чувале најчешће коришћене меморијске речи. Када процесору затреба реч, он је најпре тражи у кешу. Процесор приступа главној меморији само ако тражену реч не нађе у кешу. Ако се у кешу налази битан део најчешће коришћених речи, просечно време приступања знатно се скраћује.

Успешност проналажења речи у кешу, дакле, зависи од тога колико се меморијских речи у њему налази. Већ годинама се зна да програми не приступају меморији потпуно насумично.

Ако се већ приступало меморијској адреси A, постоји вероватноћа да ће се и следећи приступ меморији бити у околини адресе A. Најједноставнији пример пружа сам програм. Осим у случајевима гранања и приликом позивања процедура, инструкције се преузимају за узастопних меморијских локација. Штавише, програм већи део времена током извршавања проводи у петљама, у којима се ограничен број инструкција стално изнова извршава. Слично томе, програм за рад с матрицама вероватно ће више пута приступати истој матрици пре него што пређе у други задатак.

Запажање да програм у неком произвољном, кратком временском интервалу обично приступа блиским меморијским зонама зове се приступ локалности. То је основа за све системе кеширања. Замисао је следећа: када се већ приступало некој речи, она и њене суседне речи одмах се пребацују из велике и споре меморије у кеш, тако да им се следећи пут, ако затреба, може брже приступити. Уобичајен распоред процесора, кеша и главне меморије приказан је на слици 5. Ако се реч чита или уписује k пута у кратком временском периоду, рачунар ће морати да једном приступи спорој меморији и k – 1 пута брзој меморији. Што је веће k, укупне перформансе су боље [3].



Слика 5. *Кеш се логички смешта између процесора и главне меморије. Постоји више могућих места где се он може наћи физички*

Овај рачун можемо да формализујемо тако што ћемо увести време приступања кешу (c), време приступања главној меморији (m) и степен погађања (h), који представља део свих приступа меморији који може бити задовољен из кеша. Неки аутори дефинишу и степен промашивања, 1-h.

Имајући у виду ове дефиниције, можемо да израчунамо просечно време приступа на следећи начин:

- просечно време приступа = c + (1-h)*m

Када h→1, сви приступи се могу задовољити из кеша и време приступа се приближава вредности c. С друге стране, када h→0, сваки пут је потребно приступити главној меморији, па се време приступа приближава вредности c+m, јер се прво (неуспешно) приступа кешу (c), па онда главној меморији (m). У неким системима може се истовремено приступити и главној и кеш меморији, тако да је, кад уследи промашај у кешу, циклус приступања главној меморији већ започет. Међутим, ова стратегија захтева да се при успешном проналажењу инструкције у кешу прекине приступ главној меморији, што отежава њену имплементацију.

Поштујући принцип локалности, главне и кеш меморије се деле у блокове фиксне величине. Када говоримо о таквим блоковима унутар кеша, обично их зовемо редови кеша (*cache lines*). Када дође до промашаја у кешу, из главне меморије се у кеш учитава не само тражена меморијска реч, већ читав ред.

На пример, ако је ред дужине 64 бајта, а приступа се меморијској адреси 260, у кеш ће се пребацити читав ред речи од адресе 256 до адресе 319. Неке од њих ће вам, уз мало среће, убрзо затребати. Рад на овај начин је ефикаснији од преузимања појединачних речи зато што је брже преузети k речи одједном него појединачне речи k пута. Тако и одреднице у кешу садрже више речи, па је и искоришћење кеша веће.

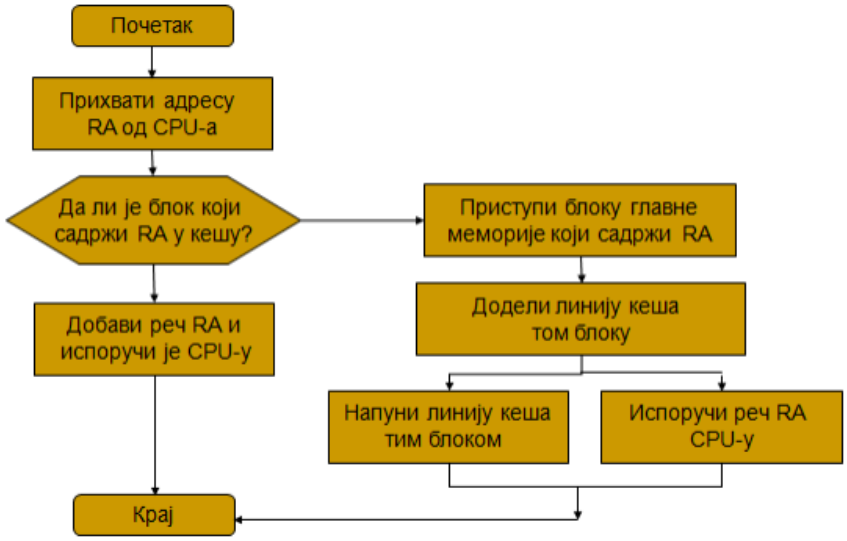
Пројектовање кеша постаје све важније за процесоре високих перформанси. Једна од важнијих особина је величина кеша. Што је кеш већи, боље су му перформансе, али је и скупљи. Друга важна особина је дужина реда кеша. Кеш од 16 KB може се поделити на 1024 реда по 16 бајтова, на 2048 редова по 8 бајтова итд. Трећа особина је организација кеша и своди се на питање: како кеш води рачуна о томе које су речи тренутно у њему?

Четврто питање које се јавља приликом пројектовања кеша гласи: да ли инструкције и податке треба држати у истом кешу или одвојено. Јединствен кеш, у коме се налазе и инструкције и подаци, једноставнији је и аутоматски одржава равнотежу између преузетих инструкција и података. Ипак, данас се тежи раздвојеном кешу, инструкције у једном, подаци у другом кешу. Таква конструкција се зове Хардверска архитектура, према старом рачунару Марк Ховарда Аикена, који је имао засебну меморију за инструкције, а засебну за податке. Оно што пројектанте гура у овом смеру јесте масовно коришћење процесора који податке обрађују паралелно. Јединица за преузимање инструкција треба да приступи инструкцији истовремено с јединицом за преузимање операнда која приступа подацима. Раздвојени кеш омогућава такав паралелан приступ, док јединствени кеш то не дезвољава. Исто тако, пошто се инструкције током извршавања не мењају. садржај кеша са инструкцијама никада се не враћа у меморију.

На крају, пето питање је број јединица кеш меморије. Данас није необично да наиђете на чип са примарним кешом на њему, секундарним кешом изван њега, али у истом пакету с чипом, и терцијарним кешом на неком другом месту [4].

3.1. Принцип рада кеш меморије

Број блокова оперативне меморије је знатно већи од броја блокова кеш меморије, тако да се у кеш меморији у истом мтренутку налазе копије малог броја блокова оперативне меморије. Када централни процесор генерише адресу меморијске локације, формира се управљачки сигнал за приступ кеш меморији. Уколико се податак са траженом адресом налази у кеш меморији, он се преноси у процесор ради даље обраде или се замењује новом вредношћу, резултатом обраде, добијеном из процесора. Уколико се у кеш меморији не налази блок са траженом адресом, активира се процедура којом се из кеш меморије један блок шаље у оперативну меморију, а на његово место се позива тражени блок из оперативне меморије и истовремено се тражени податак шаље даље процесору (слика 6) [4].



Слика 6. *Принцип рада кеш меморије*

3.2. Начин мапирања (технике пресликавања)

Потребно је водити евиденцију и о свим садржајима у кеш меморији, односно о томе који се садржај из радне меморије налази у кеш меморији. Тај поступак назива се механизам или техника пресликавања. Постоје три технике пресликавања [5]:

- Директно пресликавање,
- Асоцијативно пресликавање и
- Сет-асоцијативно пресликавање.

3.2.1. Директно пресликавање

Ово је најједноставнија техника пресликавања где се сваки блок главне меморије може пресликати само у један могући ред кеша (слика 7). Пресликавање се извршава по следећем обрасцу:

$$i = j \text{ moduo } m$$

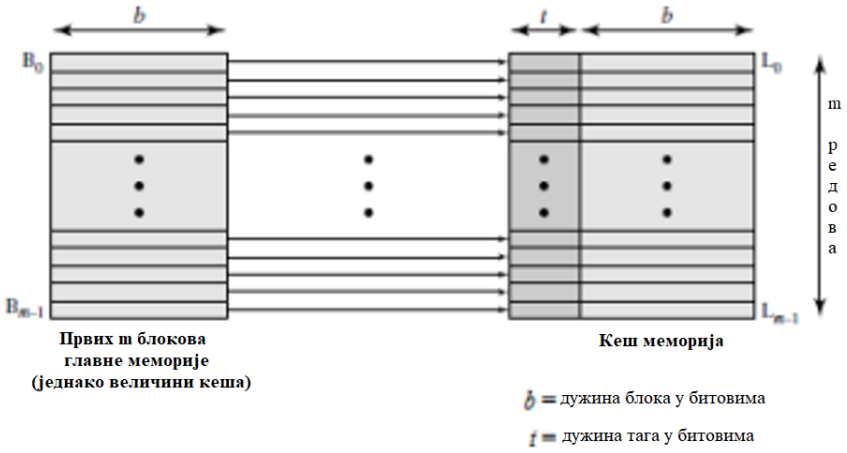
Где је:

i - број реда у кешу

j - број блока у главној меморији

m - број редова у кешу

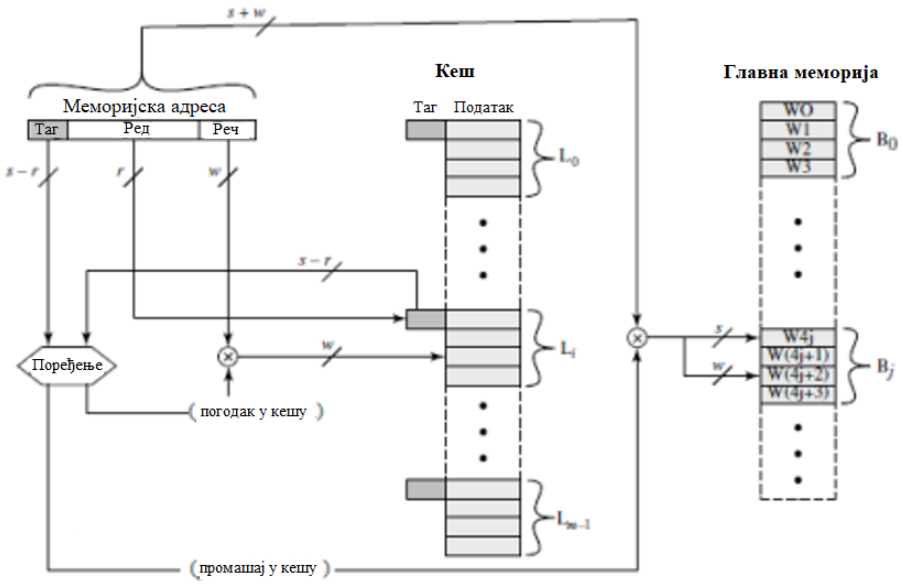
Функција пресликавања се лако имплементира користећи адресе. Када се приступа кешу, свака адреса главне меморије може да се посматра као да се састоји из три дела.



Слика 7. *Директно пресликавање кеша*

Најмање значајних w битова идентификују јединствену реч унутар блока главне меморије. То значи да један блок главне меморије садржи 2 w речи. Осталих s битова одређују 2 s блокова главне меморије. Логика кеша интерпретира тих s битова као таг од s-r битова и поље реда од r битова. Поље реда идентификује један од m=2r редова кеша.

Код директног пресликавања, прво се чита ред у адреси и таг у кешу који одговара том реду (слика 8). Таг се пореди са тагом у меморијској адреси. Ако су исти, блок се налази у кешу и реч се чита. Ако нису исти, кажемо да је дошло до промашаја у кешу и приступа се главној меморији. Блок из главне меморије у којем се налази тражена реч се пребацује у кеш меморију. Тада се чита блок s и реч на адреси w из тог блока.



Слика 8. *Организација кеша код директног пресликавања*

Резиме:

- Дужина адресе: s+w битова
- Капацитет меморије: 2^{s+w} речи
- Величина блока (реда): 2^{w} речи
- Број блокова у главној меморији: 2^{s}
- Број редова у кеш меморији: 2^{r}
- Величина тага: s-r битова

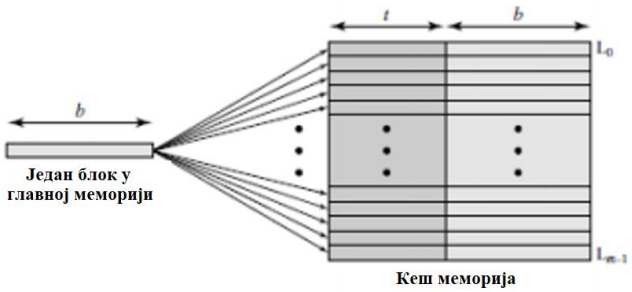
Техника директног пресликавања је једноставна и јефтина за имплементацију. Њен главни недостатак је то што постоји фиксна локација у кешу за сваки блок главне меморије. Ако се догоди да програм стално тражи речи из два различита блока који се пресликавају у исти ред, онда ће се блокови стално избацивати из кеша, а вероватноћа поготка ће бити мала. У табели 2 су дати редови кеша у које се пресликавају блокови оперативне меморије [3].

Табела 2. *Редови кеша у које се пресликавају блокови оперативне меморије*

Ред у кешу	Додељени блокови радне меморије
0	$0, m, 2m, \ldots, 2^s - m$
1	$1, m + 1, 2m + 1, \ldots, 2^s - m + 1$
⋮	⋮
$m - 1$	$m - 1, 2m - 1, 3m - 1, \ldots, 2^s - 1$

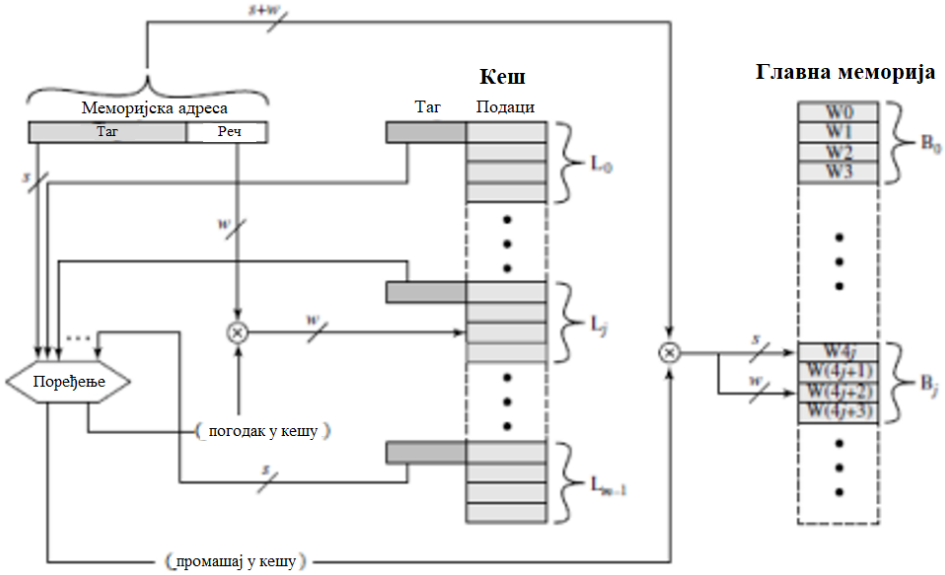
3.2.2. Асоцијативно пресликавање

Асоцијативно пресликавање превазилази недостатак директног пресликавања дозвољавајући сваком меморијском блоку да се учита у било који ред у кешу (слика 9). У том случају, управљачка логика кеша интерпретира меморијску адресу као поље за таг и поље речи. Поље за таг јединствено идентификује блок главне меморије. Да би одредила да ли је блок у кешу, управљачка логистика кеша мора истовремено да испита таг сваког реда ради проналажења подударности.



Слика 9. *Асоцијативно пресликавање*

Код асоцијативног пресликавања постоји флексибилност у погледу тога који блок треба заменити када се нови блок учитава у кеш. Основни недостатак овог пресликавања су сложена електорнска кола која су потребна да би се паралелно испитивали тагови свих редова у кешу. Организација кеша код асоцијативног пресликавања је приказана на слици 10.



Слика 10. *Организација кеша код асоцијативног пресликавања*

Дакле, код асоцијативног пресликавања важи [3]:

- Дужина адресе: n=s+w бита
- Дужина тага: s битова
- Дужина речи: w битова
- Капацитет главне меморије: 2^n речи
- Величина блока (реда): 2^w речи
- Број блокова у главној меморији: 2^s

3.2.3. Сет-асоцијативно пресликавање (Асоцијативно пресликавање скупа)

Ово је компромис између директног и асоцијативног приступа, при чему до изражаја долазе њихове предности, а умањују се њихови недостаци. У случају асоцијативног пресликавања скупа, кеш је подељен на v скупова, од којих се сваки састоји од k редова. Ово пресликавање се врши према следећем обрасцу:

$$m=v*k$$

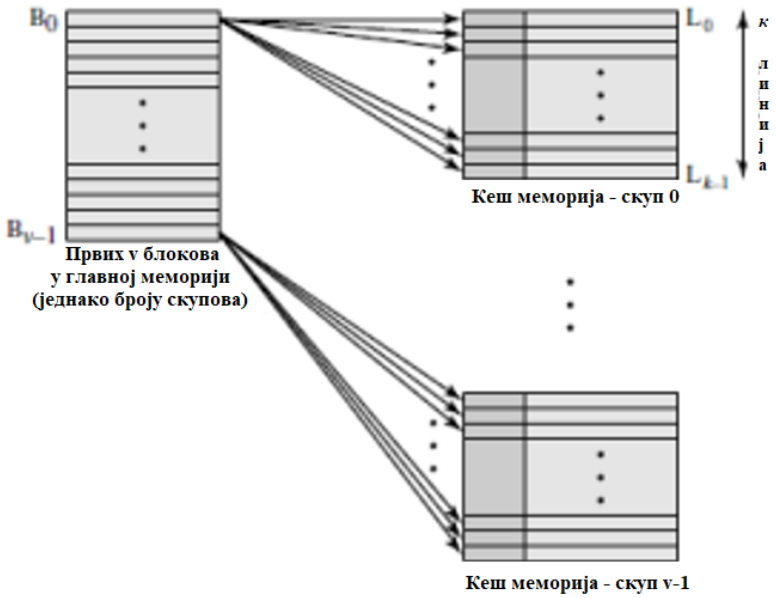
$$i=j \text{ moduov}$$

Где је:

- i – редни број скупа у кеш меморији
- j – редни број блока у главној меморији
- m – број редова у кеш меморији
- v – број скупова у кеш меморији
- k – број редова у сваком скупу

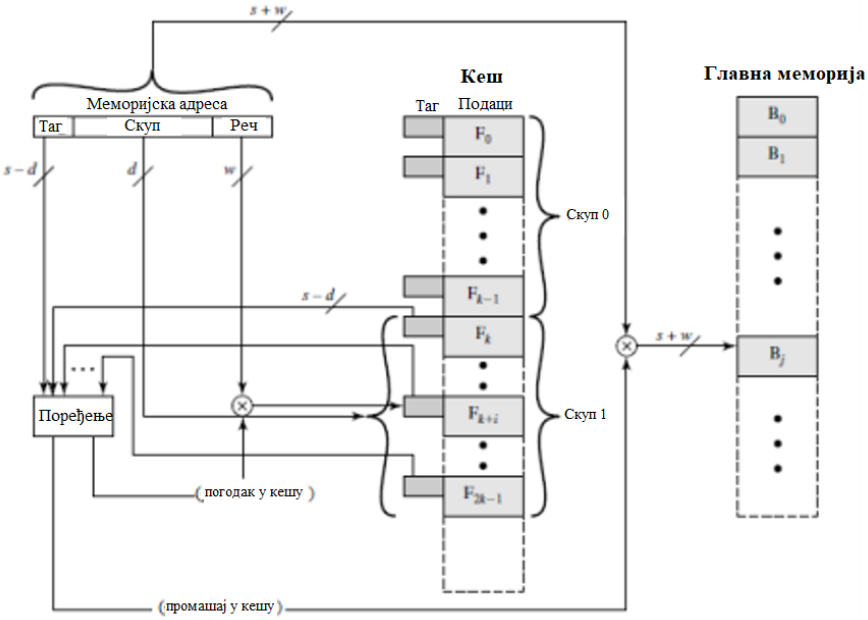
Ако један скуп садржи k редова, онда за пресликавање кажемо да је k-тоструко асоцијативно пресликавање скупа.

Код овог пресликавања, блок Bj може да се преслика у било који од редова скупа j (слика 11). У том случају, управљачка логика кеша интерпретира меморијску адресу као три поља: таг, скуп и реч. Помоћу d битова скупа се одређује један од v=2d скупова. Један од 2s блокова главне меморије се одређује помоћу s битова поља за таг и поља за скуп.



Слика 11. *Асоцијативно пресликавање скупа*

Код потпуно асоцијативног пресликавања, таг у меморијској адреси је прилично дугачак и мора да се пореди са тагом од сваког реда у кешу. Код k-тоструког асоцијативног пресликавања скупа, таг у меморијској адреси је много мањи и пореди се само са k тагова унутар једног скупа (слика 12).



Слика 12. *Организација кеша код асоцијативног пресликавања скупа*

Код асоцијативног пресликавања скупа важи следеће:

- Дужина меморијске адресе: n=s+w битова
- Капацитет меморије: 2^n речи
- Величина блока: 2^w речи
- Број блокова у главној меморији: 2^s
- Број редова у скупу: k
- Број скупова: $v=2^d$
- Број редова у кеш меморији: C=k*v

У екстремном случају када је v=m, k=1, асоцијативно пресликавање скупа постаје директно пресликавање, а када је v=1, k=m, постаје асоцијативно пресликавање. Најчешћа организација овог пресликавања је коришћење два реда по скупу (v=m/2, k=2) [3].

3.3. Алгоритам замене

При генерисању захтева за упис или читање од стране процесора може се утврдити да се блок у коме је захтевана реч не налази у блоку кеш меморије који је предвиђен одабраном техником пресликавања. Та да се један блок кеш меморије мора вратити у оперативну меморију, да би се у кеш меморији направио простор за блок из оперативне меморије у коме се захтевана реч налази. Овај блок се одређује коришћењем једног од алгоритама замене који се хардверски реализују у кеш меморији.

Код кеш меморије се директним пресликавам алгоритам замене је тривијалан јер блок за замену одређен бројем блока генерисане адресе. Код кеш меморије са асоцијативним и асоцијативним пресликавањем скупа, алгоритмом замене се за замену бира један од свих блокова кеш меморије са асоцијативним пресликавањем и један од свих блокова скупа, одређеног бројем скупа генерисане адресе, кеш меморије са асоцијативним пресликавањем скупа. Стога се алгоритам замене реализује за целу кеш меморију са асоцијативним пресликавањем, а посебно за сваки скуп кеш меморије са асоцијативним пресликавањем скупа.

При избору алгоритма замене треба водити рачуна о два захтева. Први је да он треба да обезбеди минималну вероватноћу да ће блок који је одабран за замену и враћен из кеш у оперативну меморију убрзо морати поново да се довуче из оперативне у кеш меморију. Други је да цена хардвера потребног за његову реализацију буде што је могуће нижа. Ова два захтева су контрадикторна, јер је цена хардвера алгоритма замене који боље испуњавају првизахтев виша у односу на цену хардвера алгоритама замене који то чине лошије. Овде се разматрају четири алгоритма замене и то RANDOM, FIFO, LRU и PSEUDO [6].

3.3.1. RANDOM

RANDOM алгоритмом замене за замену се случајно бира блок коришћењем једног од постојећих генератора случајних бројева. Могући начин реализације је да се користи бројач по модулу 2^n, где 2^n представља број улаза кеш меморије са асоцијативним пресликавањем и број улаза по скупу кеш меморије са асоцијативним пресликавањем скупа. Може се одабрати неки произвољан сигнал и користити за инкрементирање бројача. У тренутку када нема сагласности и треба одабрати блок за замену, тренутна вредност бројача одређује улаз за замену. После тога се продужава инкрементирање бројача до следећег тренутка када нема сагласности и када поново на основу вредности бројача треба одабрати улаз за замену. Овај алгоритам не води рачуна о томе да ће блок који јеодабран за замену и враћен из кеш у оперативну меморију можда убрзо морати поново да се довуче из оперативне у кеш меморију. Међутим, хардвер за његову реализацију је једноставан.

3.3.2. FIFO

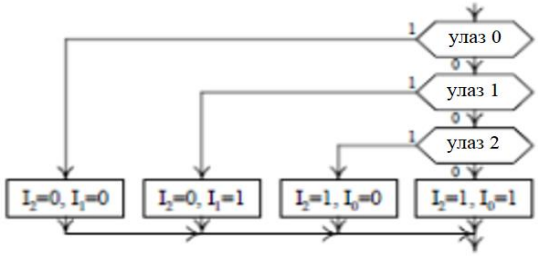
FIFO (*First in-First out*) алгоритмом замене за замену се бира блок који је најраније унет из оперативне меморије у кеш меморију. Могући нађин реализације је да се користи бројач по модулу 2^n, где 2^n представља број улаза кеш меморије за асоцијативним пресликавањем број улаза по скупу кеш меморије са асоцијативним пресликавањем скупа. У тренутку када нема сагласности и треба одабрати блок за замену, тренутна вредност бројача одређује улаз за замену. При томе се и врши инкрементирање бројача. Овај алгоритам не води рачуна о томе да ли ће блок који је одабран за замену и враћен из кеш у оперативну меморију можда убрзо морати поново да се довуче из оперативне у кеш меморију, већ за замену бира блокове по оном редоследу по коме су и довлачени из оперативне меморије у кеш меморију. Међутим, хардвер за његову реализацију је једноставан.

3.3.3. LRU

LRU (*last recently used)* алгоритмом замене за замену се бира блок коме се најдуже времена није приступало. Могући начин реализације је да се користи 2^n бројача по модулу 2^n, где 2^n представља број улаза кеш меморије са асоцијативним пресликавањем и број улаза по скупу кеш меморије са асоцијативним пресликавањем скупа. Сваком од 2^n улаза кеш мемирије додељује се један од 2^n бројача по модулу 2^n. Бројачи се током рада тако ажурирају да је у њима увек 2^n различитих вредности и да бројач улаза коме се најдуже времена није приступало има све јединице и тиме вредност 2^n-1. У тренутку када има сагласности садржај бројача улаза у коме је откривена сагласност се упоређује са садржајима бројача свих преосталих улаза. Бројачи који имају умањену вредност од бројача улаза у коме је откривена сагласност се инкрементирају, бројачи који имају већу вредност од бројача улаза у коме је откривена сагласност се не мењају и бројч улаза у коме је откривена сагласност се поставља на нулу. У тренутку када нема сагласности за замену се бира улаз чији бројач има све јединице и тиме вредност 2^n-1, бројачи свих преосталих улаза имају мању вредност од бројача улаза који је одабран за замену па се инкрементирају и бројач улаза који је одабран за замену се поставља на нулу. На почетку рада бројачи 2n улаза треба тако да се иницијализују да у њима буде 2n различитих вредности, при чему бројаче улаза треба иницијализовати на вредност 2^n-1, 2^n-2, …, 1 и 0 сагласно редоследу по коме се жели попуњавање улаза. Тиме се и за попуњавање кеш меморије и за замену блокова попуњене кеш меморије користи исти механизам.

3.3.4. PSEUDO LRU

PSEUDO LRU алгоритмом замене се покушава реализација принципа LRU алгоритма замене за једноставнијим хардвером. Могући начин реализације је дат за кеш меморије са асоцијативним пресликавањем и четири улаза и кеш меморије са асоцијативним пресликавањем скупа и четири улаза по скупу. Овај алгоритам замене често се користи код асоцијативног пресликавања скупа код кога је број улаза по скупу најчешће два, четири и осам. Улази 0 и 3 су упарени у две групе, при чему улази 0 и 1 чине једну групу на улазу, а 2 и 3 другу групу. Овим групама се додељује индикатор I_2 који се поставља на 0 кад год се приступи улазу 0 и 1 на 1 кад год се приступи улазу 2 и 3. Улазима из групе коју чине улази 0 и 1 додељује се индикатор I_1 који се поставља на 0 кад год се приступи улазу 0 и на 1 кад год се приступи улазу 1. Улазима из групе коју чине улази 2 и 3 додељује се индикатор I_0 који се поставља на 0 кад год се приступи улазу 2 и на 1 кад год се приступи улазу 3. Дијаграм тока овог алгоритма замене представљен је на слици 13.



Слика 13. *Дијаграм тока PSEUDO LRU алгоритма замене*

3.4. Начин уписа (Ажурирање оперативне меморије)

Ажурирање оперативне меморије одређује како се код операције уписа мења садржај у оперативној меморији. При томе се, код захтева за упис, могу јавити две ситуације. Прва је да у кеш меморији постоји сагласност, а друга да нема сагласности.
За случај када је у кеш меморији откривена сагласност, постоје два приступа и то **упиши кроз** (*write through* или *store through*) и **врати назад** (*write back* или *copy back*). Код приступа упиши кроз, при сваком захтеву за упис истовремено се врши упис и у кеш меморију и у оперативну меморију. Код приступа назад, при сваком захтеву за упис врши се упис само у кеш меморију, па одговарајући задржај у оперативној меморији није ажуран. Због тога се за сваки блок у кеш меморији води евиденција о томе да ли је модификован или не. Уколико је касније потребно довући нови блок из оперативне меморије на место неког блока у кеш меморији који је неким од претходни уписа модификован, потребно је, најпре, дати блок кеш меморије вратити у оперативну меморију и тиме обезбедити да и садржај у оперативној меморији буде ажуран. Поред тога, када се неком процесу одузима процесор, треба проверити који су блокови у кеш меморији модификовани, па их ради ажурирања садржаја у оперативној меморији, вратити из кеш меморије у оперативну меморију. Стога кад кеш меморија које користе овај приступ ажурирања садржаја оперативне меморије, поред захтева за читање и упис, постоје и захтеви за селективно и комплетно враћање блокова из кеш меморије у оперативну меморију.

Предност приступа упиши кроз је у томе да је оперативна меморија увек ажурна чиме је обезбеђена конзистентност садржаја оперативне и кеш меморије. Недостатак овог приступа је у обраћању оперативној меморији при сваком упису у кеш меморију, чиме се беспотребно оптерећује магистрала уписивањем међурезултата у оперативну меморију.
Предност приступа врати назад је у томе што се оперативној меморији и магистрали приступа само онда када се блок враћа из кеш меморије у оперативну меморију што резултује у мањем саобраћају на магистрали. Недостатак овог приступа је потреба да се блок који се избацује из кеш меморије мора најпре вратити у оперативну меморију, па тек онда довући нови, што знатно успорава одзив кеш меморије у случају промашаја.
Овде се види да су све предности једног приступа уједно и недостаци другог. Стога се приступ врати назад користи тамо где је магистрала уско грло система, а приступ упиши кроз где магистрала то није.

За случај када је у кеш меморији није откривена сагласност, постоје два приступа и то **довуци блок** (*write allocate*) и **не довлачи блок** (*no write allocate*). Код приступа довуци блок, блок се довлачи из оперативне у кеш меморију, чиме се обезбеђује да се сада у кеш меморији открива сагласност. Даљи поступак одговара претходно описаној ситуацији за операцију уписа и откривену сагласност, у којој се упис врши у кеш меморију, а за ажурирање садржаја оперативне меморије користи приступ упиши скроз или врати назад. Код приступа не довлачи блок, блок се не довлачи из оперативне у кеш меморију, већ се упис врши само у оперативну меморију. Обично се уз приступ врати назад (*write back* или *copy back*) користи приступ довуци блок (*write allocate*), док се уз приступ упиши скроз (*write through* или *store through*) користи приступ не довлачи блок (*no write allocate*) [6].

3.5. Нивои кеш меморије

Већина модерних десктопа и серверских CPUs имају најмање три независна кеша: инструкцију кеша ради брзине доношења извршних инструкција, податке кеша ради брзине доношења података и складиштења, и *translation lookaside buffer* (TLB) који се користи ради виртуелног-у-физички адресни превод и за извршне инструкције и податаке. Кеш подаци се обично организују као хијерархије кеша вишег-нивоа (L1, L2, итд.).

Кешеви вишег нивоа оперишу тако што прво проверавају најмањи кеш *нивоа-1* (L1); ако је погодак, процесор наставља великом брзином. Ако је мали кеш промашај, следећи већи кеш (L2) се проверава, и тако даље, пре него што се провери спољашња меморија.

Пошто латенција (кашњење) између главне меморије и брзе кеш меморије постаје све веће, неки процесори су почели да користе чак три нивоа кеша на чипу. На пример, *Alpha* 21164 (1995) је имао 1 до 64 MB ван-чипа L3 кеш; IBM POWER4 (2001) имао је ван-чипа L3 кеш од 32 MB по процесору, подељен између неколико процесора; *Itanium* 2(2003) имао је 6 MB уједињени L3 кеш на-чипу; *Itanium* 2 (2003) MX 2 *Module* обухвата два *Itanium* 2 процесора са дељеним 64 MB L4 кеш меморијом на мулти-чип модулу који је пински компатибилан са *Madison* процесором; *Intel*-ов *XeonMP* производ под кодним називом "*Tulsa*" (2006) има 16 MB на-чипу L3 кеша који учествује између два процесорска језгра; AMD *Phenom* II (2008) има до 6 MB на-чипу уједињену L3 кеш меморију; и *IntelCore* i7 (2008) има 8 MB на-чипу уједињену L3 кеш меморију која је инклузивна, коју деле сва језгра [1].

Закључак

Разлог за постојање кеш меморије је тај, што су нормални DRAM чипови сувише спори у односу на фреквенцију такта процесора, који зато мора да чека за додатне циклусе такта да би добио податке из DRAM-а. Зато се између DRAM-а и CPU-а имплементира брза статичка меморија SRAM, позната као кеш. У кеш меморији налазе се они садржаји из радне меморије којима се најчешће приступа. При читању података прво се проверава да ли се подаци налазе у кешу, а потом, уколико их ту нема приступа се радној меморији. Пошто је време приступа кеш меморији веома брзо (за ред величине мање од времена приступа радној меморији), тако од кеш меморије у највећој мери зависе перформансе процесора, па се са повећањем кеш меморије значајно повећавају перформансе и самих процесора, као и рачунара у целини [6].

Литература

[1] Hennessy, John L.; Patterson, David A. (2012). *Computer Architecture: A Quantitative Approach*

[2] Alan Jay Smith. "*Design of CPU Cache Memories*". Proc. IEEE TENCON, 1987.

[3]Andrew S. Tanenbaum, 2006, *Архитктура и организација рачунара* (превод петог издања)

[4] Душан Регодић, 2018, *Архитектура и организација рачунарских система,* Београд

[5] M. Morris Mano, Charles R. Kime, Tom Martin, *Logic and Computer Design Fundamentals* (пета едиција)

[6] Barbieri, Alberto, 2016, *Господарење меморијским простором,* завршни рад