Увод у организацију и архитектуру рачунара 2

Александар Картељ

kartelj@matf.bg.ac.rs

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

Меморија

Карактеристике и типови

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
  - KiB, MiB, GiB, TiB
    - 1KiB = 1024B, 1MiB=1024^2B, 1GiB=1024^3B, 1TiB=1024^4B
  - 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

- **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

- 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

- 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

- 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 (прпелитање и сл.)

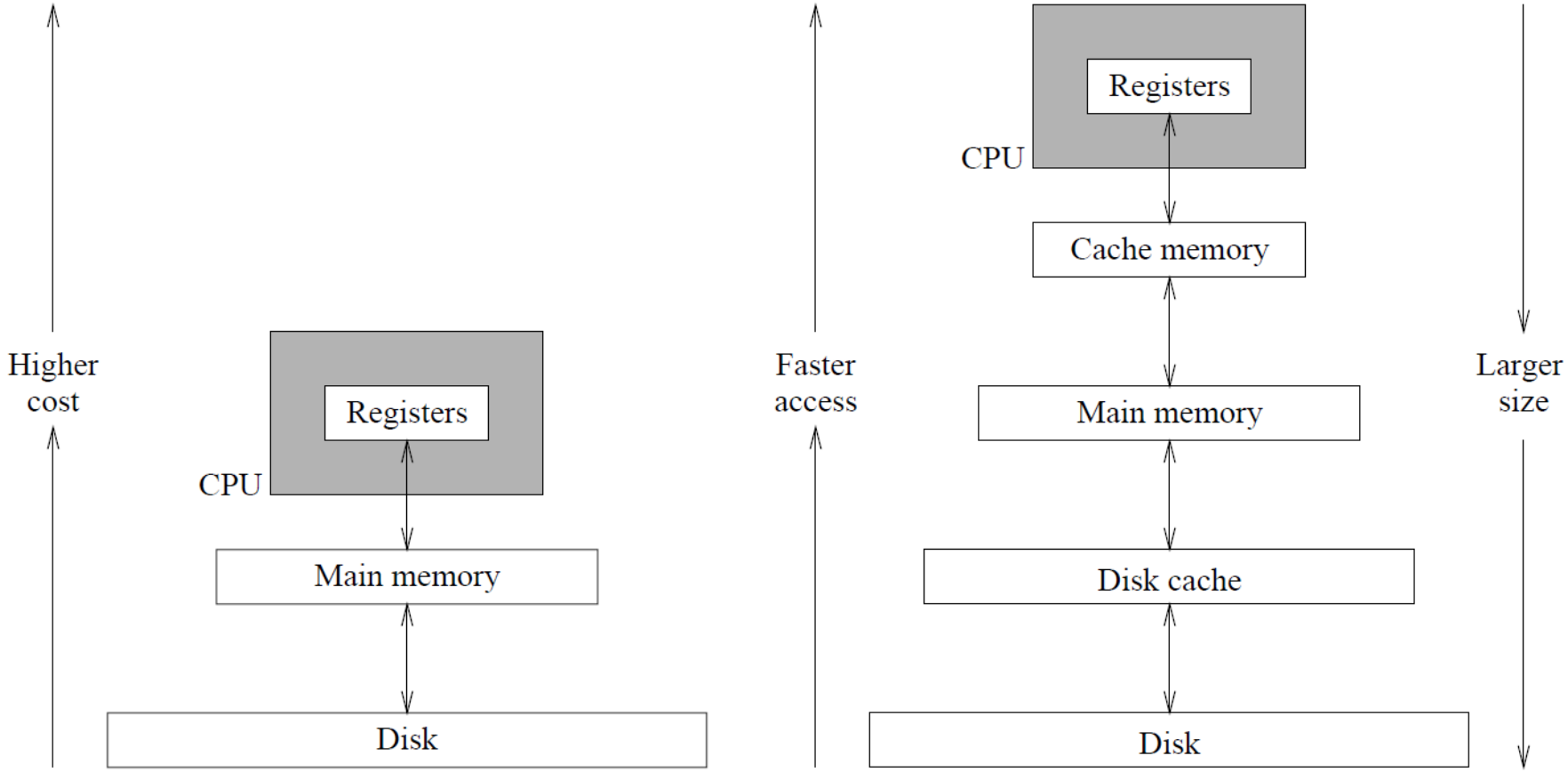
Могућност промене садржаја

- 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



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*

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

SDRAM

- Динамичкој меморији се додају компоненте израђене од статичке меморије
  - Ознака
    - *SDRAM*
  - Намена
    - кеш или
    - бафер

Неке врсте динамичког *RAM*-а

- *FPM* (*Fast Page Mode*)
- *EDO* (*Enhanced Data Out*)
- *BEDO* (*Burst EDO*)
- *ESDRAM, CDRAM* (*Enhanced SDRAM, Cache DRAM*)
- *JEDEC SDRAM* (*Enhanced SDRAM, Cache DRAM*)
- *DDR SDRAM* (*Double Data Rate SDRAM*)
- *SGRAM* (*Synchronous Graphics RAM*)
- *RDRAM* (*Rambus DRAM*)
- *SLDRAM* (*Synchronous Link DRAM*)

DDR SDRAM

- *DDR SDRAM* (*Double Data Rate SDRAM*)
  - Увео двоструку брзину рада меморије
  - Подаци се испоручују по два пута у циклусу
    - на узлазном рубу
    - на силазном рубу

DDR SDRAM

Standard name	Memory clock (MHz)	Cycle time[4] (ns)	I/O bus clock (MHz)	Data rate (MT/s)	V_{DDQ} (V)	Module name	Peak transfer rate (MB/s)	Timings (CL-tRCD-tRP)
DDR-200	100	10	100	200	2.5±0.2	PC-1600	1600	
DDR-266	133⅓	7.5	133⅓	266⅔		PC-2100	2133⅓	
DDR-333	166⅔	6	166⅔	333⅓		PC-2700	2666⅔	
DDR-400A DDR-400B DDR-400C	200	5	200	400	2.6±0.1	PC-3200	3200	2.5-3-3 3-3-3 3-4-4

DDR2 SDRAM

- *DDR2 SDRAM*
  - Подаци се испоручују по четири пута у циклусу

- Већа брзина преноса него код *DDR*
- Веће време приступа него код *DDR*

DDR2 SDRAM

Standard name	Memory clock (MHz)	Cycle time (ns)	I/O bus clock (MHz)	Data rate (MT/s)	Module name	Peak transfer rate (MB/s)	Timings[2][3] (CL-tRCD-tRP)	CAS latency (ns)
DDR2-400B DDR2-400C	100	10	200	400	PC2-3200	3200	3-3-3 4-4-4	15 20
DDR2-533B DDR2-533C	133⅓	7½	266⅔	533⅓	PC2-4200*	4266⅔	3-3-3 4-4-4	11¼ 15
DDR2-667C DDR2-667D	166⅔	6	333⅓	666⅔	PC2-5300*	5333⅓	4-4-4 5-5-5	12 15
DDR2-800C DDR2-800D DDR2-800E	200	5	400	800	PC2-6400	6400	4-4-4 5-5-5 6-6-6	10 12½ 15
DDR2-1066E DDR2-1066F	266⅔	3¾	533⅓	1066⅔	PC2-8500*	8533⅓	6-6-6 7-7-7	11¼ 13⅛

DDR3 SDRAM

- *DDR3 SDRAM*
  - Подаци се испоручују по осам пута у циклусу
- Већа брзина преноса него код *DDR*

DDR3 SDRAM

Standard name	Memory clock (MHz)	Cycle time (ns)	I/O bus clock (MHz)	Data rate (MT/s)	Module name	Peak transfer rate (MB/s)	Timings (CL-tRCD-tRP)	CAS latency (ns)
DDR3-800D DDR3-800E	100	10	400	800	PC3-6400	6400	5-5-5 6-6-6	12 1/2 15
DDR3-1066E DDR3-1066F DDR3-1066G	133⅓	7 1/2	533⅓	1066⅔	PC3-8500	8533⅓	6-6-6 7-7-7 8-8-8	11 1/4 13 1/8 15
DDR3-1333F* DDR3-1333G DDR3-1333H DDR3-1333J*	166⅔	6	666⅔	1333⅓	PC3-10600	10666⅔	7-7-7 8-8-8 9-9-9 10-10-10	10 1/2 12 13 1/2 15
DDR3-1600G* DDR3-1600H DDR3-1600J DDR3-1600K	200	5	800	1600	PC3-12800	12800	8-8-8 9-9-9 10-10-10 11-11-11	10 11 1/4 12 1/2 13 3/4
DDR3-1866J* DDR3-1866K DDR3-1866L DDR3-1866M*	233⅓	4 2/7	933⅓	1866⅔	PC3-14900	14933⅓	10-10-10 11-11-11 12-12-12 13-13-13	10 5/7 11 11/14 12 6/7 13 13/14
DDR3-2133K* DDR3-2133L DDR3-2133M DDR3-2133N*	266⅔	3 3/4	1066⅔	2133⅓	PC3-17000	17066⅔	11-11-11 12-12-12 13-13-13 14-14-14	10 5/16 11 1/4 12 3/16 13 1/8

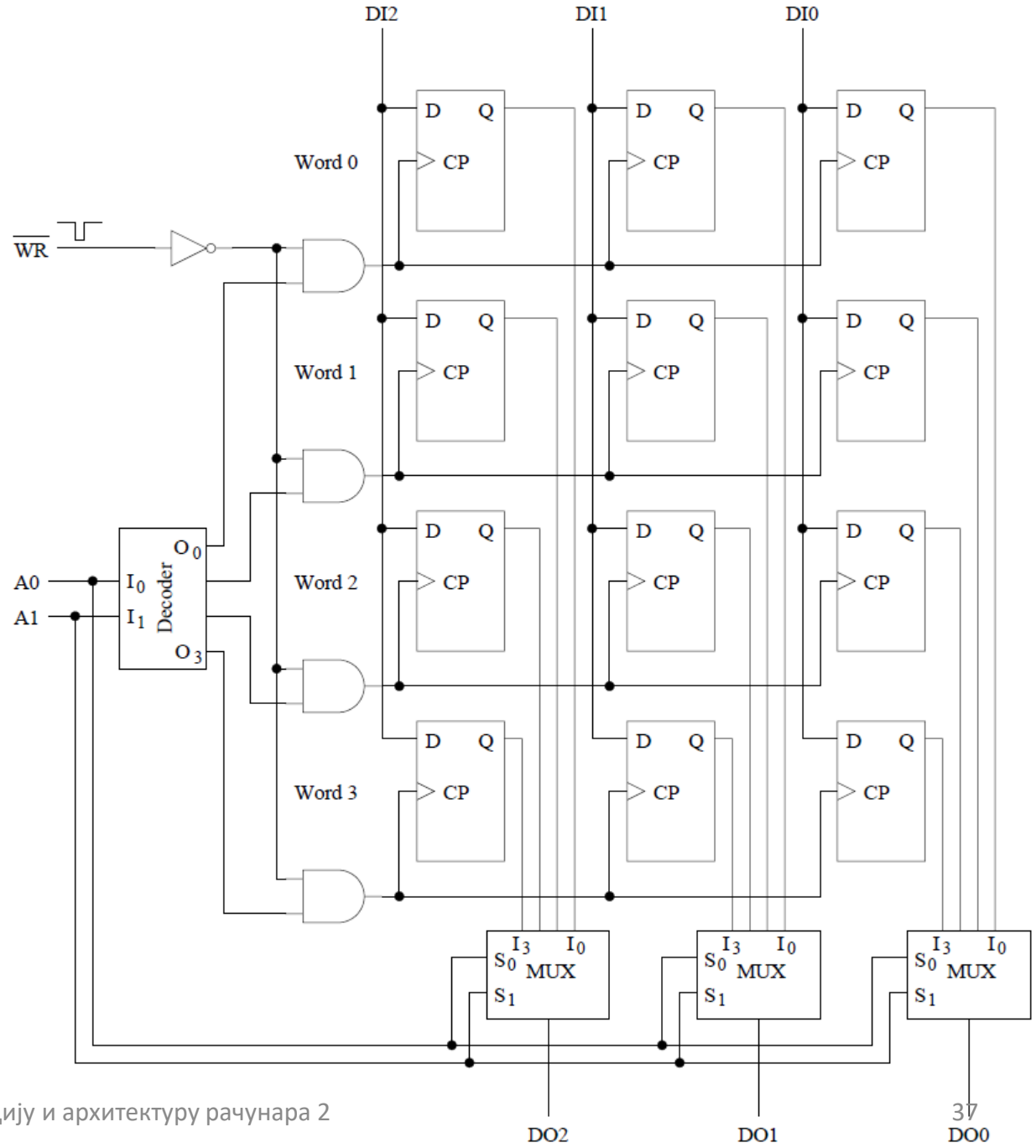
Меморија

Архитектура

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
- Не може да се користи за прављење већих меморија
  - Потребан је додатни селекторски улаз који означава да ли се блок користи или не

Повезивање на магистралу

- Потребно је узети у обзир
  - Исте линије се користе за улаз и за излаз
    - трансфер је двосмеран кроз исте линије
    - не могу се користити различите линије за улаз и излаз
  - Меморијска магистрала је дељена
    - само изабрани уређај сме стављати податке на магистралу података
    - остали се морају понашати као да нису повезани на магистралу

- Постоји више техника које се користе, а ми ћемо обрадити две:
  - Мултиплексоре
  - Бафере са три стања

Употреба мултиплексора

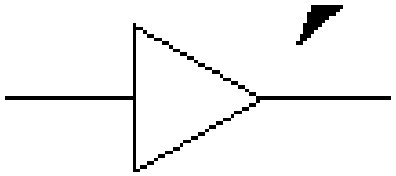
- Представљен пример почива на мултиплексорима
- Мултиплексори су проблематични јер:
  1. Не могу да се непосредно повежу улази и излази података!
  2. Не омогућавају прављење већих меморија, од више меморијских чипова, јер је потребно омогућити везу са CE (chip enable) сигналом!

Употреба бафера са три стања

- Уређаји са три стања имају 3 а не само 2 стања
  (за разлику од осталих представљаних реза и флип-флопова)
  - Имају додатни контролни сигнал
  - Ако је он активан, излаз је са високом импеданцом (активан) независно од улаза
- Данас уобичајено решење

Бафер

- Најпре да размотримо обичан “бафер”, који наизглед не служи ничему



- Међутим, он има функцију
  - Бафер је *активан* елемент

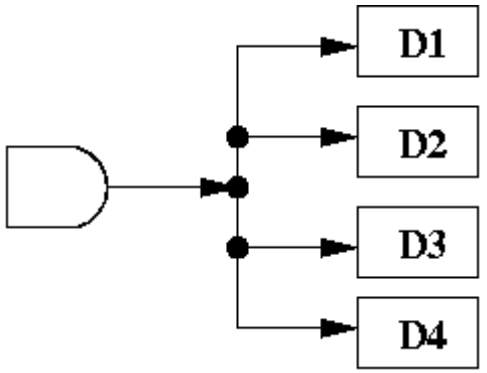
Бафер (2)

- Бафер је *активан* елемент
  - ако је на улазу *позитиван* потенцијал,
    он обезбеђује да је на излазу *пун* позитиван потенцијал
  - ако је на улазу *приближно нулти* потенцијал,
    он обезбеђује да је на излазу *нулти* потенцијал

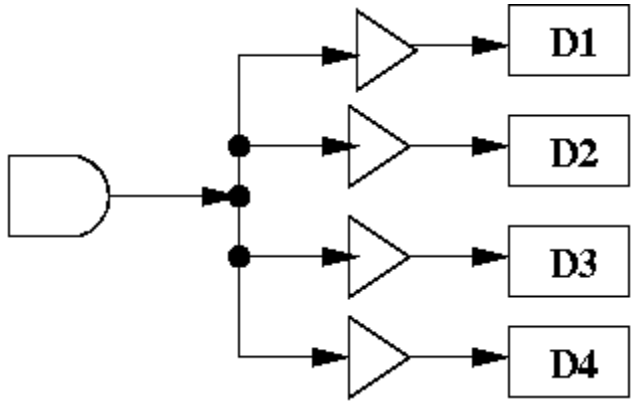
- Понаша се као појачавач сигнала
  - сваки потенцијал *изнад* прага функционисања транзистора појачава се до пуног интензитета позитивног потенцијала
  - сваки потенцијал *испод* прага функционисања транзистора “појачава” се до нултог потенцијала

Бафер (3)

- У овом примеру, сваки од елемената *D1-D3* ће добити свега по ¼ потребног потенцијала



- Употреба бафера обезбеђује да елементи *D1-D3* добију пун потребнан потенцијал



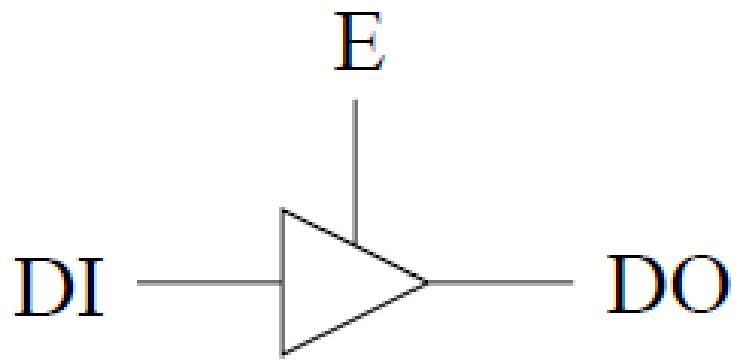
Три стања?

- До сада смо подразумевали да сваки елемент има за излаз једну од две вредности: 0 или 1
- У логичком систему то и јесте тако

- У имплементираним системима имамо потребу за трећим стањем:
  - 0 – на излазу се поставља нулти потенцијал (уземљење) и омогућава проток струје
  - 1 – на излазу се поставља позитиван потенцијал (напајање) и омогућава проток струје
  - **Z – не утиче се на стање на излазу и онемогућава се проток струје**

Бафер са три стања

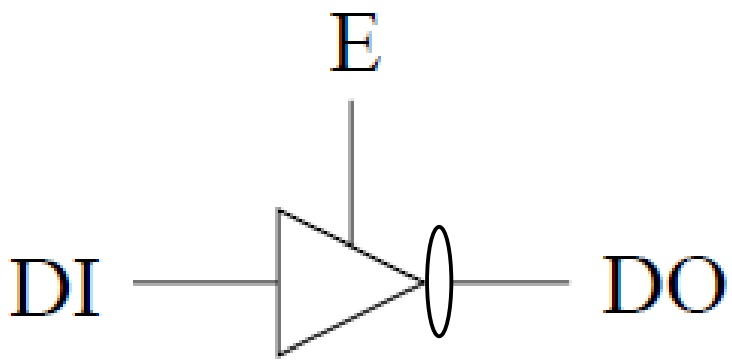
- Бафер са три стања се понаша попут **вентила**
- Ако се на “вентил” *E* који допушта проток (енгл. *enable*) доведе 0, онда се не утиче на стање на излазу
- Ако се на “вентил” *E* који допушта проток доведе 1, онда се сигнал са улаза *X* пропагира на излаз



Inputs		Output
E	DI	DO
1	0	0
1	1	1
0	X	Z

Инвертор са три стања

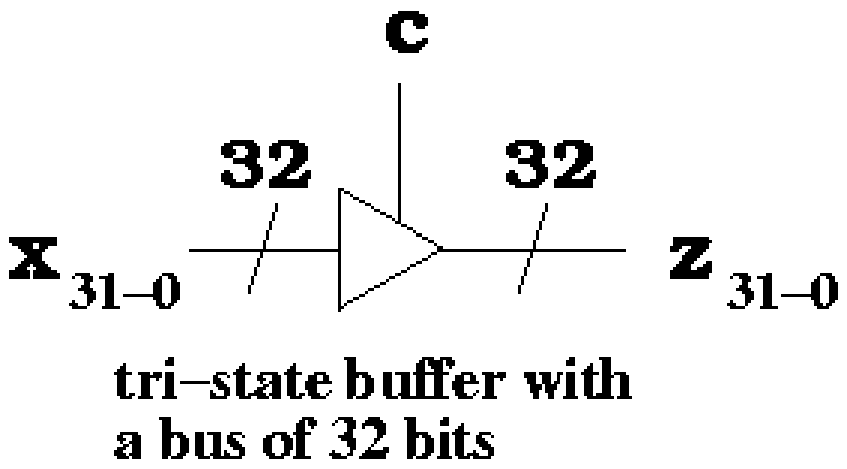
- Инвертор са три стања се понаша као негација са вентилом
- Ако се на “вентил” *E* који допушта проток (енгл. *enable*) доведе 0, онда се не утиче на стање на излазу
- Ако се на “вентил” *E* који допушта проток доведе 1, онда се сигнал са улаза *X* инвертује и пропагира на излаз



Inputs		Output
E	DI	DO
1	0	1
1	1	0
0	X	Z

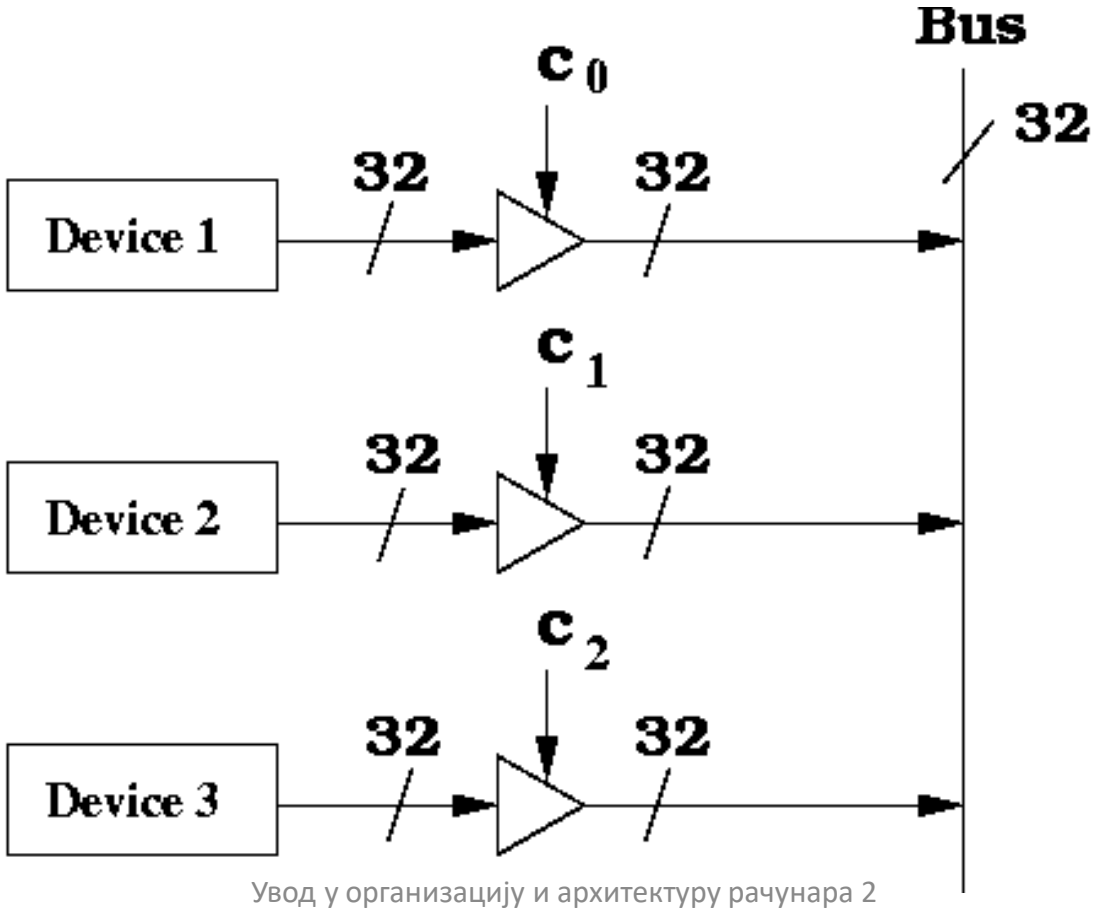
Бафери са три стања и магистрале

- На магистралу се преко бафера са три стања обично ставља истовремено већи број битова
  - у зависности од ширине магистрале
- Ради једноставнијег представљања често се користи поједностављен симбол за представљање низа бафера са три стања:



Бафери са три стања и магистрале (2)

- Пример везивања уређаја на магистралу
  - ради се о излазним подацима уређаја

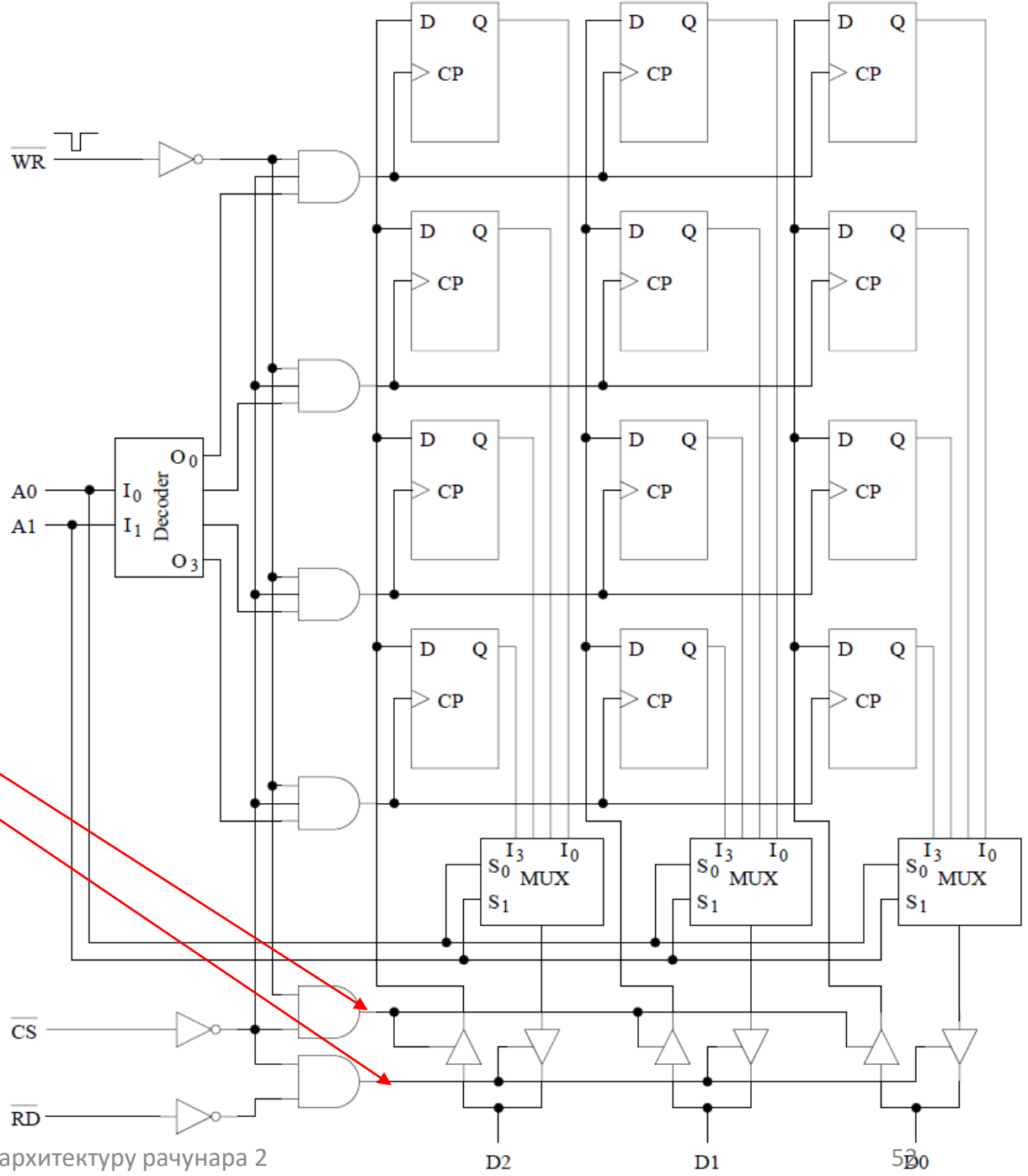


Меморијски блок

- Потребне промене у односу на претходни дизајн:
  - Додајемо селекторски улаз
    - улазни сигнал који одређује да ли се меморијски блок употребљава или не
    - повезује се као улаз на конјункције за избор адресе
  - Спајамо улазне и излазне сигнале података
    - ако се пише, онда помоћу бафера са три стања усмеравамо податке са магистрале на улазе флип-флопова
    - ако се чита, онда помоћу бафера са три стања усмеравамо излазе из флип-флопова на магистралу

Меморија 4x3 *D* флип-флопа употребом бафера са три стања

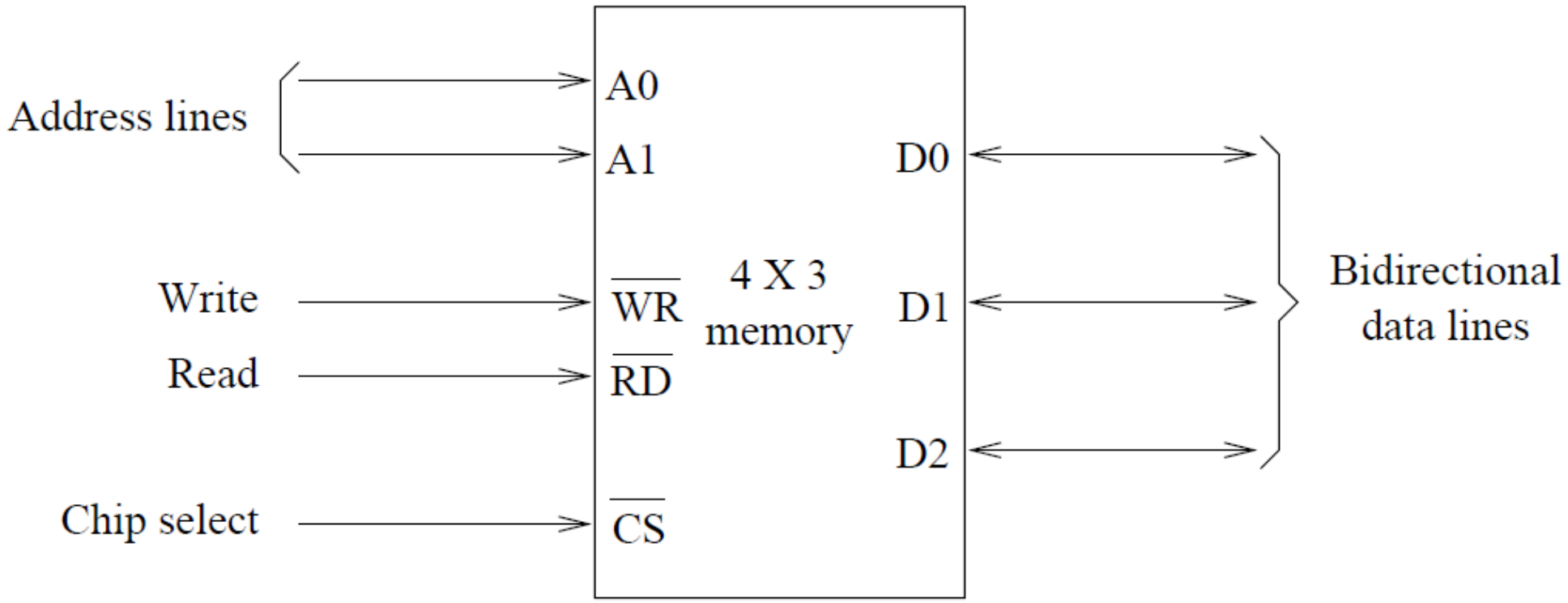
- CS’ у конјукцији са WR’ или RD’ се такође везује као вентил за бафер са три стања
- Ако чип није одабран, излаз ка магистрали ће бити у стању високе импеданце, па неће реметити рад других уређаја на магистрали
- Кад су CS’=0 и WR’=0, биће активан смер писања
- Када су CS’=0 и RD’=0, биће активан смер читања



Меморијски блок (2)

- Да резимирамо:
- Потребан је додатни контролни сигнал који означава операцију читања
- Селекторски улаз укључује/искључује контролне сигнале за читање и писање
- Баферима са три стања се
  - сигнал са магистрале података пропушта до улаза на флип-флопове, акко су активни и селекторски сигнал и контролни сигнал операције читања
  - сигнал са излаза флип-флопова се пропушта на магистралу података, акко су активни и селекторски сигнал и контролни сигнал операције писања
  - искључени бафер (контролни сигнал 0) има високу импеданцу и не представља сметњу функционисању укључених бафера са истим излазом/улазом

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)

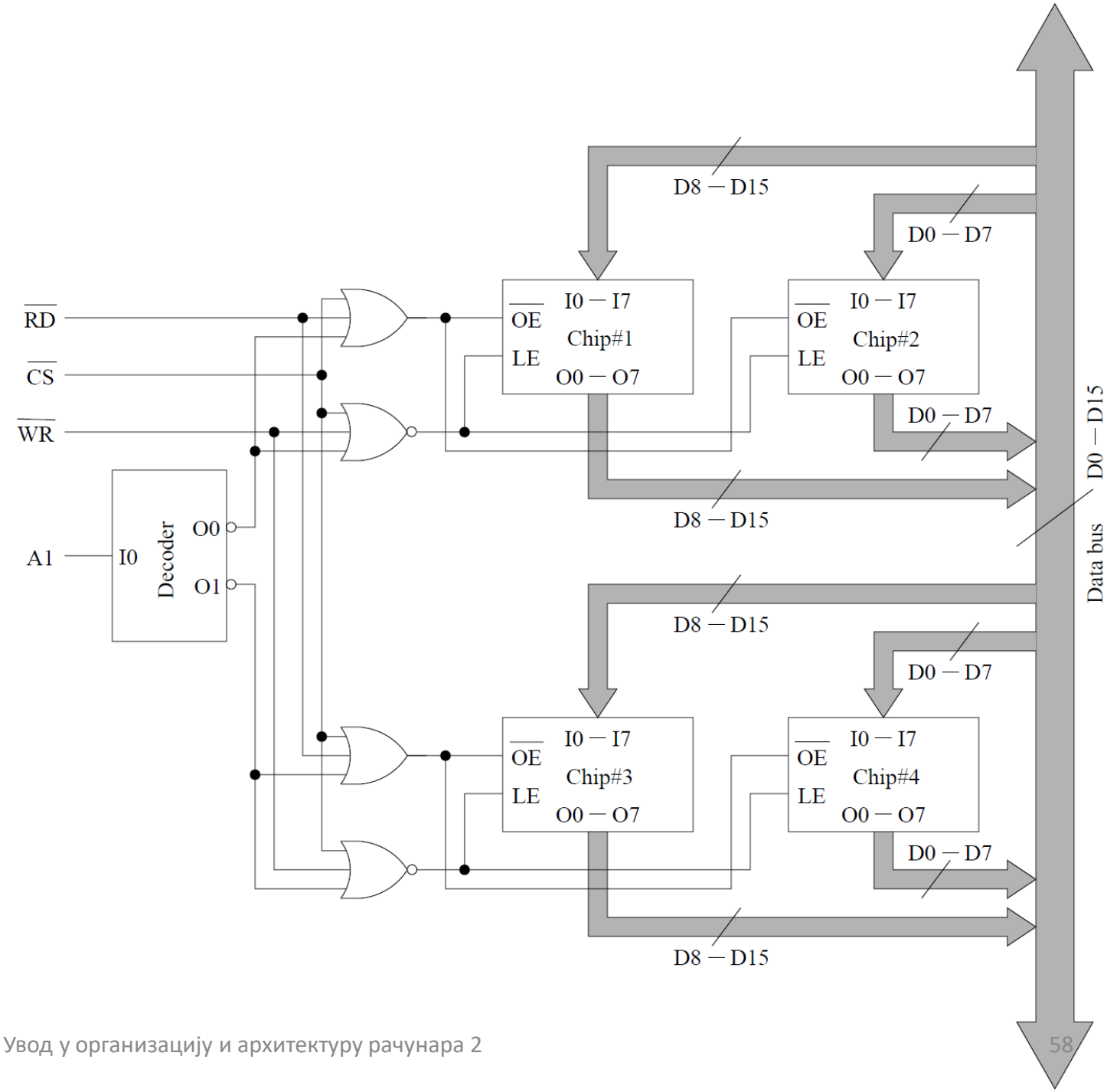
- Помоћу 4 чипа димензија 1x8 може се направити меморијски блок 2x16
  - један чип може да чува 8 бита, па се користи матрица од 2x2 чипа

- Повезивањем више чипова повећава се ширина меморијске речи (*хоризонтална експанзија*)
  - контролни улази два 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
    - нпр. од 8 чипова се може добити 64-битна реч

Прављење већих меморија (3)

- Додавањем редова повећавамо величину меморије (*вертикална експанзија*)
  - сваки ред чува по једну реч
  - помоћу декодера се врши одабир активног реда
  - на контролне сигнале излаза чипова (*OE*) се везује конјункција
    - контролног селектора, контролног сигнала читања, излаза декодера
    - због инвертованог улаза (*OE#*) уместо конјункције се примењује дисјункција инвертованих улаза
  - на контролне сигнале улаза чипова (*LE*) се везује конјункција
    - контролног селектора, контролног сигнала писања, излаза декодера
    - због већ инвертованих аргумената, уместо конјункције се примењује дисјункција инвертованих улаза са инвертором на излазу

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*

Организација чипова

- Иста количина меморије се може различито организовати и имплементирати
  - у време уских магистрала (8-16 битова) *DRAM* се израђивао у ширини 1 бита
    - данас то није практично због велике ширине речи
- Предност широких чипова је што их је потребно мање за веће меморије
  - *Pentium* је 32-битни процесор са 64-битном магистралом података
  - меморија 16M x 64 може се направити
    - од једног реда са 4 чипа 16M x 16
    - али не и од 8 чипова 32M x 8 (добијамо 32M x 64)
  - од уских чипова се ни не могу добити неке мање меморијске јединице

Дизајн већих меморија

- На примеру *DRAM*-а

- Основно питање је да ли је меморијски адресни простор (*memory address space – MAS*) адресибилан на нивоу појединачних бајтова или не
  - дужина адресибилне речи већине савремених процесора јесте 1 бајт

Дизајн већих меморија (2)

- Затим се одлучује о конфигурацији чипова
  - при изабраној циљној величини и величини чипова, не мења се укупан број чипова, али се мења њихов распоред
  - 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*Mib*
  - циљна конфигурација је 64*Mi* x 32*b*
  - користимо чипове 16*Mi* x 16*b*
  - матрица чипова је 4 x 2

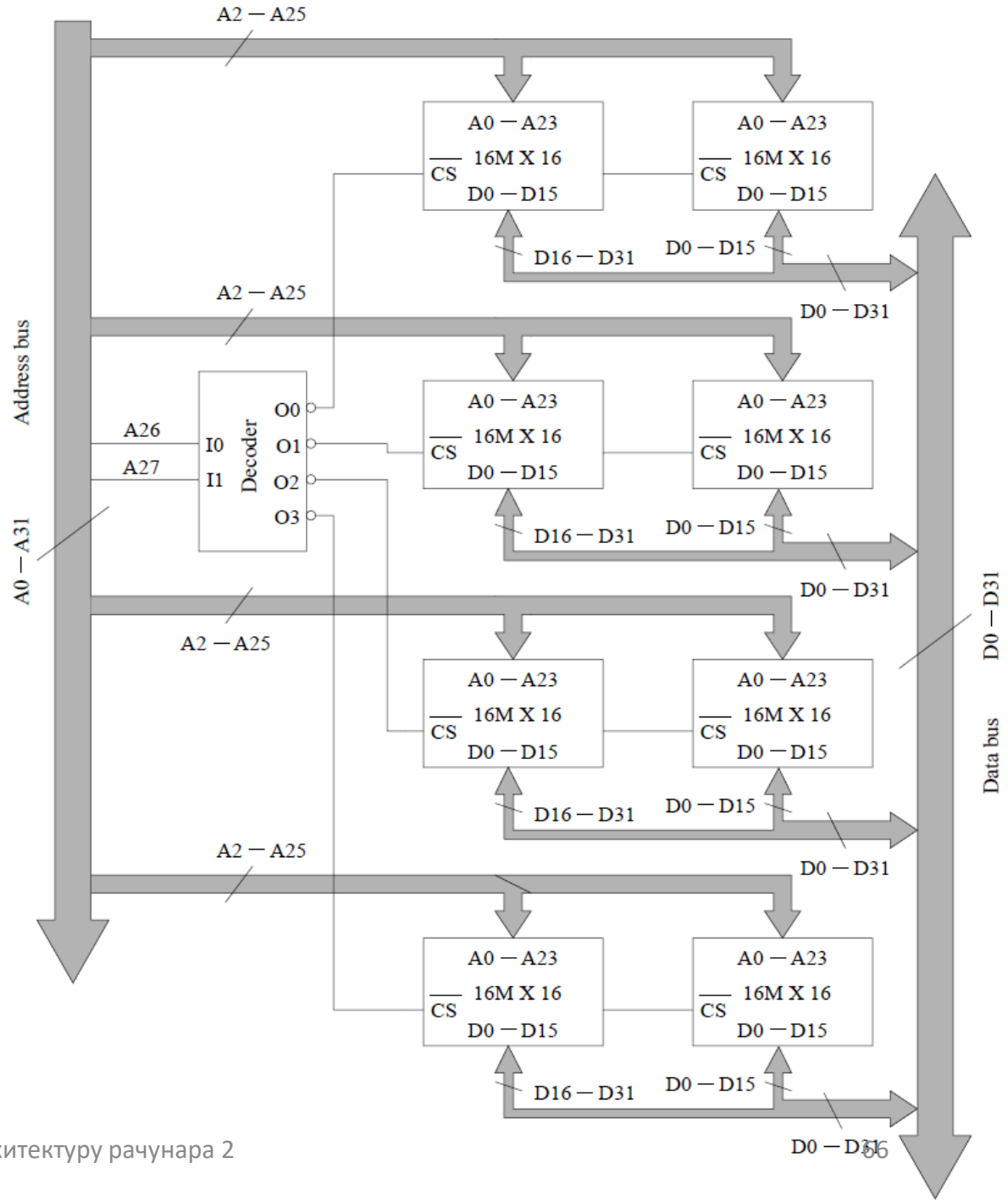
Дизајн већих меморија (3)

- 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, јер декодер приступа речи
    - ширина меморије је 32 бита (4 бајта), 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 (4)

- 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



Пресликавање адреса

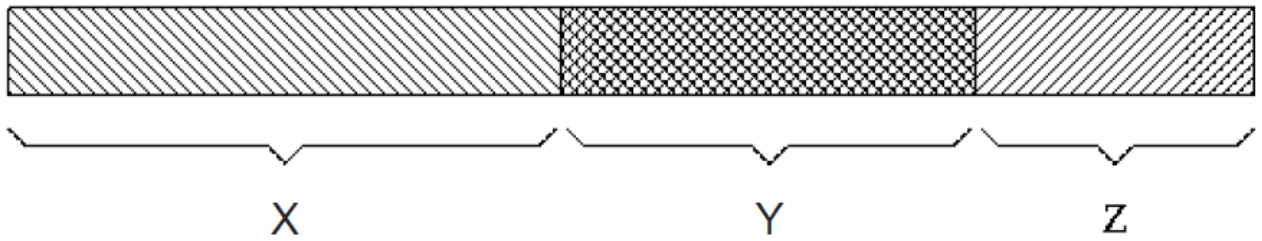
- Пресликавање адреса је поступак којим се физичка меморија лоцира у адресном простору рачунара

- На пример:
  - процесор *Pentium* има адресни простор величине 4*GiB*
    - 32 адресне линије
  - ако рачунар има 128*MiB* меморије, она се може пресликати у различите адресибилне области

- Пресликавање може бити
  - пуно и
  - делимично

Врсте адресних линија

- Адресне линије се деле у три групе:
  - *X* – највише адресне линије које одређују чип
  - *Y* –адресне линије које се прослеђују чиповима
  - *Z* – најниже адресне линије које се занемарују



Пуно пресликавање адреса

- *Пуно пресликавање адреса* је оно пресликавање ког кога је функција пресликавања меморијских адреса у меморијске локације 1-1
  - за сваку меморијску локацију постоји највише једна адреса која јој одговара
- Све адресне линије *X* се користе при декодирању ради добијања сигнала за избор модула
- Све адресне линије се деле у две групе:
  - линије *Y* и *Z* одређују бајт у меморијском модулу
  - линије *X* се користе за израчунавање сигнала за избор чипа *CS* (*chip selector*)

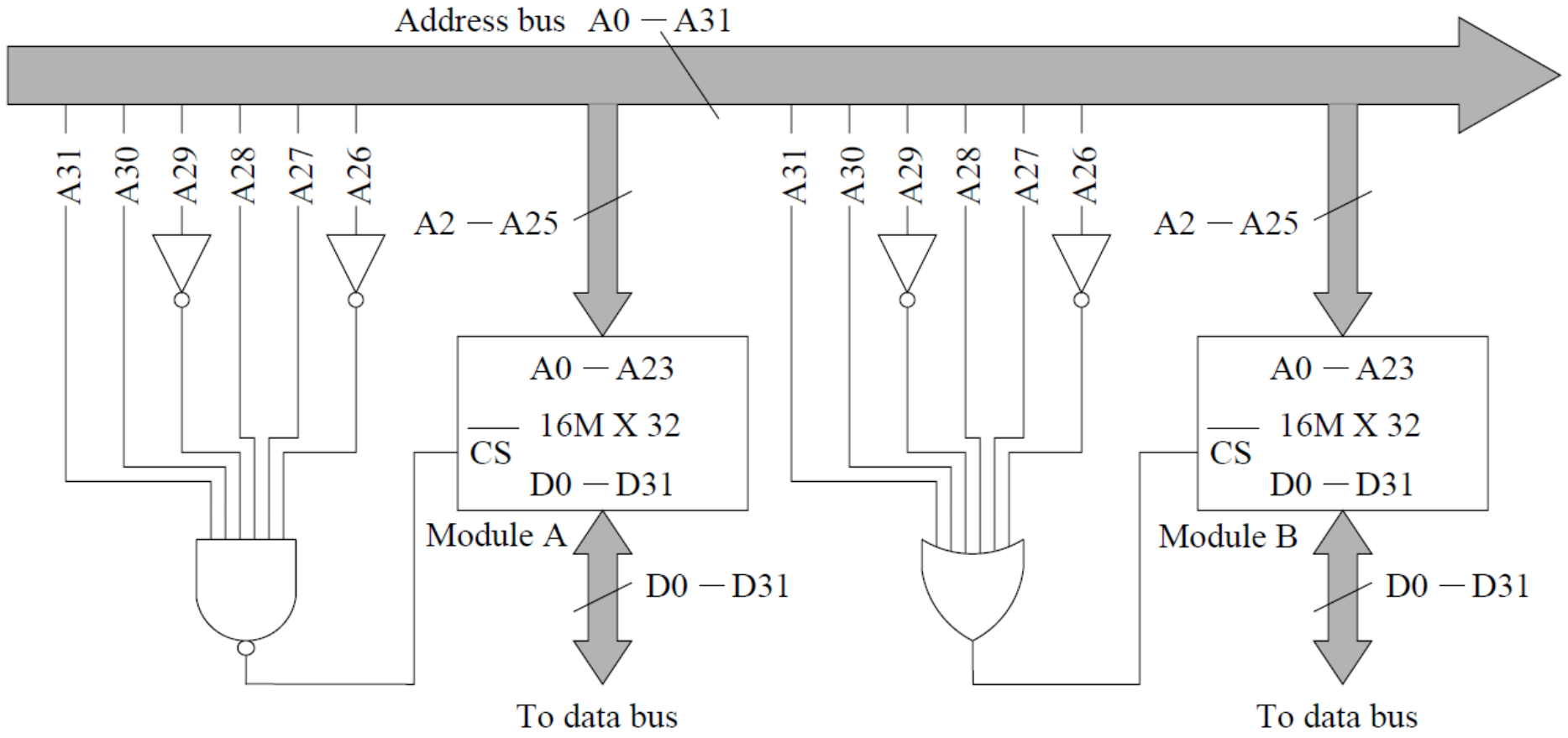
Пример пуног пресликавања адреса

- Нпр., користимо 2 модула 16*Mi* x 32 у адресном простору од 4*GiB*

- Делимо 32 адресне линије на две групе:
  - нижих 26 линија (*Y, Z*) се користе за одређивање бајта у оквиру модула 16*Mi* x 32*b*
  - виших 6 линија (*X*) се користе за одређивање сигнала *CS*
    - модулу *A* одговарају (на пример) адресе *X* = 110110
      - опсег адреса: D8000000H – DBFFFFFFH
    - модулу *B* одговарају (на пример) адресе *X* = 001001
      - опсег адреса: 24000000H – 27FFFFFFH
    - остале вредности адресе *X* не одговарају ниједном модулу
    - опсези се не преклапају, па је ово пуно пресликавање

(не представљамо контролне линије, ради прегледности)

Пример пуног пресликавања адреса



Делимично пресликавање адреса

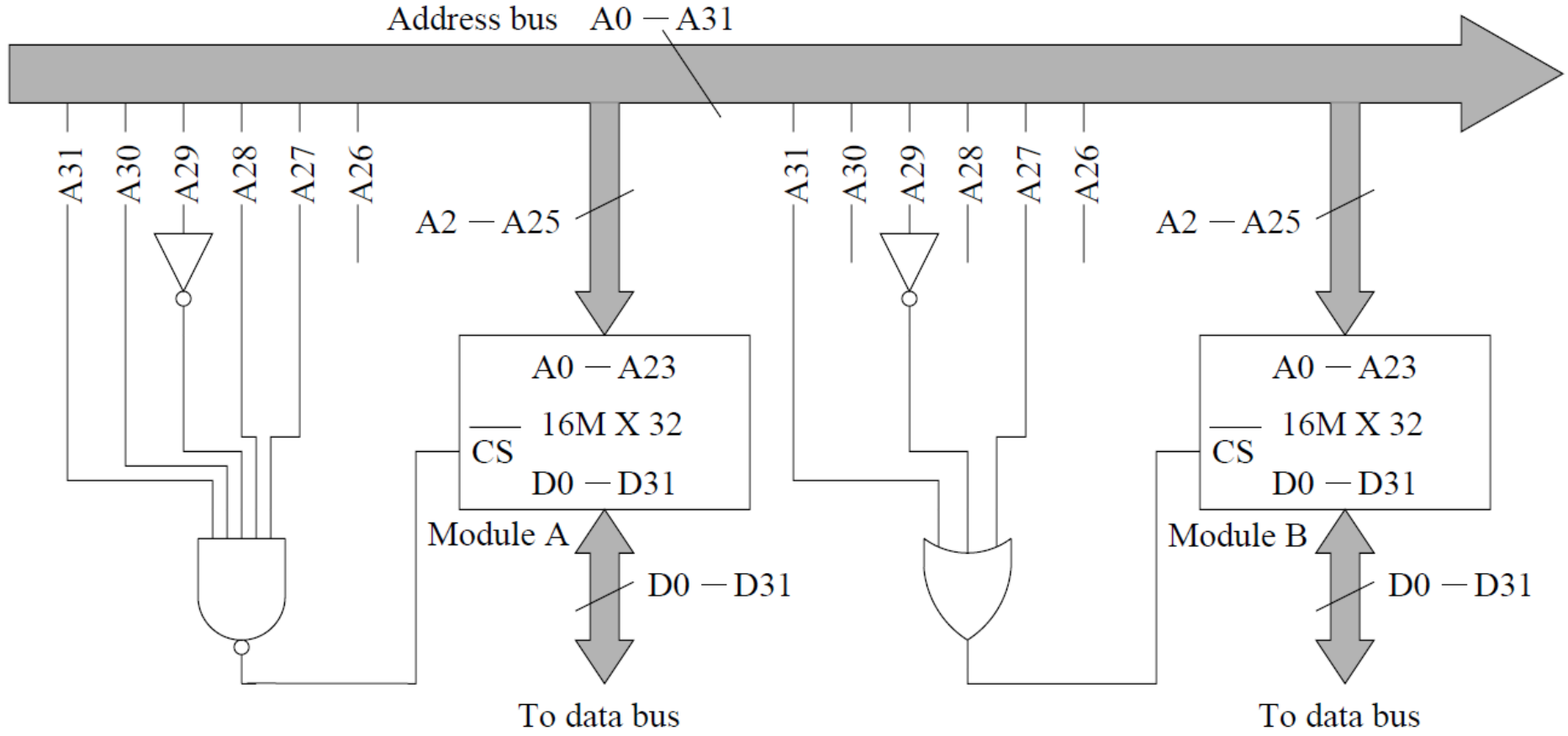
- *Делимично пресликавање адреса* је оно ког кога функција пресликавања меморијских адреса у меморијске локације није 1-1
  - неким меморијским локацијама може да одговара више адреса
- Циљ је поједностављивање логике одређивања селекторског сигнала
- Може се примењивати када је број меморијских локација значајно мањи од броја адресибилних локација

Пример делимичног пресликавања адреса

- Слично претходном примеру

- Модулима се додељују вишеструке адресе
  - модулу *A* одговарају (на пример) адресе *X* = 110110 и *X* = 110111
    - скраћено *X* = 11011*d*
    - двоструки опсег адреса:
      - D8000000H – DBFFFFFFH
      - DC000000H – DFFFFFFFH
  - модулу *B* одговарају (на пример) адресе *X* = 1*d*0*d*1*d*
  - опсези се преклапају, па је ово делимично пресликавање

Пример делимичног пресликавања адреса (2)

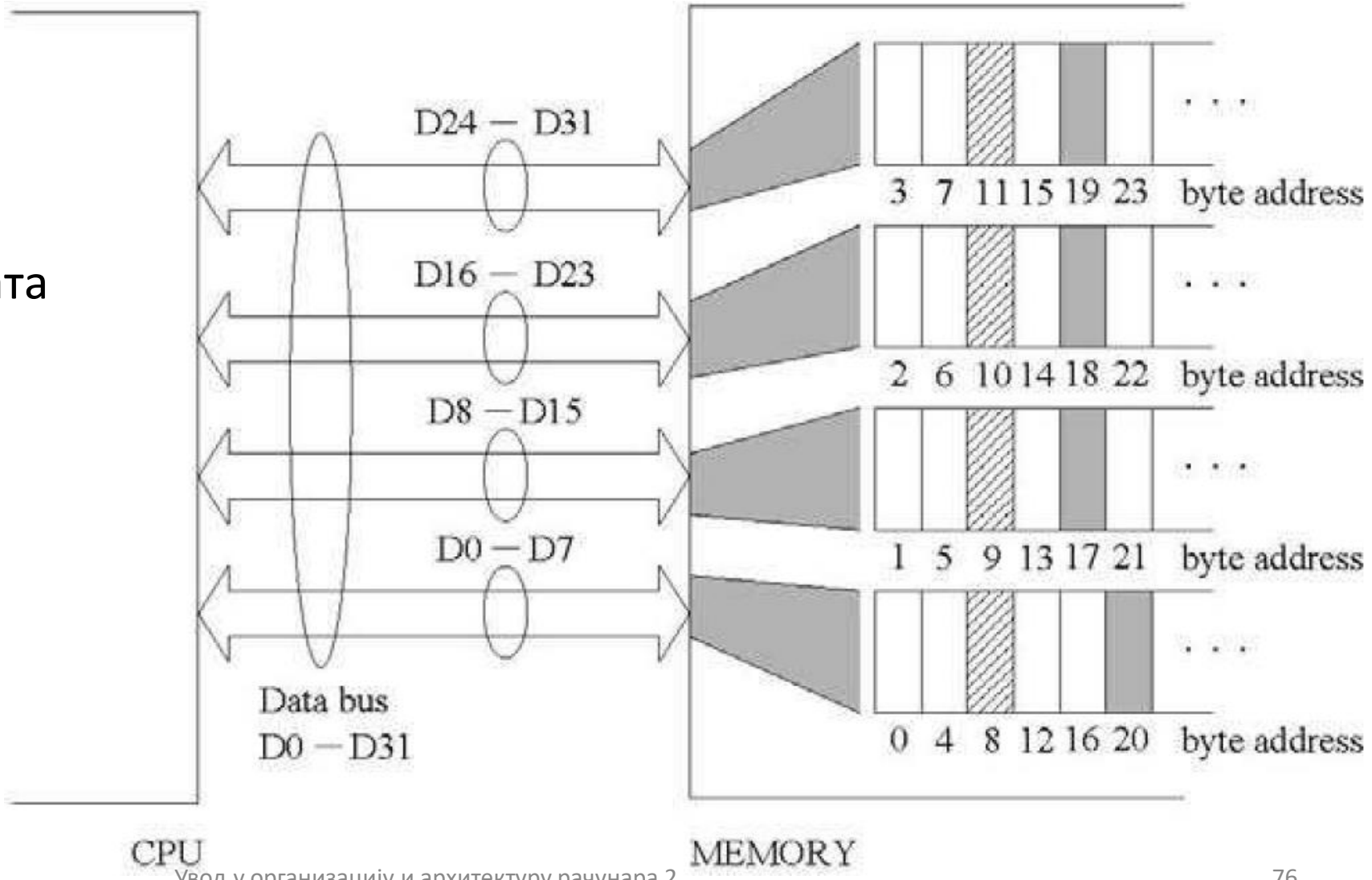


Поравнавање података

- Ако имплементација меморије почива на 32-битним модулима, онда
  - ако је адреса поравната са 32-битним речима, читање 32-битне речи се одвија у једном циклусу
    - модул може вратити целу 32-битну реч одједанпут
  - ако адреса није поравната са 32-битним речима, читање 32-битне речи се одвија у два циклуса
    - модул не може вратити целу 32-битну реч одједанпут

Пример поравнатих и непоравнатих адреса

- Адреса 8 је поравната
  - 32-битни податак се чита у једном циклусу
- Адреса 17 није поравната
  - 32-битни податак се чита у два циклуса

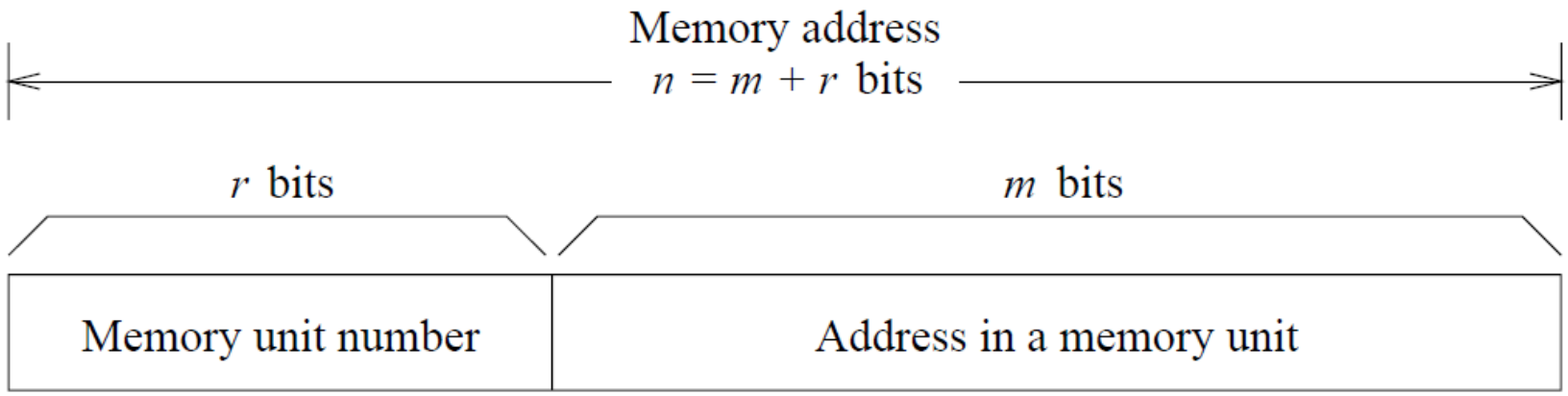


Испреплетане меморије

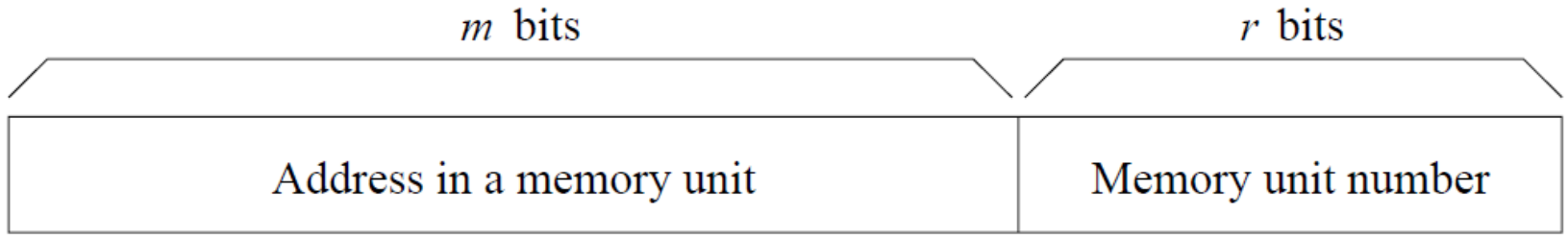
- Као што смо видели, уобичајено је да се виших *r* адресних линија употребљавају за препознавање модула,
  а нижих *m* за адресирање у модулу

- Код испреплетаних меморија то се мења,
  како би се узастопне речи налазиле у различитим модулима

Испреплетане меморије (2)



(a) Normal memory address mapping



(b) Interleaved memory address mapping

Испреплетане меморије (3)

- Раније представљен дизајн меморије омогућава њено једноставно повећавање
- Сваки захтев се извршава током више циклуса (нпр. 4)
- Ако бисмо желели да прочитамо 8 узастопних речи, то би захтевало 8 x 4 = 32 циклуса
- Испреплетане меморије омогућавају да се скрати време читања узастопних речи

Испреплетане меморије (4)

- Меморијски модули се у контексту преплитања називају *меморијским банкама*
- Адресе се додељују банкама наизменично:
  - нека имамо *B* банака
  - банка се одређује као *addr MOD B*
- Имплементирају се на два начина:
  - синхронизованим приступом и
  - независним приступом

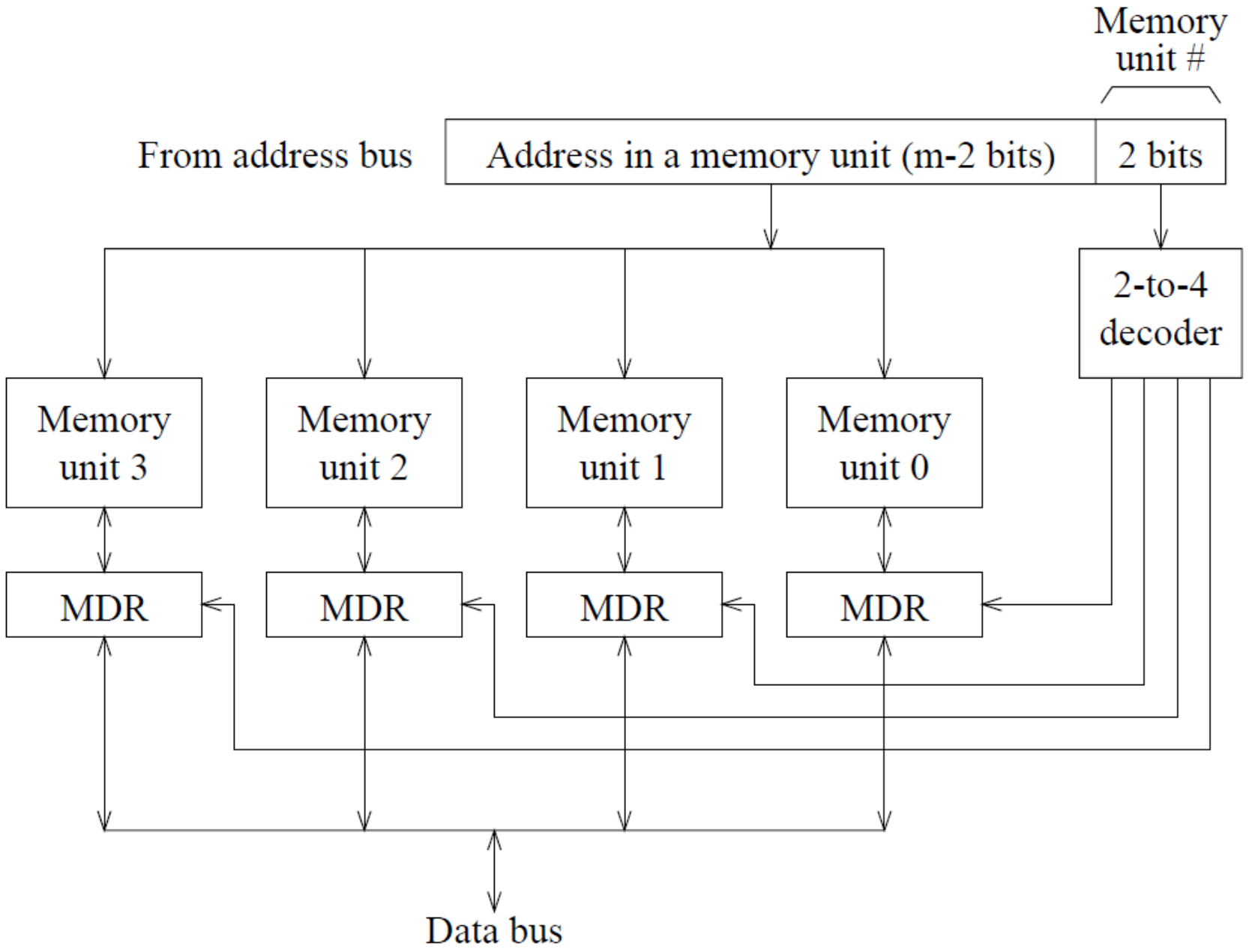
Синхронизован приступ

- Горњих *r* адресних линија се симултано доводе свим модулима
  - Све банке истовремено започињу своје операције
  - Након 4 циклуса, свака од меморија је довршила читање
    - (претпоставимо да читање захтева 4 циклуса)
- Прочитани подаци се уписују у четири меморијска регистра (*MDR – memory data register*)
  - имплементирани помоћу бафера са три стања

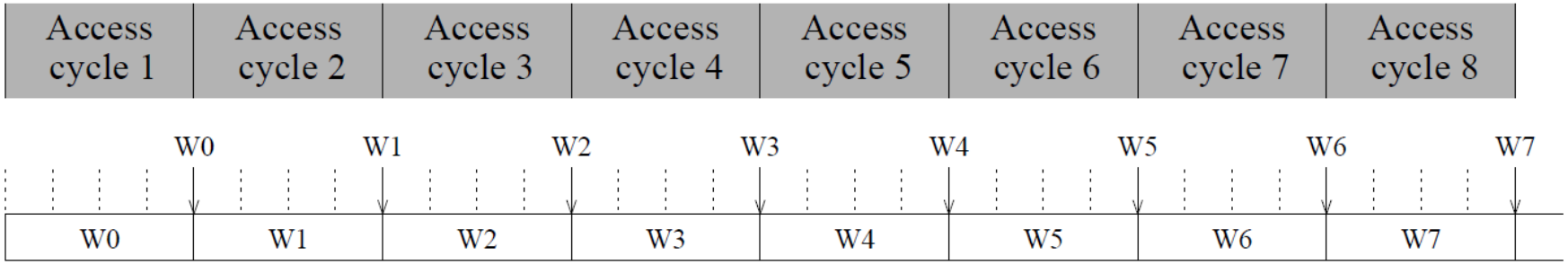
Синхронизован приступ (2)

- Током наредних *B* циклуса се ови подаци преносе на магистралу
  - декодером се бирају редом регистри
- Док се ови подаци преносе, наредних *B* речи се читају из меморије
- За читање првих *B* речи је потребно 4 циклуса
- Свака следећа реч захтева мање

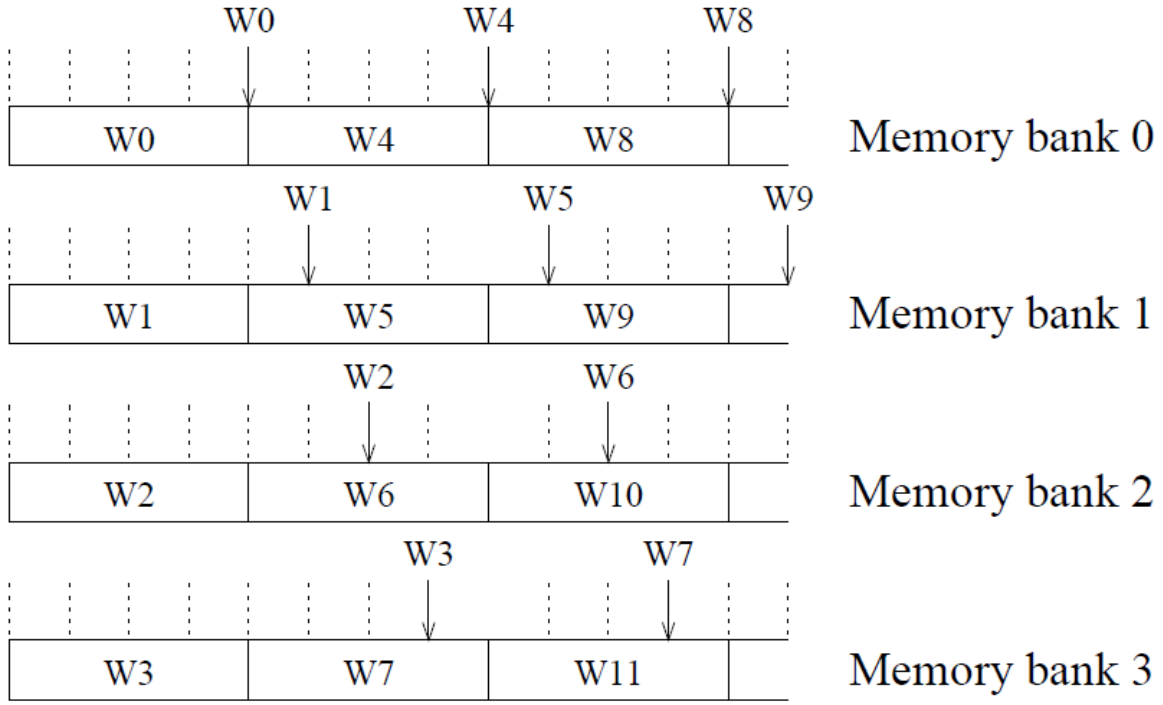
Испреплетана меморије са 4 банке синхронизован приступ



Без преплитања



Са преплитањем



- За читање 8 речи
- Без преплитања
  - 32 циклуса
- Са преплитањем
  - 11 циклуса

- За читање 12 речи
- Без преплитања
  - 48 циклуса
- Са преплитањем
  - 15 циклуса

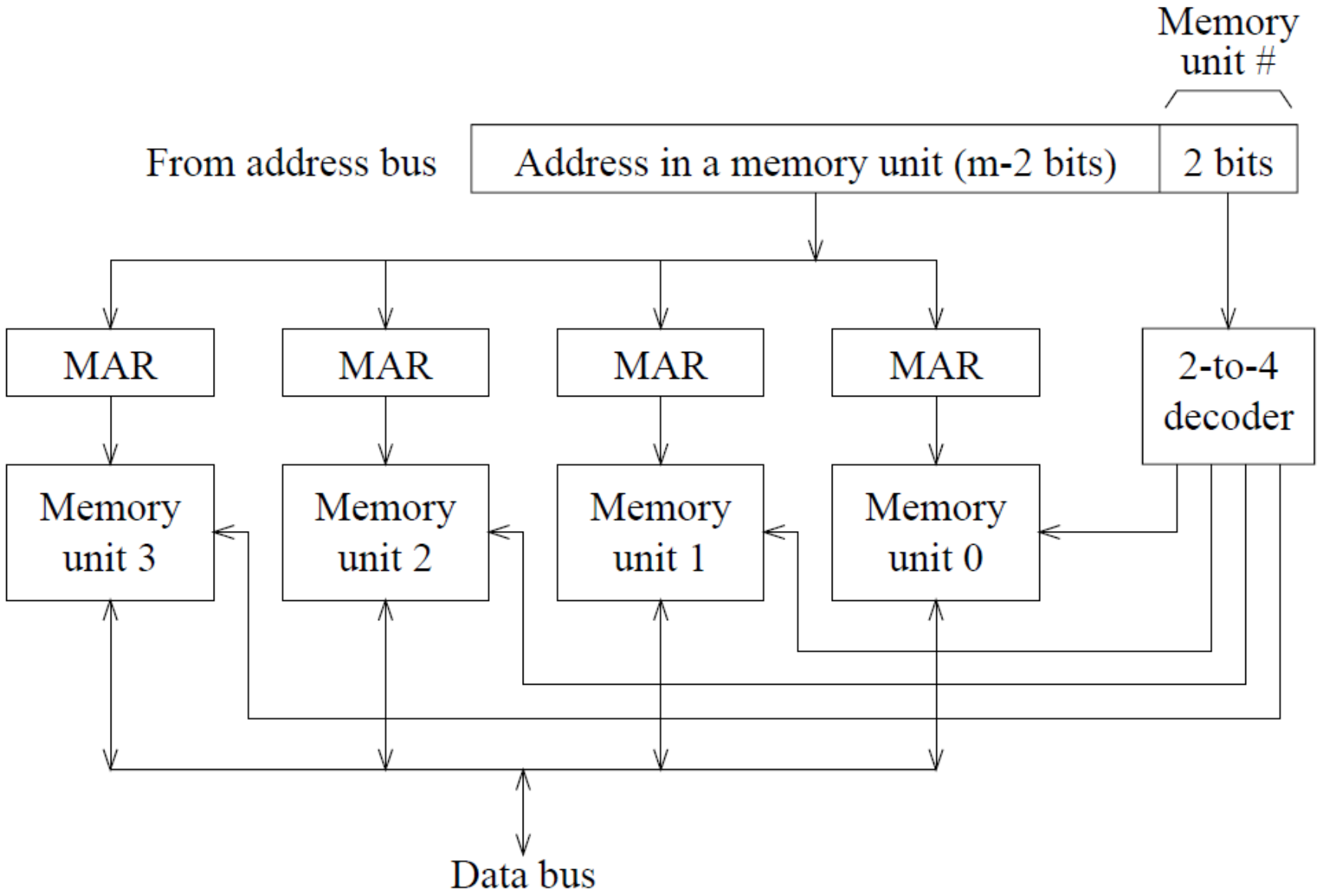
Независан приступ

- Проблем са синхронизованим приступом је да доноси убрзања само у случају секвенцијалног приступа
- Независан приступ омогућава преклопљено (*pipelined*) извршавање операција на произвољним адресама
- Свакој банци се додаје меморијски регистар адресе (*MAR*) који чува адресу коју користи та банка
- Нису потребни додатни регистри података (*MDR*), већ се подаци читају непосредно из меморије

Независан приступ (2)

- Независан приступ има исти ниво преклапања као и синхронизован
  - преклапање се постиже на нивоу адреса, а не на нивоу података
  - у сваком циклусу по једна адреса иде у одговарајући *MAR*
  - током 4 циклуса се чита податак и испоручује на магистралу

Испреплетана меморије са 4 банке независан приступ



Број банака

- Број банака би требало да буде бар једнак броју циклуса
  - Због тога је претходно синхронизован приступ илустрован примером са 4 банке и 4 циклуса

Проблеми у вези преплитања

- Преплитање захтева сложенију имплементацију меморијских кола
  - додатни регистри (података или адресе)
  - додатна контролна кола за постављање података у регистре и читање из њих
  - слаба толеранција грешака
    - ако је једна банка у кварy, читава меморија не ради
  - умањена флексибилност повећавања меморије