Увод у организацију и архитектуру рачунара 2

Александар Картељ

kartelj@matf.bg.ac.rs

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

Улазно/излазни уређаји

Основни концепти

Улазно/излазни уређаји

- Рачунарски систем обично има више различитих улазних и излазних уређаја
  - називају се и *периферни* уређаји, зато што се налазе на периферији рачунарског система
- Обезбеђују две основне функције
  - комуникацију рачунарског система са спољашњим светом и
  - чување података

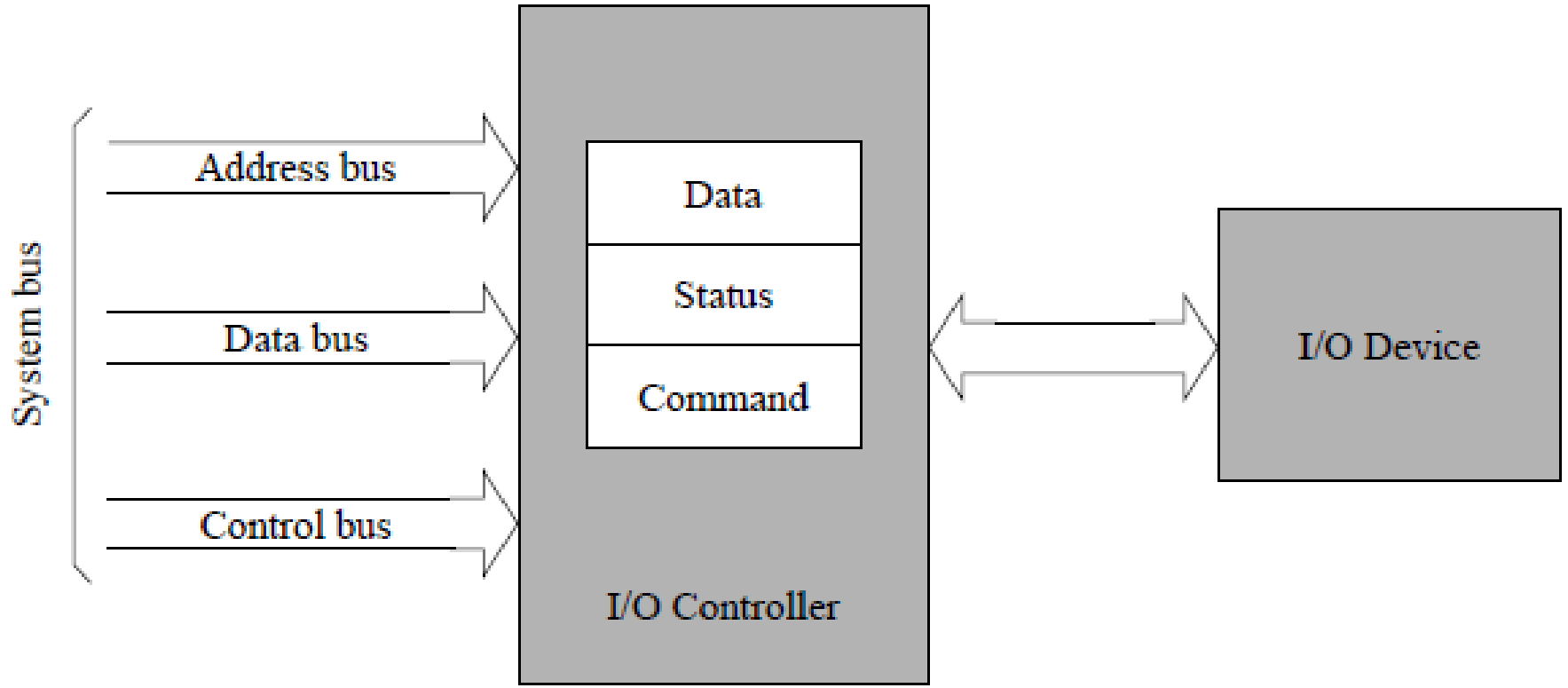
Принципи рада

- Независно од врсте уређаја, принципи рада улазно/излазних уређаја су исти:
  - сви У/И уређаји се на системску магистралу повезују посредством одговарајућег У/И контролера
  - први разлог је у различитости уређаја
    - различити уређаји имају различите протоколе комуникације
    - уместо да процесор и системска магистрала “уче” како да комуницирају са различитм врстама уређаја, те специфичности се препуштају контролерима
    - контролери имају улогу *моста* између центра и периферије
  - други је у техничким ограничењима
    - магистрала ради на високим фрквенцијама
    - да се не би сувише грејала, ради под врло ниским напонима
    - низак напон и висока фреквенција могу да функционишу без сметњи само на врло кратким растојањима (неколико *cm*)
    - УИ уређаји захтевају дуже каблове, јачи напон и нижу фреквенцију

У/И контролери

- Процесор се никада не обраћа непосредно уређајима, већ само одговарајућим контролерима
- У/И контролери имају улогу *моста* између центра (процесор, меморија и системска магистрала) и периферије (периферни уређаји рачунарског система)
- Контролери уобичајено имају три врсте интерних регистара:
  - регистар података
  - командни регистар
  - статусни регистар

Блок-дијаграм уопштеног У/И контролера



Улазно/излазни уређаји

Начини употребе, пример рада

Употреба У/И уређаја

- Два основна начина употребе У/И уређаја су
  - пресликавање портова у меморију
  - изоловани У/И

Пресликавање портова у меморију

- Сви У/И портови се пресликавају у меморијски адресни простор
- Није потребан никакав посебан интерфејс процесора
- Процесор користи уређај као меморију
- Сваки процесор може употребљавати уређаје путем пресликавања портова у меморију
- Неки процесори подржавају само овакав начин рада:
  - *PowerPC, MIPS*

Изоловани У/И

- Изоловани У/И подразумева посебан У/И адресни простор, независан од меморијског адресног простора
- *Intel x86* процесори подржавају овакав начин рада
- Системи засновани на процесорима који подржавају изоловани У/И омогућавају избор метода
  - нпр: штампачи се користе путем изолованог У/И,
    а графички подсистем путем пресликавања у меморију

Приступ портовима (*x86*)

- Наредно излагање је на примеру процесора *Intel x86*
- Адресни простор за изоловани У/И је 64*KiB*
  - подржани су 8-битни, 16-битни и 32-битни портови али само док свеукупно не прелазе 64*KiB* адресног простора
    - 64Ki 8-битних портова
    - 32Ki 16-битних портова
    - 16Ki 32-битних портова
    - различите комбинације
  - изоловани У/И адресни простор је линеаран – не подлеже ни сегментацији ни страничењу

Приступ портовима (*x86*) (2)

- Постоје два типа инструкција
  - регистарске
    - за преношење појединачних података (8, 16 или 32 бита) између регистара и портова
    - *in* – чита податак са улазног порта
    - *out* – уписује податак на излазном порту
  - блоковске
    - за преношење блокова података између меморије и портова
    - *ins* – чита блок података са улазног порта
    - *outs* – уписује блок података на излазном порту

Пример – тастатура

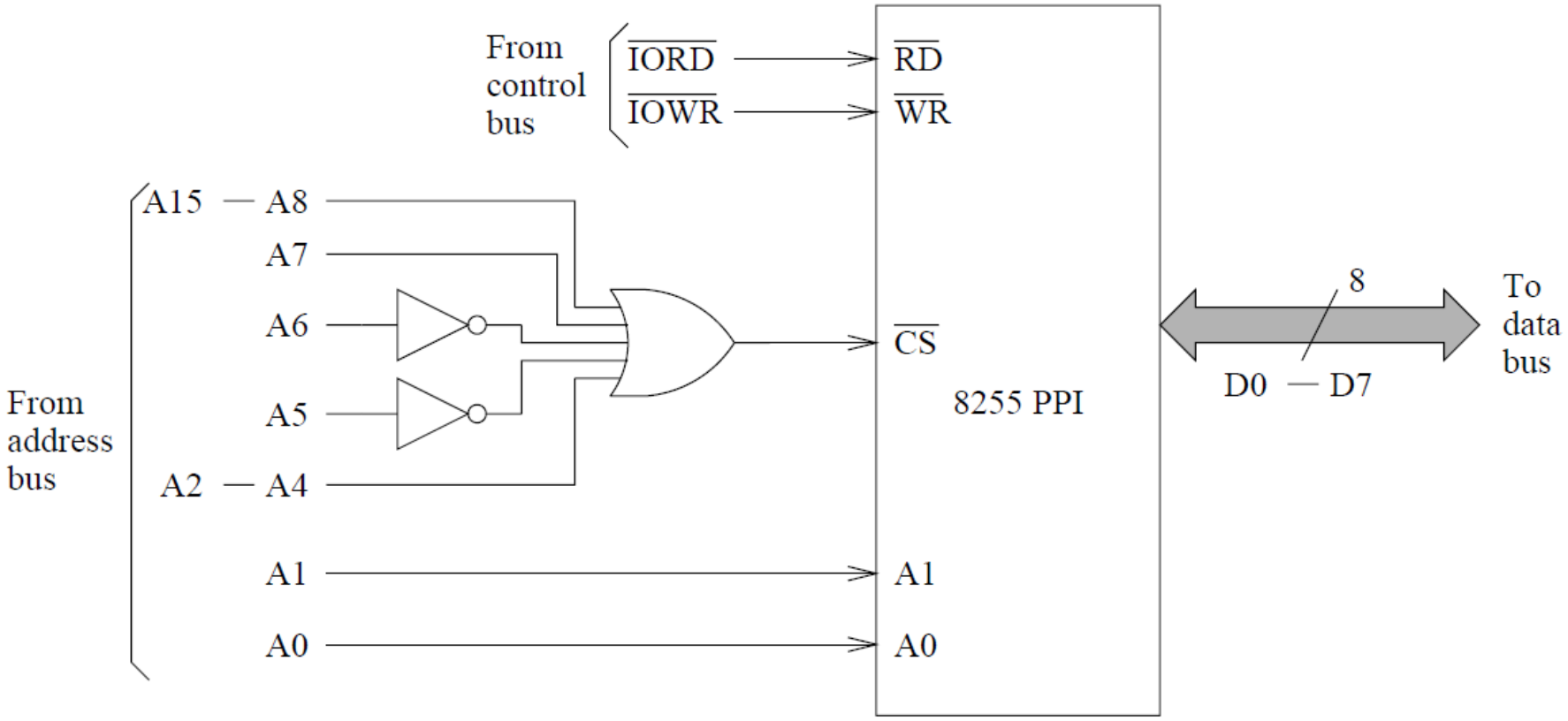
- Контролер тастатуре испитује тастатуру и извештава о притискању и отпуштању тастера у виду тзв. *кодова тастера* (енгл. *scan code*)
  - код тастера је идентификациони број који се додељује тастеру на основу његове локације
  - улазни потпрограми преводе код тастера у одговарајуће кодове карактера

Пример – тастатура (2)

- Као контролер се (на пример) може користити чип са следећим карактеристикама:
  - чип располаже са три 8-битна регистра (*PA, PB, PC*)
  - контролер испоручује код тастера путем регистра *PA*
    - битови *PA0-PA6* чине код тастера
    - бит *PA7* означава притисак (0) или отпуштање (1) тастера
  - нека су, нпр., ови регистри адресирани од адресе 60*H*:

8255 register	Port address
PA (input port)	60H
PB (output port)	61H
PC (input port)	62H
Command register	63H

Пример – тастатура (3)



- Пресликавање магистрала адресе и података је слично као у случају меморијских модула
- Основна разлика је у томе што се користе контролни сигнали за рад са У/И портовима *IORD* и *IOWR* уместо сигнала за рад са меморијом

  (нису приказане везе контролера и тастатуре)

Пренос података

- Пренос података између “система” и УИ уређаја подразумева размену података између меморије (или регистара) и УИ уређаја
- Пренос података чине две фазе:
  - фаза преноса података
  - фаза обавештавања о крају
- Неки аутори додају као посебну фазу
  - иницијализацију преноса података

Фаза преноса података

- Током ове фазе се преносе подаци између меморије и УИ уређаја
- Пренос се остварује путем
  - *програмираног У/И* или
  - *непосредног приступа меморији* (енгл. *direct memory access - DMA*)

Фаза обавештавања о крају

- Током ове фазе процесор се информише да је пренос података довршен

- Информисање се остварује путем
  - *система прекида* или
  - *програмираног У/И*

Улазно/излазни уређаји

Програмирани УИ и DMA

Пренос података (2)

- Три значајне технике:
  - програмирани У/И
  - непосредан приступ меморији (*DMA*)
  - система прекида

- Уобичајено:
  - ако се подаци преносе путем *DMA*, онда се обавештавање о крају одвија путем система прекида
  - ако се подаци преносе путем програмираног У/И, онда се и обавештавање о крају остварује истим путем

Илустрација техника

- На примеру радника и шефа:
  - програмирани У/И
    - шеф задаје посао
    - “непрестано” проверава да ли је посао довршен
    - када је довршен, шеф узима резултат рада и ради даље
  - непосредан приступ меморији (*DMA*) / систем прекида
    - шеф задаје посао раднику и наставља са својим послом
    - радник самостално обавља посао
    - када заврши посао, радник обавештава шефа о завршетку
    - шеф реагује на обавештење и затим наставља са својим послом

Програмирани У/И

- У основи програмираног У/И је петља у којој се чека да се доврши задати посао да би се могло наставити са радом
  - чекање се изводи кроз вишеструко проверавање (узорковање, тестирање) да ли је уређај достигао очекивано стање

Програмирани У/И – пример

- Пример тастатуре:
  - очитава се регистар *PA*
  - ако бит 7 има вредност 1, значи да није притиснут тастер, па се понавља претходни корак
  - ако бит 7 има вредност 0, значи да је притиснут тастер и прекида се петља
  - затим се прочитани код тастера може даље употребљавати, тј. преводити у код карактера

Програмирани У/И – пример (2)

- Исечак програмског кода:

```
        ...
key_up_loop:
        in    AL,60H
        test  AL,80H
        jnz   key_up_loop
        ...
```

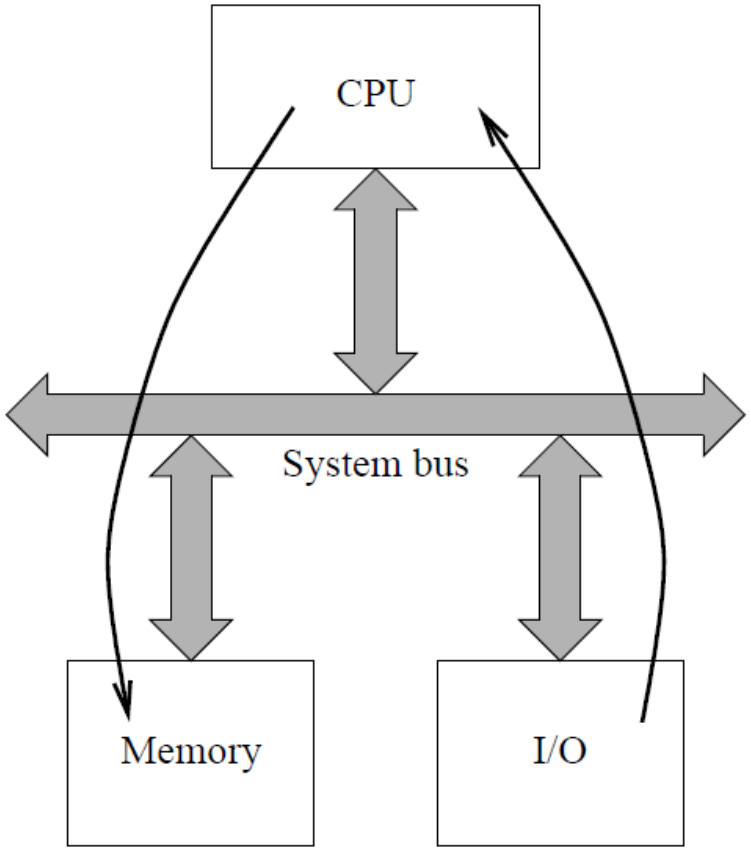
DMA

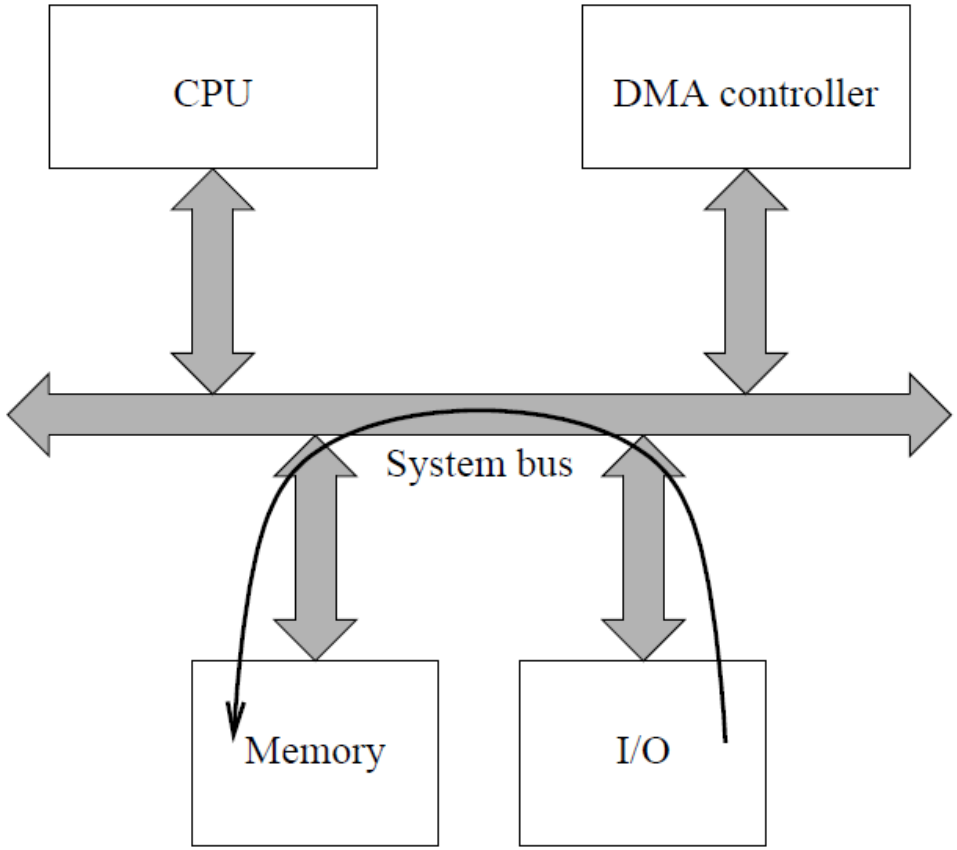
- У случају уређаја који преносе веће количине података или раде временски захтевне послове, програмирани У/И води великом утрошку времена на чекање
- *DMA* има за циљ да се процесор ослободи старања о преносу података и посвети другим стварима

DMA (2)

- *DMA* се имплементира помоћу контролера *DMA*
  - Контролер се понаша као подређен уређај у односу на процесор и прима инструкције за пренос података од процесора
  - Затим преузима контролу над магистралом и остварује пренос података
- Контролер уобичајено подржава већи број уређаја
  - сваки се повезује на посебан канал *DMA*

Однос програмираног У/И и непосредног приступа меморији





Кораци операције *DMA*

1. Иницијализација канала

2. Преношење података

3. Обавештавање процесора

Кораци операције *DMA*

1. Иницијализација канала:

- процесор иницира контролер *DMA* и шаље му:
  - број уређаја
  - адресу простора у меморији
  - број бајтова који се преносе
  - смер преношења података
- након иницијализације канал је спреман за преношење података

2. Преношење података

3. Обавештавање процесора

Кораци операције *DMA*

1. Иницијализација канала:

2. Преношење података

- када У/И уређај буде спреман за преношење података, обавештава о томе контролер *DMA*.
- контролер започиње операцију преноса:
  - контролер захтева магистралу (и добија је уобичајеним поступком арбитраже)
  - поставља меморисјку адресу и одговарајући сигнал (читање или писање) на магистралу
  - довршава пренос и ослобађа магистралу
  - ажурира адресу и бројач
  - ако има још података за преношење, понавља поступак

3. Обавештавање процесора

Кораци операције *DMA*

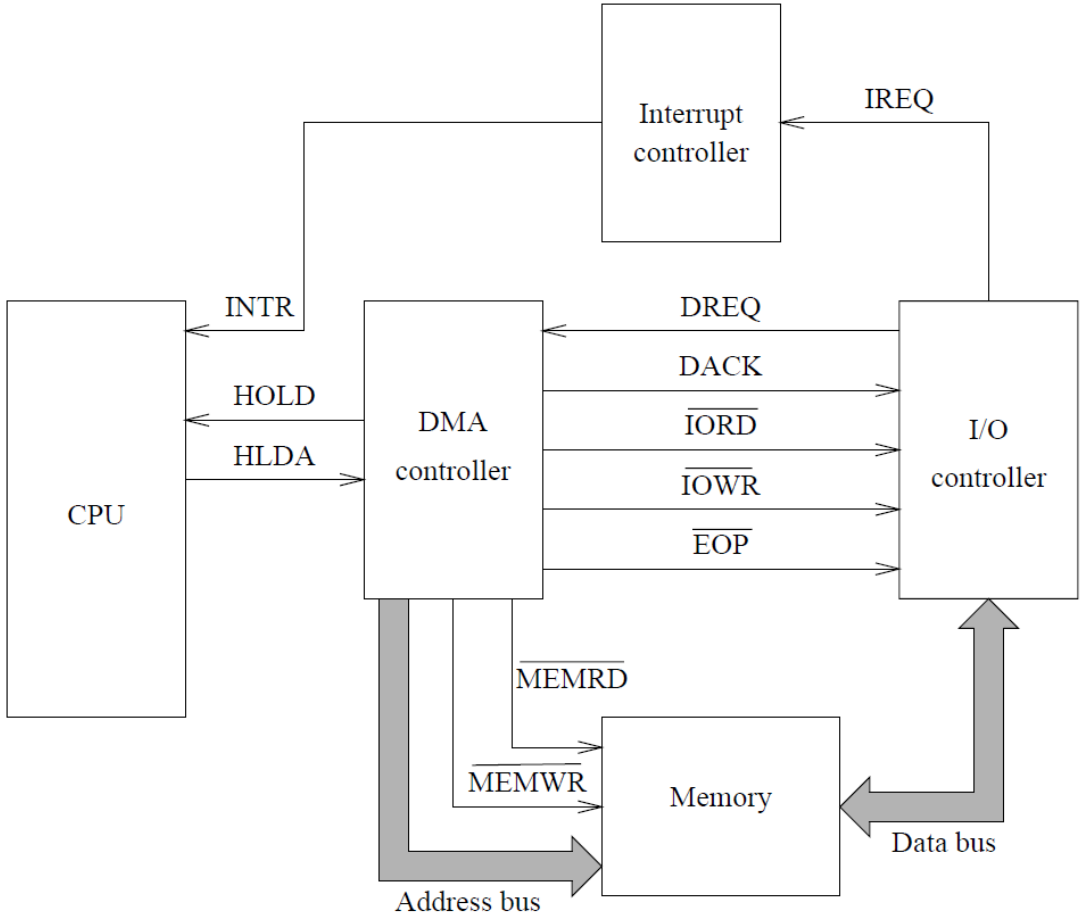
1. Иницијализација канала:

2. Преношење података

3. Обавештавање процесора

- након довршеног преноса обавештава се процесор
- уобичајено се за то користи систем прекида
- процесор затим проверава стање преноса

Поједностављени дијаграм употребе контролера *DMA*



Пример преноса преко *DMA*

- У примеру претпостављамо да се ради о операцији читања путем *DMA*
  - одговарајућим командама процесора *DMA* контролеру иницијализован је канал *DMA* да подржи конкретан У/И уређај (или контролер)
  - подаци се са У/И уређаја и уписују у меморију
  - претпостављамо да се преносе две речи података
  - претпостављамо да је у питању брз пренос који потпуно задржава магистралу до краја процеса

Пример преноса преко *DMA* (2)

- Када У/И уређај буде спреман да преноси податке,
  шаље захтев контролеру *DMA* путем линије *DREQ*
- Када прими сигнал *DREQ*, контролер *DMA* шаље процесору захтев за добијање магистрале, путем сигнала *HOLD*
- Након завршетка текуће инструкције, процесор ослобађа магистралу
  и то јавља контролеру *DMA* путем сигнала *HLDA*
  - процесор надаље не користи магистралу
    већ одлаже све операције које захтевају постављање сигнала на магистралу
- По пријему сигнала *HLDA* контролер *DMA* обавештава У/И уређај
  (путем сигнала *DACK*) да може да почне пренос података
  - након тога сигнал *DREQ* се уобичајено склања

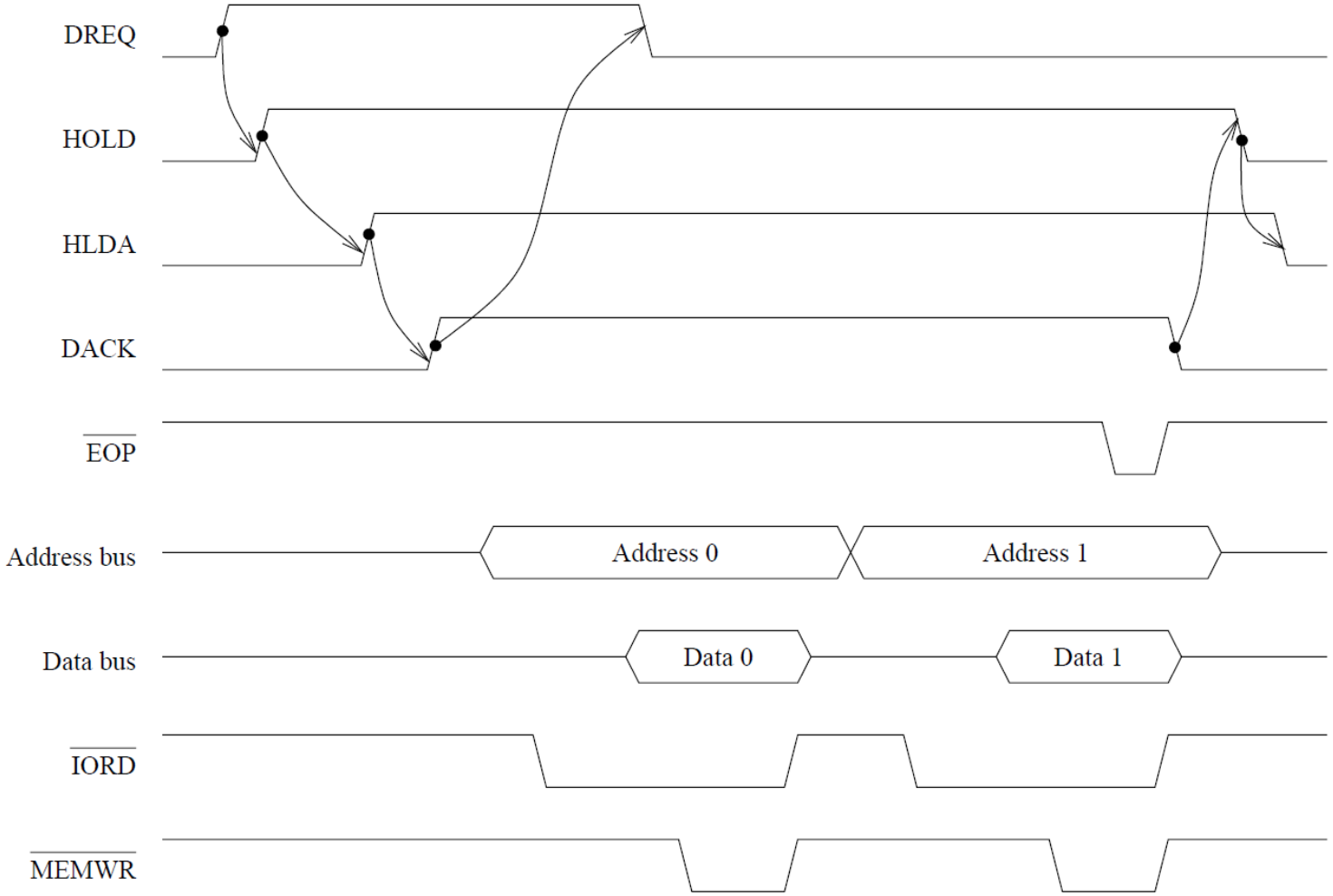
Пример преноса преко *DMA* (3)

- Контролер *DMA* је одговоран за постављање одговарајућих контролних сигнала за читање са уређаја (*IORD'*) и писање у меморију (*MEMWR'*)
  - У/И уређај одговара на сигнал читања и поставља податке на магистралу података
    - не користи се адреса, зато што је уређај и канал везе иденитификован иницијализацијом *DMA*
  - меморија одговара на сигнал писања и уписује податке са магистрале података на одговарајућој адреси
  - бројач пренесених речи се смањује
  - адреса се увећава за величину речи
  - ако је бројач различит од нуле, понавља се овај циклус

Пример преноса преко *DMA* (4)

- Када се преношење података доврши, контролер *DMA* означава да је операција завршена постављањем сигнала *EOP'* (“*end of process*”)
  - том приликом контролер склања и сигнал *DACK*
- Затим контролер *DMA* склања сигнал *HOLD* чиме враћа магистралу на употребу процесору
- Процесор одговара склањањем сигнала *HLDA*
- Поступак је тиме довршен

Дијаграм употребе магистрале током преноса путем *DMA*



Пример преноса преко *DMA* (5)

- У случају споријих уређаја пренос се одвија у другачијем режиму
  - контролер *DMA* за свако појединачно преношење података понавља захтевање и ослобађање магистрале
  - сваки циклус почиње сигналом *DREQ* од стране У/И уређаја, а завршава ослобађањем магистрале
  - крај процеса се означава сигналом *EOP'* који контролер *DMA* шаље У/И уређају

Улазно/излазни уређаји

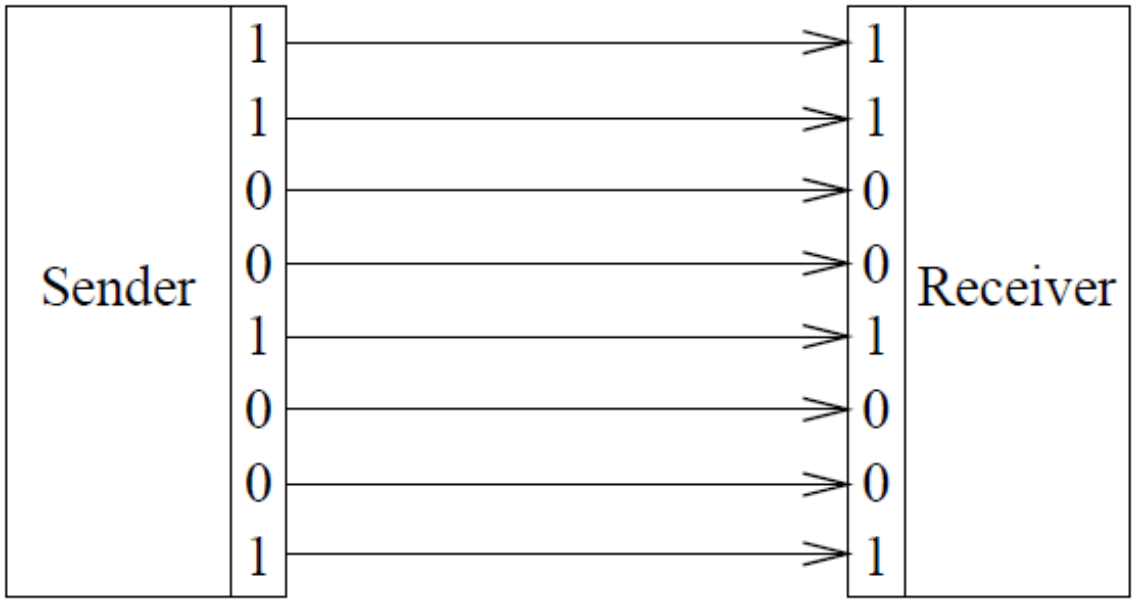
Серијски и паралелни пренос

Врсте техника преноса података

- Пренос података може бити
  - Паралелан
  - Серијски
    - Синхрони
    - Асинхрони

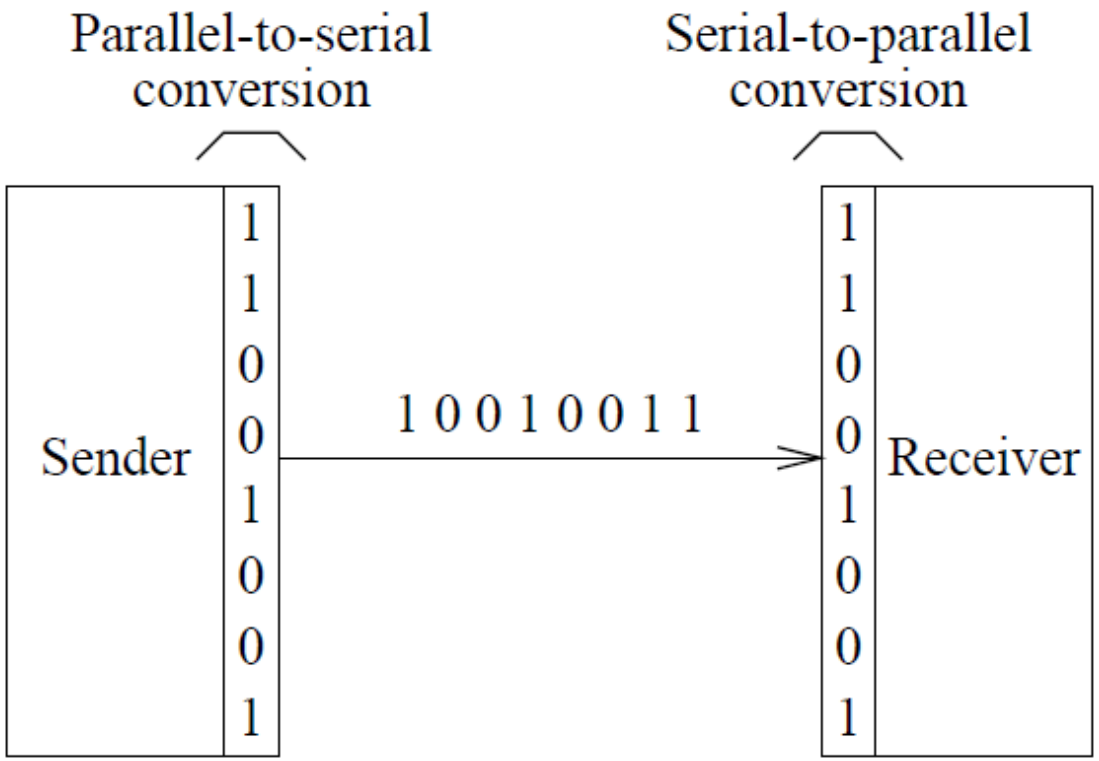
Паралелан пренос података

- Неколико битова се преноси истовремено кроз паралелне водове



Серијски пренос података

- За пренос података се користи један вод
- Нема паралелног преношења података



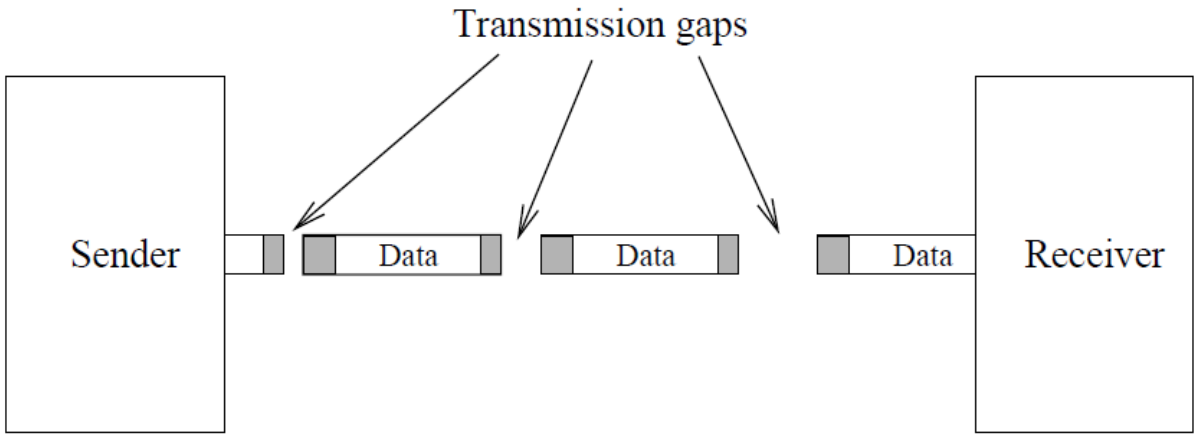
Однос паралелног и серијског преноса

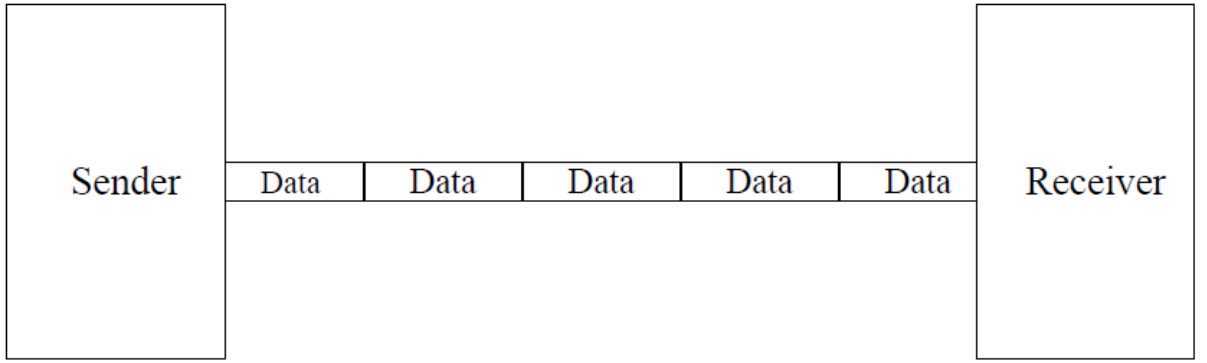
- Парелни пренос
  - кроз *n* паралелних водова се може истовремено преносити *n* битова
  - бржи је
  - скупљи је за имплементацију
  - са повећавањем дужине водова и брзине рада расте вероватноћа појављивања *искривљења* (*дисторзије*)
    - неки битови стижу раније или касније, несинхронизовано са осталим битовима
  - користи се углавном на мањим даљинама
- Серијски пренос
  - јевтинији
  - нема могућности искривљења података

Серијски пренос података

- Може да буде
  - синхрони
    - часовници пошиљаоца и примаоца су синхропнизовани пре преношења података
  - асинхрони
    - сваки бајт се енкодира за пренос тако да часовници пошиљаоца и примаоца не морају да буду усклађени
    - сваки пакет (нпр. бајт) је окружен тзв. почетним (старт) битовима и зауставним (стоп) битовима
- Синхрони пренос је
  - ефикаснији
  - скупљи

Асинхрони и синхрони пренос података

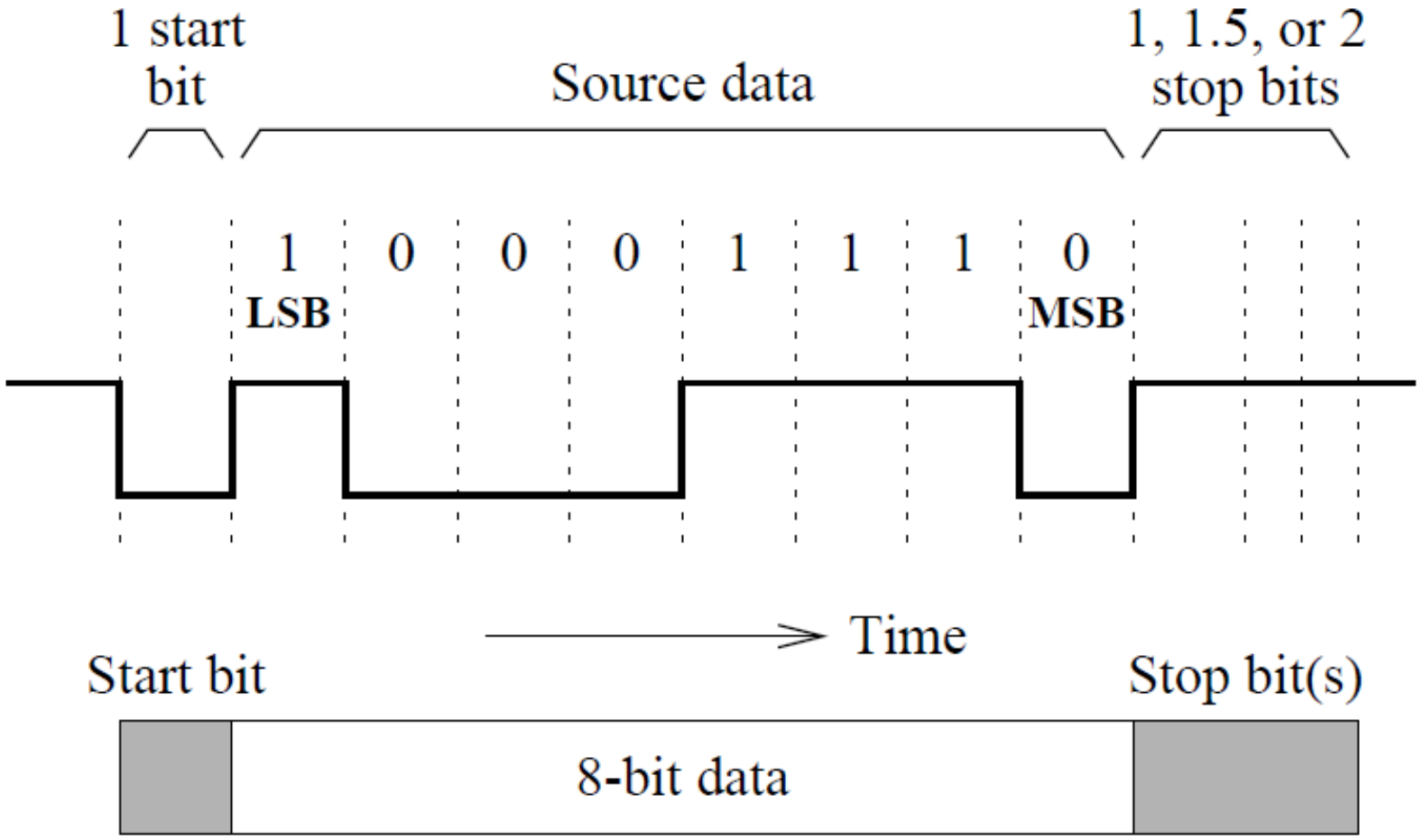




Асинхрони пренос

- Комуникациона линија је у активном стању док не постоји пренос
- При преношењу података почетни бит спушта стање линије
  - тако прималац препознаје почетак пакета
- Прималац
  - зна дужину битова
  - зна и величину пакета (нпр. 8 битова)
  - чита стања линије у срединама периода
  - игнорише почетни бит и слаже пакет (бајт)
- Зауставни бит има две намене
  - на крају пакета не сме остати 0, зато што прималац не би могао да препозна почетак наредног пакета
  - оставља довољно времена примаоцу да сложи пакет од примљених битова
- Уобичајено се користи 1, 1.5 или 2 зауставна бита

Сигнали при асинхроном преносу података



Пример паралелног интерфејса

- Интерфејс за штампаче (*Centronics*)
  - 25 линија, познат и под именом (*DB-25*)
- Интерфејс је једноставан:
  - 8 линија за податке
  - 1 линија резе за податке (као часовник)
  - руковање (*handshaking*) се остварује сигналом *ACK*
    - након сваког испорученог бајта рачунар чека да прими сигнал *ACK* пре слања наредног бајта
  - 5 линија статуса
    - *busy, out of paper, online/offline, autofeed, fault*
  - 2 контролна сигнала
  - 8 линија уземљења

Пример паралелног интерфејса (2)

Pin #	Signal	Signal direction	Signal function
1	STROBE	PC ⟹ printer	Clock used to latch data
2	Data 0	PC ⟹ printer	Data bit 0 (LSB)
3	Data 1	PC ⟹ printer	Data bit 1
4	Data 2	PC ⟹ printer	Data bit 2
5	Data 3	PC ⟹ printer	Data bit 3
6	Data 4	PC ⟹ printer	Data bit 4
7	Data 5	PC ⟹ printer	Data bit 5
8	Data 6	PC ⟹ printer	Data bit 6
9	Data 7	PC ⟹ printer	Data bit 7 (MSB)
10	ACK	printer ⟹ PC	Printer acknowledges receipt of data
11	BUSY	printer ⟹ PC	Printer is busy
12	POUT	printer ⟹ PC	Printer is out of paper
13	SEL	printer ⟹ PC	Printer is online
14	AUTO FEED	printer ⟹ PC	Autofeed is on
15	FAULT	printer ⟹ PC	Printer fault
16	INIT	PC ⟹ printer	Clears printer buffer and resets printer
17	SLCT IN	PC ⟹ printer	TTL high level
18–25	Ground	N/A	Ground reference

Universal Serial Bus – USB

- Серијски пренос
- Изворно развијен 1995.
- Основни циљ
  - омогућити повезивање рачунарских периферија једнако једноставно као што се повезују телефон или пегла
- Стандарди
  - 1.0 из 1996.
    - *ниска* брзина 1.5Mbps, *пуна* брзина 12Mbps
  - 1.1 из 1998.
    - први широко распрострањен стандард
  - 2.0 из 2000.
    - *висока* брзина, до 480 Mbps
  - 3.0 спецификације објављене 2008.
    - брзина до 4800 Mbps

USB – Неки од најважнијих циљева

- Обезбеђивање хомогеног рачунарског интерфејса
  - идеја је да се мноштво различитих и некомпатибилних интерфејса замене једним универзалним интерфејсом
- Превазилажење проблема са ресурсима система
  - додавање нових уређаја доноси проблеме разрешавања конфликата око адреса или прекида
  - *USB* уређаји не захтевају ни меморију ни адресни простор ни прекиде
- Једноставнија инсталација и конфигурација
  - потпуна аутоматизација (*plug-and-play*) за разлику од старијих уређаја, који су често захтевали мануелно конфигурисање
- Повезивање током рада рачунара
  - старији начини повезивања уређаја су захтевали искључивање рачунара пре повезивања
  - повезивање *USB* уређаја се може обављати без искључивања рачунара

USB – Додатне напредне особине

- Напајање уређаја кроз интерфејс
  - кабл за повезивање има линије напајања,
    чиме је омогућено да се уређаји напајају струјом напона +5V и јачине 100-500mA
- Двосмерна контрола уређаја
  - подаци могу да теку у оба смера
- Проширивост помоћу *хабова*
  - на једно прикључно место се може повезати хаб (разделник) ради повећавање броја прикључака или постављања прикључака на жељено место
- Уштеда енергије
  - *USB* уређаји аутоматски улазе у примирено (суспендовано) стање
    ако нема активности на магистрали 3ms
- Препознавање и отклањање грешака
  - користи се алгоритам *CRC* за препознавање грешака у комуникацији

USB – Пренос података

- Кабл има четири линије:
  - две служе за напајање уређаја
  - две служе за преношење сигнала

- За преношење података се употрбљава схема енкодовања *NRZI* (*nonreturn to zero-inverted*)

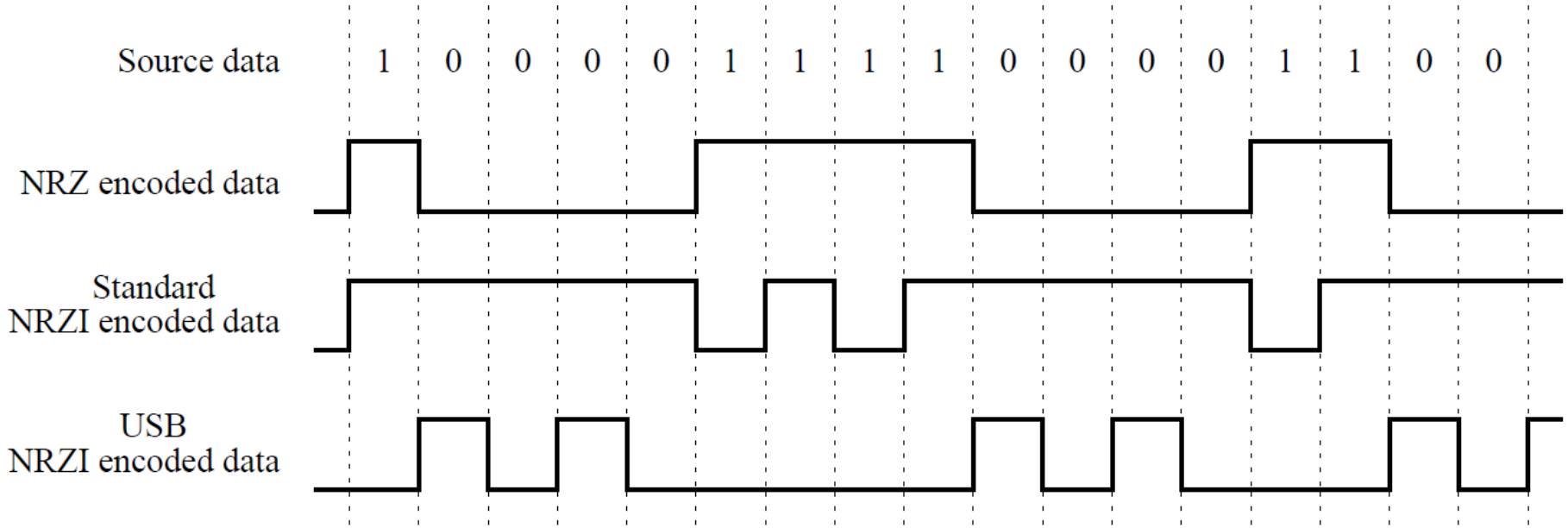
Схема енкодовања *NRZ*
(*nonreturn to zero*)

- 0 је низак а 1 висок ниво сигнала
- Проблеми:
  - Нема промена сигнала ако се преносе дуги низови нула или јединица
    - промене су важне за примаоца због синхонизације и препознавања података
  - У случају шумова, тешко је препознати нивое 0 и 1,
    али се промене ипак лако препознају

Схема енкодовања *NRZI*
(*nonreturn to zero – inverted*)

- Решава неке од проблема схеме *NRZ*
- Не користи апсолутне нивое за представљање података, већ само промене стања
- Стандардно енкодовање *NRZI* подразумева:
  - сигнал се мења ако је наредни бит 1
  - сигнал се не мења ако је наредни бит 0
  - смер промене није значајан
- Енкодовање *NRZI* у случају *USB*-а је другачије:
  - сигнал се мења ако је наредни бит 0
  - сигнал се не мења ако је наредни бит 1
  - смер промене није значајан

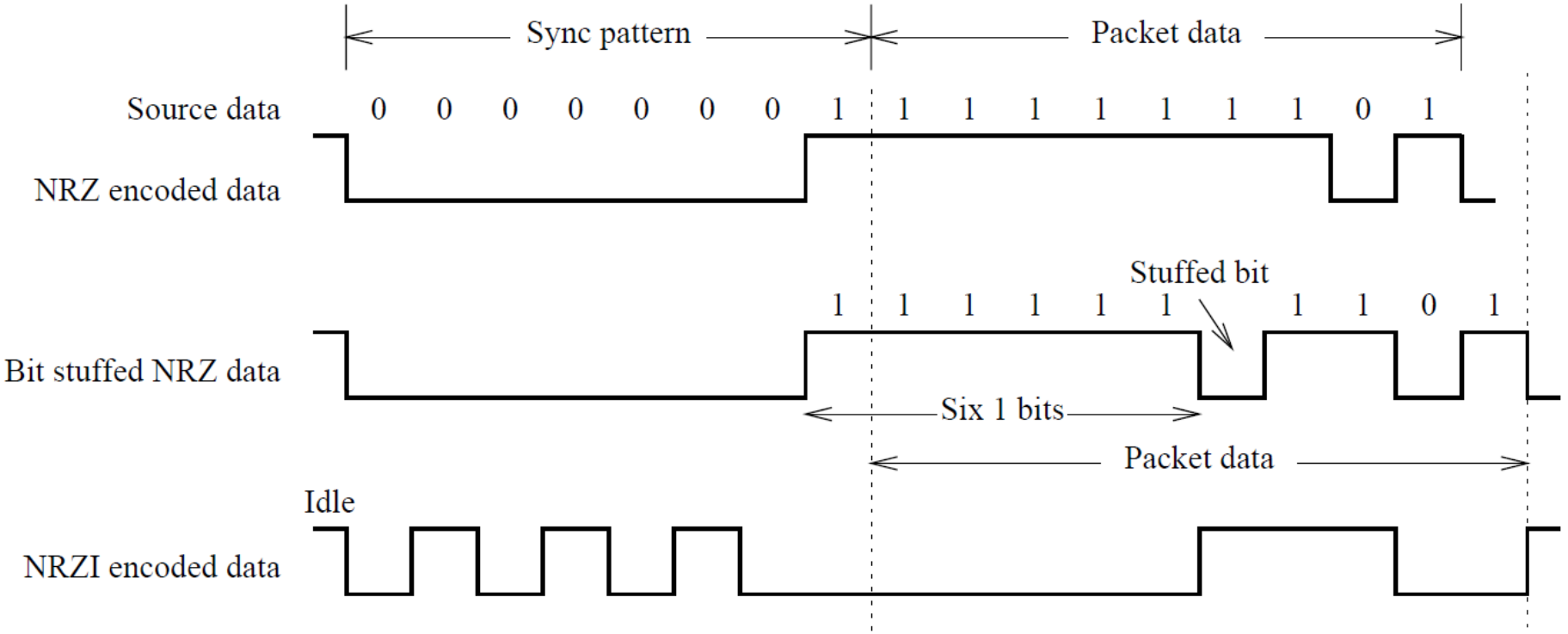
NRZI


Source data
1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 0 0
NRZ encoded data
Standard
NRZI encoded data
USB
NRZI encoded data

Уметање битова

- Описана схема енкодовања *NRZI* решава неке од проблема:
  - ниво сигнала не игра главну улогу
    - посматрају се само транзиције
  - отклањају се дугачки низови непромењених стања у случају низова јединица у оригиналним подацима
- Преостаје проблем:
  - опстају дугачки низови непромењених стања у случају јединица
- Решење – предузима се тзв. *уметање битова*
  - након сваких 6 узастопних јединица се умеће једна 0
  - уметање је на нивоу података, а не сигнала

Уметање битова (2)



Архитектура *USB* повезивања

- Основни хардвер чине
  - матични контролер *USB*-а (*host controller*)
    - служи да иницира трансакције
  - корени хаб (*root hub*)
    - служи да успостави везу контролера са циљним уређајем

Пример архитектуре *USB* повезивања