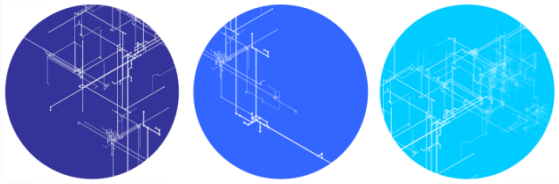
PB153
OPERAČNÍ SYSTÉMY A JEJICH ROZHRANÍ



I/O systém

12

HARDWARE

- HW pro I/O je značně rozmanitý
- Existují však určité běžně používané prvky
  - port
  - sběrnice (bus)
  - řadič (host adapter, controller)
- I/O zařízení jsou řízeny I/O instrukcemi (IN, OUT)
- Adresy I/O zařízení
  - uváděné přímo v I/O instrukcích (např. IN AL, DX : DX port, AL získaný bajt)
  - I/O se mapuje na přístup k paměti (např. grafická karta, videopaměť)
- Základní způsoby ovládání I/O
  - polling, programované I/O operace
    - aktivní čekání na konec operace
  - přerušení
  - DMA

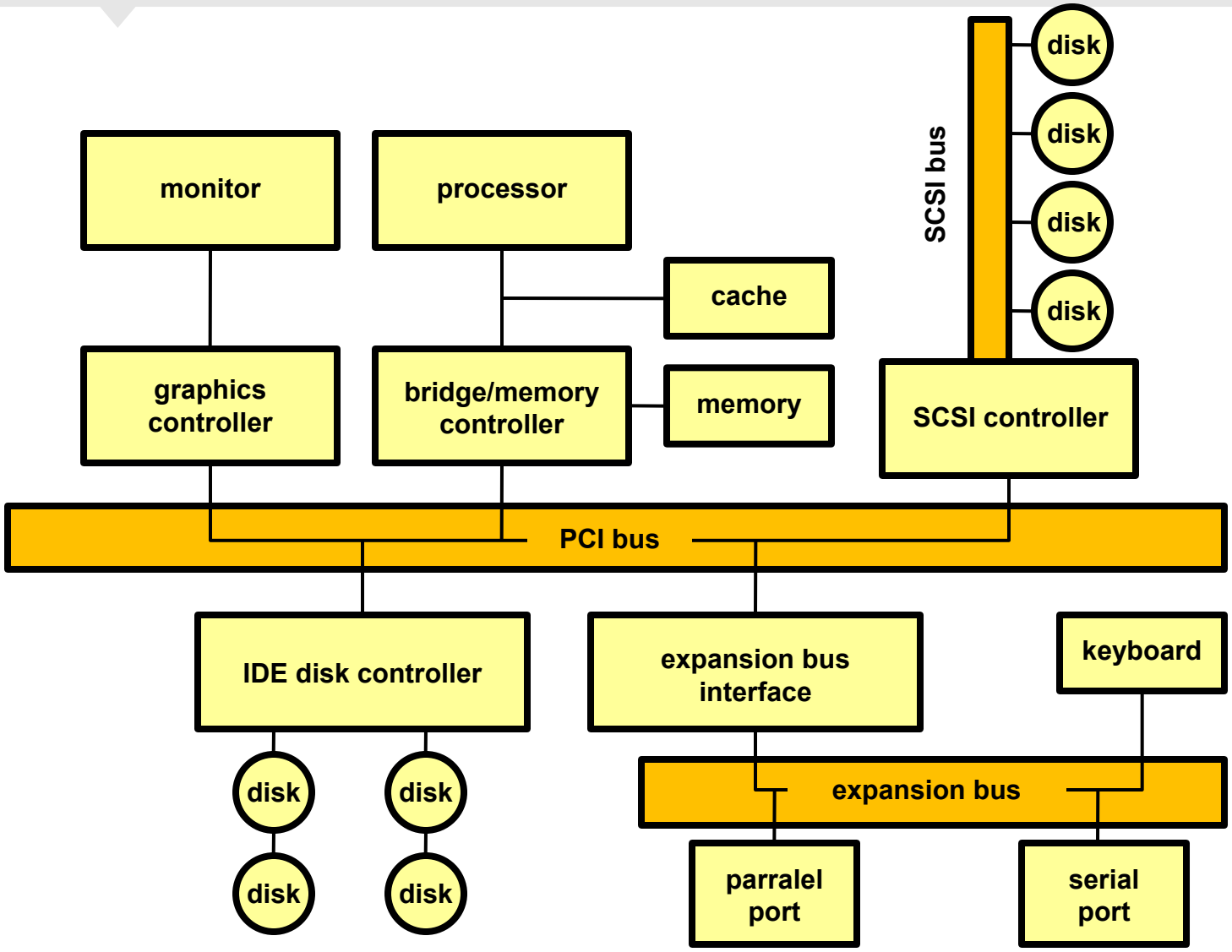
PŘÍKLAD: INSTRUKCE IN

IN-Input from Port

Optcode	Instruction	Op/En	64-Bit Mode	Compat/ Leg Mode	Description
E4 ib	IN AL, i*mm8*	A	Valid	Valid	Input byte from i*mm8* I/O port adress into AL.
E5 ib	IN AX, i*mm8*	A	Valid	Valid	Input word from i*mm8* I/O port adress into AX.
E5 ib	IN EAX, i*mm8*	A	Valid	Valid	Input dword from i*mm8* I/O port adress into EAX.
EC	IN AL,DX	B	Valid	Valid	Input byte from I/O port in DX into AL.
ED	IN AXDX	B	Valid	Valid	Input word from I/O port in DX into AL.
ED	IN EAX,DX	B	Valid	Valid	Input doubleword from I/O port in DX into EAX.

Copies the value from the I/O port specified with the second operand (source operand) to the destination operand (first operand). The source operand can be a byte-immediate or the DX register; the destination operand can be register AL, AX, or EAX, depending on the size of the port being accessed (8, 16, or 32 bits, respectively). Using the DX register as a source operand allows I/O port addresses from 0 to 65,535 to be accessed; using a byte immediate allows I/O port addresses 0 to 255 to be accessed.

SBĚRNICE PC


monitor
processor
cache
graphics controller
bridge/memory controller
memory
SCSI bus
disk
disk
disk
disk
SCSI controller
PCI bus
IDE disk controller
expansion bus interface
keyboard
disk
disk
disk
disk
expansion bus
parralel port
serial port

ROZMÍSTĚNÍ I/O PORTŮ V PC

I/O address range (hexadecimal)	device
000-00F	DMA controller
020-021	interrupt controller
040-043	timer
200-20F	game controller
2F8-2FF	serial port (secondary)
320-32F	hard-disk controller
378-37F	parallel port
3D0-3DF	graphics controller
3F0-3F7	diskette-drive controller
3F8-3FF	serial port (primary)

I/O PORTY

- I/O port se obvykle skládá ze 4 registrů
  - Data-in
    - Čtení vstupu od zařízení
  - Data-out
    - Pro zápis výstupu do zařízení
  - Status
    - Aktuální stav (data připravena, chyba, …)
  - Control
    - Ovládání zařízení, konfigurace, příkazy,…

PŘÍKLAD: SÉRIOVÝ PORT

- COM1 – porty od 3F8, COM2 – porty od 2F8

8250 SCC Registers

I/O Address (hex)	Description
3F8/2F8	Receive/Transmit data register. Alsto the L. O. byte of the Baud Rate Divisor Latch Register.
3F9/2F9	Interrupt Enable Register. Also the H. O. byte of the Baud Rate Divisor Register.
3FA/2FA	Interrupt Identification Register (read only).
3FB/2FB	Line Control Register
3FC/2FC	Modem Control Register.
3FD/2FD	Line Status Register (read only).
3FE/2FE	Modem Status Register (read only).
3FF/2FF	Shadow Receive Register (read only not available on original PCs)

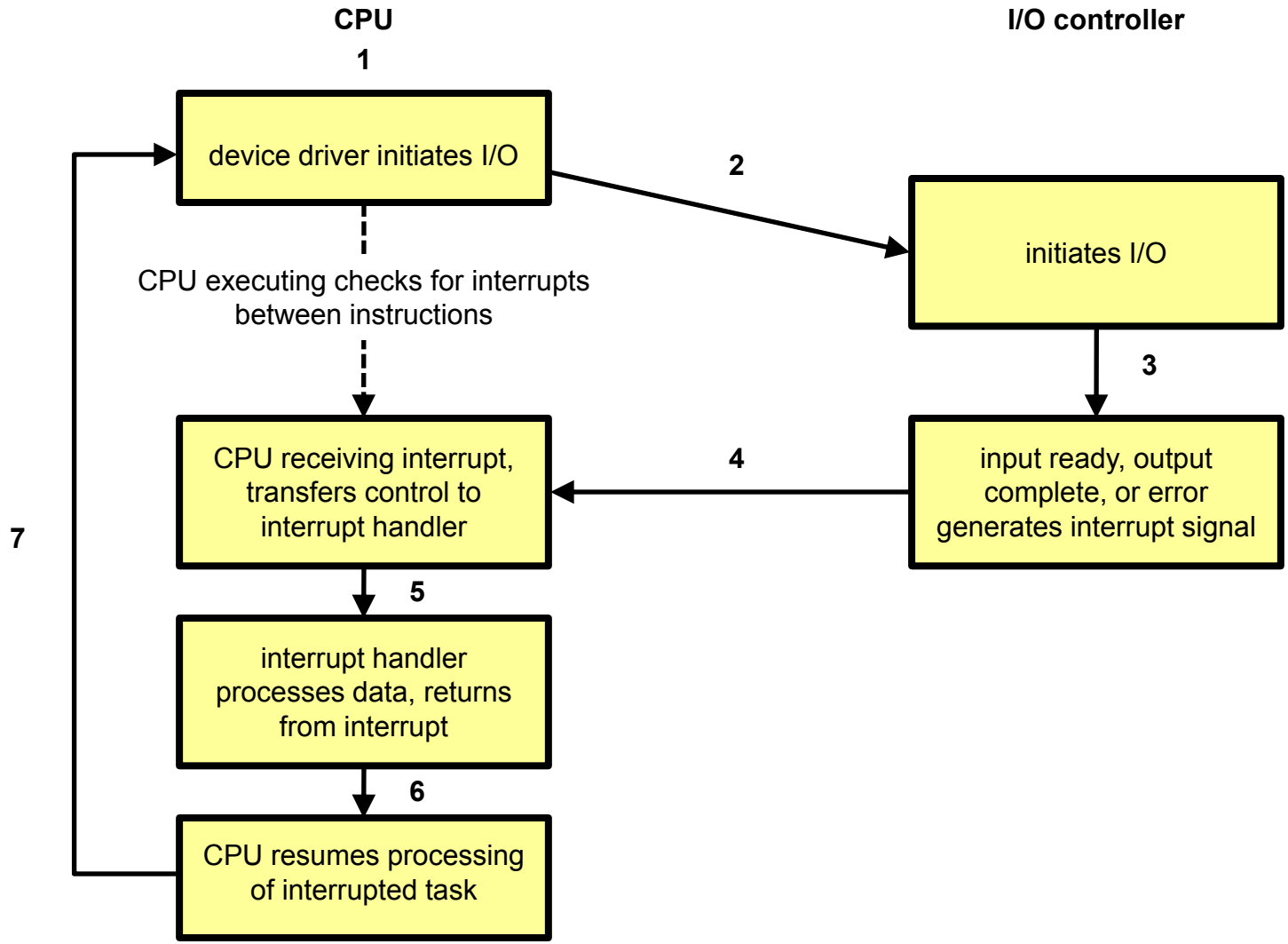
TECHNIKY PROVÁDĚNÍ I/O

- Programovaný I/O (busy-waiting)
  - opakovaně se ptám na stav zařízení
    - připraven
    - pracuje
    - chyba
- I/O řízený přerušením
  - zahájení I/O pomocí I/O příkazu
  - paralelní běh I/O s během procesoru
  - I/O modul oznamuje přerušením konec přenosu
- Direct Memory Access (DMA)
  - kopírování bloků mezi pamětí a I/O zařízením na principu kradení cyklů paměti
  - přerušení po přenosu bloku (indikace konce)

PŘERUŠENÍ

- Přerušení obsluhuje ovladač přerušení (kód OS)
- Maskováním lze některá přerušení ignorovat nebo oddálit jejich obsluhu
- Patřičný ovladač přerušení se vybírá přerušovacím vektorem
  - některá přerušení nelze maskovat
  - přerušení mohou být uspořádána podle priorit
- Přerušení se používá i pro řešení výjimek (nejsou asynchronní)

I/O CYKLUS ŘÍZENÝ PŘERUŠENÍM



VEKTOR PŘERUŠENÍ PROCESORU INTEL PENTIUM

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid optcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–*255*	maskable interrupts

DMA

- Přímý přístup do paměti
  (Direct Memory Access - DMA)
  - nahrazuje programovaný I/O při velkých přesunech dat
  - vyžaduje speciální DMA řadič
  - při přenosu dat se obchází procesor, přístup do paměti zajišťuje přímo DMA řadič
  - procesor a DMA soutěží o přístup k paměti

DMA: PŘÍKLAD

1. device driver is told to transfer disk data to buffer at address X

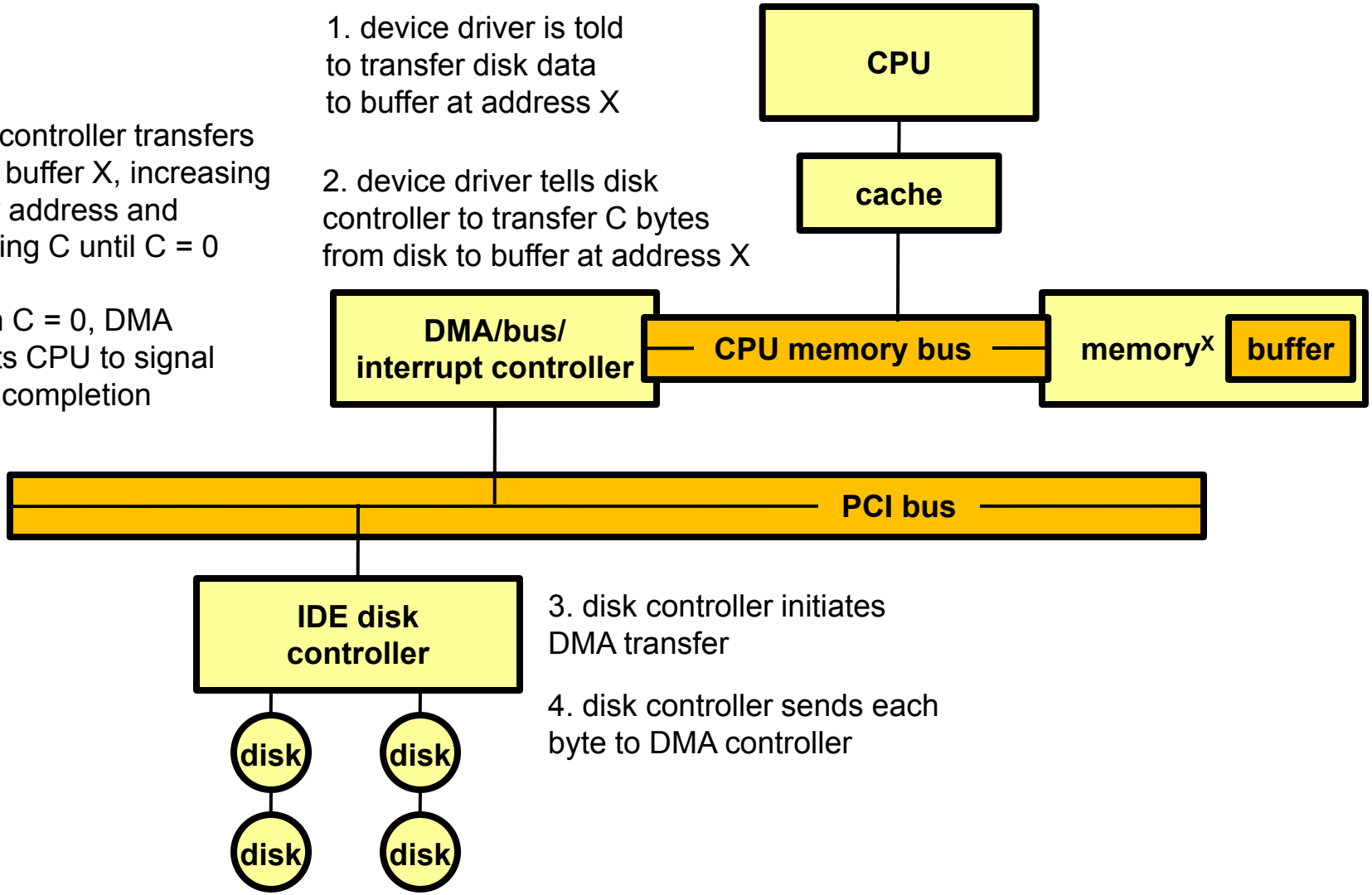
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X

3. disk controller initiates DMA transfer

4. disk controller sends each byte to DMA controller

5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0

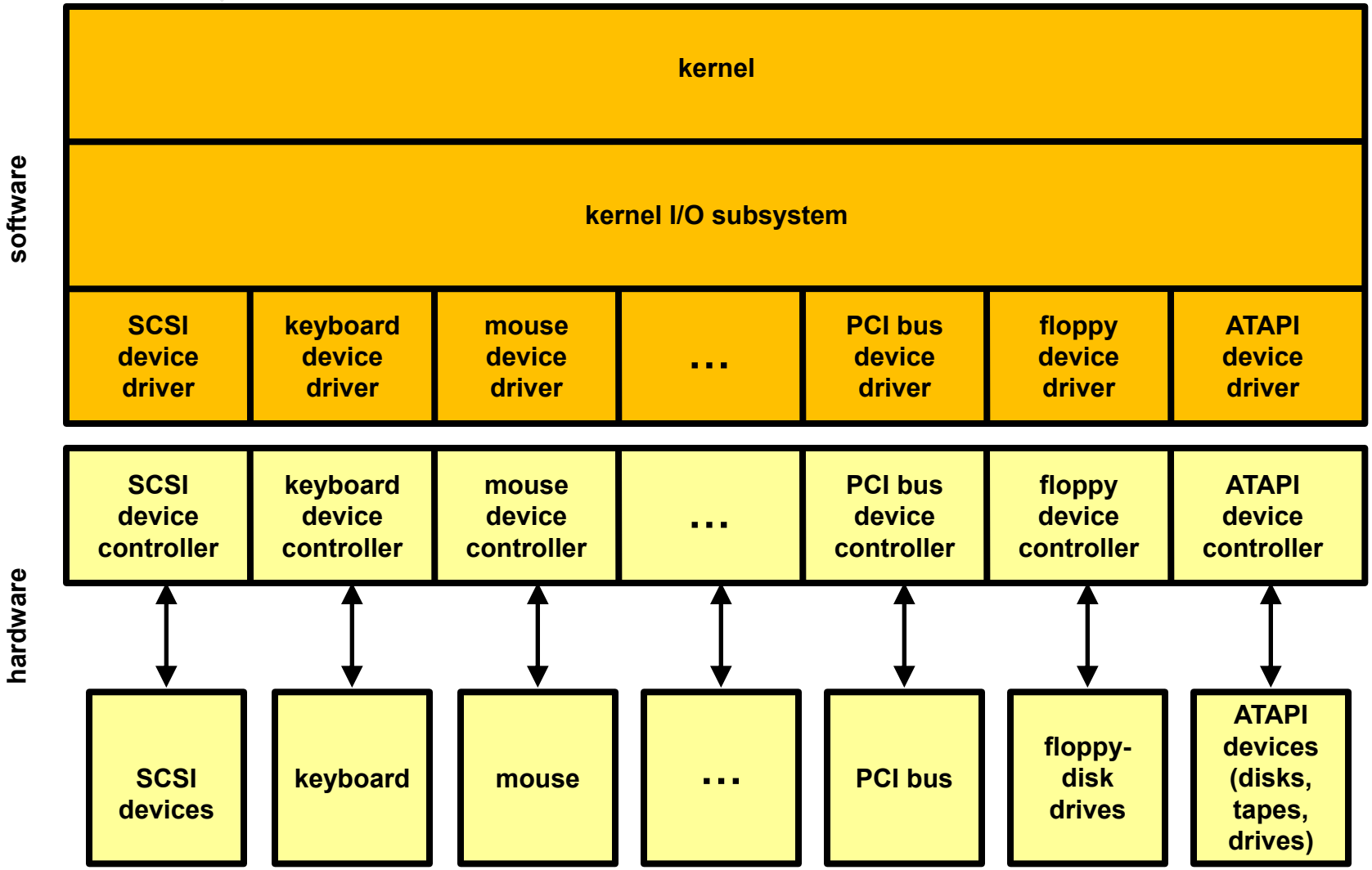
6. When C = 0, DMA interrupts CPU to signal transfer completion



APLIKAČNÍ ROZHRANÍ I/O

- Jádro OS se snaží skrýt rozdíly mezi I/O zařízeními a programátorům poskytuje jednotné rozhraní
- Dále vrstva ovladačů ukrývá rozdílnost chování I/O řadičů i před některými částmi jádra
- Některé vlastnosti I/O zařízení
  - mód přenosu dat: znakové (terminál) / blokové (disk)
  - způsob přístupu: sekvenční (modem) / přímý (disk)
  - sdílené/dedikované: klávesnice / páska
  - rychlost přenosu: vystavení, přenos, ...
  - read-write, read only, write only

I/O V JÁDŘE A HW



BLOKOVÁ A ZNAKOVÁ ZAŘÍZENÍ

- Bloková zařízení – typicky disk
  - příkazy: read, write, seek
  - logický způsob přístupu: obecný I/O nebo souborový systém
  - možný přístup formou souboru mapovaného do paměti
- Znaková – klávesnice, myš, sériový port
  - příkazy: get, put
  - nad nimi knihovní podprogramy pro další možnosti (např. řádková editace)

LINUX: SPECIÁLNÍ ZAŘÍZENÍ

```
NAME
       mknod - make block or character special files

SYNOPSIS
       mknod [OPTION]... NAME TYPE [MAJOR MINOR]

DESCRIPTION
       Create the special file NAME of the given TYPE.

       Mandatory arguments to long options are mandatory for short options too.

       -m, --mode=MODE
              set file permission bits to MODE, not a=rw - umask

       -Z, --context=CTX
              set the SELinux security context of NAME to CTX

       --help display this help and exit

       --version
              output version information and exit

       Both  MAJOR and MINOR must be specified when TYPE is b, c, or u, and they must be omitted when TYPE is
       p.  If MAJOR or MINOR begins with 0x or 0X, it is interpreted as hexadecimal; otherwise, if it  begins
       with 0, as octal; otherwise, as decimal.  TYPE may be:

       b      create a block (buffered) special file

       c, u   create a character (unbuffered) special file

       p      create a FIFO
```

LINUX: SPECIÁLNÍ ZAŘÍZENÍ

```
brw-rw----   1 root disk      1,   6 Mar 20 10:46 ram6
brw-rw----   1 root disk      1,   7 Mar 20 10:46 ram7
brw-rw----   1 root disk      1,   8 Mar 20 10:46 ram8
brw-rw----   1 root disk      1,   9 Mar 20 10:46 ram9
crw-rw-rw-   1 root root      1,   8 Mar 20 10:46 random
drwxr-xr-x   2 root root           60 Mar 20 10:45 raw
lrwxrwxrwx   1 root root            5 Mar 20 10:46 root -> sdbd1
lrwxrwxrwx   1 root root            4 Mar 20 10:46 rtc -> rtc0
crw-rw----   1 root root    254,    0 Mar 20 10:46 rtc0
lrwxrwxrwx   1 root root            3 Apr 24 12:24 scd0 -> sr0
brw-rw----   1 root disk     65, 160 Apr 24 03:11 sdaa
brw-rw----   1 root disk     65, 176 Apr 24 03:11 sdab
brw-rw----   1 root disk     65, 192 Apr 24 03:11 sdac
brw-rw----   1 root disk     65, 208 Apr 24 03:11 sdad
brw-rw----   1 root disk     65, 224 Apr 24 03:11 sdae
brw-rw----   1 root disk     65, 240 Apr 24 03:11 sdaf
brw-rw----   1 root disk     66,   0 Apr 24 03:11 sdag
brw-rw----   1 root disk     66,  16 Apr 24 03:11 sdah
brw-rw----   1 root disk     66,  32 Apr 24 03:11 sdai
brw-rw----   1 root disk     66,  48 Apr 24 03:11 sdaj
brw-rw----   1 root disk     66,  64 Apr 24 03:11 sdak
brw-rw----   1 root disk     66,  80 Apr 24 03:11 sdal
brw-rw----   1 root disk     66,  96 Apr 24 03:11 sdam
brw-rw----   1 root disk     66, 112 Apr 24 03:11 sdan
brw-rw----   1 root disk     66, 128 Apr 24 03:11 sdao
brw-rw----   1 root disk     66, 144 Apr 24 03:11 sdap
brw-rw----   1 root disk     66, 160 Apr 24 03:11 sdaq
brw-rw----   1 root disk     66, 176 Apr 24 03:11 sdar
brw-rw----   1 root disk     66, 192 Apr 24 03:11 sdas
brw-rw----   1 root disk     66, 208 Apr 24 03:11 sdat
brw-rw----   1 root disk     66, 224 Apr 24 03:11 sdau
brw-rw----   1 root disk     66, 240 Apr 24 03:11 sdav
brw-rw----   1 root disk     67,   0 Apr 24 03:11 sdaw
brw-rw----   1 root disk     67,  16 Apr 24 03:11 sdax
```

WINAPI: PRÁCE S DISKY

When opening a physical drive *x:*, the *lpFileName* string should be the following form: "\\.\PhysicalDrive*X*". Hard disk numbers start at zero. The following table shows some examples of physical drive strings.

String	Meaning
"\\.\PhysicalDrive0"	Opens the first physical drive.
"\\.\PhysicalDrive2"	Opens the third physical drive.

To obtain the physical drive identifier for a volume, open a handle to the volume and call the **DeviceIoControl** function with **IOCTL_VOLUME_GET_VOLUME_DISK_EXTENTS**. This control code returns the disk number and offset for each of the volume's one or more extents; a volume can span multiple physical disks.

For an example of opening a physical drive, see Calling DeviceIoControl.

When opening a volume or removable media drive (for example, a floppy disk drive or flash memory thumb drive), the *lpFileName* string should be the following form: "\\.*X*:". Do not use a trailing backslash (\), which indicates the root directory of a drive. The following table shows some examples of drive strings.

String	Meaning
"\\.\A:"	Opens floppy disk drive A.
"\\.\C:"	Opens the C: volume.
"\\.\C:\"	Opens the file system of the C: volume.

You can also open a volume by referring to its volume name. For more information, see Naming a Volume.

PŘÍKLAD: LINUX

- Mapování souboru do paměti

```
#include <sys/mman.h>

void *mmap(void *start, size_t length, int prot, int flags, int fd, off_t offset);

int munmap(void *start, size_t length);
```

The mmap() function asks to map length bytes starting at offset offset from the file (or other object) specified by the file descriptor fd into memory, preferably at address start. The prot argument describes the desired memory protection (and must not conflict with the open mode of the file). It is either PROT_NONE or is the bitwise OR of one or more of the other PROT_* flags…

SÍŤOVÁ ZAŘÍZENÍ

- Přístup k nim se značně liší jak od znakových, tak od blokových zařízení
  - proto mívají samostatné rozhraní OS
- Unix i Windows obsahující rozhraní nazývané „sockets“
  - separují síťové protokoly od síťových operací
  - přístup jako k souborům (včetně funkce *select*)
- Existuje celá řada přístupů k síťovým službám
  - Pipes (roury), FIFOs, streams, queues, mailboxes

BLOKUJÍCÍ A NEBLOKUJÍCÍ I/O

- Blokující
  - z hlediska procesu synchronní
  - proces čeká na ukončení I/O
  - snadné použití (programovaní), snadné porozumění (po provedení operace je hotovo to co jsem požadoval)
  - někdy však není dostačující (z důvodu efektivity)
- Neblokující
  - řízení se procesu vrací co nejdříve po zadání požadavku
  - vhodné pro uživatelské rozhraní, bufferovaný I/O
  - bývá implementováno pomocí vláken
  - okamžitě vrací počet načtených či zapsaných znaků
- Asynchronní
  - proces běží souběžně s I/O
  - konec I/O je procesu hlášen signály
  - obtížné na programovaní, složité používání, ale v případě vhodně promyšleného programu velice efektivní

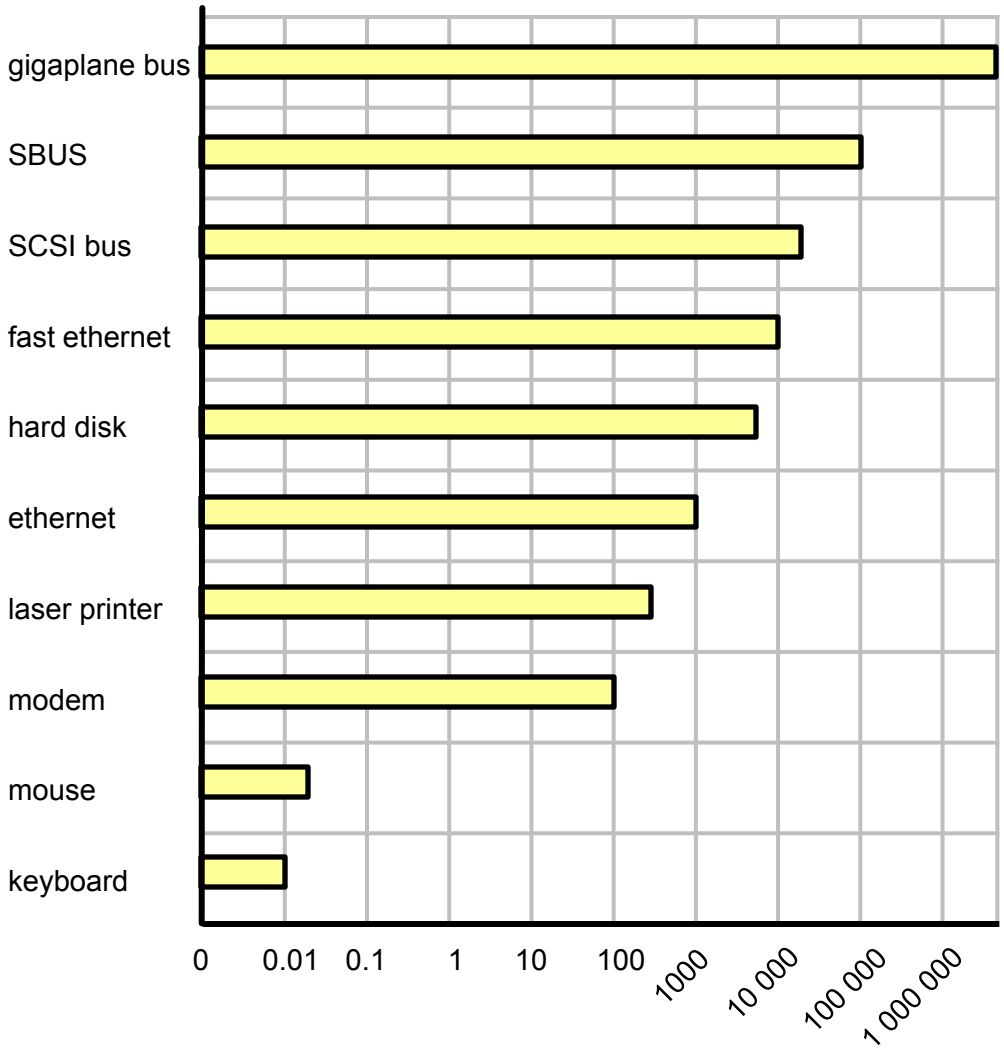
PŘÍKLAD: LINUX

- Blokující čtení
  - ssize_t read(int fd, void *buf, size_t count);
- Neblokující volání
  - int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);
- Asynchronní čtení
  - int aio_read(struct aiocb *aiocbp);
  - int aio_error(const struct aiocb *aiocbp);
  - ssize_t aio_return(struct aiocb *aiocbp);

I/O SUBSYSTÉM V JÁDRU

- Plánování
  - některé I/O operace požadují řazení do front na zařízení
  - některé OS se snaží o „spravedlnost“
- Vyrovnání (vyrovnávací paměti), buffering
  - ukládání dat v paměti v době přenosu k/ze zařízení
  - řeší rozdílnost rychlosti
  - řeší rozdílnost velikosti datových jednotek

PŘÍKLAD: SUN ENTERPRISE 6000



I/O SUBSYSTÉM V JÁDRU

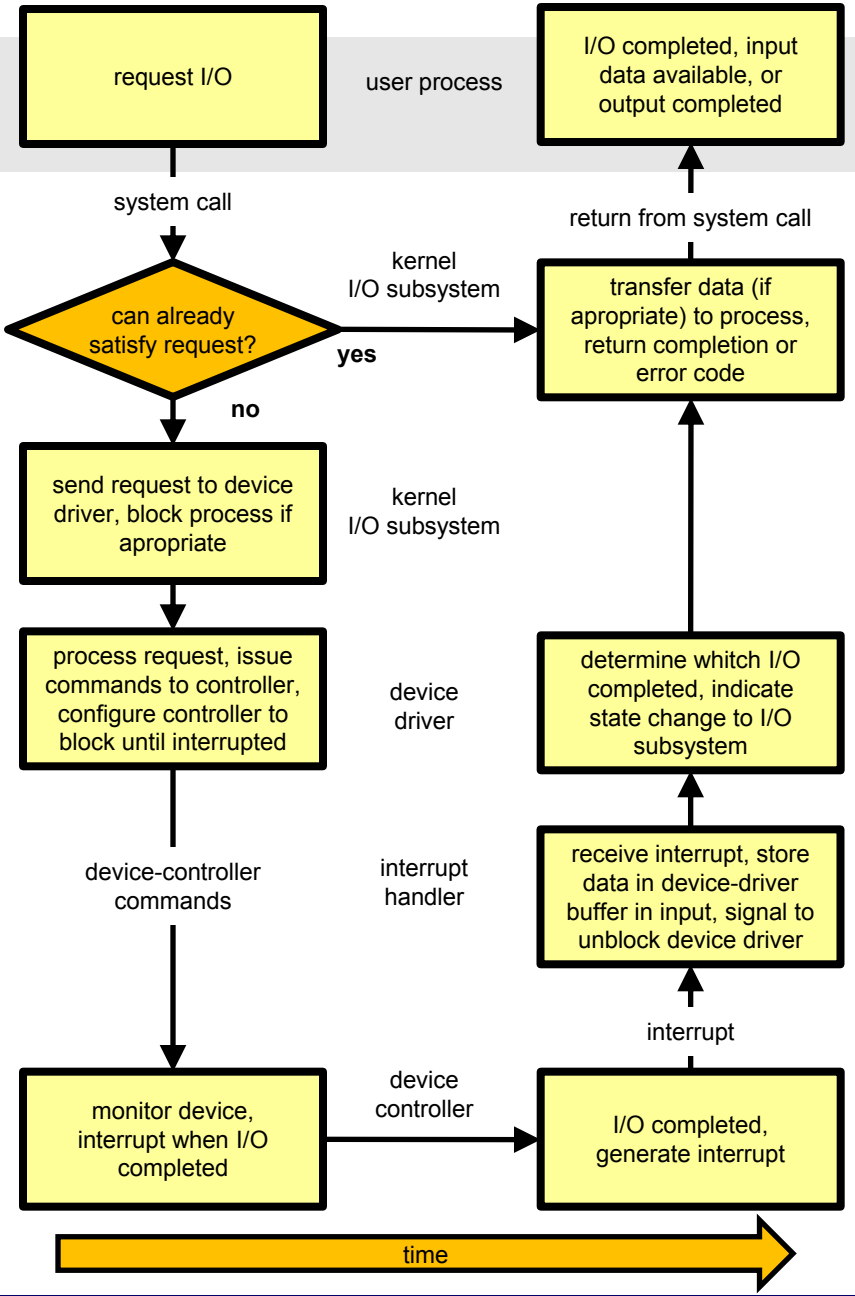
- Caching
  - rychlá paměť udržuje kopii dat
  - vždy pouze *kopii*
  - caching je klíčem k dosažení vysokého výkonu
- Spooling
  - udržování *fronty dat* určených k výpis na zařízení
  - pokud zařízení může vyřizovat požadavky pouze sekvenčně
  - typicky tiskárna
- Rezervace zařízení
  - exkluzivita přístupu k zařízení pro proces
  - rezervace / uvolnění – volání systému
  - pozor na uváznutí (deadlock)

CHYBOVÉ ŘÍZENÍ

- Vzpamatování se po poruše při chybě čtení z disku, zjištění nedostupnosti zařízení, po náhodné chybě zápisu, …
- Volání požadující I/O operaci získá číslo chyby
  - typicky záporná hodnota
- Udržuje se záznam o chybách v systému
  - pro následné analýzy
  - syslog, events (viewer), ...

PŘ.: I/O POŽ.

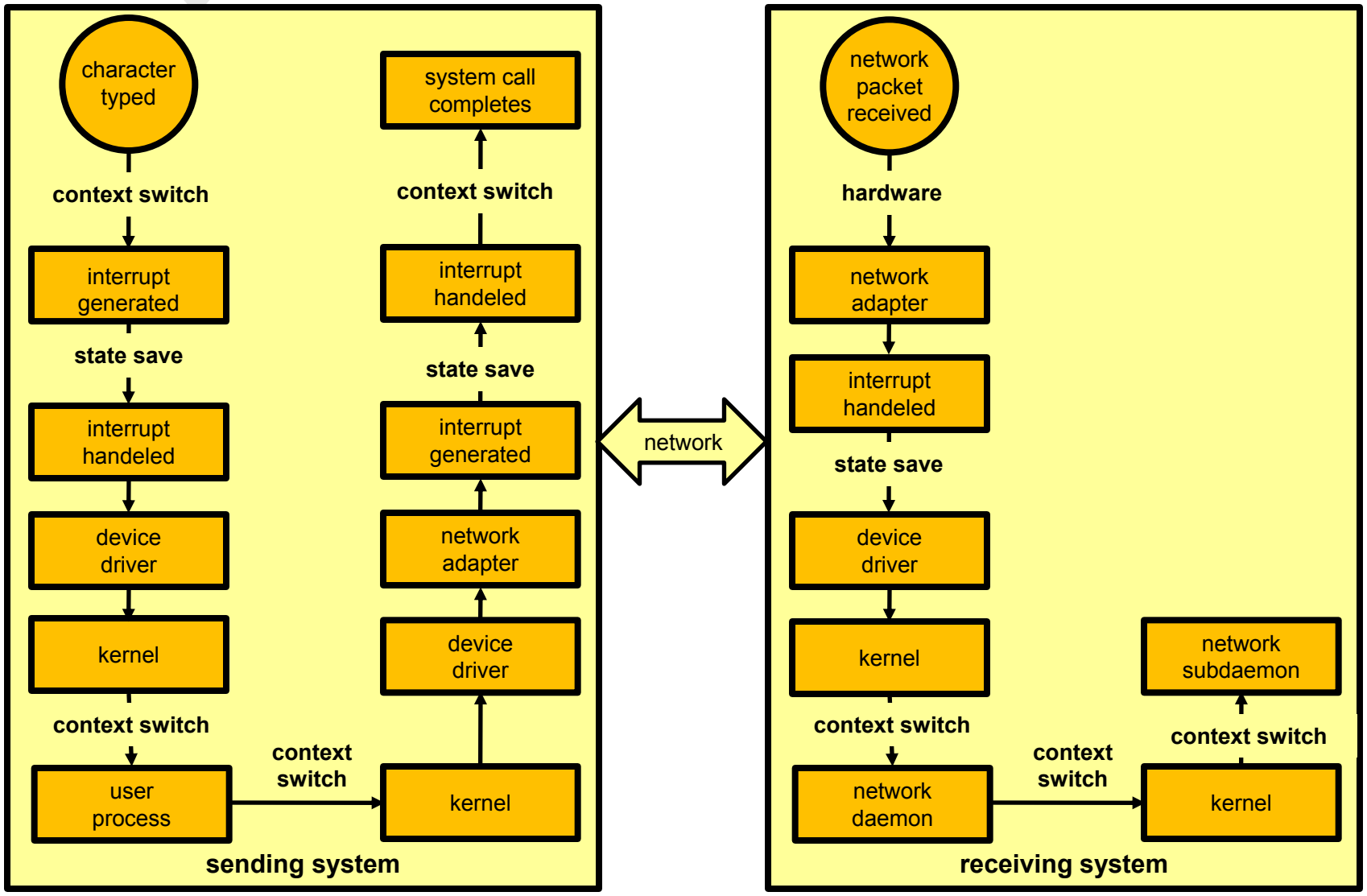
- Příklad čtení souboru z disku:
  1. Zjistíme, které zařízení obsahuje soubor
  2. Převedeme jméno na reprezentaci na zařízení
  3. Fyzicky přečteme data z disku do bufferu
  4. Získaná data zpřístupníme procesu
  5. Vrátíme řízení procesu



VÝKON

- I/O je nejvýznamnějším faktorem výkonu celého systému
  - CPU musí provádět ovladače a programy I/O části jádra
  - při přerušení se přepíná kontext
  - provádí se kopírování dat
  - zvláště významný je síťový provoz

PŘÍKLAD: SÍŤOVÁ APLIKACE



ZVYŠOVÁNÍ VÝKONU

- Omezujeme počet přepnutí kontextu
- Omezujeme zbytečné kopírování dat
- Omezujeme počet přerušení tím, že přenášíme delší bloky
- Využíváme všech výhod (funkcí) moderních řadičů
- Používáme co nejvíce DMA
- Všechny komponenty kombinujeme s cílem dosažení co nejvyšší propustnosti
  - CPU, paměť, sběrnice, I/O zařízení

Výukovou pomůcku zpracovalo

Servisní středisko pro e-learning na MU

CZ.1.07/2.2.00/28.0041
Centrum interaktivních a multimediálních studijních opor pro inovaci výuky a efektivní učení

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ