

# IV. Počítače a sítě

Update: 4. května 2018

## Obsah

1	Architektura univerzálních procesorů. Principy urychlování činnosti procesorů.	2
2	Základní vlastnosti monolitických počítačů a jejich typické integrované periférie. Možnosti použití.	9
3	Struktura OS a jeho návaznost na technické vybavení počítače.	12
4	Protokolová rodina TCP/IP.	21
5	Metody sdíleného přístupu ke společnému kanálu.	22
6	Problémy směrování v počítačových sítích. Adresování v IP, překlad adres (NAT).	23
7	Bezpečnost počítačových sítí s TCP/IP: útoky, paketové filtry, stavový firewall. Šifrování a autentizace, virtuální privátní síť.	24

# 1 Architektura univerzálních procesorů. Principy urychlování činnosti procesorů.

**Architektura počítačů** je náčrt struktury a funkčnosti systému. Je charakterizována výčtem **registrů** a jejich funkcí, vnitřních a vnějších **sběrnic**, způsobem **adresování** a **instrukčním souborem**.

**Registr** je malé úložiště dat v mikroprocesoru s rychlým přístupem, které slouží jako **pracovní paměť** během výpočtů.

**Sběrnice** je soustava vodičů pro **přenos informací** mezi více účastníky na principu „jeden vysílá, ostatní přijímají.“ Podle typu přenášené informace je dělíme na *datové*, *adresové* a *řídící*. V praxi však díky multiplexu může jít o jedny dráty.

## 1.1 Procesory CISC a RISC

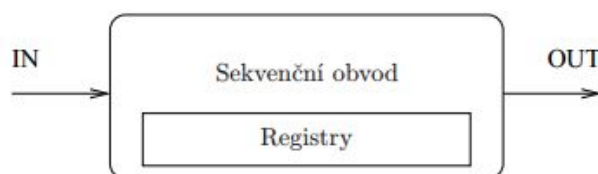
V dnešní době se ustálilo dělení počítačů do dvou základních kategorií podle typu používaného procesoru:

- **CISC** – počítač se složitým souborem instrukcí (*Complex Instruction Set Computer*)
- **RISC** – počítač s redukováným souborem instrukcí (*Reduced Instruction Set Computer*)

### 1.1.1 CISC

- procesory s **komplexním instrukčním souborem**
- instrukce mají **proměnlivou délku i dobu vykonání**
- **vysoká složitost instrukcí** → nutný systematický návrh řadiče procesoru
- vykonání strojové instrukce probíhá posloupností mikrooperací (předepsána mikroinstrukcí v řídící paměti)
- procesor obsahuje relativně **nízký počet registrů**
- operace provedená i **složenou instrukcí** (např. násobení) může být **nahrazena** sledem jednodušších strojových instrukcí (*sčítání a bitové posuny*) → mohou být ve výsledku vykonány rychleji, než hardwarově implementovaná složená varianta
- Označení CISC bylo zavedeno jako **protiklad** až poté, co se prosadily procesory RISC, které mají instrukční sadu naopak maximálně redukovanou (pouze jednoduché operace, tj. žádné složené, jsou stejně dlouhé a jejich vykonání trvá stejnou dobu).
- Obvyklou chybou je domněnka, že procesory CISC mají více strojových instrukcí, než procesory RISC. Ve skutečnosti nejde o absolutní počet, ale o počet různých druhů operací, které procesor sám přímo umí vykonat na hardwarové úrovni (tj. již z výroby). Procesor CISC tak může například paradoxně obsahovat jen jednu strojovou instrukci pro danou operaci (např. *logické operace*), zatímco procesor RISC může tuto operaci

obsahovat jako několik strojových instrukcí, které stejnou operaci umí provést nad různými registry.



Obrázek 1: Procesor (CISC) jako sekvenční obvod

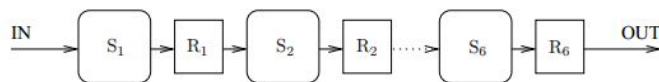
Krok		Význam
1.	<b>VI</b>	Výběr Instrukce
2.	<b>DE</b>	Dekódování
3.	<b>VA</b>	Výpočet Adresy
4.	<b>VO</b>	Výběr Operandu
5.	<b>PI</b>	Provedení Instrukce
6.	<b>UV</b>	Uložení Výsledku

	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>4</sub>	T <sub>5</sub>	T <sub>6</sub>	T <sub>7</sub>	T <sub>8</sub>	T <sub>9</sub>	T <sub>10</sub>	T <sub>11</sub>	T <sub>12</sub>	T <sub>13</sub>
VI	I <sub>1</sub>						I <sub>2</sub>						...
DE		I <sub>1</sub>						I <sub>2</sub>					
VA			I <sub>1</sub>						I <sub>2</sub>				
VO				I <sub>1</sub>						I <sub>2</sub>			
PI					I <sub>1</sub>						I <sub>2</sub>		
UV						I <sub>1</sub>						I <sub>2</sub>	

Tabulka 4: Postup provádění instrukcí procesorem CISC

### 1.1.2 RISC

- počet instrukcí a způsobů adresování je malý, ale zůstává úplný, aby bylo možno provést vše → v tomhle se liší od CISC
- instrukce jsou vytvořeny pomocí obvodu → jednodušší na výrobu než CISC
- širší sběrnice, rychlejší tok instrukcí a dat do procesoru
- instrukce jen nad registry
- navýšený počet registrů → delší program
- instrukce mají **jednotný formát** – délku i obsah
- komunikace s pamětí pouze pomocí instrukcí **LOAD / STORE**
- **každý strojový cyklus znamená dokončení jedné instrukce**
- používá se **zřetězené zpracování instrukcí**
- řešení problémů s frontou instrukcí
- mikroprogramový řadič může být nahrazen rychlejším obvodem
- přenášejí složitost technologického řešení do programu (překladače)
- představitelé *ARM, MOTOROLA 6800, INTEL i960, MIPS R6000*



Obrázek 2: Zfetěžené zpracování (v procesoru RISC)

	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>4</sub>	T <sub>5</sub>	T <sub>6</sub>	T <sub>7</sub>	T <sub>8</sub>	T <sub>9</sub>	T <sub>10</sub>	T <sub>11</sub>	T <sub>12</sub>
VI	I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	I <sub>4</sub>	I <sub>5</sub>	I <sub>6</sub>	I <sub>7</sub>	...				
DE		I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	I <sub>4</sub>	I <sub>5</sub>	I <sub>6</sub>	I <sub>7</sub>				
VA			I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	I <sub>4</sub>	I <sub>5</sub>	I <sub>6</sub>	I <sub>7</sub>			
VO				I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	I <sub>4</sub>	I <sub>5</sub>	I <sub>6</sub>	I <sub>7</sub>		
PI					I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	I <sub>4</sub>	I <sub>5</sub>	I <sub>6</sub>	I <sub>7</sub>	
UV						I <sub>1</sub>	I <sub>2</sub>	I <sub>3</sub>	I <sub>4</sub>	I <sub>5</sub>	I <sub>6</sub>	I <sub>7</sub>

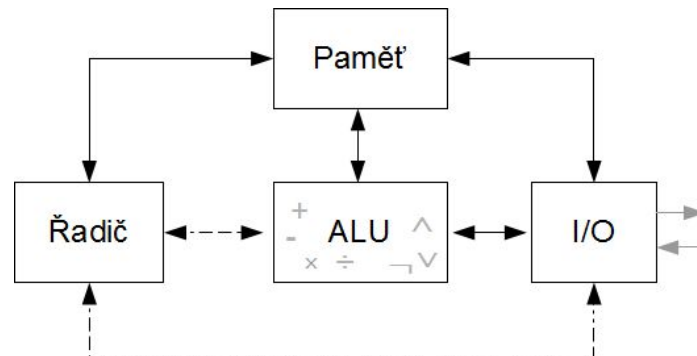
Tabulka 5: Zfetěžené provádění instrukcí procesorem RISC

## 1.2 Von Neumannovo schéma počítače

John Von Neumann definoval v roce **1945** základní koncepci počítače (EDVAC) **řízeného obsahem paměti**. Od té doby se objevilo několik odlišných modifikací, ale v podstatě se **počítače v dnešní době** konstruují podle tohoto modelu. Ve svém projektu si von Neumann stanovil určitá kritéria a principy, které musí počítač splňovat, aby byl použitelný univerzálně. Můžeme je ve stručnosti shrnout do následujících bodů:

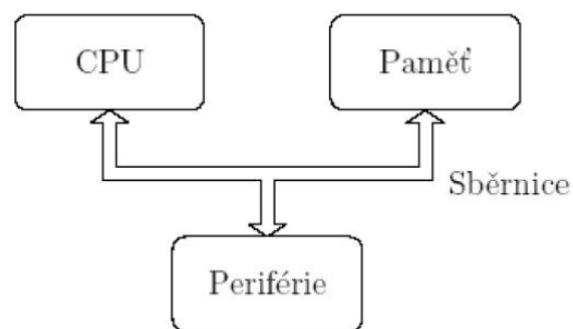
- Počítač se skládá z paměti, řídicí jednotky, aritmeticko-logické jednotky, vstupní a výstupní jednotky.
  - **ALU** - aritmeticko-logická jednotka (arithmetic-logic unit) - jednotka provádějící veškeré aritmetické výpočty a logické operace. Obsahuje sčítačky, násobičky a komparátory.
  - **Operační paměť** - slouží k uchování zpracovávaného programu, zpracovávaných dat a výsledků výpočtu
  - **Řídicí jednotka** - řídí činnost všech částí počítače. Toto řízení je prováděno pomocí řídicích signálů, které jsou zasílány jednotlivým modulům. Řadiči jsou pak zpět zasílané stavové hlášení. Dnes řadič spolu s ALU tvoří jednu součástku, a to procesor neboli CPU (Central Processing Unit).
  - **Vstup/ Výstup** - zařízení určené pro vstup dat, a výstup zpracovaných výsledků.
- Struktura pc je **nezávislá na typu řešené úlohy** (univerzálnost), počítač se programuje obsahem paměti.
- Následující krok počítače je závislý na kroku předešlém.
- **Instrukce** a **data** jsou v téže paměti.
- Paměť je rozdělena do **paměťových buněk stejné velikosti (Byte)**, jejichž pořadová čísla se využívají jako adresy.
- Program je tvořen posloupností instrukcí, které se vykonávají jednotlivě v pořadí, v jakém jsou zapsány do paměti.

- Změna pořadí prováděných instrukcí se provádí **skokovými instrukcemi** (podmíněné nebo nepodmíněné skákání na adresy).
- Čísla, instrukce, adresy a znaky se značí v **binární soustavě**.



### Nevýhody Von Neumannovy koncepce ve srovnání s dnešními PC

- Podle von Neumannova schématu počítač pracuje **vždy nad jedním programem**. Toto vede k velmi špatnému využití strojového času. Dnes je obvyklé, že počítač **zpracovává paralelně více programů** zároveň - tzv. **multitasking**
- Počítač může mít i více jak jeden procesor.
- Podle Von Neumanova schématu mohl počítač pracovat pouze v tzv. **diskrétním režimu**, kdy byl do paměti počítače zaveden program, data a pak probíhal výpočet. V průběhu výpočtu již nebylo možné s počítačem dále interaktivně komunikovat.
- Dnes existují **vstupní/výstupní** zařízení, např. pevné disky a páskové mechaniky, které umožňují vstup i výstup.
- Program se do paměti nemusí zavést celý, ale je možné zavést pouze jeho část a ostatní části zavádět až v případě potřeby.



### Výhody

- + **Rozdělení paměti** pro kód a data určuje programátor, řídicí jednotka přistupuje pro data i instrukce jednotným způsobem.

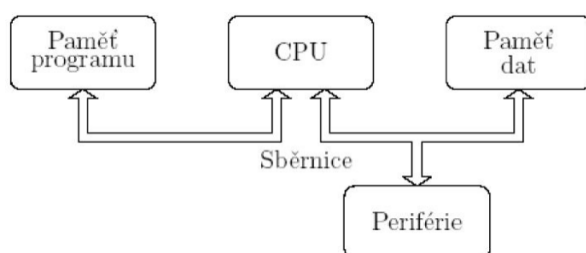
- + **Jedna sběrnice** -> jednodušší levnější výroba.

### Nevýhody

- **Společné uložení dat a kódu** může mít za následek přepsání vlastního programu
- **Jedna sběrnice** je omezující.

## 1.3 Harvardské schéma počítače

Několik let po von Neumannovi, přišel vývojový tým odborníků z Harvardské univerzity s vlastní koncepcí počítače, která se sice od Neumannovy příliš nelišila, ale odstraňovala některé její nedostatky. V podstatě jde pouze o **oddělení paměti pro data a program**. Abychom si mohli obě koncepce porovnat, můžeme vycházet ze zjednodušených schémat.



### Výhody

- + **Program se nepřepíše** (oddělené paměti pro data a program).
- + Dvě sběrnice umožňují **paralelní** načítání instrukcí a dat.
- + Paměti mohou být vyrobeny **odlišnými technologiemi** a každá může mít jinou nejmenší adresovací jednotku (8 bitů pro instrukce a 8, 16 nebo 32 pro data).

### Nevýhody

- 2 sběrnice mají **vyšší nároky na vývoj** řídicí jednotky a jsou také dražší a složitější na výrobu.
- Paměť je **rozdělena** už od **výrobce**.
- Nevyužitou část dat **nelze využít** pro program a obráceně.

## 1.4 Principy urychlování činnosti procesorů

Techniky urychlování výpočtu v hardwaru:

- speciální kódování dle potřeby dané úlohy
- speciální výpočetní jednotky dle potřeby dané úlohy (FFT – rychlý fourierova transformace)

- paralelní zpracování (násobné výpočetní jednotky)
- zřetězové zpracování instrukcí (pipelining)

#### 1.4.1 Paralelní zpracování

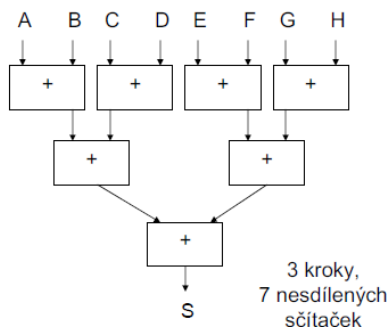
Zpracování více elementárních úloh běží současně.

$$\text{Př. } S = A + B + C + D + E + F + G + H$$

SW: **sekvenčně**

```
S = A + B;
S = S + C;
S = S + D;
S = S + E;
S = S + F;
S = S + G;
S = S + H;
```

7 kroků,  
1 sdílená  
sčítačka



#### 1.4.2 Zřetěžené zpracování instrukcí (pipelining)

Princip zřetěžení se značně překrývá s principy procesorů RISC. Základní myšlenkou je **rozdělení zpracování jedné instrukce** mezi různé části procesoru a tím i dosažení možnosti **zpracovávat více instrukcí** najednou. Pro dosažení tohoto zřetěžení je nutné rozdělit úlohu do posloupnosti dílčích úloh, z nichž každá může být vykonána **samostatně**, např. oddělit načítání a ukládání dat z paměti od provádění výpočtu instrukce a tyto části pak mohou běžet souběžně. To znamená že musíme osamostatnit jednotlivé části sekvenčního obvodu tak, aby každému obvodu odpovídala jedna fáze zpracování instrukcí. Všechny fáze musí být **stejně časově náročné**, jinak je rychlost **degradována** na nejpomalejší z nich. Fáze zpracování je rozdělena minimálně na 2 úseky:

- Načtení a dekódování instrukce.
- Provedení instrukce a případné uložení výsledku.

Zřetěžení se stále vylepšuje a u novějších procesorů se již můžeme setkat stále s více řetězci rozpracovaných informací (více pipelines), dnes je standardem 5 pipelines.

#### Problém

Největší problém spočívá v **plnění zřetěžené jednotky**, hlavně při provádění podmíněných skoků, kdy během stejného počtu cyklů se vykoná více instrukcí. U pipelingu se instrukce následující po skoku vyzvedává dřív, než je skok dokončen. **Primitivní implementace** vyzvedává vždy **následující instrukci**, což vede k tomu, že se vždy mýlí, pokud je skok nepodmíněný. Pozdější implementace mají **jednotku předpovídání skoku (1bit)**, která

vždy správně **předpoví nepodmíněný skok** a s použitím cache se záznamem předchozího chování programu se pokusí předpovědět i cíl podmíněných skoků nebo skoků s adresou v registru nebo paměti. V případě, že se predikce nepovede, bývá nutné vyprázdnit celou pipeline a začít vyzvedávat instrukce ze správné adresy, což znamená relativně **velké zdržení**. Související problémem je přerušení.

### Plnění fronty instrukcí

Pokud se dokončí skoková instrukce, která odkazuje na jinou část kódu, musejí být instrukce za ní zahozeny (*problém plnění fronty instrukcí*)

- u malého zřetězení **neřešíme**
- používání bublin na vyprázdnění pipeline, **naplnění prázdnými instrukcemi**
- **predikce skoku** – vyhrazen jeden bit předurčující, zda se skok provede či nikoliv
  - **Statická** – součást instrukce → řeší programátor nebo kompilátor
  - **Dynamická**
    - **jednobitová** – zaznamenává jestli se skok provedl, či ne (1/ 0)
    - **dvoubitová** – metoda zpožděného skoku → v procesoru řeší se např. tabulkou s 4 kB instrukcí

Zřetězené zpracování přináší urychlení výpočtu nejen v procesorech, ale i jiných číslicových obvodech (např. pro zpracování obrazu, bioinformatických dat apod.). Pokud použijeme zřetězené zpracování, musíme dodat řadu podpůrných obvodů a řešit řadu nových problémů. Moderní procesory používají kromě zřetězení i další koncepty:

- **superskalární architektura** (zdvojení) – když nastane podmíněný skok, začnou se vykonávat instrukce obou variant, nepotřebná část se pak zahodí. Tento způsob, pak vyžaduje vyřešit ukládání výsledku.
- **VLIW procesory** – má více ALU – tzn. může zároveň dělat více operací → k tomu složí dlouhé instrukce.
- **vektorové procesory** – je navržený tak, aby dokázal vykonávat matematické operace nad celou množinou čísel v daném čase. Je opakem skalárního procesoru, který vykonává jednu operaci s jedním číslem v daném čase.
- **multivláknové procesory**



## 2 Základní vlastnosti monolitických počítačů a jejich typické integrované periférie. Možnosti použití.

### Monolitické počítače (mikroprocesory)

- Mikroprocesory, mikrokontroléry, minipočítače jsou další názvy pro monolitické počítače.
- Jsou to malé počítače integrované v jediném pouzdře (all in one).
- Mají širokou oblast využití.
- Využívá se Harvardské koncepce, což umožňuje aplikovat paměti pro data a program různých technologií.
- Zjednodušené rysy architektury RISC.
- INTEL 8051 (standart), ATMEL, MICROCHIP PIC.
- V monolitických počítačích můžeme najít dva základní typy periférií (vstupní/výstupní).
- **Rozdělení pamětí:**
  - **pro data** – používáme většinou paměti energeticky závislé typu **RWM–RAM** (*Read–Write Memory–Random Access Memory*), tedy paměť s libovolným přístupem pro čtení i zápis. Jsou vyráběny jako statické (uchování paměti po celou dobu napájení), jejich paměťové buňky jsou realizovány jako klopné obvod.
  - **pro program** – se používají paměti typu **ROM** (*Read–Only Memory*) určené především ke čtení (paměť je uchována i po odpojení napájení). Mezi nejčastěji používané paměti patří **EPROM**, **EEPROM** (*Electrically Erasable Programmable Read–Only Memory*), **PROM** (*Programmable Read Only Memory*) a **Flash**.

### Organizace paměti

- **Střadačové (pracovní) registry** - ve struktuře procesoru jsou obvykle **1-8-16** základních pracovních registrů, jsou nepoužívanější. Ukládají se do nich **aktuálně zpracovávaná data** a jsou nejčastějším operandem strojových instrukcí (to na co se instrukce v závorkách odkazují). A také se do nich nejčastěji ukládají výsledky operací. Nejsou určeny pro dlouhodobé ukládání dat. Nejrychlejší.
- **Univerzální zápisníkové registry** – jsou jich desítky až stovky. Slouží pro ukládání **nejčastěji používaných dat**. Instrukční soubor obvykle dovoluje, aby se část strojových instrukcí prováděla přímo s těmito registry. Formát strojových instrukcí ovšem obvykle nedovoluje adresovat velký rozsah registru, proto se implementuje několik stejných skupin registru vedle sebe, s možností mezi skupinami přepínat - **registrové banky**.
- **Paměť dat RWM** - slouží pro ukládání **rozsáhlejších** nebo **méně používaných dat** (z těch předešlých nejméně používaný). Instrukční soubor obvykle nedovoluje s

obsahem této paměti přímo manipulovat, kromě instrukcí přesunových. Těmi se data přesunou např. do pracovního registru. Některé procesory dovolují, aby data z této paměti byla použita jako druhý operand strojové instrukce, výsledek ale nelze zpět do této paměti uložit přímo. Nejpomalejší.

## Zdroje synchronizace

- krystal (křemenný výbrus) – jsou drahé ale přesné
- keramický rezonátor
- obvod RC – snadno integrovatelný
- obvod LC – méně časté

## Ochrana proti rušení

Na prvním místě jde o ochranu **mechanickou**. Odolávat náhodným nárazům, nebo i trvalým vibracím nebo elektromagnetickým vlivům z okolí. Pro odstranění chyb, které nastanou působením vnějších vlivů nebo chyby programátora, je v mikropočítačích implementován speciální obvod nazývaný **WATCHDOG** → provede reinicializaci mikropočítače pomocí vnitřního RESETu, například při zacyklení. Watchdog (WDT) se řadí mezi elektrické ochrany, mezi které můžeme zařadit **BROWN-OUT** – ochrana proti podpěti.

## Typické periferie

**Periferie** - obvody, které zajišťují komunikaci mikropočítače s okolím.

1. **Vstupní a výstupní brány** - Nejjednodušší a nejčastěji používané rozhraní pro vstup a výstup informací je u mikropočítačů **paralelní brána - port**. Bývá obvykle organizována jako **4** nebo **8 jednobitové vývody**, kde lze současně zapisovat i číst logické informace 0 a 1. U většiny bran lze jednotlivě **nastavit**, které bitové vývody budou sloužit jako **vstupní** a které jako **výstupní**. Na vstupu je **Schmittův klopný obvod**. U mnoha mikropočítačů jsou brány implementovány tak, že s nimi instrukční soubor může pracovat jako s množinou vývodu, nebo jako s jednotlivými bity.
2. **Čítače a časovače** - Do skupiny nejpoužívanějších periférií mikropočítače určitě patří čítače a časovače. *Časovač* se od *čítače* příliš neliší. Není, ale **inkrementován vnějším signálem**, ale přímo *vnitřním hodinovým signálem* používaným pro řízení samotného mikropočítače. Lze tak podle přesnosti zdroje hodinového signálu zajistit řízení událostí a chování v reálném čase. Při přetečení časovače se i zde může automaticky předávat signál do přerušovacího podsystému mikropočítače.
3. **Sériové linky**: Sériový přenos dat je v praxi stále více používán. Dovoluje efektivním způsobem přenášet data na relativně velké vzdálenosti při použití minimálního počtu

vodičů. Hlavní nevýhodou je však nižší přenosová rychlost, a to že se data musí kódovat a dekódovat.

- **USART (RS232)** +/-12V je transformována na TTL /RS422/RS485
  - **I2C** (Philips) komunikace mezi integrovanými obvody (přenos dat uvnitř elektronického zařízení)
  - **SPI**
4. **A/D a D/A převodníky** - Fyzikální veličiny, které vstupují do mikropočítače, jsou většinou reprezentovány **analogovou formou** (napětím, proudem, nebo odporem). Pro zpracování počítačem však potřebujeme informaci v digitální (číselné) formě. K tomuto účelu slouží analogově-číslíkové převodníky.
5. **Obvody reálného času** (RTC - Real Time Clock) - V mnoha aplikacích s použitím mikropočítačů je potřeba dodržovat přesnou časovou souvislost řízených událostí. Jde tedy o řízení v reálném čase. Ne vždy, ale taková posloupnost dostačuje a je nutno pro potřebu řízení udržovat skutečný čas, tedy hodiny, minuty, sekundy a případně i zlomky sekund. Pro tyto účely slouží obvody **RTC**. Při jejich použití je obvykle nutné vyřešit dva základní problémy:
- **záložní zdroj** - je třeba zajistit záložní zdroj pro udržení nepřetržité činnosti obvodu (může dojít k výpadku proudu a tak i k ztrátě skutečného času).
  - **čtení dat** - čas je hodnota neustále se měnící. Např. pokud zahájíme čtení hodnoty v čase 10:59:59, může se stát, že po přečtení prvních dvou hodnot, v našem případě hodin, se čas posune na 11:00:00 a čtení dalších hodnot bude neplatné (řešení technicky pomocnými registry v RTC obvodu, nebo vhodným programovým řešením).

## I<sub>2</sub>C

- Dvoudrátová, dvou vodičová sběrnice se sériovým přenosem.
- Obsahuje slave a master obvody.
- Lze propojit až 128 zařízení. (Master, slave)
- **Adresa zařízení:** skládá se ze 7 bitů (horní 4 určuje výrobce, dolní 3 jdou nastavit libovolně)
- **Signály** - SCL (synchronous clock), SDA (synchronous data)

### 3 Struktura OS a jeho návaznost na technické vybavení počítače.

Operační systém je zpravidla tvořen tzv. **jádrem** (kernel), **ovladači I/O zařízení** (driver), příkazovým procesorem (shell) a podpůrnými systémovými programy.

- **Jádro** – po **zavedení** do paměti **řídí** činnost počítače, poskytuje procesům služby a řeší správu prostředků a správu procesů.
- **Ovladač** – zvláštní (pod)program pro **ovládání konkrétního zařízení** standardním způsobem. Použití strategie s ovladači umožňuje snadnou konfigurovatelnost technického vybavení.
- **Příkazový procesor** – program, který **umožňuje** uživatelům **zadávat příkazy** ve speciálním, obvykle jednoduchém jazyce.
- **Podpůrné programy** – do této kategorie jsou mnohdy zahrnovány i překladače (jazyk C v OS UNIX) a sestavující programy. Stojí na stejném místě jako aplikační programy.

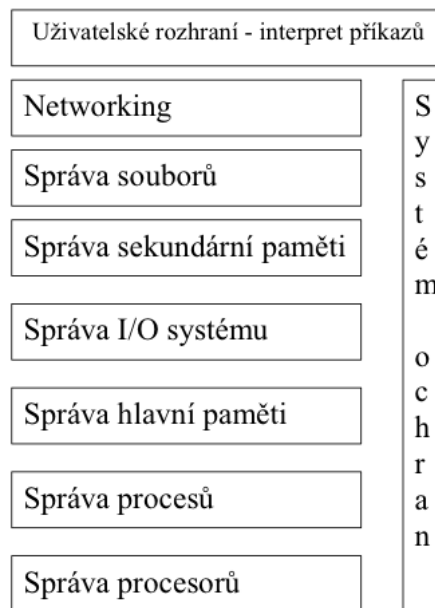
#### 3.1 Jádro

Jádro se zpravidla dělí na dvě podstatné části:

1. **Správa procesů** – řeší problematiku aktivování a deaktivování procesů podle jejich priority resp. požadavků na prostředky (prakticky není u jednoduchých OS)
2. **Správa prostředků** – zajišťuje činnost I/O zařízení, přiděluje paměť, případně procesory. Velmi důležitou částí správy prostředků je **správa souborů** – způsob ukládání souborů a přístupu k nim. Moderní OS zajišťují jednotný pohled na soubory a zařízení. Zařízení jsou považovány za soubory se speciálním jménem.

#### 3.2 Generické komponenty OS

- Správa procesorů
- Správa procesů (proces – činnost řízená programem)
- Správa vnitřní (hlavní) paměti
- Správa souborů
- Správa I/O systému
- Správa vnější (sekundární) paměti
- Networking, distribuované systémy
- Systém ochran
- Interpret příkazů



### 3.2.1 Správa procesorů/procesů

Správce procesoru má tyto funkce:

- sleduje prostředek (procesor a stav procesů),
- rozhoduje, komu bude dána možnost užít procesor,
- přiděluje procesoru prostředek, tj. procesor,
- požaduje vrácení prostředku (procesoru).

Pojem **proces** (task) je nějaká činnost řízená programem. Proces potřebuje pro svou realizaci jisté zdroje:

- dobu procesoru,
- paměť,
- I/O zařízení, atd.

OS je z hlediska **správy procesů** zodpovědný za:

- vytváření a rušení procesů,
- potlačení a obnovení procesů,
- poskytnutí mechanismů pro synchronizaci procesů a pro komunikaci mezi procesy.

OS je z hlediska **správy procesorů** zodpovědný za výběr procesu běžícího na volném procesoru.

### 3.2.2 Správa (hlavní, operační) paměti

Jedná se o úložiště připravených tj. rychle dostupných dat sdílených procesorem a I/O zařízeními. Hlavní paměť je pole samostatně adresovatelných slov nebo bytů, zpravidla energeticky závislá.

OS je z hlediska správy (**hlavní**) **paměti** odpovědný za:

- vedení přehledů kdo a kterou část paměti v daném okamžiku využívá,
- rozhodování, kterému procesu uspokojit jeho požadavek na prostor paměti po uvolnění,
- přidělování a uvolňování paměti dle potřeby,
- řízení virtuální paměti.

Z hlediska těchto zodpovědností správce operační paměti:

- udržuje přehled o přidělované a volné paměti,
- ve spolupráci se správou procesů rozhoduje o tom, kterému procesu, kolik, kde a kdy má přidělit operační paměť,
- provádí přidělení volné části paměti,
- určuje strategii odnímání dříve přidělené operační paměti procesům (opět po předchozí domluvě se správou procesů).

### 3.2.3 Správa I/O systému

Správce periferních (I/O systému) má tyto funkce:

- sleduje stav prostředků (periferních zařízení, jejich řídicích jednotek),
- rozhoduje o efektivním způsobu přidělování prostředku – periferního zařízení,
- přiřazuje prostředek (periferní zařízení) a zahajuje I/O operaci,
- požaduje navrácení prostředku.

Z hlediska funkce OS lze správce I/O systému chápat jako:

- úložiště vyrovnávacích pamětí,
- univerzální rozhraní ovladače I/O zařízení,
- ovladače jednotlivých hardwarových I/O zařízení.

Do správy I/O systému patří i **správa vnější (sekundární) paměti**. Počítačový systém musí poskytnout pro zálohování hlavní paměti sekundární paměť (HDD, SSD). OS je z hlediska správy vnější (sekundární) paměti odpovědný za:

- správu volné paměti,
- přidělování paměti,
- plánování činnosti disku.

### 3.2.4 Správa souborů

Správce souborů má tyto funkce:

- sleduje prostředek (soubor), jeho umístění, užití, stav atd.,
- rozhoduje, komu budou prostředky **přiděleny**, realizuje požadavky na **ochranu informací uložených** v souborech a realizuje operace přístupu k souborům,
- **přiděluje** prostředek, tj. otevírá soubor,
- **uvolňuje** prostředek, tj. uzavírá soubor.

Pod pojmem soubor chápeme jak programy, tak data. OS je z hlediska správy souborů odpovědný za:

- vytváření a rušení souborů,
- vytváření a rušení adresářů (katalogů, složek),
- podporu primitivních **operací** pro manipulaci se soubory a adresáři,
- archivování souborů na energeticky nezávislé média.

### 3.2.5 Networking, distribuované systémy

Distribuovaný systém je **kolekce procesorů**, které **nesdílejí** ani **fyzickou paměť** ani **hodiny**, synchronizující činnost procesoru. Každý procesor má svoji **lokální paměť** a **lokální hodiny**. Procesory distribuovaného systému jsou **propojeny** komunikační sítí. Komunikace jsou řízeny protokoly. Distribuovaný systém uživateli zprostředkovává přístup k různým zdrojům systému.

### 3.2.6 Systém ochran

Jsou to mechanismy pro **řízení přístupu** k **systémovým** a **uživatelským zdrojům**. Systém ochran musí:

- **rozlišovat** mezi **autorizovaným** a **neautorizovaným** použitím,
- specifikovat problém vnucovaného řízení,
- poskytnout prostředky pro své prosazení.

### 3.2.7 Uživatelské rozhraní - interpret příkazů

Interpret příkazů je program, umožňující vykonávat příkazy pro:

- správu a vytváření procesů – služby OS poskytované interpretem příkazů slouží k **provedení programu**, tj. k schopnosti OS zavést program do hlavní paměti a spustit jeho běh,
- ovládání I/O zařízení – uživatelský program **nesmí** provádět I/O operace **přímo**, OS musí poskytovat prostředky k provádění I/O operací,

- správu sekundární paměti – manipulace se systémem souborů, schopnost číst, zapisovat, vytvářet a rušit soubory,
- správu hlavní paměti,
- zpřístupňování souborů,
- ochranu – tj. **detekci chyb v procesoru a paměti**, I/O zařízeních a v programech uživatelů pro zajištění správnosti výpočtu,
- práci v síti – **výměna informací** mezi procesy realizována buďto v rámci jednoho počítače nebo mezi různými počítači pomocí sítě, tj. implementace sdílenou paměti nebo předávání zpráv.

Uživatelská rozhraní jsou realizovaná **znakově** (někdy označované řádkově) nebo **graficky**. Znakově orientovaným interpretům zadáváme příkazy pomocí klíčových slov, graficky orientovaným pomocí poklepání myši nebo dotykem na dotykové obrazovce na ikonu, pomocí dialogů apod.

### 3.2.8 Vnitřní služby operačního systému

Vnitřní služby OS **nejsou** určeny k tomu, aby **pomáhaly uživateli**, v první řadě slouží pro **zabezpečení efektivního provozu systému**, tj. slouží pro:

- Přidělování prostředků (zdrojů) mezi více souběžně operujících uživatelů nebo úloh.
- Účtování a udržování přehledu o tom, kolik jakých zdrojů systému který uživatel používá. Cílem je účtování za služby a sběr statistik pro plánování.
- Ochranu tj. péči o to, aby veškerý přístup k systémovým zdrojům byl pod kontrolou.

Vnitřní služby OS jsou obecně **realizovány souborem systémových programů** vytvářejících určité systémové struktury tzv. virtuální stroje. Typickými službami jsou programy pro:

- práci se soubory, editaci souborů, katalogizaci souborů, modifikaci souborů,
- získávání, definování a údržbu systémových informací,
- podporu jazykových prostředí,
- zavádění a provádění programů,
- komunikace a řízení aplikačních programů.

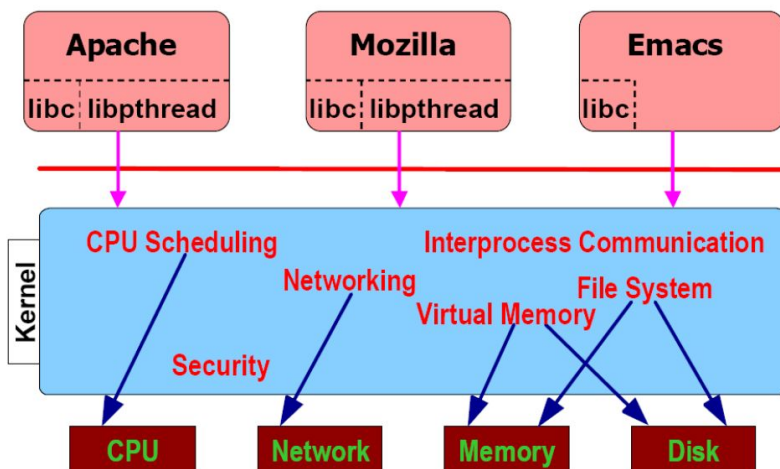
## 3.3 Struktura OS podle jádra

- Monolitická jádra (monolithic kernel)
- Otevřené systémy (Open systems)
- Microkernel



### 3.3.1 Monolitický OS

OS je na jedno místě (v obrázku pod „červenou čarou“). Aplikace používá dobře definované rozhraní systémového volání pro komunikaci s jádrem.



Příklady OS – Unix, Windows NT / XP, Linux, BSD.

Monolitické jádro je specifické pro komerční systémy.

#### Výhody

- + dobrý výkon
- + dobře pochopená koncepce
- + jednoduché pro vývojáře jádra
- + vysoká úroveň ochrany mezi aplikacemi

#### Nevýhody

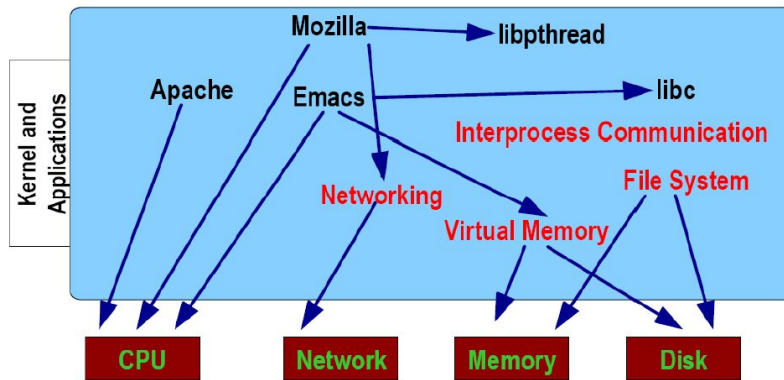
- žádná ochrana mezi komponentami jádra
- není jednoduše a bezpečně rozšiřitelné jádro
- celková struktura je ve výsledku komplikovaná, neexistující hranice mezi moduly jádra

### 3.3.2 Otevřené Systémy

Aplikace, knihovny i jádro jsou všechny v jednom adresovaném prostoru.

Příklady OS – MS-DOS, Mac OS 9 a starší, Windows ME, 98, 95, 3.1, Palm OS.

Tato koncepce bývala velmi běžná.



### Výhody

- + velmi dobrý výkon
- + velmi dobře rozšiřitelné
- + výhodné pro jednouživatelské OS

### Nevýhody

- žádná ochrana mezi jádrem a aplikacemi
- nepřiliš stabilní
- skládání rozšíření může vést k nepředvídatelnému chování

### 3.3.3 Microkernel OS

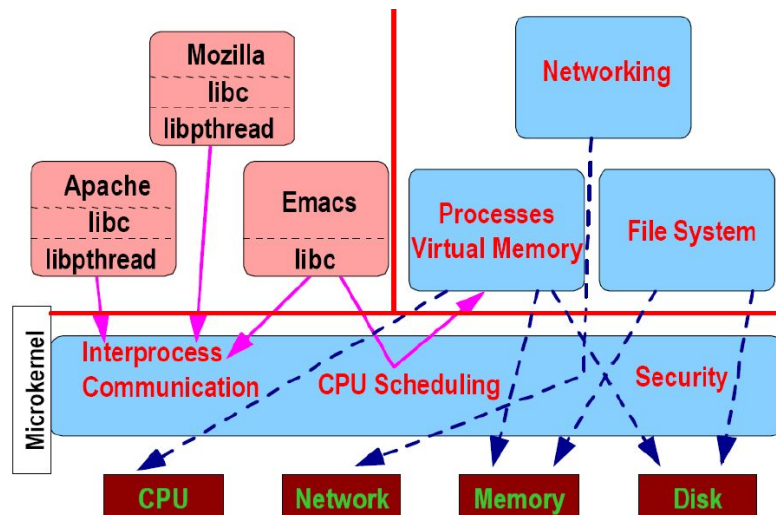
Filosofie návrhu – chráněné jádro obsahuje pouze minimální (malou, čistou, logickou) množinu abstrakcí:

- procesy a vlákna
- virtuální paměť
- komunikace mezi procesy

Všechno ostatní jsou server-procesy běžící na uživatelské úrovni.

Příklady OS – Marc, Chorus, QNX, GNU Hurd

Zkušenosti s tímto návrhem jsou smíšené.



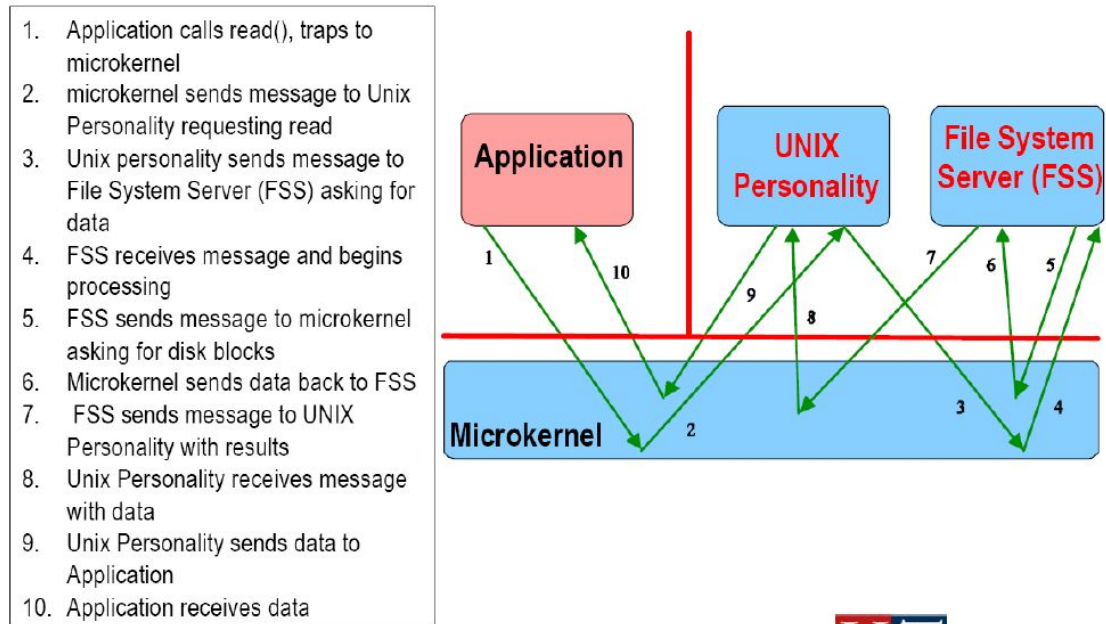
## Výhody

- + přidáním server-procesu se rozšíří funkcionality OS
- + jádro nespécifikuje prostředí OS
- + servery na uživatelské úrovni se nemusí zabývat hardwarem
- + silná ochrana OS i sám proti sobě (části OS jsou oddělené servery)
- + jednoduché rozšíření na distribuovaný nebo multiprocesorový systém

## Nevýhody

- výkon (systémová volání, viz příklad pod touto kapitolou)
- špatná minulost

### 3.3.4 Příklad systémového volání



## 4 Protokolová rodina TCP/IP.

## 5 Metody sdíleného přístupu ke společnému kanálu.

## 6 Problémy směrování v počítačových sítích. Adresování v IP, překlad adres (NAT).

- 7 Bezpečnost počítačových sítí s TCP/IP: útoky, paketové filtry, stavový firewall. Šifrování a autentizace, virtuální privátní síť.