

Влез и излез

I/O Систем

- ▶ Принципи кај I/O хардверот
- ▶ Принципи кај I/O софтверот
- ▶ I/O софтверски нивоа

I/O уреди

- ▶ Генерална поделба:
 - Блок уреди: HD, CD-ROM, USB меморија
 - Пренесуваат блокови со фиксна големина (512B – 32kB), секој со своја адреса.
 - Знакови уреди (не се адресабилни и нема seek операција): печатари, мрежен интерфејс, глумче итн.

Принципи кај I/O хардверот

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

Типични податочни рати

Transfer rates

100 Gigabit Ethernet (100GBASE-X) 10×/4×	100 Gbit/s	12.5 GB/s	2010/2018
Infiniband EDR 12× ^[29]	300 Gbit/s	37.5 GB/s	2014
IEEE 802.11ad (maximum theoretical speed)	7.14–7.2 Gbit/s	892.5–900 MB/s	2011
PCI Express 3.0 (×8 link) ^[43]	64 Gbit/s	7.877 GB/s ^[y]	2011
PCI Express 1.0 (×32 link) ^[41]	80 Gbit/s	8 GB/s ^[z]	2001
HyperTransport 3.1 (3.2 GHz, 32-pair)	409.6 Gbit/s	51.2 GB/s	2008
NVLink	640 Gbit/s	80 GB/s	2016
SATA revision 3.2 – SATA Express	16000 Mbit/s	2000 MB/s	2013
Fibre Channel 32GFC (28.05 GHz) ^[48]	26424 Mbit/s	3303 MB/s ^[b]	2016
USB 3.1 SuperSpeed+	10 Gbit/s	1250 MB/s	2013
Thunderbolt 3	40 Gbit/s	5000 MB/s	2015
PC3-21300 DDR3 SDRAM	170.4 Gbit/s	21.3 GB/s	
PC3-24000 DDR3 SDRAM	192 Gbit/s	24 GB/s	
PC4-25600 DDR4 SDRAM	204.8 Gbit/s	25.6 GB/s	
DisplayPort 1.3 (4-lane High Bit Rate 3)	32.4 Gbit/s	4.05 GB/s ^[a]	2014
superMHL	36 Gbit/s	4.5 GB/s	2015
Thunderbolt 3	40 Gbit/s	5 GB/s	2015
HDMI 2.1 ^[68]	48 Gbit/s	6 GB/s ^[b]	2017

Контролери на уреди

- ▶ I/O уредите се составени од:
 - Механички компоненти
 - Електронски компоненти
- ▶ Електронската компонента (адаптер) е контролерот на уредот
 - Може да управува и со повеќе уреди
 - IDE, SATA, SCSI, USB интерфејс
- ▶ Функции на контролерот
 - Конверзија на сериски податоци во паралелни
 - Детекција и корекција на грешки

Контролни регистри

- ▶ Контролерите имаат контролни регистри за комуникација со CPU
 - типично се состои од 4 регистри:
 - **статус** (зафатен? податоци спремни? грешка?)
 - **контрола** (команда за извршување)
 - **податок во** (податок што се праќа од уредот кон CPU-то)
 - **податок од** (податок што се праќа од CPU-то кон уредот)

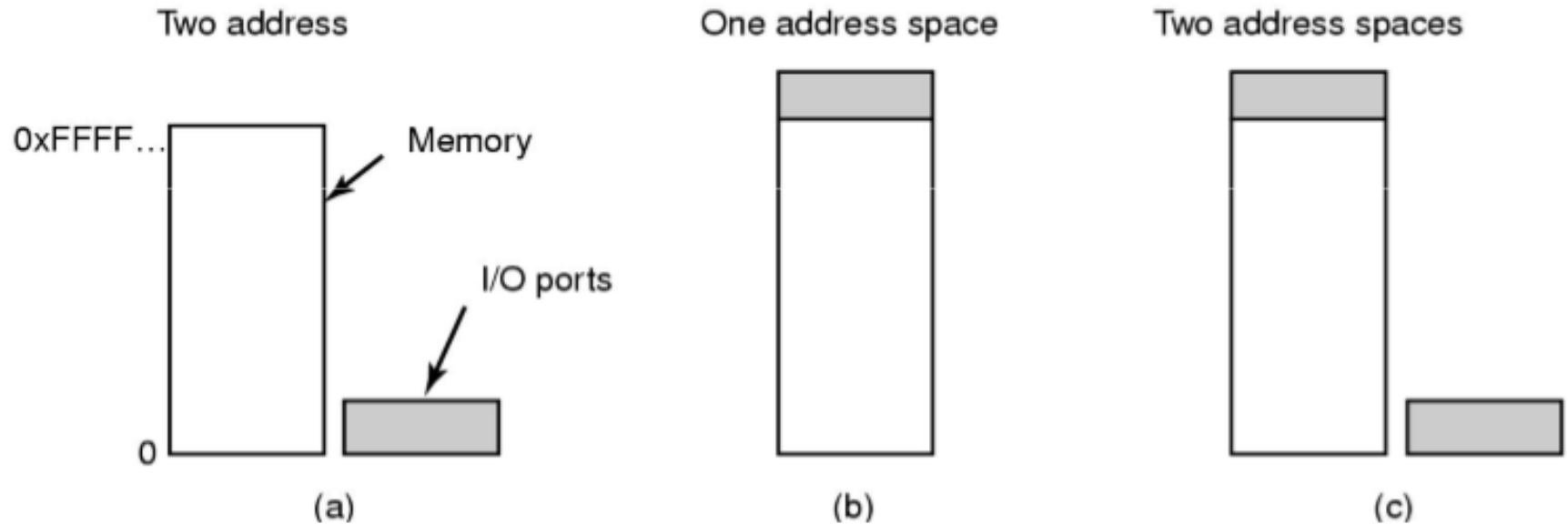
Начини на пристап

- ▶ CPU може да ги достапи овие регистри преку специјални I/O инструкции (in/out во Intel assembler) или преку мемориско пресликување
 - Мемориското пресликување го изведува меморискиот управувач и е побрзо од експлицитни I/O инструкции.
- ▶ Во интеракција со контролер на уред CPU може да чека на одговор преку прозивка на статусните регистри, т.е преку периодично проверување дали статусот на уредот се сменил

Мемориски пресликан I/O

- ▶ Пресликани (мапирани) се контролните регистри во меморискиот простор
- ▶ На секој контролен регистар му се доделува единечна мемориска адреса во која не се доделува меморија
 - Најчесто овие адреси се на почетокот на меморискиот простор
 - Хибридните шеми имаат мемориски пресликани I/O податочни бафери и посебни I/O порти за контролните регистри

Мемориски пресликан I/O



- (a) различен I/O и мемориски адресен простор
- (б) мемориски мапиран I/O
- (c) хибридно решение (x86)

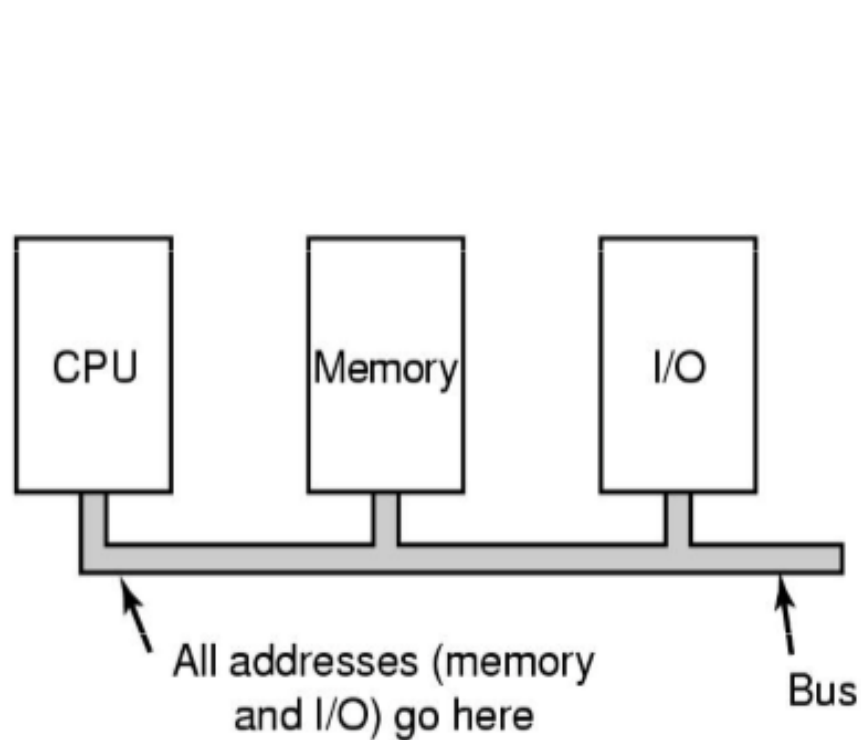
Предности

- ▶ Драјверите може да се пишуваат во C, не мора да се користи асемблер
- ▶ Механизам за заштита
 - соодветниот адресен простор со контролните регистри на уредот нема да биде доделен на ниту еден виртуелен адресен простор
- ▶ Може да се користат истите инструкции кои се користат и за меморија (TEST)
 - Се намалува бројот на инструкции

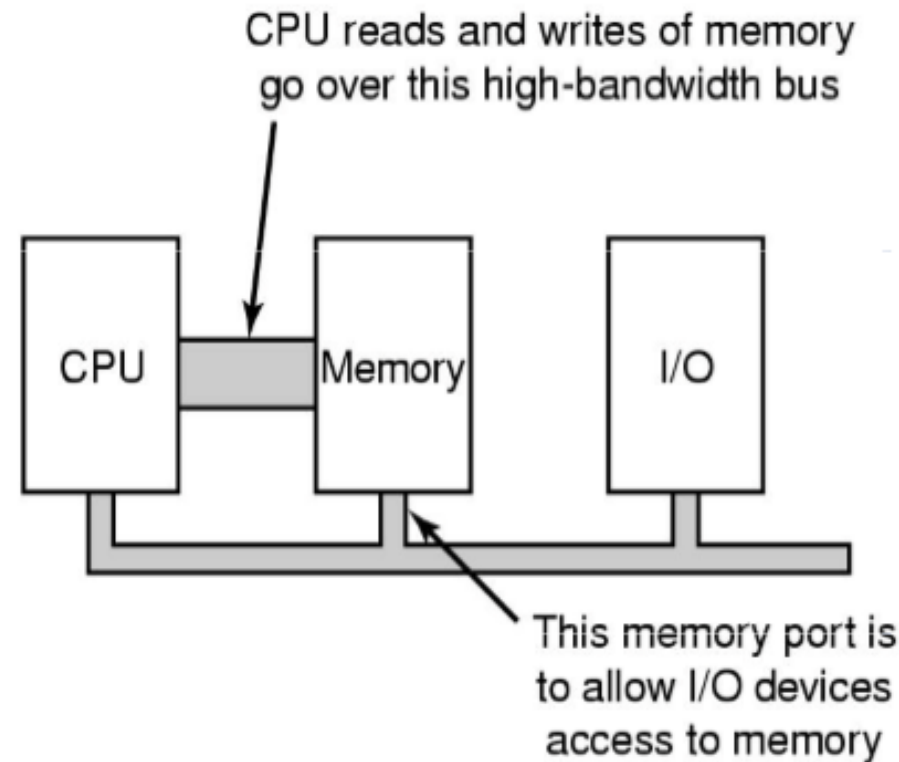
Недостатоци

- ▶ Проблеми со кеширање (дали е подготвен уредот)
 - Проблемот се надминува селективно оневозможување на кеширањето
 - Дополнителна комплексност
- ▶ Проблем со одредување дали дадена мемориска референца е за меморија или за мемориски мапиран I/O

Архитектури



(a)

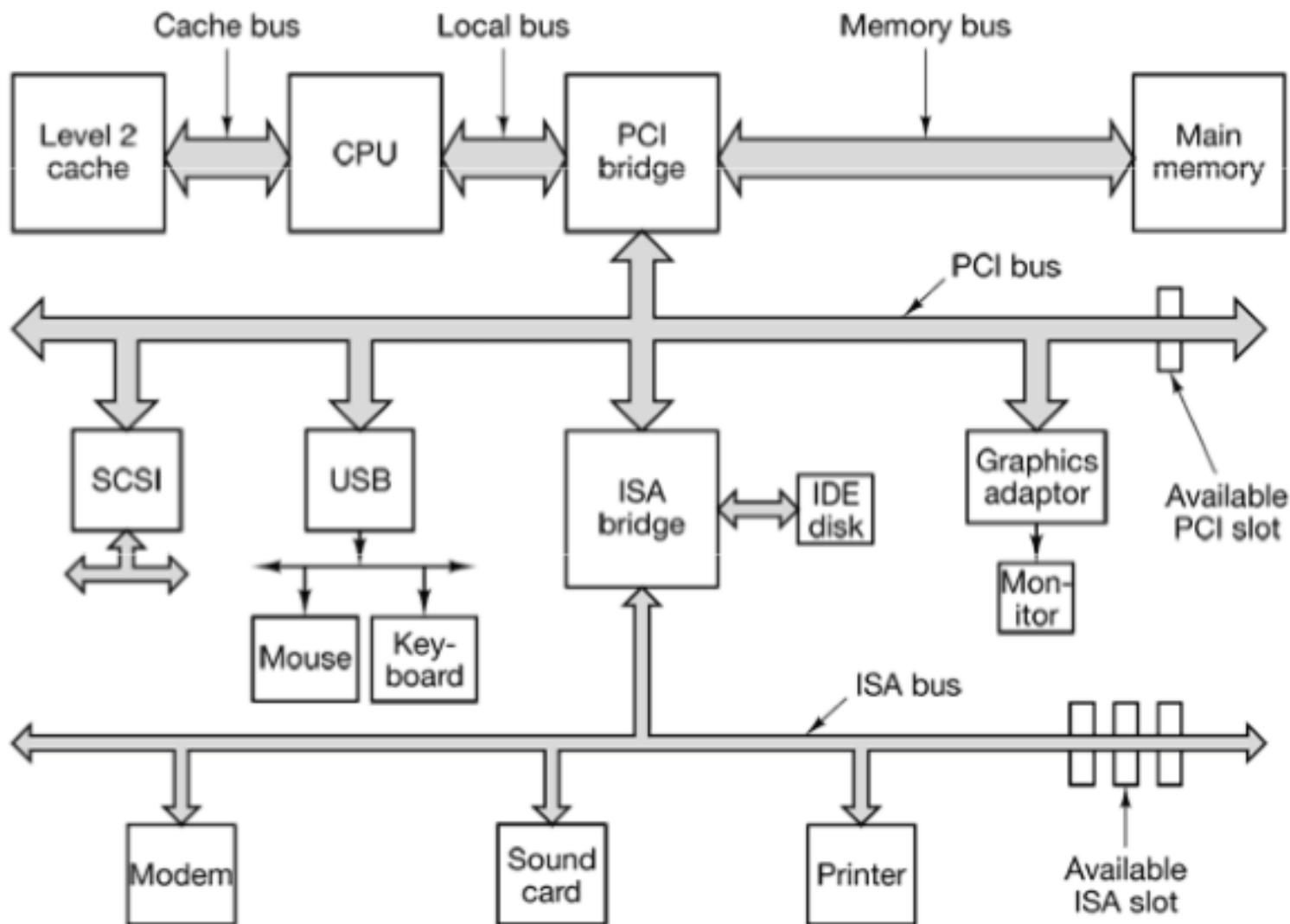


(b)

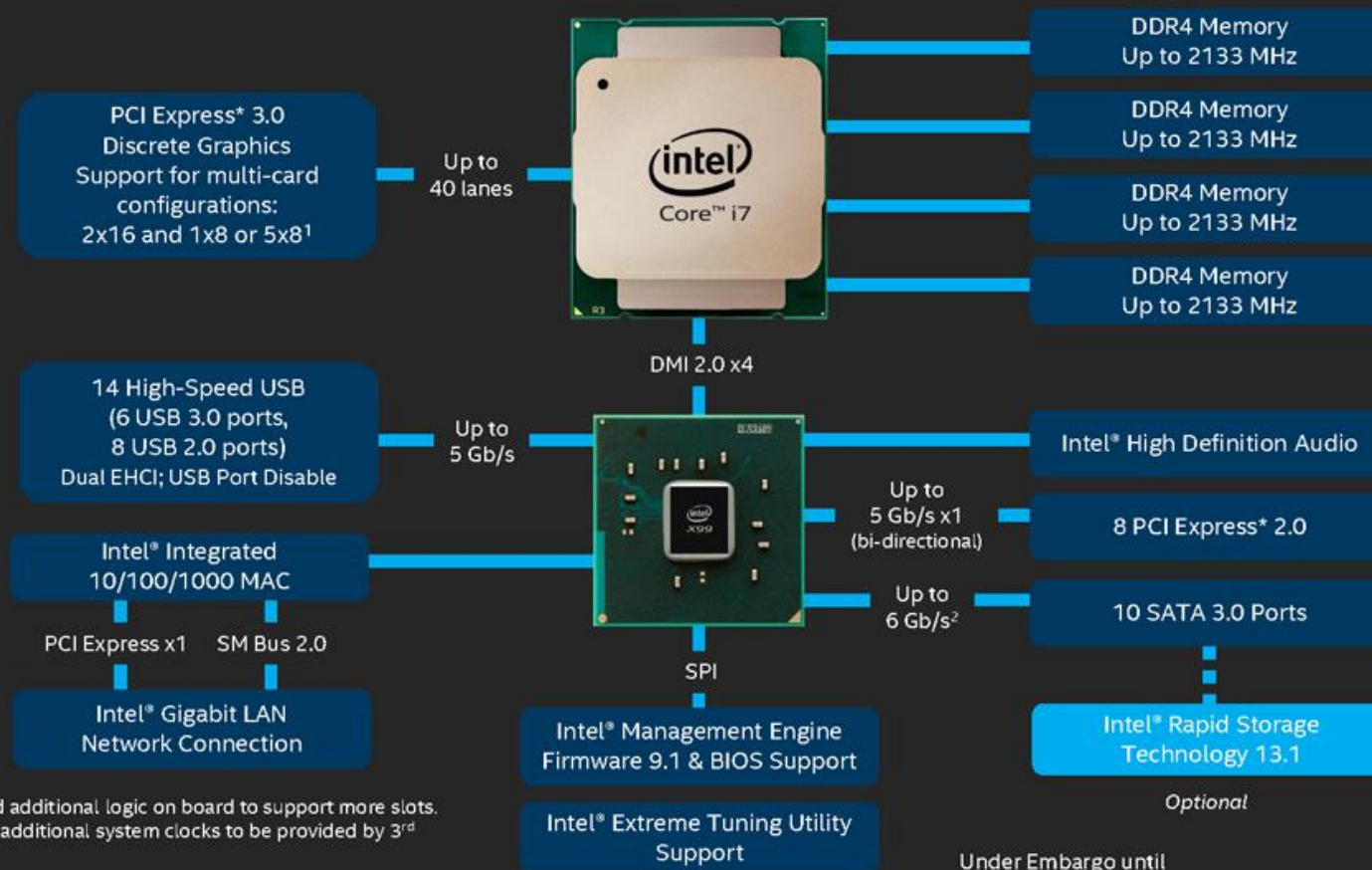
(a) Архитектура со една магистрала

(b) Архитектура со две магистрали

Структура на Pentium системот



Intel® Core™ i7 High End Desktop Platform Overview



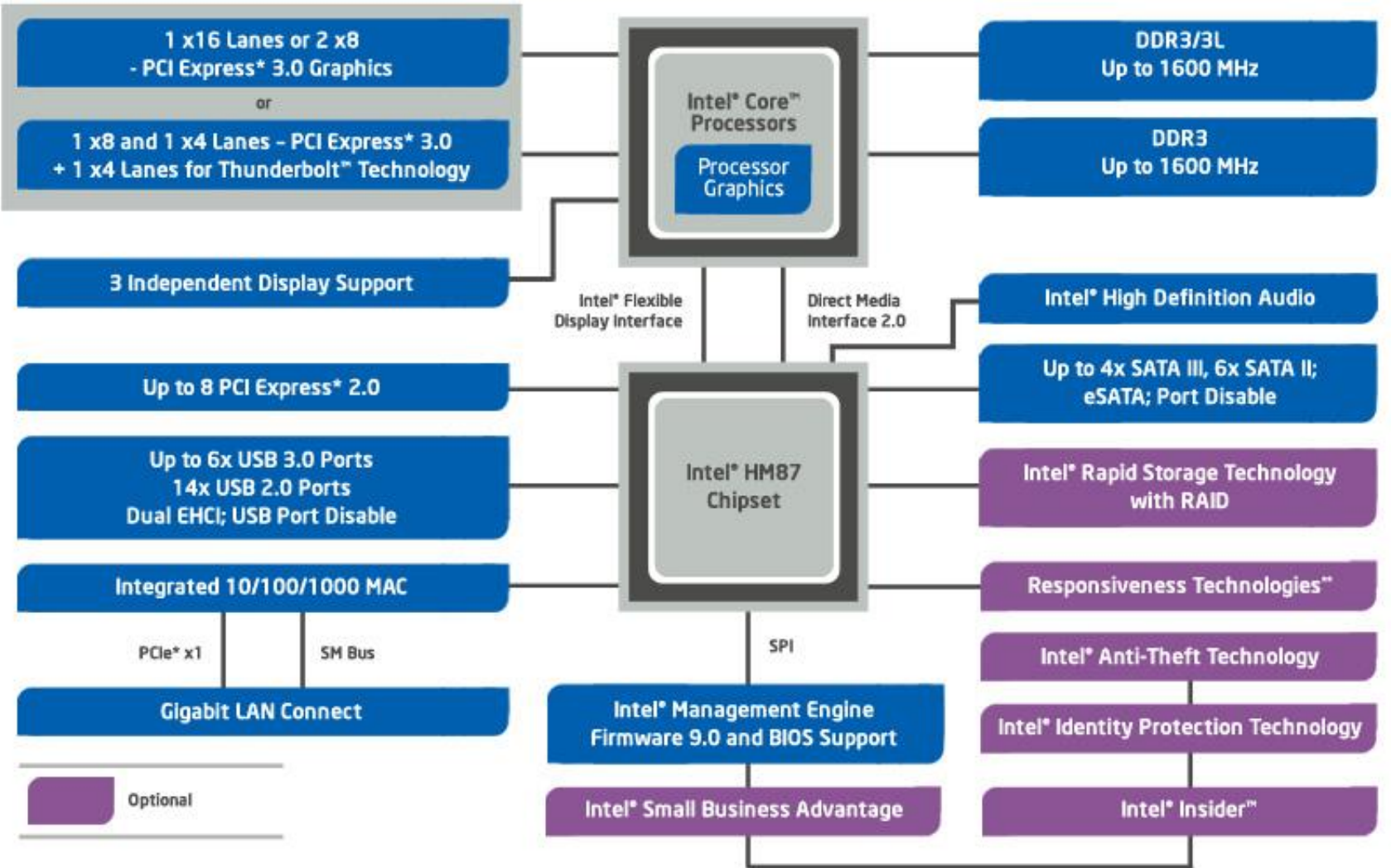
¹ 3 slots available, but need additional logic on board to support more slots. 5x8 configuration requires additional system clocks to be provided by 3rd party components.

² All SATA ports capable of 6 Gb/s.

Under Embargo until
9:00am PST, August 29, 2014



Mobile Intel® HM87 Chipset Block Diagram




*Includes Intel® Smart Response Technology, Intel® Rapid Start Technology, and Intel® Smart Connect Technology.

Типови на комуникација со уредите

- ▶ Programmed I/O (програмски воден)
- ▶ Interrupt-Driven I/O (прекински воден)
- ▶ I/O со DMA

В/И управувачки методи (1)

- ▶ Програмиран I/O
 - еден byte/збор во единица време
 - користи *polling*  троши CPU циклуси
 - добар за уреди со специјална намена
микропроцесор-контролирани уреди
- ▶ Interrupt-водени I/O
 - многу I/O уреди го користат овој пристап како добра алтернатива на *polling*
- ▶ **Direct Memory Access (DMA)**
 - за блок трансфери
 - CPU минимално учество само на почеток и на крај на трансфер)

Програмиран В/И

Polling:

- ▶ CPU-то работно чека додека статусот на контролерот не стане **idle**
- ▶ CPU -то го поставува командниот регистер (status **ready**) и податоците (ако е излезна операција)
- ▶ Контролерот реагира на статусот **ready** и поставува статус **busy**
- ▶ Контролерот ја чита командата од командниот регистер и ја извршува праќајќи или примајќи податоци
- ▶ По завршувањето на операцијата, контролерот го менува статусот во **idle**
- ▶ CPU-то го гледа новиот статус на контролерот и ги презема податоците (ако е влезна операција)

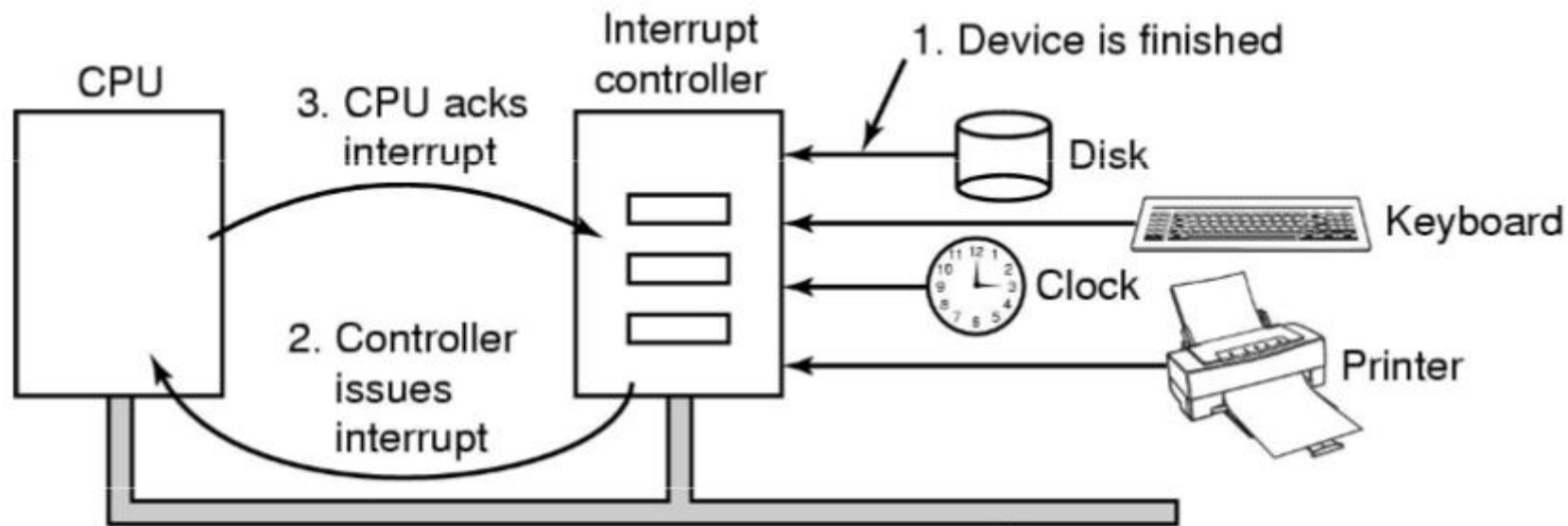
Преземањето на податоците мора да биде брзо, инаку тие можат да бидат изгубени (модем, тастатура)

В/И водени со прекини

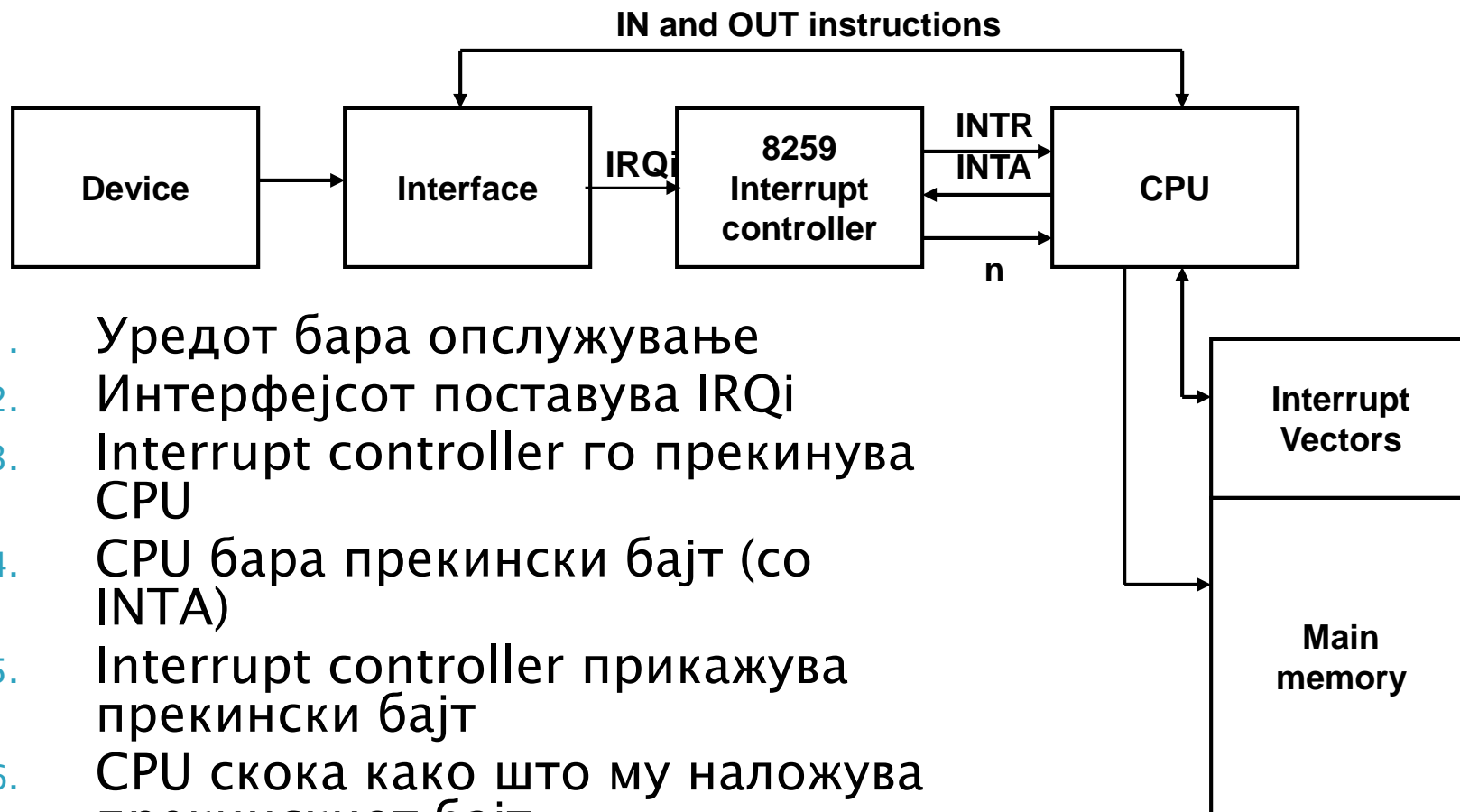
Interrupts:

- ▶ Наместо CPU-то работно да чека додека уредот да стане слободен, уредот може да го **прекине** CPU-то кога ќе комплетира една операција (предизвикува прекин)
- ▶ При В/И прекин:
 - CPU-то одредува кој уред го предизвикува прекилот
 - Ако операцијата е влезна, ги презема податоците од регистрите на контролерот
 - Започнува следна операција со уредот

Прекини



По барање за прекин



1. Уредот бара опслужување
2. Интерфејсот поставува IRQ_i
3. Interrupt controller го прекинува CPU
4. CPU бара прекински бајт (со INTA)
5. Interrupt controller прикажува прекински бајт
6. CPU скока како што му наложува прекинскиот бајт
7. CPU комуницира со интерфејсот користејќи PIO

Рутини за сервисирање прекин – Interrupt Service Routines

Рутина дизајнирана за да се справи со
прекин кој доаѓа од некој уред

Треба да е невидлива за прекинатиот програм

Сервисира уред кој користи програмиран I/O

Ако фатиш прекин, треба да го отстраниш пред да ја терминираш
апликацијата

Пример

ISR за нов интерфејс



Услови при влегување во ISR

- ▶ Понатамошните прекини од уреди се оневозможени
- ▶ На стекот се снимени IP (instruction pointer), CS (code segment) и FLAGS
- ▶ Сите други CPU регистри не се модифицирани
 -
- ▶ Прекинатиот програм може да продолжи од точката каде што е прекинат, користејќи ја специјалната CPU инструкција 'IRET'
- ▶ 'IRET' прави рор на трите најгорни збора на стекот и ги враќа назад во IP, CS, и FLAGS

Зачувување на контекстот на прекинатиот програм

- ▶ Од големо значење е прекинатата програма да може да продолжи точно од истата CPU состојба во која беше пред да настане прекинот
- ▶ CPU снима само минимална количина од контекстот на програмот (FLAGS, CS, и IP)
- ▶ На одговорност на програмерот на соодветната ISR е да ги зачува другите регистри кои би му требале да ги модифицира внатре во ISR

Генерална структура

1. Операцијата на прекин снима РС и знаменца
2. Зачувај околина
Push сите регистри кои планираш да ги користиш
3. I/O акцијата
Добива статус на уредот
Извршува соодветна I/O акција
4. Ресторирај околина
Pop на регистрите кои претходно ги зачува на стек (push)
5. IRET

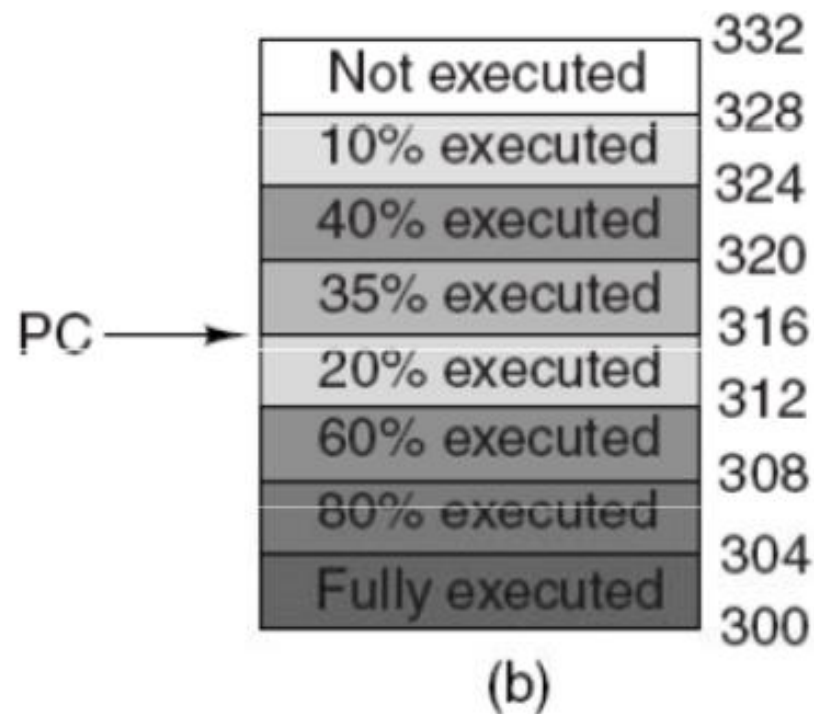
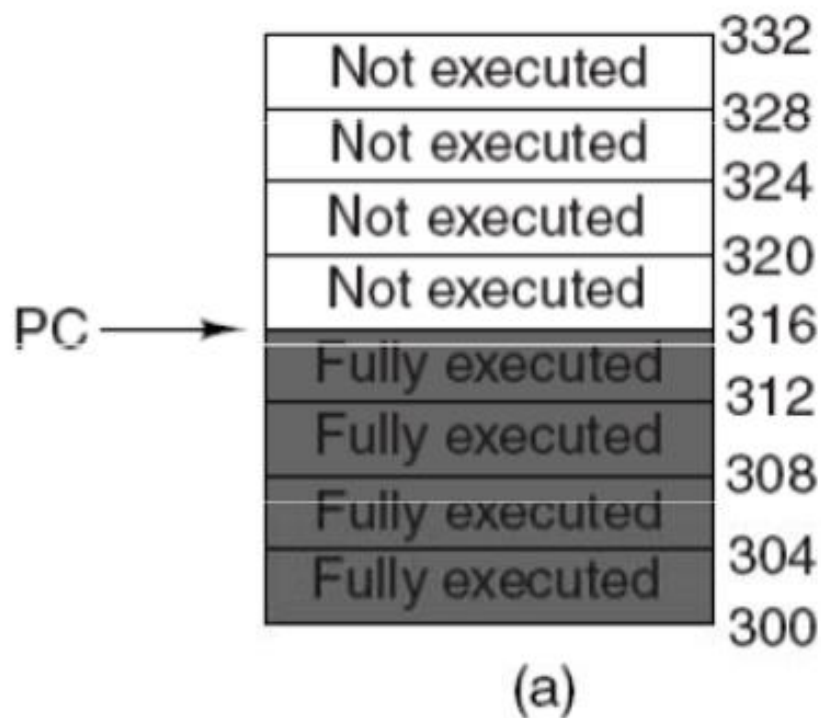
Ако фаќаш прекин (пр. hotkey)

Знај што имаш на стекот пред да скокнеш на оригиналната ISR

Прецизни и непрецизни прекини

- ▶ Својства на прецизни прекини:
 1. Програмскиот бројач се чува на познато место
 2. Сите инструкции пред онаа на која покажува програмскиот бројач се целосно извршени
 3. Ниту една инструкция после програмскиот бројач не е извршена.
 4. Состојбата на инструкцијата на која што покажува РС е познат

Прецизни и непрецизни прекини



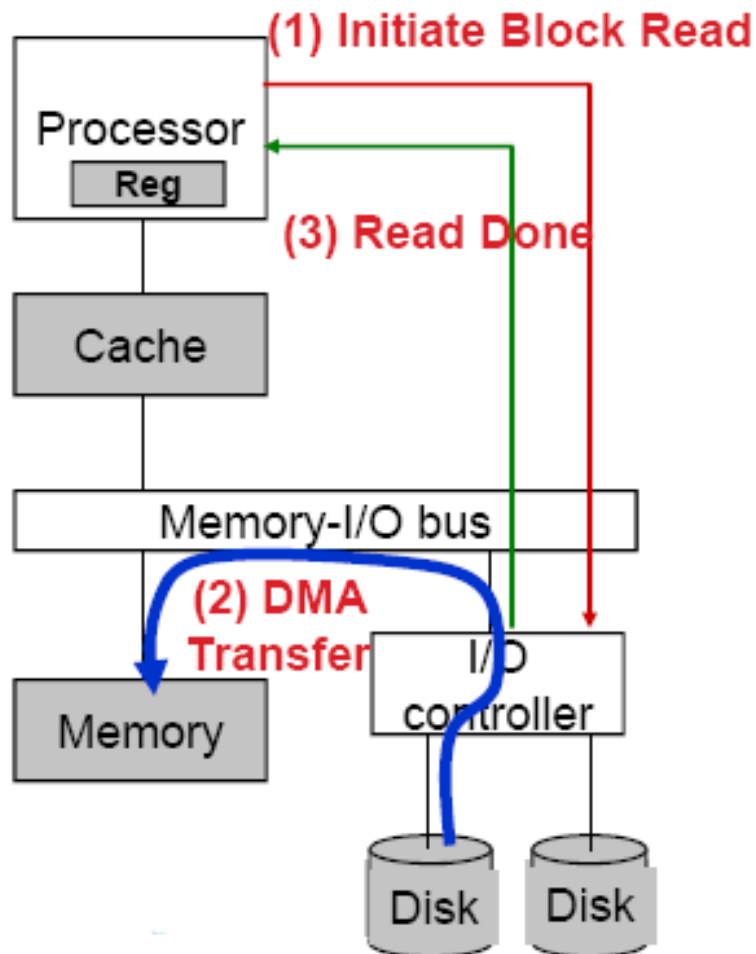
Непрецизни прекини

- ▶ Голема содржина се чува во стекот
- ▶ Pentium заради компатибилност работи и со прецизни прекини
 - Се зголемува големината на чипот и комплексноста на дизајнот
- ▶ Со непрецизните оперативниот систем е покомплексен и побавен

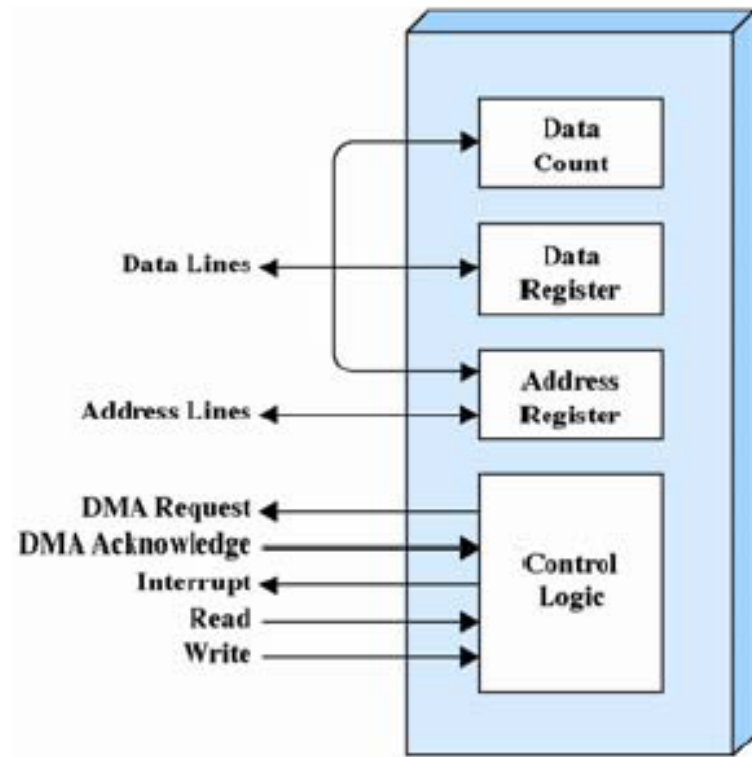
Директен пристап до меморијата (DMA)

- ▶ DMA е контролер, кој го ослободува CPU-то од пренос на податоци, но на ниско ниво
- ▶ DMA контролерот може директно да запишува во меморијата наместо во регистрите за влез и излез од CPU -то
- ▶ CPU -то му кажува на DMA-то (покрај диск-адресата на блокот што се чита уште два параметра)
 - мемориска адреса каде што ќе се запише блокот
 - Број на бајти што треба да се пренесат (бројач)

DMA



типичен DMA блок
дијаграм



DMA (2)

Како чита дискот СО DMA:

► контролерот

- го чита целиот блок од уредот (дискот) во својот бафер и ја верифицира точноста;
- го копира првиот бајт или збор во главната меморија на адреса зададена од DMA-то
- Ја зголемува адресата на DMA-то
- Го намалува бројачот на DMA-то за бројот на пренесените бајти
- Процесот се повторува додека бројачот на DMA-то не стане 0
- Предизвикува прекин на CPU

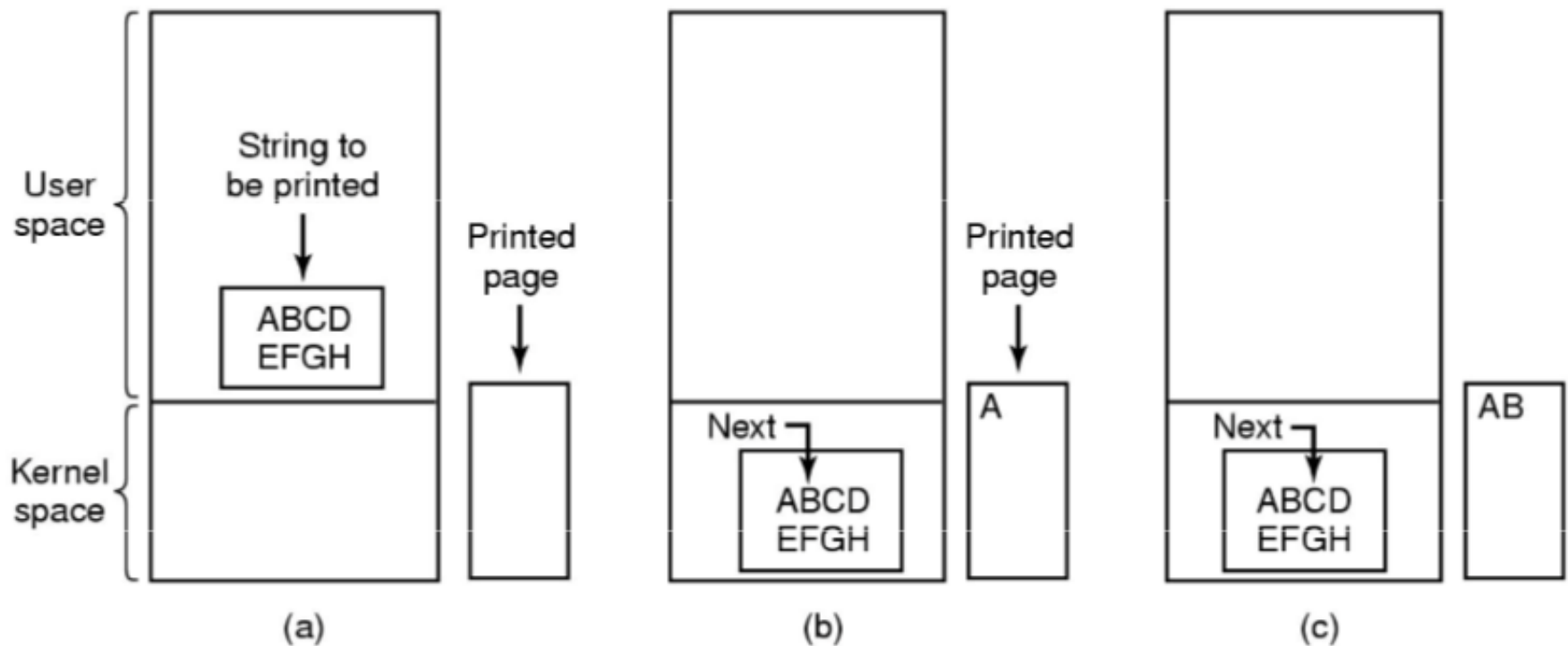
DMA (3)

- ▶ Со DMA-то, работата на CPU -то се прекинува дури кога ќе заврши трансферот
- ▶ DMA-контролерот и CPU-то се натпреваруваат за системската магистрала што делумно го успорува CPU-то, но многу помалку во споредба со случајот на постојано прекинување на неговата работа

DMA режими

- ▶ Режими на DMA трансфер
 - Крадење на циклуси
 - Burst режим
- ▶ Крадење на циклуси (*Cycle stealing*) се користи за да се пренесат збор по збор податоците на системската магистрала, т.е инструкциски циклус е суспендиран за да се пренесат податоци
- ▶ DMA контролерот му кажува на уредот да ја побара магистралата, направи низа трансфери и ја ослободи (bursts).
 - Поголеми трансфери
 - Подоцна добива CPU достап до магистралата

Чекори при печатење на текст



Програмиран I/O

```
copy_from_user(buffer, p, count);
for (i = 0; i < count; i++) {
    while (*printer_status_reg != READY) ;
    *printer_data_register = p[i];
}
return_to_user();
```

/ p is the kernel bufer */*
/ loop on every character */*
/ loop until ready */*
/ output one character */*

- Скица на програма за програмски воден I/O (polling или busy waiting)

Прекински воден I/O

```
copy_from_user(buffer, p, count);  
enable_interrupts( );  
while (*printer_status_reg != READY) ;  
*printer_data_register = p[0];  
scheduler( );
```

(a)

```
if (count == 0) {  
    unblock_user( );  
} else {  
    *printer_data_register = p[i];  
    count = count - 1;  
    i = i + 1;  
}  
acknowledge_interrupt( );  
return_from_interrupt( );
```

(b)

- ▶ Пример за печатење на текст со interrupt-driven I/O (a) Код кој се извршува кога е направен повик на процедурата за печатење (б) Сервисната рутина за прекилот

I/O со DMA

```
copy_from_user(buffer, p, count);  
set_up_DMA_controller( );  
scheduler( );
```

(a)

```
acknowledge_interrupt( );  
unblock_user( );  
return_from_interrupt( );
```

(b)

- ▶ Пример за печатење на текст со DMA (a) Код кој се извршува кога е направен повик на процедурата за печатење (b) Сервисната рутина за прекинот

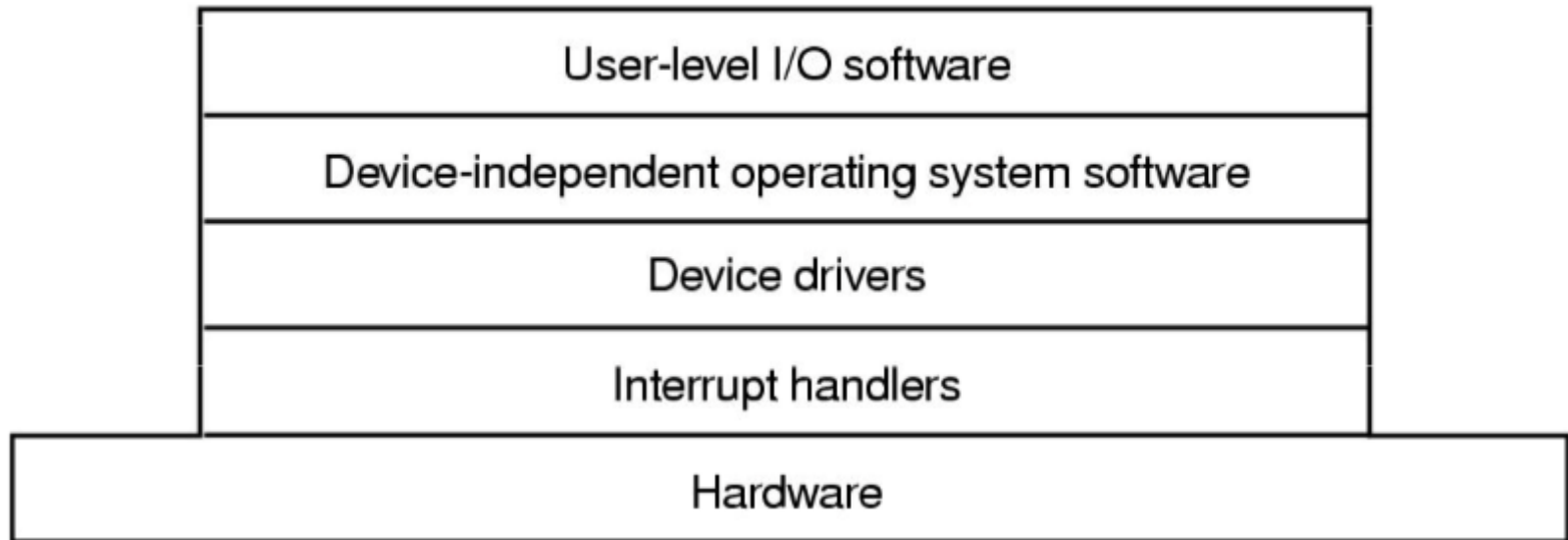
Принципи на I/O софтверот

- ▶ Независност од уредите
 - Програмите можат да пристапат до било која група на I/O уред на ист начин
 - Пример: floppy, hard drive, или CD-ROM
 - sort < input > output
- ▶ Униформно именување
 - Имињата на уредите и датотеките се во ист формати (string или integer)
 - Не зависи од типот на уредот
- ▶ Справување со грешките
 - Справувањето со грешките треба да биде што е можно поблиску до хардверот (Таму кај што се случила грешката)

Принципи на I/O софтверот

- ▶ Синхрон и Асинхрон трансфер
 - Блокирачки трансфер и прекински воден трансфер
- ▶ Баферирање
 - Податоците кои доаѓаат/одат од/кон уредите треба да се сместат на некоја привремена локација
- ▶ Деливи и Не деливи уреди
 - Дисковите се деливи (може повеќе процеси да им пристапуваат во исто време)
 - Траката не е делив уред

I/O софтверски нивоа



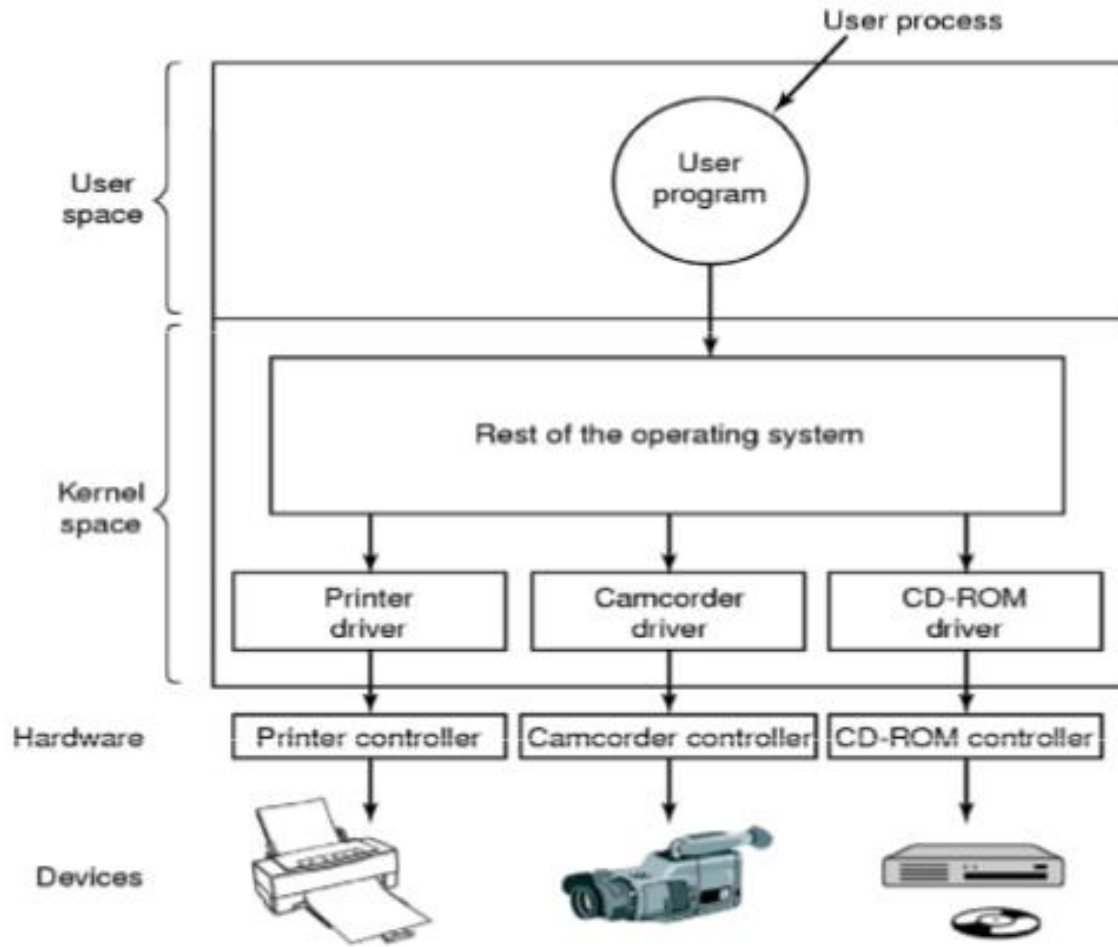
Обработка на прекин од ОС

1. Се снимаат регистрите.
2. Се нагодува контекстот за процедурата за опслужување на прекилот (TLB, MMU и табелата на страници)
3. Се нагодува магацинската меморија за процедурата за опслужување на прекин.
4. Се праќа Ask на контролерот на прекини и се овозможуваат прекините
5. Се стартува процедурата за опслужување
6. Се нагодува MMU контекстот за наредниот процес
7. Се вчитуваат регистрите на новиот процес.
8. Се стартува процесот

Драјвери

- ▶ Код потребен за управување до уредот
 - ОС специфичен и доставен од производителите
- ▶ Прави достап до контролните зборови за управување со уредот

Логички поглед кон драјвери



Карактеристики

- ▶ Еден драјвер за еден уред
 - Или за група поврзани уреди (SCSI драјвер)
- ▶ Драјвери во јадрениот простор
 - Може и во корисничкиот простор (MINIX 3)
- ▶ Оперативните системи дефинираат што треба да подржуваат блок ориентираните, а што знак ориентираните драјвери

Функции на драјверите

- ▶ Треба да обезбедат начин за
 - Читање
 - Запишување
 - Иницијализација на уредот
 - Управување со енергетските барања
- ▶ Исто така и можност за
 - Детекција
 - Корекција
 - Пријава на грешки кон погорните нивоа

Структура на драјверите

- ▶ Проверка на влезните параметри
- ▶ Конверзија на параметрите во хардверски специфичен формат
 - За HDD глава, сектор, цилиндер, трака
- ▶ Проверка дали уредот моментално се користи
 - Ако се користи барањето се става во ред на чекање
- ▶ Креирање на соодветна низа од команди за уредот во зависност од операцијата која треба да се заврши
- ▶ Испраќање на командите до уредот
- ▶ Драјверот чека да се заврши бараната акција од уредот

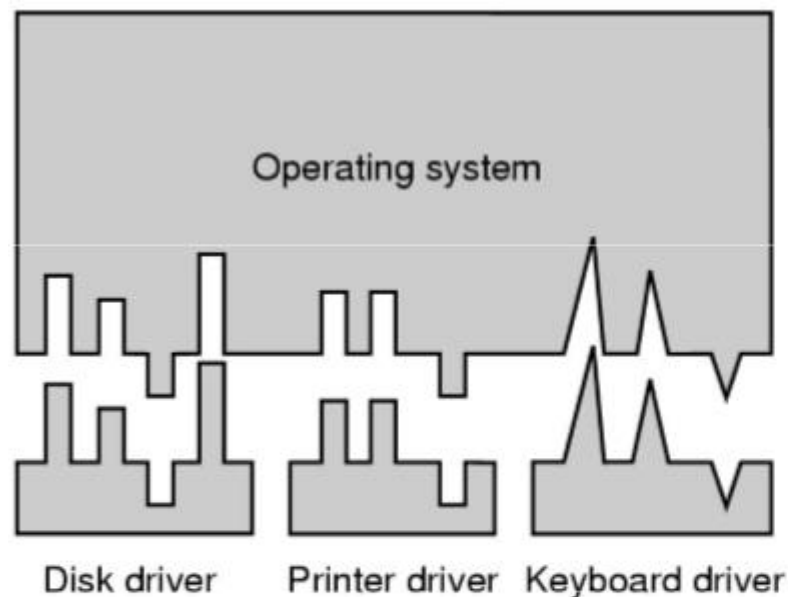
Проблеми на кои треба да се отпорни драјверите

- ▶ Стартување на втора инстанца од драјверот додека се уште не завршила со работа првата инстанца (reentrant driver)
- ▶ Дополнителни проблеми се јавуваат во Hot pluggable системите (уредите може да се додаваат и вадат додека работи компјутерот)

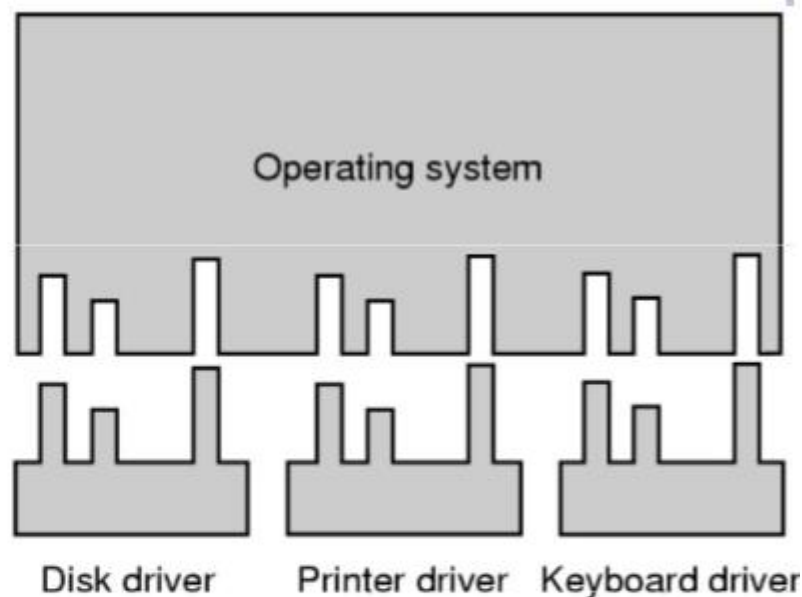
I/O Софтвер независен од Уредите

- ▶ Униформен интерфејс за драјверите за уредите
- ▶ Баферирање
- ▶ Обработка и пријавување на грешки
- ▶ Алокација и ослободување на неделивите уреди уреди
- ▶ Механизам за независност од големината на блокот кај уредот

Потреба од стандардизација



(a)



(b)

(a) Without a standard driver interface

(b) With a standard driver interface

Начин на функционирање

- ▶ Оперативниот систем дефинира множество на функции за секој драјвер
 - Читање, запишување, вклучување, исклучување, форматирање (за тврд диск)
- ▶ Драјверот има табела со функциски покажувачи
- ▶ Кога се вчитува драјверот, ОС ја запаметува адресата на оваа табела
- ▶ Табелата претставува интерфејс меѓу ОС и драјверот
- ▶ Сите драјвери од одредена класа мора да се придржуваат на интерфејсот

Мапирање

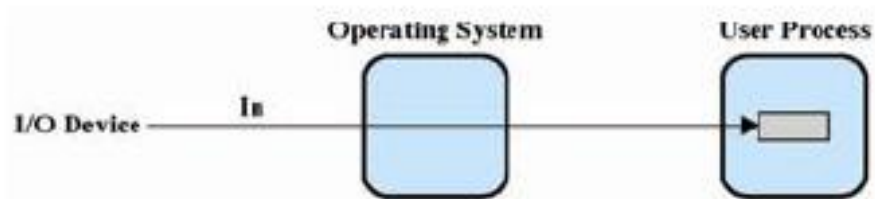
- ▶ Мапирање од симболичко име (/dev/disk0) во соодветен драјвер (Major device number)

brw-rw-----

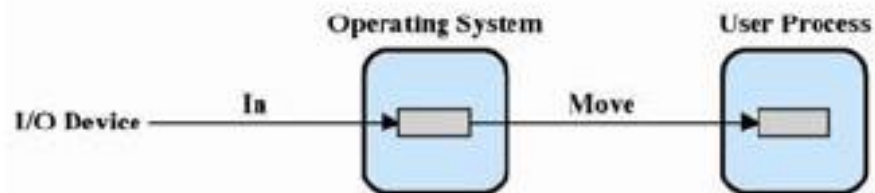
1 rootdisk8,0May51998/dev/sda

- ▶ Важи истата заштита како за обични датотеки

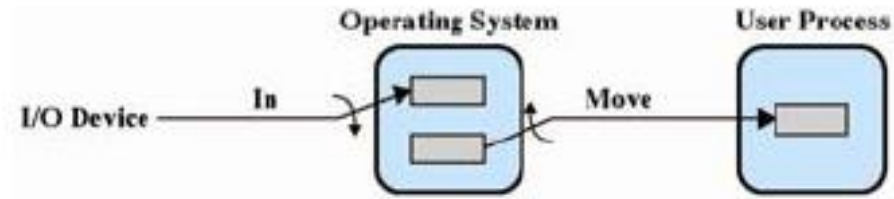
Шеми за I/O (input)



(a) No buffering



(b) Single buffering



(c) Double buffering



(d) Circular buffering

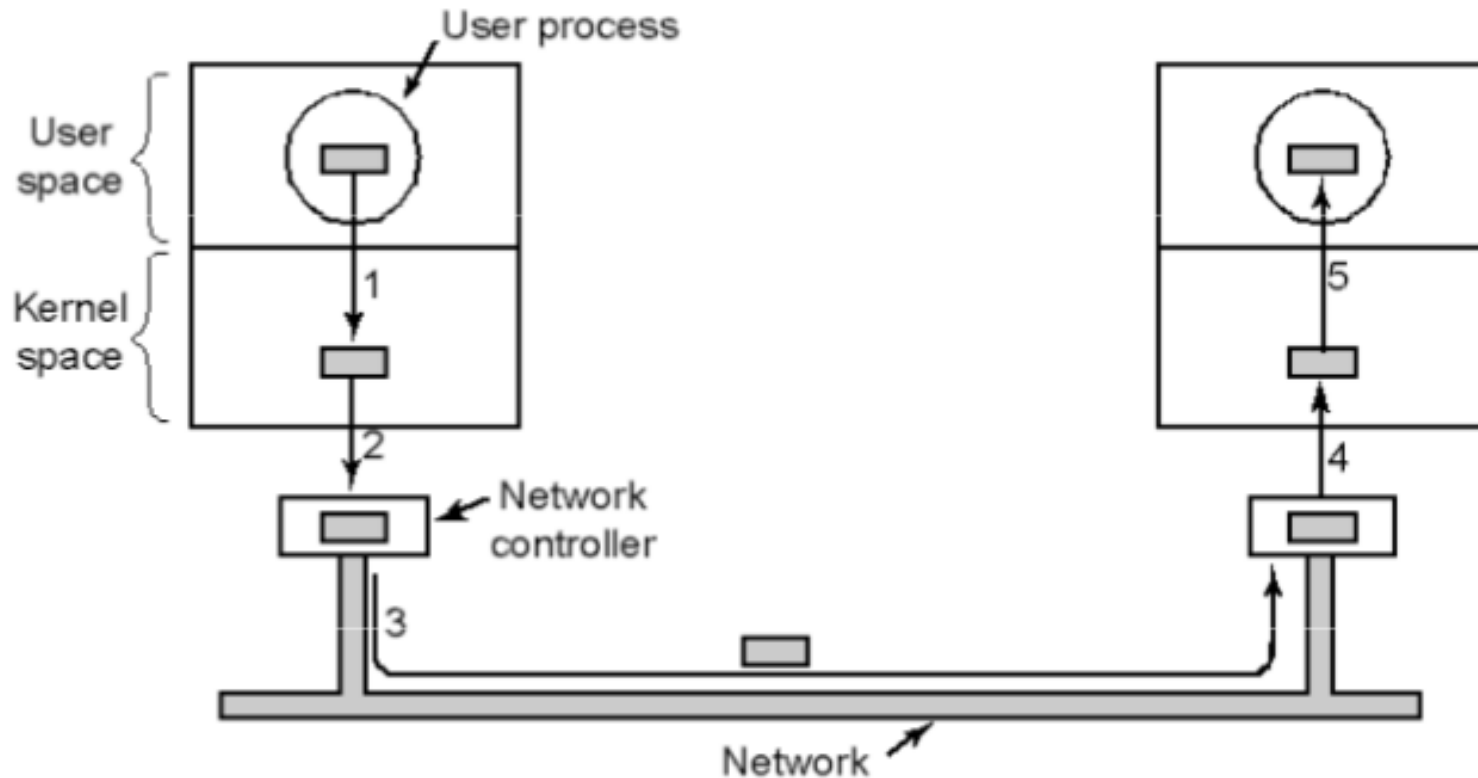
Единечно баферирање

- ▶ ОС доделува бафер во главната меморија за едно I/O барање
- ▶ Влезните трансфери се сместуваат во баферот
- ▶ Блокот се пренесува во корисничкиот простор кога е потребен
- ▶ Корисничкиот процес може да обработи еден податочен блок додека друг блок на податоци се преместува во баферот
- ▶ Замена (swapping) на кориснички процес е овозможен бидејќи внесот се прави во системската меморија, а не во корисничката меморија

Двојно и циркуларно баферирање

- ▶ Двојно баферирање
 - Користи 2 системски бафери наместо еден
 - Процесот може да пренесува податоци од еден бафер, додека ОС го празни или полни другиот
- ▶ Циркуларен бафер
 - Повеќе од 2 бафера се користат
 - Секој индивидуален бафер е една единка од циркуларниот бафер
 - Се користи кога I/O операцијата мора да држи чекор со процесите

Проблеми при баферирање (перформанси)



Останати ф-ии на I/O софтверот независен од уредите

- ▶ Обработка и пријавување на грешки
 - Програмски грешки (запишување на тастатура, читање од печатар, непостоечки уред)
 - I/O грешки (запишување на расипан блок на дискот или читање од исклучен уред)
- ▶ Алокација и ослободување на неделивите уреди (CD-ROM снимач, еден процес)
 - Директно отворање на специјалните датотеки за уредите (доколку не успее, се отфрла барањето)
 - Или блокирачки режим додека не се ослободи уредот (со редици)
- ▶ Механизам за независност од големината на блокот кај уредот

I/O софтвер во корисничкиот простор

- ▶ Библиотеки кои вршат I/O операции
`count = write(fd, buffer, nbytes);`
 - Пример: библиотеки за форматирање на текст во C/C++
- ▶ Процес за Spooling
 - Се справува со неделивите уреди
 - Најчесто се користи за печатач
 - Даemon процес и spooling именик

Преглед на комуникацијата помеѓу нивоата во I/O софтверот

