Увод у организацију и архитектуру рачунара 1

Александар Картељ

kartelj@matf.bg.ac.rs

Напомена: садржај ових слајдова је преузет од проф. Саше Малкова

Меморија

Преглед, подела и карактеристике

Меморија

- Меморија је уређај који омогућава чување (записивање) и читање података у рачунару

Основне карактеристике

- Трајање записа
- Тип носиоца
- Капацитет
- Јединица преноса
- Адресибилност
- Цена
- Могући начини приступа
- Перформансе
- Могућност промене садржаја

Трајање записа

- Меморије са сталним записом
- Меморије са привременим записом

Тип носиоца

- Полупроводничке
- Са магнетном површином
- Оптичке

Капацитет

- Капацитет се изражава у бајтовима или речима
- Уобичајене дужине речи
  - 8, 16, 32, 64, 128 битова
  - 1, 2, 4, 8, 16 бајтова

- Капацитет уобичајено у бајтовима
  - KB, MB, GB, TB
    - $1KB = 10^3B$, $1MB=10^6B$, $1GB=10^9B$, $1TB=10^{12}B$

Јединица преноса

- За унутрашње меморије јединица преноса је обично *реч*
  - 1,2,4,8,16 бајтова
- За спољашње меморије јединица преноса је обично *блок*
  - од 512B до неколико MB

Адресибилност

- Адресибилна меморија
  - ако се може адресирати свака појединачна меморијска локација (реч)
- Полуадресибилна меморија
  - ако се адресом приступа групи бајтова, која је већа од речи
- Неадресибилна меморија
  - ако се садржају не приступа путем адресе

Цена

- Цена меморије се пореди у односу на одређен капацитет и перформансе

Могући начини приступа

- Секвенцијалан приступ
- Непосредан приступ
- Произвољан приступ
- Асоцијативан приступ

Могући начини приступа (2)

- **Секвенцијалан приступ**
  - подаци су организовани у јединице – *слогове*
  - слогови се међусобно раздвајају контролним информацијама
  - пише се редом
  - чита се редом, како је вршено писање
  - пример: магнетна трака
- Непосредан приступ
- Произвољан приступ
- Асоцијативан приступ

Могући начини приступа (3)

- Секвенцијалан приступ
- **Непосредан приступ**
  - постоји зависност адресе слога и његове физичке локације (не мора да буде пуна)
  - на основу адресе се приступа непосредно слогу или његовој околини
  - време за приступ није фиксно
  - пример: магнетни диск
- Произвољан приступ
- Асоцијативан приступ

Могући начини приступа (4)

- Секвенцијалан приступ
- Непосредан приступ
- **Произвољан приступ**
  - свака адресибилна локација има механизам приступа подацима
  - време за приступ је фиксно
  - пример: главна меморија рачунара
- Асоцијативан приступ

Могући начини приступа (5)

- Секвенцијалан приступ
- Непосредан приступ
- Произвољан приступ
- **Асоцијативан приступ**
  - подврста меморије са произвољним приступом
  - податку се приступа на основу неког узорка (маске) адресе или податка
  - пример: кеш меморија

Перформансе

- Време приступа
  - трајање операције читања или писања
  - од неколико *ns* до неколико *ms*
- Трајање временског циклуса
  - обухвата време приступа
  - и додатно време за ослобађање магистрале и припрему за наредну операцију
- Брзина преноса
  - време за које већа количина података може да се прочита или упише
  - узима у обзир време приступа али и архитектуру

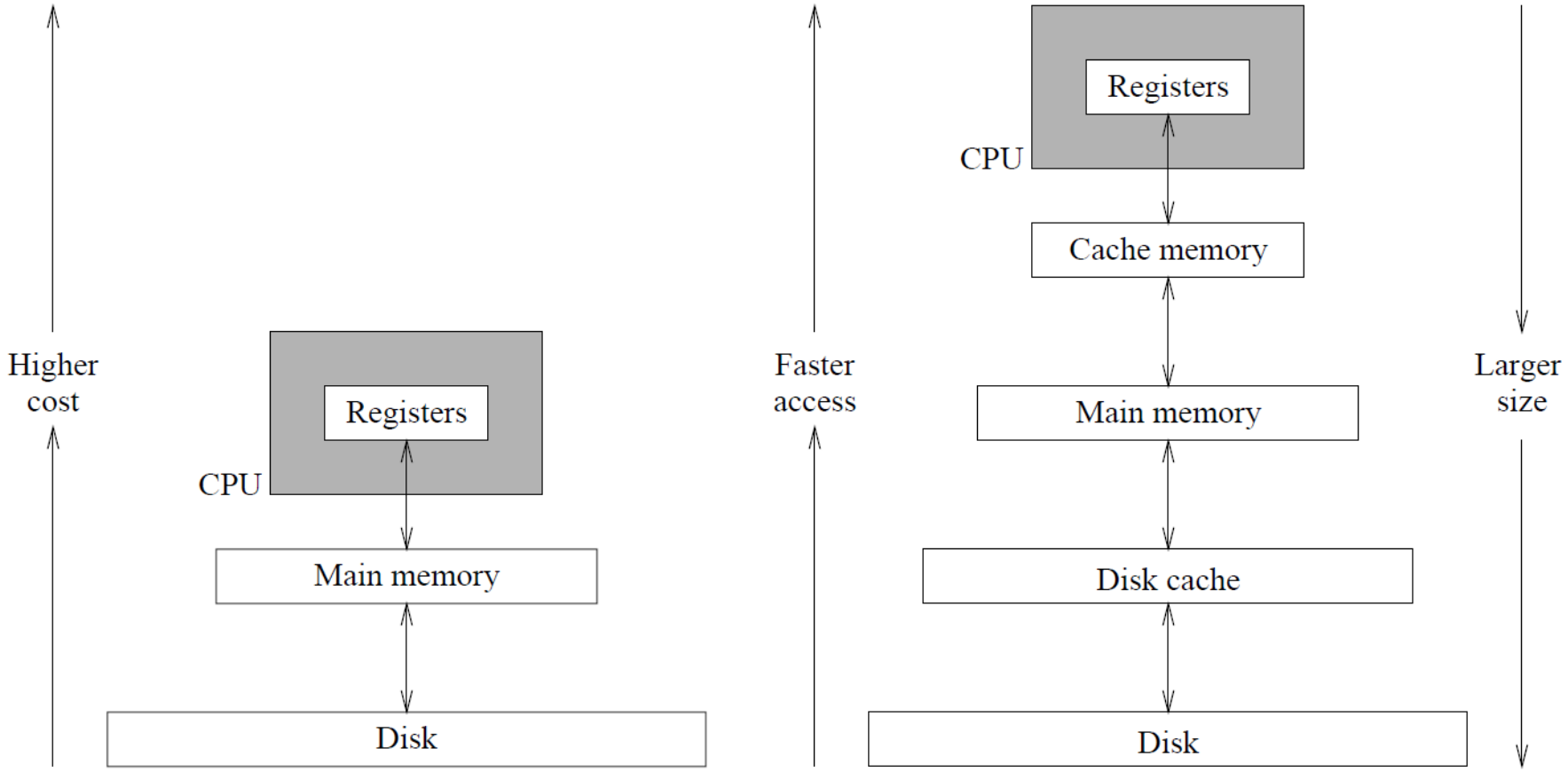
Могућност промене садржаја

- Меморија само за читање
- Меморија за читање и писање

Хијерархија меморија

- За меморију по правилу важи:
  - што је краће време приступа, цена је већа
  - што је већи капацитет, време приступа је дуже
  - што је већи капацитет, цена по биту је нижа
  - нове технологије доносе нижу цену по биту уз очување претходних односа

Хијерархија меморија (2)



Основне врсте меморије

- Два основна типа меморија:
  - меморије само за читање
    - (*read-only memory*)
    - *ROM*
  - меморије за читање и писање
    - (*read-write memory*)
    - *RAM*
    - назив (*random access memory*) је неисправан, зато што и *ROM* и *RAM* омогућавају произвољан приступ

ROM

- *ROM*
  - меморија само за читање
  - (*read-only memory*)
  - не захтева напајање за одржавање садржаја
  - могу чувати податке док је рачунар искључен
  - обично се употребљава за подизање рачунарског система (*boot*)

Врсте *ROM*-а

- Фабрички програмиран
  - прави се у случају масовне потребе
- Програмибилан
  - *PROM* (*programmable ROM*)
  - на пример, корисник може да спаљује осигураче по избору
- Вишекратно програмибилан
  - *EPROM* (*erasable programmable ROM*)
  - излагањем ултраљубичастом светлу се брише садржај *EPROM*-а
- Вишекратно програмибилан са ел. брисањем
  - *EEPROM* (*electrically erasable programmable ROM*)
  - омогућава да се селективно бише садржај

RAM

- Две основне врсте *RAM* меморија су
  - статички *RAM* и
  - динамички *RAM*

Меморија

Главна (радна) меморија

Статички *RAM*

- *SRAM*
- Имплементира се помоћу резе или флип-флопа
- Не захтева освежавање да би чувао садржај

- Предности:
  - Једноставност употребе
  - Брзина

- Употребљава се за кеш меморије

Динамички *RAM*

- *DRAM*
- Имплементира се помоћу малих кондензатора
- Захтева периодично освежавање да би чувао садржај
- Читање нарушава садржај
  - неопходно *писање-после-читања*
- Предности
  - Нижа цена
  - Мање загревање
  - Већа густина паковања
- Употребљава се за радну меморију рачунара

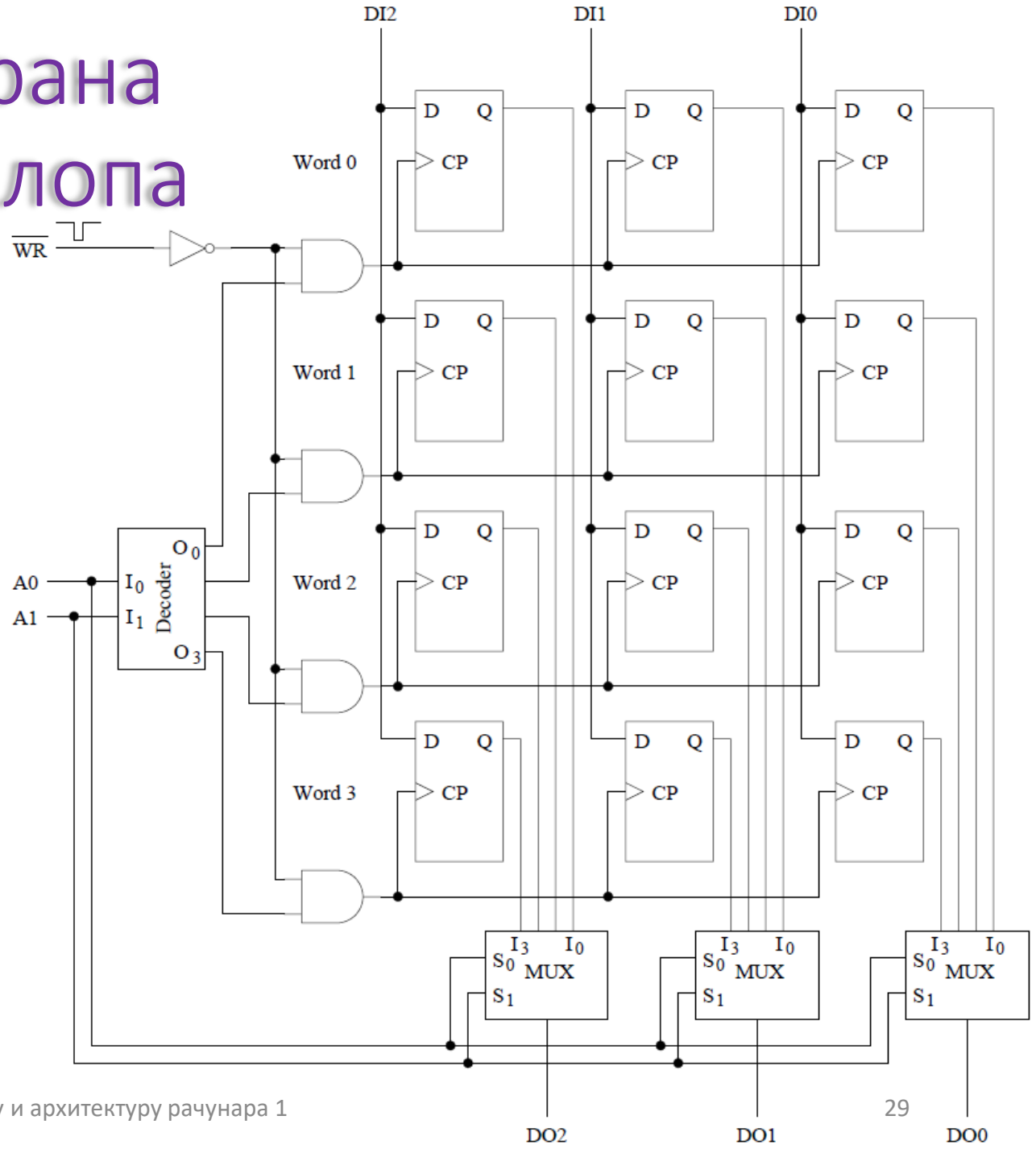
Технологије израде *DRAM*-а

- Динамичка меморија се дели на
  - асинхрону
    - меморија одређује време одзива за сваку од операција
    - ако часовник ради брже, уводе се стања чекања
    - брзина се мери временом одзива, нпр. 20*ns*
  - синхрону
    - рад меморије се синхронизује са часовником
    - зна се број циклуса за сваку од операција
    - брзина се мери брзином часовника и бројем циклуса

Меморија од *D* флип-флопова

- Употребљава се дводимензиони низ *D* флип-флопова
  - сваки ред чува једну реч
  - број колона одговара броју битова у речи
    - *хоризонтално ширење* је повећавање број битова у речима
  - број редова одговара броју речи у меморији
    - *вертикално ширење* је повећавање броја речи
  - оба броја су обично неки степени броја 2
  - меморија *M* x *N* има *M* речи од по *N* битова

Меморија имплементирана матрицом 4x3 *D* флип-флопа



Меморија од *D* флип-флопова (2)

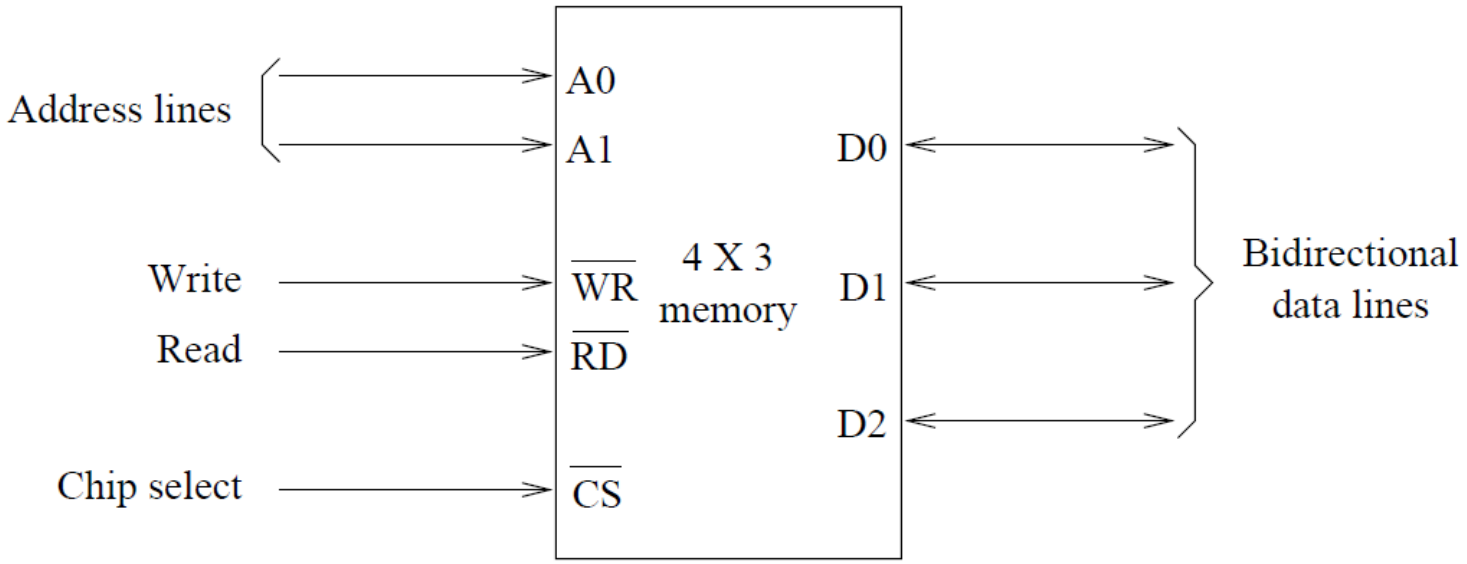
- Декодер одређује тачно један ред на основу улазне адресе
  - адреса је кодирана са две линије
  - декодер са И-елементима гради демултиплексор који усмерава сигнал на одговарајући ред
- Активан сигнал часовника ће добити само изабрани ред и то само у случају писања
- Сви флип-флопови у једној колони добијају исти улазни сигнал

- За читање се употребљава 4-1 мултиплексор
  - адресне линије се користе као селектори

Ограничења и проблеми

- Није прилагођена повезивању на магистралу
  - потребно је да исте линије носе улазне и излазне податке
  - овај проблем се решава употребом бафера са три стања, али је ван домета курса...
- Не може да се користи за прављење већих меморија
  - Потребан је додатни селекторски улаз који означава да ли се блок користи или не

Блок дијаграм меморије 4x3



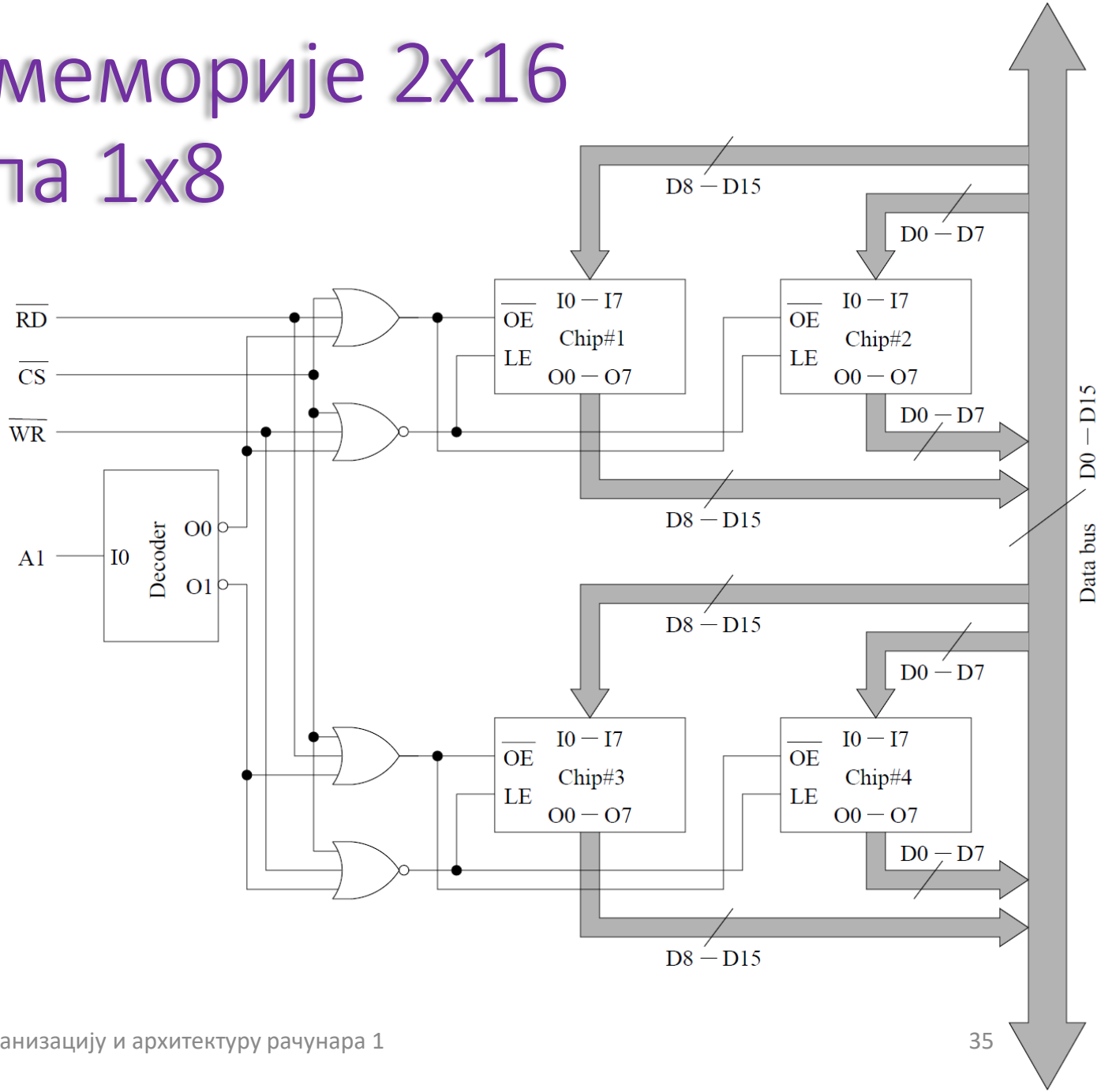
Прављење већих меморија

- Од меморијских блокова који имају контролне селекторе (*CS*) могу се правити већи меморијски блокови
- Први корак је прављење независне меморијске јединице која није чврсто везана за специфичне адресе у адресном простору
  - Нешто као већа верзија претходно представљеног меморијског блока
- Други корак је везивање оваквих независних меморијских јединица за конкретан адресни простор

Прављење већих меморија (2)

- Повезивањем више чипова повећава се ширина меморијске речи (*хоризонтална експанзија*)
  - контролни улази нпр. два осмобитна чипа се везују заједно како би представљали 16-битну целину
  - улазе и излазе сваког чипа везујемо на одговарајуће линије магистрале података
  - на сличан начин се може добити и шира реч
- Додавањем редова повећавамо величину меморије (*вертикална експанзија*)
  - сваки ред чува по једну реч
  - помоћу декодера се врши одабир активног реда

Логички дијаграм меморије 2x16 изграђене од 4 чипа 1x8



Примери меморијских чипова

- Постоји велики број чипова који се употребљавају за израду већих меморија
  - примери *SRAM* и *DRAM* чипова фирме *Micron*
- *SRAM*
  - 8Mb чип, у три конфигурације:
    - 512K x 18
    - 256K x 32
    - 256K x 36
  - додатни битови служе за препознавање и отклањање грешака
  - време приступа 3.5*ns*
  - чип 512K x 18 има 19 адресних линија
  - чипови 256K x 32 и 256K x 36 имају по 18 адресних линија

Примери меморијских чипова (2)

- *DRAM*
  - синхрони *DRAM*
  - 256Mb чип, у три конфигурације:
    - 64M x 4, 26 адресних линија
    - 32M x 8, 25 адресних линија
    - 16M x 16, 24 адресне линије
  - трајање циклуса је око 7*ns*

Дизајн већих меморија

- Ако је циљ меморија *M* x *N*, а користе се чипови *D* x *W*:
  - број колона је *N*/*W*
  - број редова је *M*/*D*
  - број чипова је (*M* x *N*)/(*D* x *W*)
- У примеру
  - правимо меморију од 256*MB*
  - циљна конфигурација је 64*M* x 32*b*
  - користимо чипове 16*M* x 16*b*
  - матрица чипова је, дакле, 4 x 2

Дизајн већих меморија (2)

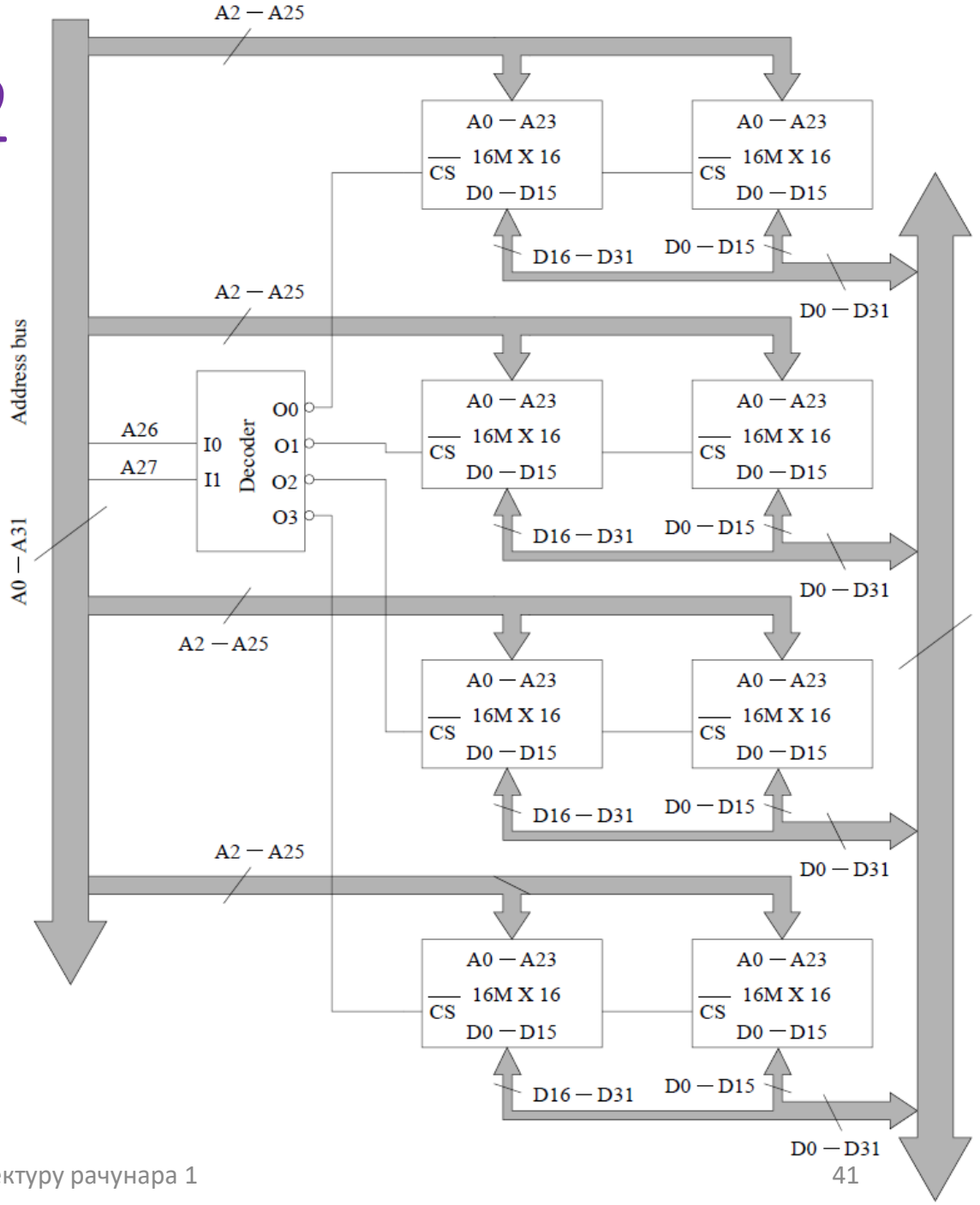
- Везивање на магистралу података је непосредно
  - сваком чипу у реду одговара део линија података
- Ако се у једном кораку чита *N* битова (>8)
  - *Z* најнижих битови адресе се игноришу
  - $Z = \log_2(N/8)$

- У примеру
  - 256M се адресира са 28 битова адресе
  - један чип има 24 бита адресе
  - најнижа 2 бита адресе A0, A1 се не користе
    - ширина меморије је 32 бита, а адресирање по бајтовима
  - на чипове се везују 24 бита A2 до A25
  - битови адресе A26 и A27 се користе за бирање реда чипова

Дизајн већих меморија (3)

- Контролни сигнали за читање и писање свих чипова се повезују међусобно и на контролну магистралу
  - (изостављено из наредног дијаграма ради једноставности)

Дизајн меморије 64M x 32 од чипова 16M x 16



Меморија

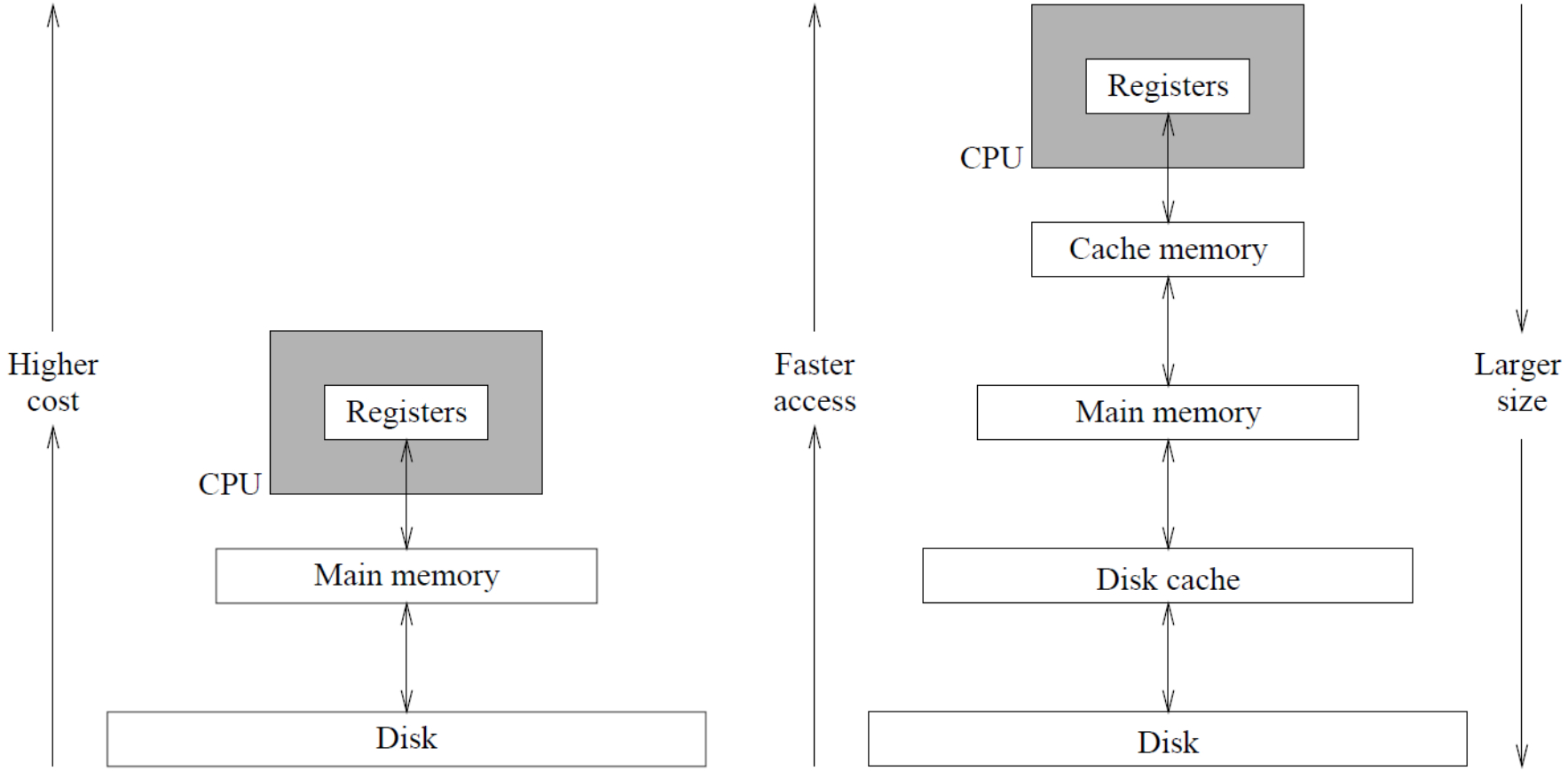
Кеш меморија (само функционални опис, без реализације)

Намена кеша

- Процесор садржи регистре, као најефикаснију меморију у систему
- Радна меморија рачунара је релативно велика и спора
- Разлика у брзини ова два слоја је довољно велика да може да значајно ослаби перформансе система

- Кеш се додаје као међуслој између регистара (тј. процесора) и радне меморије
  - брзина између регистара и радне меморије
  - величина између регистара и радне меморије
- Ако је разлика превелика, додаје се више слојева кеш меморије

Хијерархија меморија (подсећање)



Примери брзина и величина кеш меморија

CPU Type	Pentium	Pentium Pro	Pentium II	Pentium III	Pentium 4
CPU speed	233MHz	200MHz	450MHz	1.4GHz	3.6GHz
L1 cache speed	4.3ns (233MHz)	5.0ns (200MHz)	2.2ns (450MHz)	0.71ns (1.4GHz)	0.28ns (3.6GHz)
L1 cache size	16KiB	32KiB	32KiB	32KiB	20KiB
L2 cache type	onboard	on-chip	on-chip	on-die	on-die
CPU/L2 speed ratio		1/1	1/2	1/1	1/1
L2 cache speed	15ns (66MHz)	5ns (200MHz)	4.4ns (225MHz)	0.71ns (1.4GHz)	0.28ns (3.6GHz)
L2 cache size	varies	256K	512K	512K	1M
CPU bus speed	66MHz	66MHz	100MHz	133MHz	800MHz
Memory bus speed	60ns (16MHz)	60ns (16MHz)	10ns (100MHz)	7.5ns (133MHz)	1.25ns (800MHz)

Принцип рада кеша

- Основни принцип рада је захватање података из радне меморије унапред (*prefetch*), пре него што заиста затребају процесору
  - ако је успешно предвиђено који ће подаци бити потребни процесору у блиској будућности, њих ће процесор читати из кеша а не из меморије
  - тиме се значајно подижу перформансе система

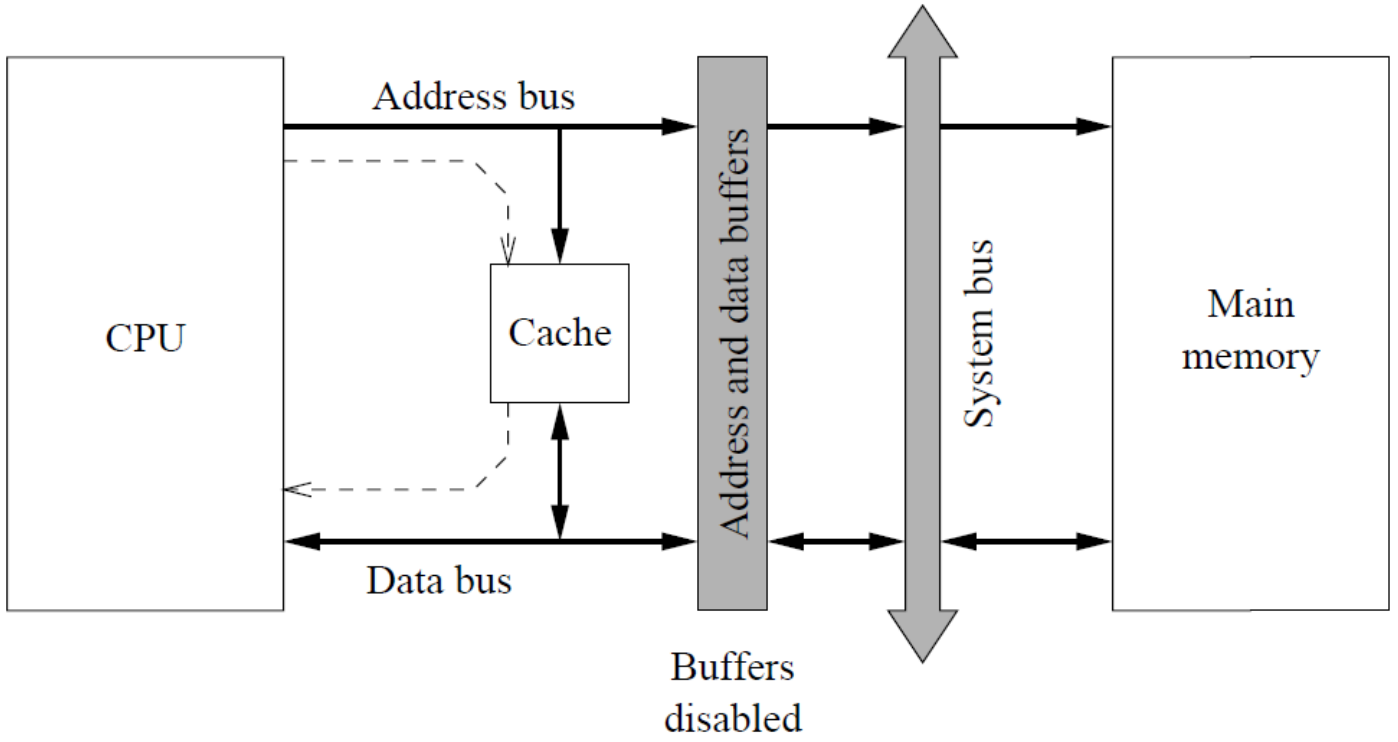
Основне операције кеша

- Кеш се користи у случају две основне меморијске операције
  - читање и
  - писање

- У оба случаја постоје по две варијанте
  - када су подаци присутни у кешу
    - тзв. *погодак* (*hit*)
  - када подаци нису присутни у кешу
    - тзв. *промашај* (*miss*)

Читање у случају поготка

- Ако се потребни подаци налазе у кешу, онда се одатле и читају



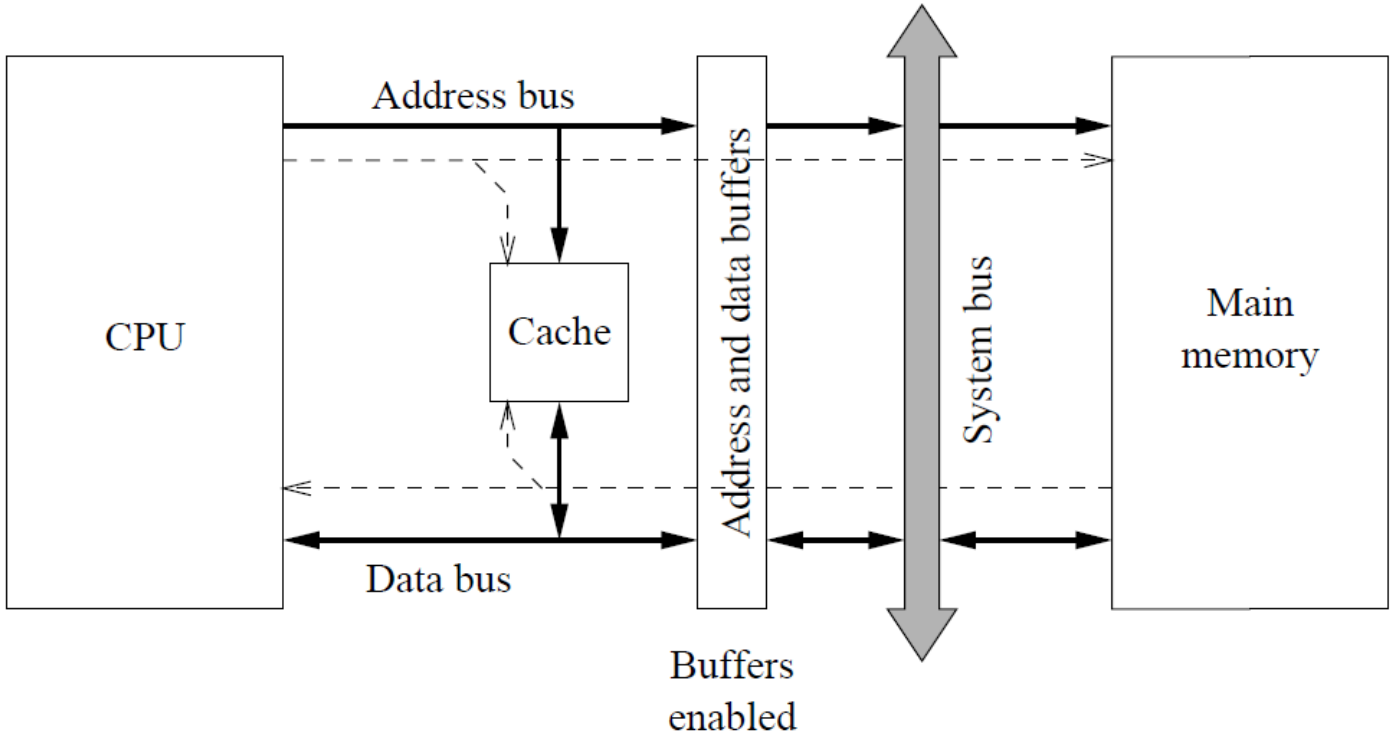
Читање у случају поготка (2)

- У случају поготка, линије адресе и података према меморији се блокирају
- Размена информација се одвија искључиво са кешом

- Читање са поготком је значајно брже од читања из меморије без примене кеша

Читање у случају промашаја

- Ако се потребни подаци не налазе у кешу,
  онда се читају из меморије и истовремено уписују у кеш



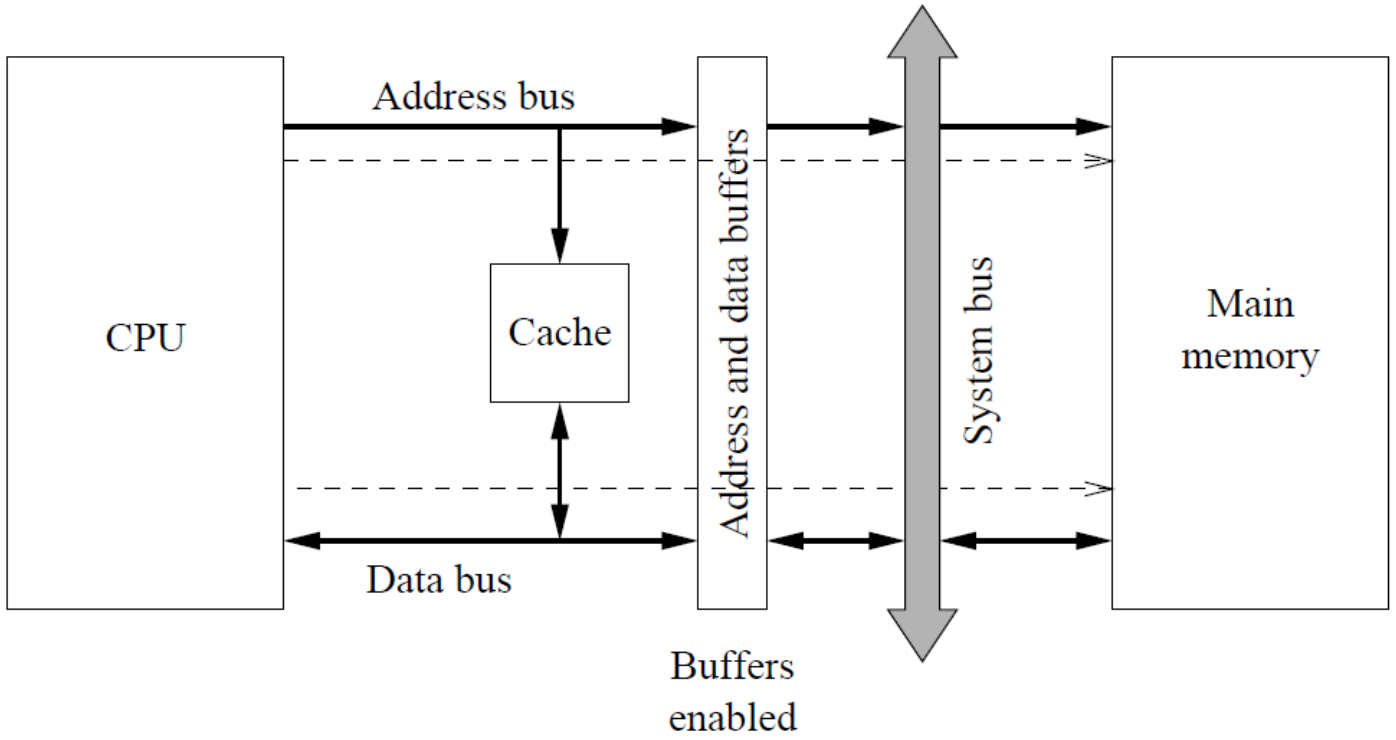
Читање у случају промашаја (2)

- У случају промашаја, линије адресе и података према меморији су активне
- Одвија се уобичајено (као да нема кеша) читање података из меморије
  - додатно се прочитани подаци уписују и у кеш

- Читање са промашајем је нешто спорије од читања из меморије без примене кеша
  - због неопходног проверавања да ли податак постоји у кешу или не

Писање у случају промашаја

- У случају промашаја подаци се уписују само у меморију, зато што не постоје у кешу

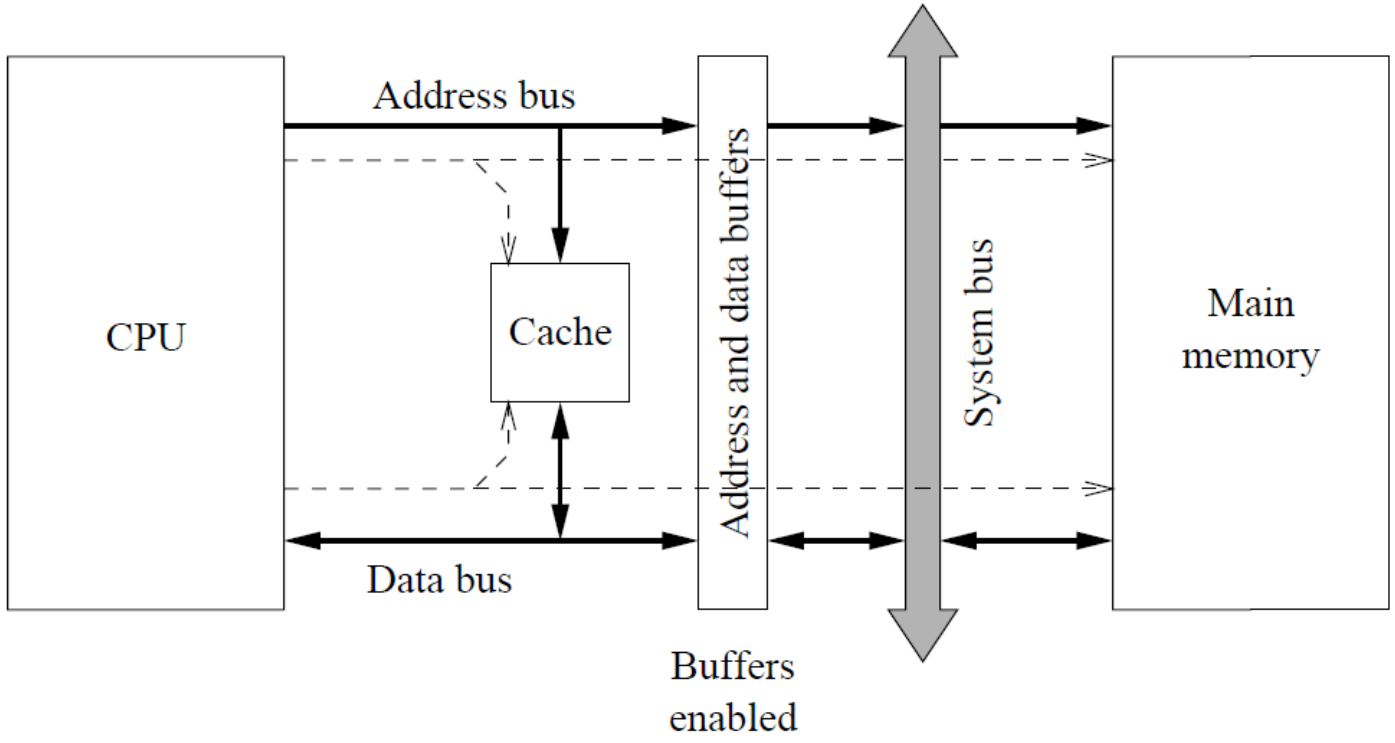


Писање у случају поготка

- У случају поготка постоје две основне могућности:
  - писање се обавља само у кеш или
  - писање се обавља и у кеш и у меморију

Писање у случају поготка (2)

- Случај писања и у меморију и у кеш



Зашто кеш ради?

- Практичан пример: увећавање свих елемената матрице (нпр. *double*) за *K*:

```
for(int i=0; i<M; i++)
   for(int j=0; j<N; j++)
      X[i][j] = X[i][j] + K;
```

Зашто кеш ради? (2)

- Први фактор за успешну примену кеша је *поновљена употреба* истих података или делова кода
  - Наредба у петљи се понавља *M*x*N* пута
  - Ако је наредба записана у кешу, њено извршавање је значајно убрзано зато што се чита из брзог кеша а не из споре меморије

Зашто кеш ради? (3)

- Други фактор је планско пуњење кеша подацима и инструкцијама пре него што их процесор затражи
  - Планско пуњење подиже перформансе из два разлога:
    - маскира кашњење спорије главне меморије
    - пуњење се одвија у блоковима, а пренос блокова података је вишеструко бржи него пренос појединачних података

Зашто кеш ради? (4)

- Посматрамо пример и разматрамо само читање, без писања
  - При читању податка *X*[0][0] долази до промашаја
    - Приступа се главној меморији
    - Податак се чита и истовремено уписује у кеш
    - Осим њега, у кеш се уписују и *X*[0][1], *X*[0][2] и *X*[0][3]
      - (претпостављамо да се ради са блоковима по 32 бајта)
  - Наредне три итерације се читају из кеша, без употребе главне меморије
  - Иако нема поновљене употребе, добитак у перформансама се остварује захваљујући успешном предвиђању који ће подаци бити потребни

Понашање програма

- У случају већине програма се испољава бар један од фактора за успешност кеша
  - понављање и/или
  - предвидиво приступање подацима

Принцип локалности

- Принцип локалност реферисања тврди да програми имају тенденцију да у неком датом периоду времена реферишу само неки подскуп података и инструкција и то често уз понављање

- Разликујемо
  - просторну локалност
  - и временску локалност

Просторна локалност

- Програми имају тенденцију да податке и инструкције користе секвенцијално
  - Локални подаци у функцијама су записани на стеку, блиско једни другима
  - Сложени подаци заузимају секвенцијалне области у меморији
  - Већину времена инструкције се читају и извршавају секвенцијално
    - скокови ремете секвенцијалност, али између два скока обично је више инструкција

Временска локалност

- Програми имају тенденцију да поновљено користе податке и инструкције у одређеним периодима времена
  - Типичан пример су петље
    - исте инструкције се извршавају више пута
    - исте локалне променљиве се користе више пута
    - неки делови сложених података се користе више пута