

# Počítačové systémy

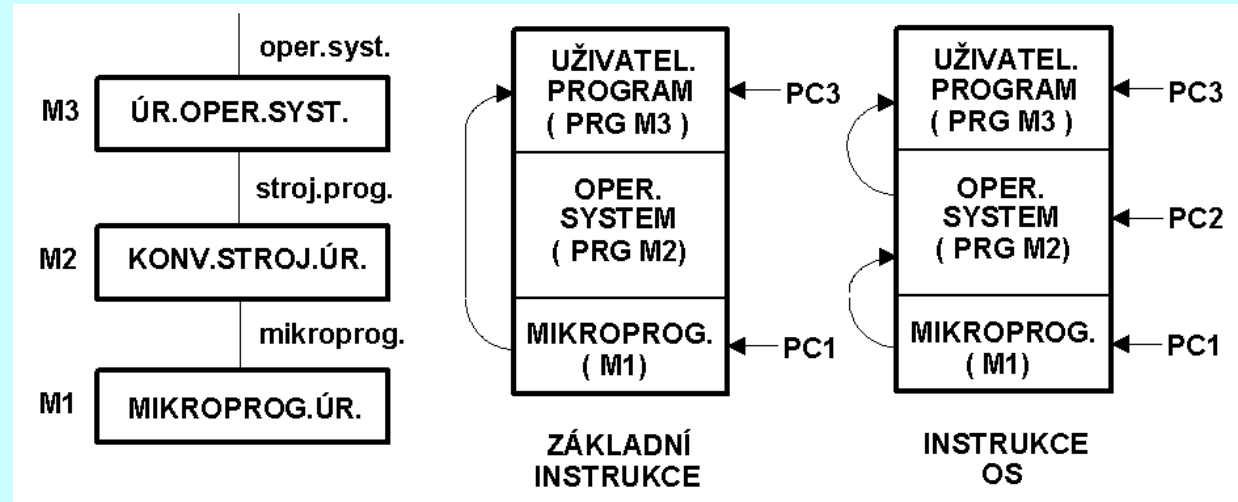
## 8 Mnohaúrovňová organizace počítače 2

# Úroveň operačního systému

## M3 Úroveň operačního systému

Kromě základních instrukcí úrovně M2 obsahuje spec.instrukce OS

### Interpretace instrukcí OS



- OS běží na pozadí, hlídá INT, privileg. instr., ochr. pam., perif., sdílení prg.
- pod OS většinou multiprogramové prostředí (sdílení, batch)
- OS má zajistit paralelní funkci několika virt. strojů úrovně 3.
- základní otázkou je řízení (managing) funkce virt.strojů, uživ. komfort

**Správci OS** (jádro OS): úloh, přerušení, paměti, komunik. a synchr. úloh

**Programové zabezpečení:**

1. jazyk řídicích příkazů OS
2. virtuální instrukce OS

# Úroveň operačního systému

## Jazyk řídicích příkazů OS (JOB CONTROL LANGUAGE)

Jednoduchý jazyk pro řízení uživatelských služeb OS. (**Instr + param.**)

- Zajišť. služby:**
1. překlad programu speciálním překladačem
  2. linkování procedur (knihovna) a progr.modulů
  3. exekuce sestaveného programu
  4. definice souboru
  5. zapsání souboru do adresáře
  6. vyjmutí souboru z adresáře
  7. požadavek připojení periferie (MG páska, disk)
  8. výpis obsahu paměti (oktal, hexa, ASCII)
  9. vytvoření kopie souboru
  10. uchování stavu procesu pro jeho pokračování

## Virtuální instrukce OS

Makroinstrukce OS interpretované úrovní M2. Zajišťují uživat. komfort OS a některé komplexní funkce virtuálního stroje.

# Úroveň operačního systému

## Virtuální I/O instrukce

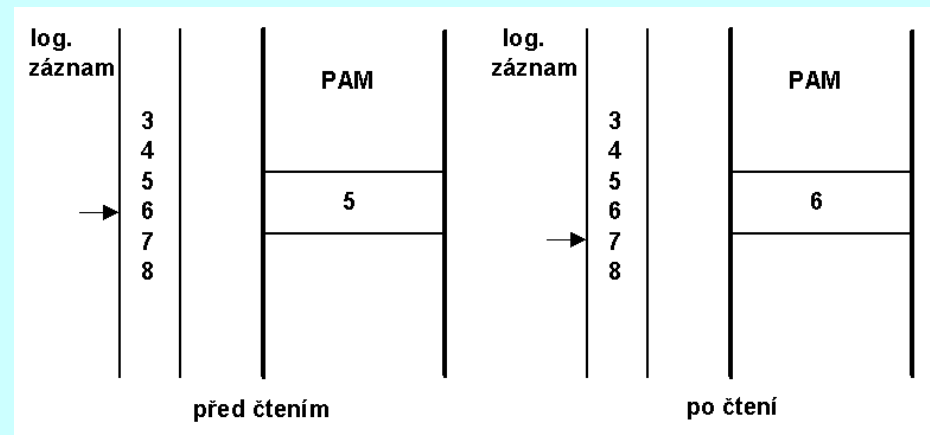
Komplexní instrukce s vazbou na perif. zařízení (disk, páska, aj.)

- R/W, test chyb, rozbor parametrů aj. Instrukce se obrací k **souboru** (file).

### 1. Sekvenční soubor

sekvence log. zápisů - info. def. programátor (znak, číslo, pole, řetěz....)

- délka soub.: **pev/prom**
- neexistuje adr. dat
- instrukce specifikuje:
  1. jméno souboru
  2. adr. OP pro R/W
- v prgm se uchovává **ukazatel sekv. zápisu** (následující zápis)



- Virt. instrukce:**
- |             |   |
|-------------|---|
| OPEN n      | - otevření souboru n pro čtení nebo zápis |
| CLOSE n     | - zavření souboru n                       |
| READ n,adr  | - čtení z n násl. logického zápisu do adr |
| WRITE n,adr | - zápis do n násl. " " z adr              |
| REWIND n    | - přesun ukazatele souboru n na začátek   |

# Úroveň operačního systému

## 2. Soubor s libovolným přístupem (RANDOM)

Soubory mají adresaci jednotlivých zápisů (i asociativní)

Instrukce musí specifikovat:

1. jméno souboru
2. adresu OP pro R/W
3. adresu zápisu uvnitř souboru

**Virt. instrukce:**

OPEN n	- otevření souboru n pro čtení nebo zápis
CLOSE n	- zavření souboru n
READ n,a1,a2	- čtení záznamu soub. n z a1 do BUFF a2
WRITE n,a1,a2	- zápis záznamu do soub. n a1 z a2 BUFF
GET n,adr	- čtení souboru n do fronty z adr
PUT n,adr	- zápis z fronty do souboru n na adr.

## Použití virt. I/O instrukcí

**Nosičem info:** - děrná páska, mg. páska, disk. v poč.vyr. pam. (BUFFER)

**Param. I/O instr.:**

- jméno souboru, jednotka, pevná/prom. délka, typ záznamu, počet záznamů, délka souboru, alokace aj.
- adr. vyr. paměti, její délka, počítadlo čtených zápisů, ukazatel kruh. paměti, adresy procedur EOF, ERR aj.

# Úroveň operačního systému

## DISK

- **adr. záznamu** na disku def.: číslo disku, povrch, stopa, sektor
- informace **v sektoru** většinou sekvenčně
- umístění logických záznamů na disku: - **sekvenčně/random**

**Přístup k zápisům:** z M3 - **log.adr.** - název + číslo záz., (alokace transp.)  
z M2 - **fyz.adr.** - přesná identifikace místa záznamu

## Převod log. adresy (M3) na fyz. adresu (pro M2)

### 1. Tabulka záznamů souboru

- každý soubor má tab. fyz. adres pro každý záznam - **file index**

soubor	záznam	povrch	stopa	sektor
AAAA	0	0	3	12
	1	1	35	3
	...	...	...	...

### 2. Tabulka souborů

- každý soubor má v tab. poč. adr., každý záznam má na konci adr. pokrač. **linked list**

soubor	povrch	stopa	sektor
AAAA	0	3	12
BBBB	1	45	3
...	...	...	...

# Úroveň operačního systému

Pro **rozšiřování souborů** je třeba znát **volná místa na disku**

## 1. Tabulka volných sektorů

- udává počáteční volný a počet volných

povrch	stopa	sektor	počet vol.
0	0	5	4
0	0	10	2
...	...	...	...

## 2. Bitová mapa

- udává přímou obsazenost sektorů (0-volné, 1-obsaz.)

povrch	stopa	sektor										
		0	1	2	3	4	5	6	7	8	9	.....
0	0	0	0	1	1	1	0	0	1	1	0	.....
0	1	1	1	1	1	1	0	0	0	0	.....	
...	...	.....										

## **BUFFER** - vyr.paměť

- spoluprac. místo v počítači.
- lineární nebo kruhový.
- pro instr. slouží ukazatelé

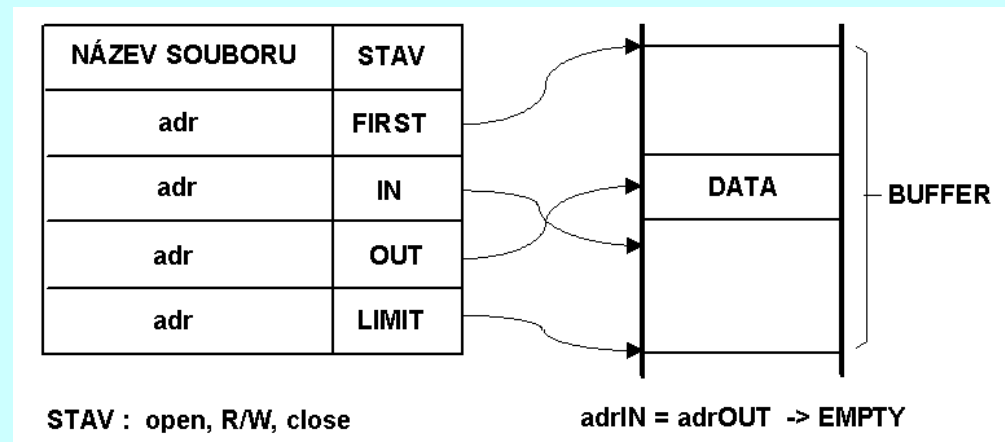
## Virtuální instrukce

**FIRST** - první adr. BUF

**IN** - R soub. do BUF + ukaz.

**OUT** - W do soub. z BUF + ukaz.

**LIMIT** - koncová adr. BUF



# Úroveň operačního systému

## Instrukce práce se soubory

Instrukce pracují s celými soubory na rozdíl od práce se záznamy

Soubory: - **dočasné** (mažou se), **stálé** (pamatují se do instr. práce se souborem)

Uživatel má:

- **adresář souborů**  
(**directory**)

OS má:

- **správce souborů**  
(**Directory manage.**)

- **strom adresářů**  
zákl. výběr podle  
uživatele (větev)

jméno uživatele číselný kód umístění paměti
další informace (blokování, kódování a j.)
SOUBOR 1 TAB SOUBOR 2 TAB SOUBOR 3 TAB SOUBOR 4 TAB ... ... ...

jméno souboru	(název)
umístění na disku	(povrch, stopa, sektor)
délka	(byte)
typ	(zdroj, SYS, HEX,...)
číslo	(kód. označení souboru)
povolení přístupu	(jméno, R, R/W)
datum vytvoření	(datum)
poslední změna	(datum)
počet přístupů	(suma)
kopie	(počet)

### Struktura adresáře:

přístup vyžaduje kód uživ., porovnává se s podmínkami v DIR.

### Virt.instrukce práce s adresářem:

DIR - výpis adresáře

DELETE n - výmaz soub. z adr.

RENAME n1,n2 - změna jména soub. z n1 na n2

CHANGE STATUS - změna stavu povolení přístupu (LOCK, RO, R/W, USR aj.)

### Virt.instrukce práce se soubory:

LOAD n - čtení souboru

SAVE n - zápis souboru (i do adr)



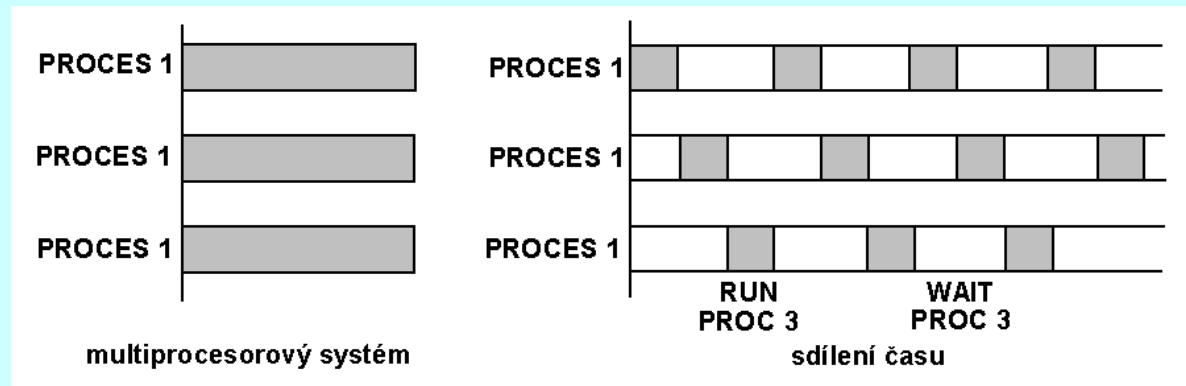
# Úroveň operačního systému

## Instrukce paralelního běhu programů

Simultánní běh programů v multiprocessorovém syst. nebo sdílením času.

OS:

- multitask
- multiuser



## Virt.instrukce:

- |         |                                   |
|---------|-----------------------------------|
| CREATE  | - vytvoření (konfigurace) procesu |
| DESTROY | - zrušení procesu                 |
| EXAMINE | - spuštění procesu                |
| STOP    | - zastavení procesu               |
| RESTART | - znovuspuštění procesu           |

Proces je určen stavem a adr. prostorem pro prgm a data – **virt.stroj 3.úr.**

Paralelně spouštěné procesy - **TASK** - úloha

- Typy OS:**
- **pevný poč. úloh** - spouštějí/zastavují se všechny najednou
  - **prom. poč. úloh** - jedn. procesy se dají spouštět a zastavovat

# Úroveň operačního systému

## Synchronizace paral. běžících asynchronních procesů

Speciálně se jedná o sekvenční činnost, spolupráci procesů.

### 1. Synchronizace podmínkami vybuzení (RACE CONDITION)

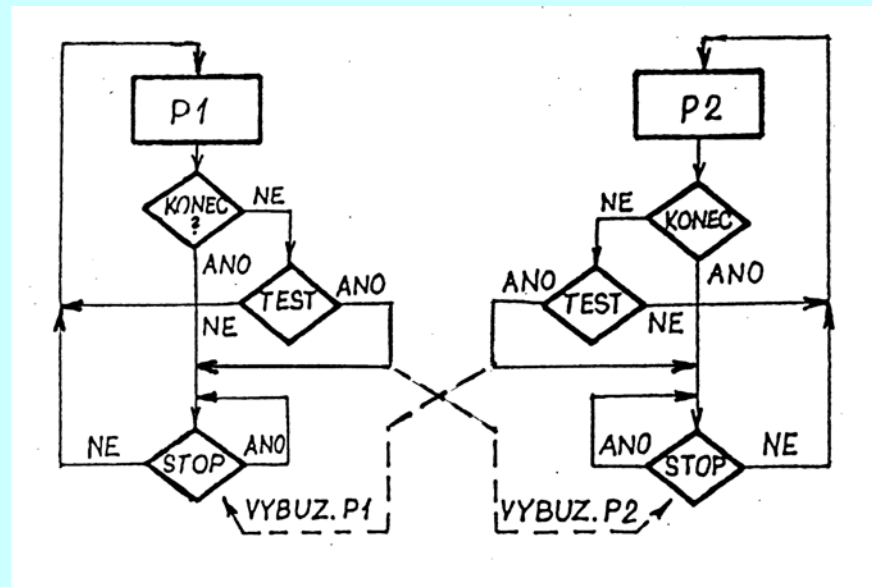
V programech test ukazatelů  
- podle výsledku generace  
vybuzení druhého procesu  
(realiz. stav. bitem vybuzení)

#### TEST stav. bitu procesu

spouští proces a současně  
stav. bit nuluje.

#### Kolize:

test nesprávných obsahů  
ukazatelů (během přepisu)



Stavový bit procesu - 0,1. (**WAKE UP/WAITING bit**) - určuje násl. činnost

proces běží + vybuzení -> st. bit ->1    proces běží    -> st. bit =1  
proces stojí + vybuzení -> st. bit ->1    proces startuje, -> st. bit ->0

Nevýhodné pro komunikaci více programů - **pro N procesů N-1 stav. bitů**

# Úroveň operačního systému

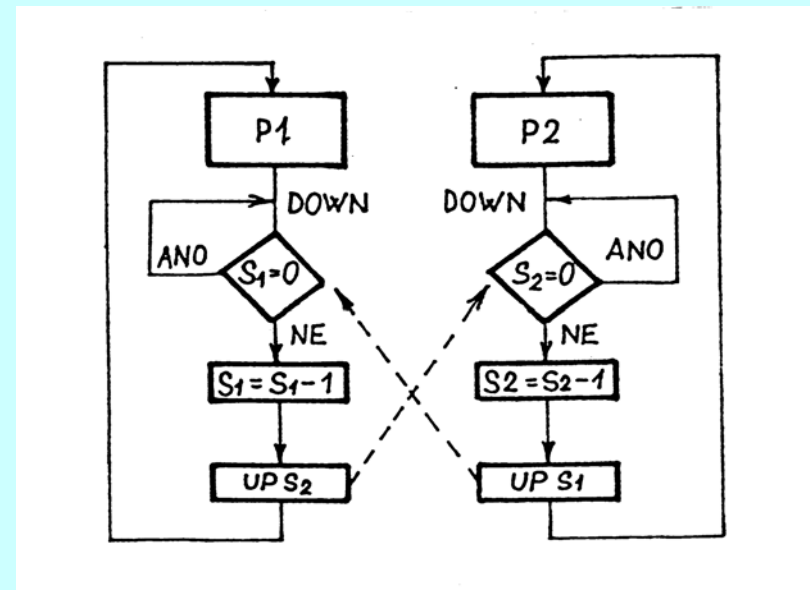
## 2. Synchronizace pomocí semaforů

### Semafor

proměnná (integer) příslušná k danému procesu, sumarizuje jednotlivá vybudení procesu, určuje chování v násl. čase.

### Instrukce práce se semaforem:

- UP - inkrement semaforu spoluprac. procesu
- DOWN - dekrement vlastního semaforu



- semaforey se **inicializují** počtem průchodů bez UP druhého procesu (naplnění BUFF).
- instrukce se semaforem musí být **nedělitelné (LOCK sběrnice)**.
- pokud na semafor **čeká více procesů**, pak po UP použije semafor pouze jeden z nich.

INSTR	S=0	S>0
UP	S=S+1 proces dokončuje DOWN a pokračuje v programu	S=S+1
DOWN	proces se zastavuje, dokud jiné procesy nenastaví semafor instrukcí UP	S=S-1

# Úroveň operačního systému

## Instrukce pro komunikaci mezi procesy

Specielně se jedná o předávání dat mezi spolupracujícími procesy

Používá se vyr. paměť (**BUFFER a semafor**) - nazývá se **SCHRÁNKA**

- schránce je přiřazen semafor
- hodnota semaforu odpovídá počtu zpráv ve schránce.

### Virt. instrukce:

**SEND** - zápis zprávy do schránky + inkr. semaforu  
**GET** - výběr zprávy ze schránky + dekr. semaforu

Pokud ve schránce není zpráva, proces s instr. **GET** čeká na zaslání zprávy

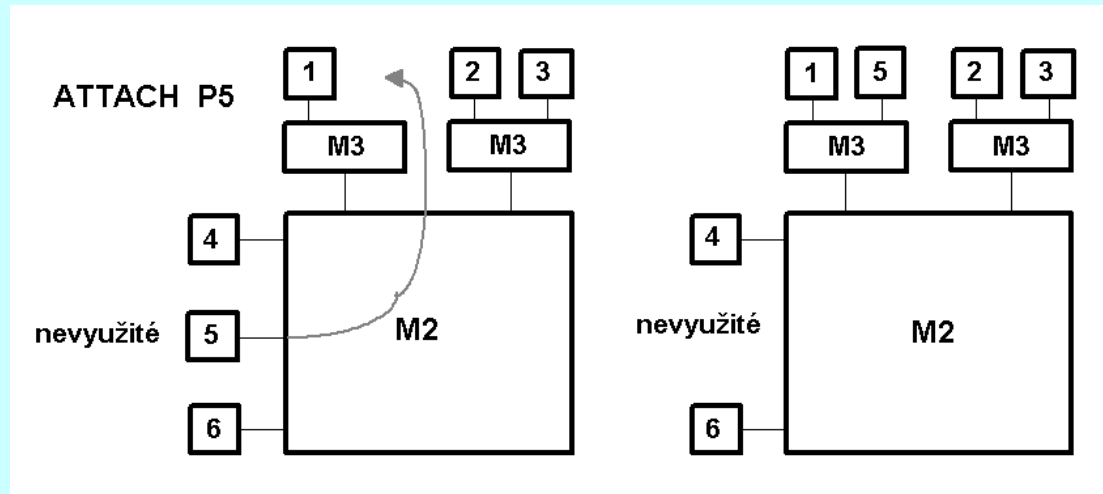
**SEND** - odpovídá UP  
**GET** - odpovídá DOWN

Instrukce **GET** a **SEND** - souč. synchronizace procesů i předávání zpráv

# Úroveň operačního systému

## Instrukce rekonfigurace stroje úrovně M3

umožňují připojit nevyužité periferie úr. M2 k některému virt. stroji úr. M3.



### Virt. instrukce:

**MOUNT per** - připojení per. úr. M2 k virt. stroji úrovně M3 (ATTACH per)

**DISMOUNT per** - odpojení per. virt. stroje úr. M3 a její předání úr. M2

Pokud **úr.3 žádá** o připojení periferie a požadovaná **v úr.2 neexistuje**

- proces čeká do té doby, dokud jiný proces periferii nevrátí
- možnost **deadlock** – všichni čekají, nikdo nevrací

# Úroveň operačního systému

## Virtuální paměť

Požadavek velké paměti pro rozsáhlé programy. Program se dělí na části (overlay), které se vykonávají postupně. Nepoužívané části na sek. pam.

**Realizace:**

- **programem** - realizuje programátor (instr. CHAIN, SWAP aj.)
- **OS** - pro programátora transparentní (**správce paměti**)

**Virtuální paměť** - tvoří OP a všechny sek. paměti počítače.

- adresování **virt. adresou** (velký adr. prostor, **symb. jména**)

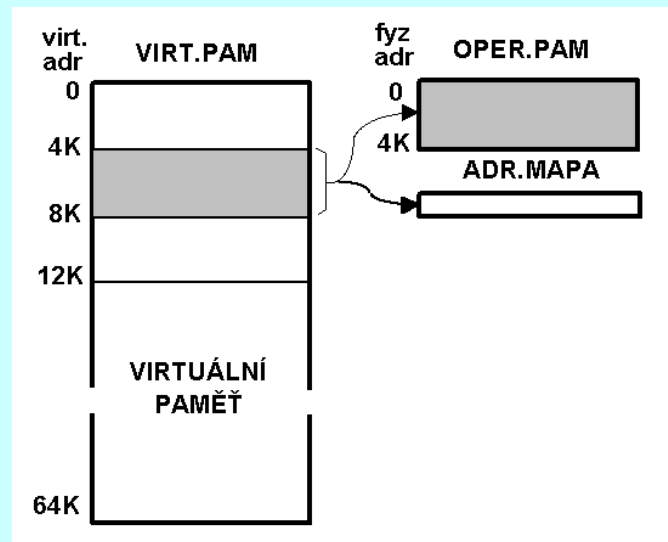
**Adr. prostor poč.** - max. rozsah adres programů počítače

**Fyzický** - reálný adr. prostor OP

**Virtuální**

- úplný rozsah pam. poč. včetně sek.pam. (HDD, FDD, mag. páska, .. a j.).
- adr. prostor virt. stroje

Př.: 64KB program, 4KB OP -> **výměna**



OP -> sek.pam., pož.blok -> OP, změna adr.mapy, spolupr. s pož.adr. 14

# Úroveň operačního systému

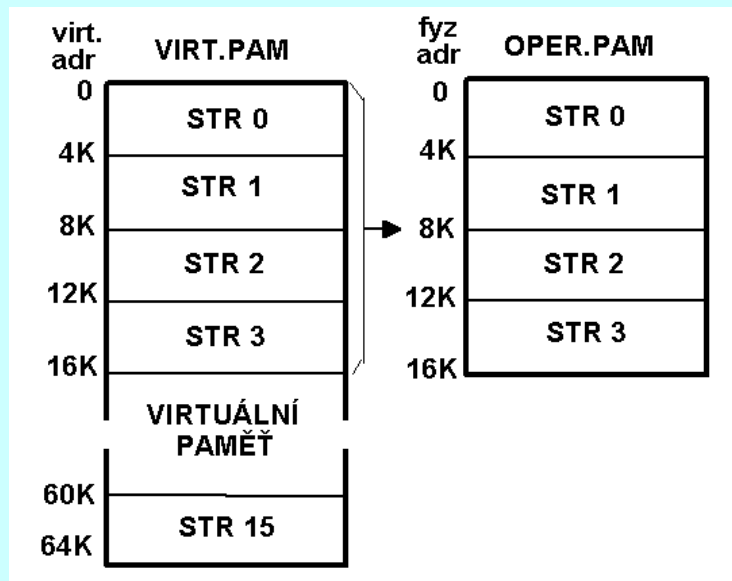
## Mapování virtuální paměti

**Adres. mapa** - doplňuje adr. OP na úplnou **virt. adr.**, pro poč. (horní bity).

**Mapování** - přiřazení virt. a reál. adr - realizace pomocí **stránek** nebo **segmentů**.

### 1. Stránková verze virtuální paměti

Virt. paměť je rozdělena na **stránky** (adr. rozsah je < rozsah OP poč.).



### Virtuální adresa:



### Stránka:

- 256 až 4096 adres
- na požádání se stránky **vyměňují**
- mechanismus je **transparentní** (OS)

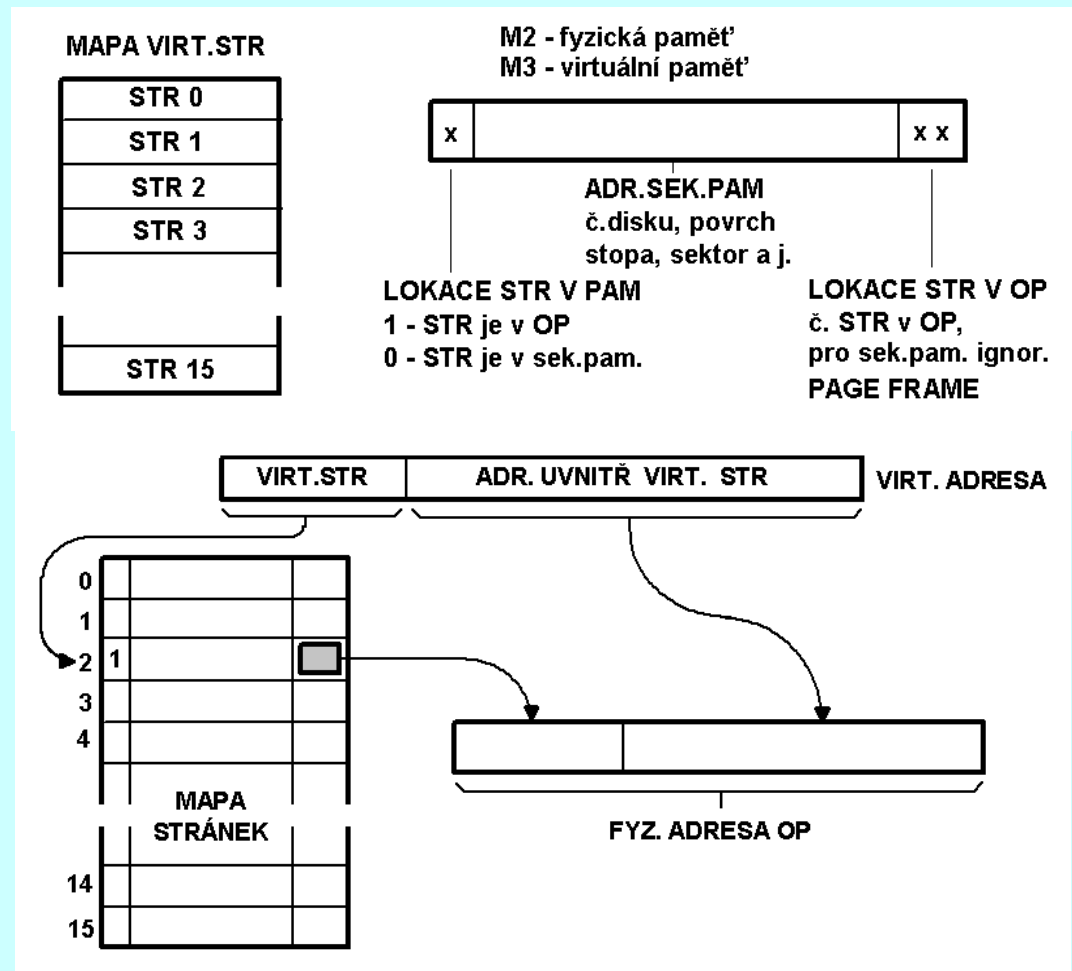
### Výměna:

str. OP -> sek.pam., pož.str. -> OP, změna adr.mapy, spolupr. s pož.adr.

# Úroveň operačního systému

## Mapa stránek (page table)

- **přiřazuje** virtuál.str. reálný adr. prostor (v OP i v sek. paměti)
  - provádí vzájemnou **transformaci** virt. a reál. (fyz.) adr. pam.
  - lokace stránek v OP není lineární
  - po výměně stránek se mapa str. **aktualizuje**
- Def. **fyzické adresy OP** z virt. adresy





# Úroveň operačního systému

**Stránkování** - požadavek výměny stránek

když exist. adr. reference na str., která není v OP - **chyba stránky** (page fault)

**OS musí:**

1. do OP načíst požadovanou stránku
2. aktualizovat mapu stránek
3. zopakovat instrukci, která způsobila chybu stránky

**Výměna stránek :**

- **následné stránkování** (**demand paging**) - výměna až po požadavku (chyba str) **krize** při výměně celého obsahu
- **pracovní soubor** (**working set**) - množina nejvíce používaných str. (nahrává se najednou při restartu úlohy)

**Algoritmy výměny stránek** (při plné OP)

- **LRU** (**Least Recently Used**) - v poslední době má str. **nejméně přístupů** (čítač)
- **FIFO** - str., která je v OP **nejdéle** (každá virt. str. má čítač času)

**Modifikační bit** (**Dirty bit**) str. - indikuje změnu str. v OP (pokud ne - nepřepisuje se)

**Transformace** virt.adr. -> fyz. adr. - mapa str. jako spec. registry + HW.

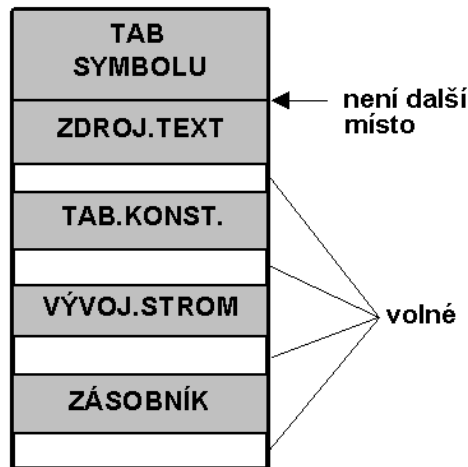
- mapa str jako spec. registry +  $\mu$ program 17

# Úroveň operačního systému

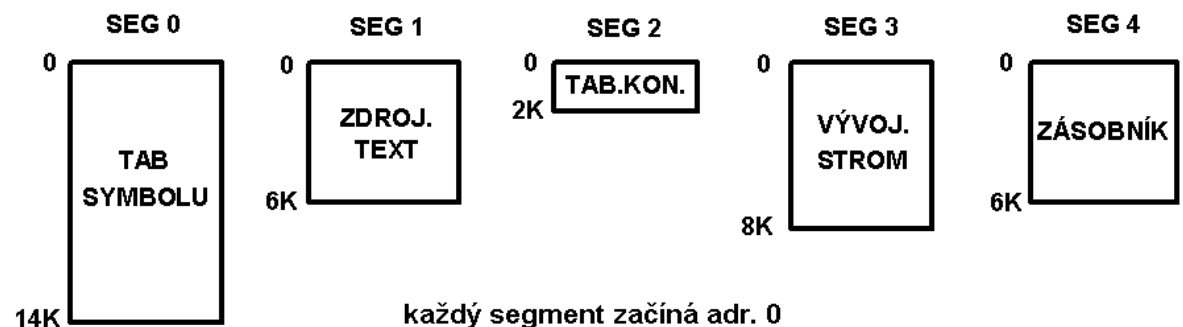
## 2. Segmentová verze virtuální paměti

Požadavek **několika oddělených virt. prostorů** (překladače, datové strukt., syst/uživat. program vybavení), které na sobě nejsou závislé.

překlad ve stránkové verzi



překlad v segmentové verzi



**Segment** - samostatný virt. adr. prostor určený pro spec.data procesu (prgm, data, zásobník aj.). Začíná **adr. 0**, velikost se může průběžně **měnit**

Virt. paměť **dvoudimenzionální** - adr. z **č.segmentu** a **adr. uvnitř segmentu**.

**Výhody:** - na sobě **nezávislé** paměťové prostory

- snadná **změna délky** segmentu a jeho umístění ve virt. paměti
- po proběhnutí procesu možnost **slinkování paměti** bez mezer
- při sdílení paměti správce **dynamicky přiděluje** procesům segment
- jednoduchá **ochrana** spec. segmentů (syst/uživ -> R/RW)

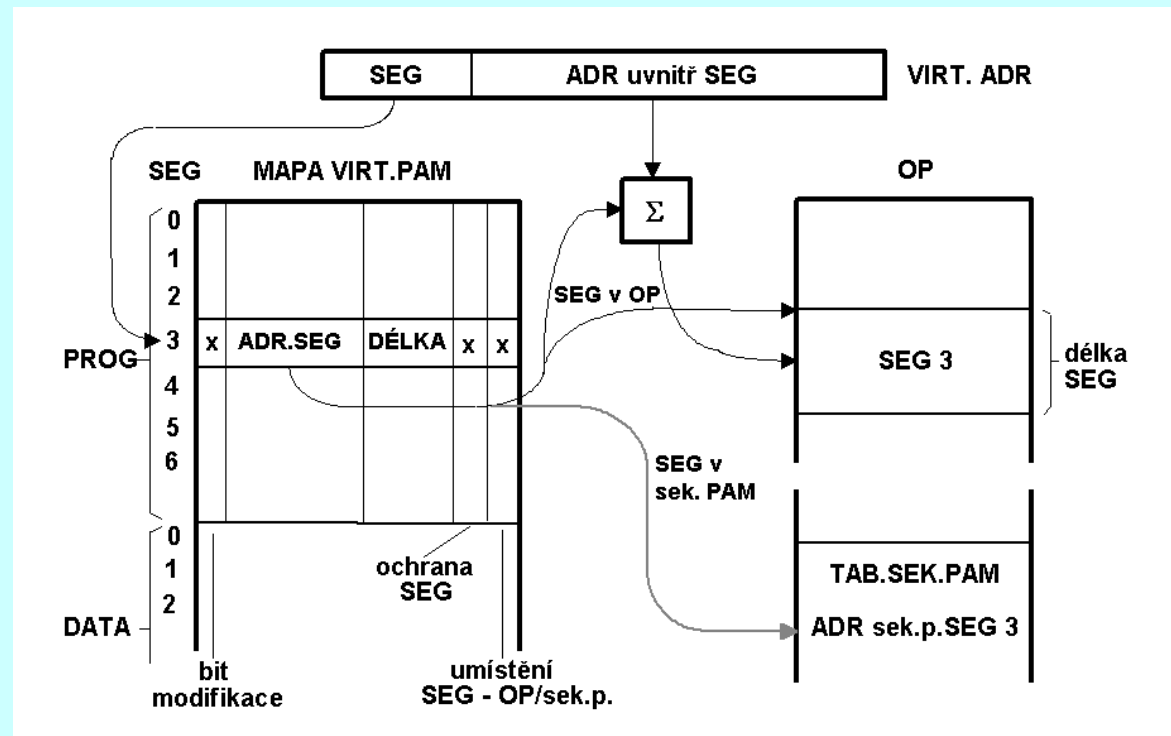
# Úroveň operačního systému

## Mapování paměti

Mapa segmentů se rozšiřuje o **další bity**

Většinou samostatný seg. pro **prgm / data, syst. / uživat. prostor**

Při **výměně segm.** OS zajišťuje opt. využití paměti - **linkuje moduly.**



**Výměna segmentů** závisí na výskytu chyb:

- chyba segmentu
- chyba délky
- chyba ochrany
- seg. není v OP
- virt. adr. je větší než max. adr. seg.
- nedovolený přístup

**Algoritmy výměny:** LRU, FIFO

# Úroveň operačního systému

## 3. Kombinace segmentové a stránkové verze

Využívá výhod segm. a str. verze - velká pružnost práce s adr. prostorem

**Výměna** při chybě:

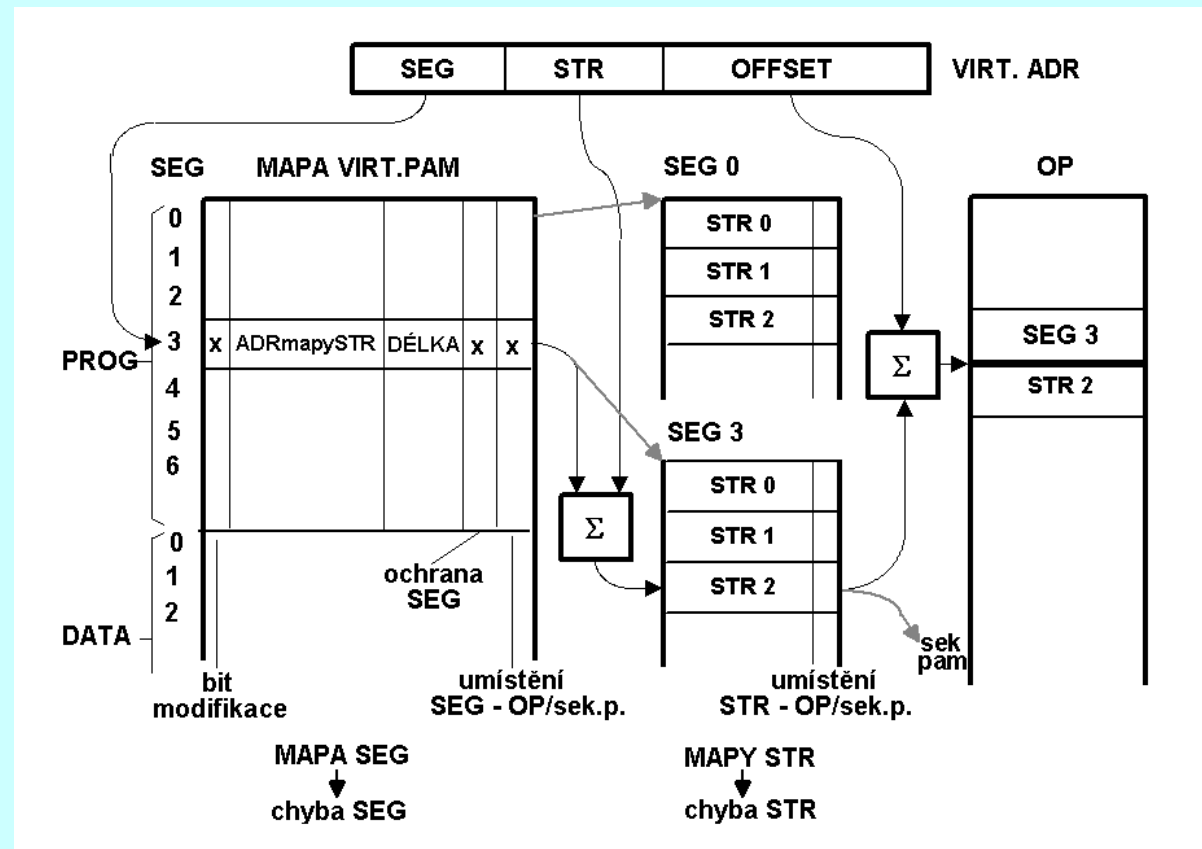
- segmentu
- délky segmentu
- ochrany
- stránky

**Algoritmy** výměny:

- LRU
- FIFO

**Segm.virt. paměť  
a soubory I/O**

každému souboru  
svůj segment ->  
přístup k záznamům  
soub. jako k virt.pam.



**Dynamické linkování** - lze přistupovat k souborům symbolickým jménem.

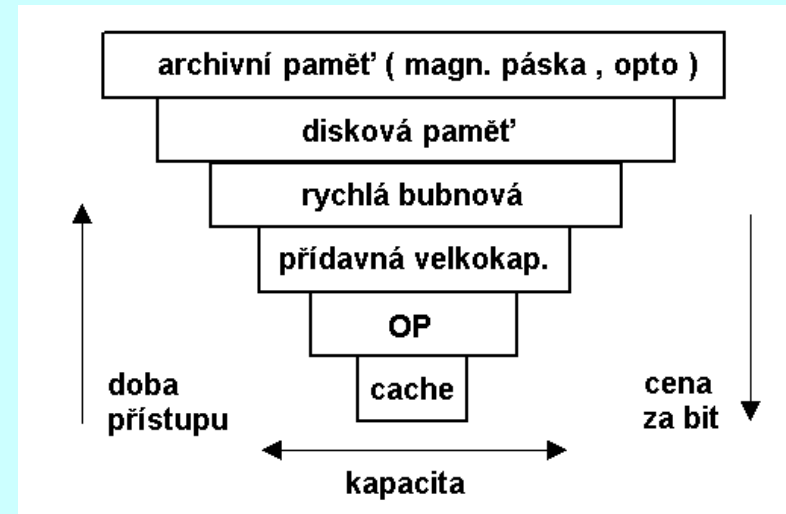
# Úroveň operačního systému

## Hierarchická organizace paměti

**Rozsah doby přístupu** -  $10^9$  až  $10^{10}$   
(10ns - cache, 50s - mag.pásku)

**Rozsah kapacity** -  $10^8$  až  $10^9$   
( $10^3$  - cache,  $10^{11}$  - mag. pásce nebo  
v optoelektrických pamětech)

S rychlostí paměti roste **cena/bit**



**Cache** - spolupracuje s CPU. Program v OP (stránky) je dělen do **bloků**, nejvíce aktivovaný blok je **přepisován do cache**.

Pož. přepisu je **chyba bloku**, doba přepisu -  $\mu$ s -  $\mu$ programem.

Dvojnásobná úroveň **transformace virtuální adresy**:

virt. adr. sek. paměti -> virt. adresa OP

virt. adr. OP -> reál. adresa cache

povídá OS

provádí  $\mu$ program

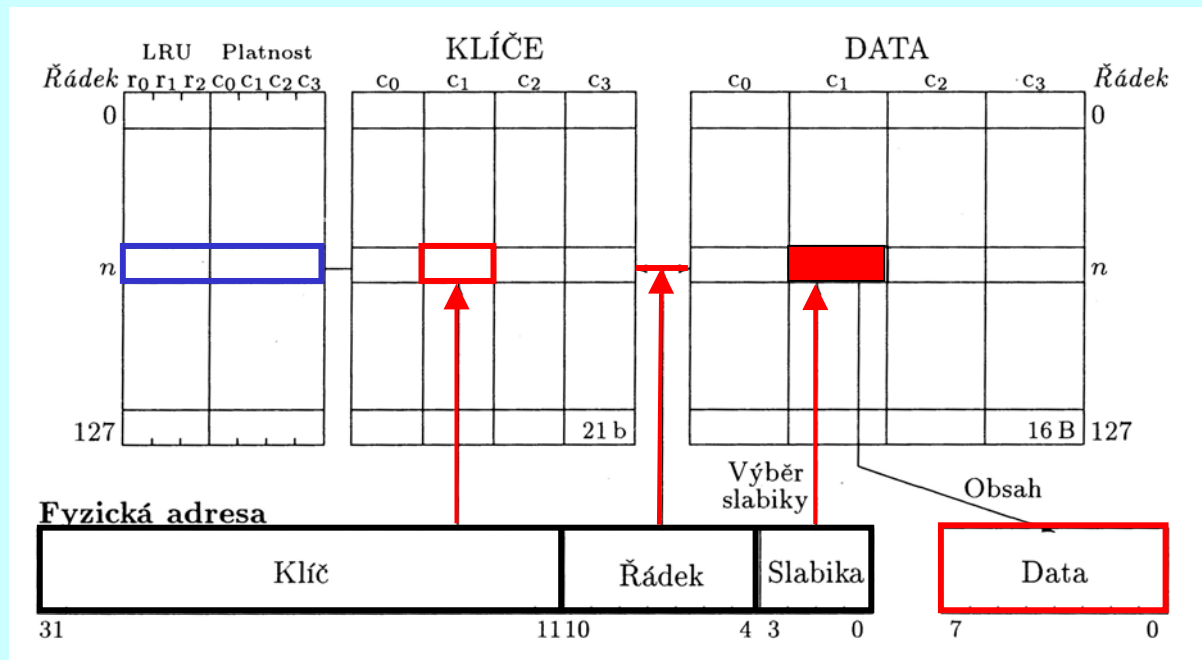
Transformace jsou **transparentní**

# Úroveň operačního systému

## Hierarchická organizace paměti

### Cache 486

- struktura asociativního přístupu
- virt. adr. OP -> reál. adresa cache
- výměna obsahu algoritmem LRU



# Vyšší úrovně počítače

Programy vyšších úrovní poč. mohou být **překládány** nebo **interpretovány** úrovní nižší (záleží na úrovni cílového jazyka).

<b>Překlad</b>	- převedení prgm psaného v jednom jazyce do jiného jaz.
<b>Překladač</b>	- program pro provedení překladu
<b>Zdrojový jazyk</b>	- v jazyce je psán program
<b>Cílový jazyk</b>	- jazyk, do kterého se překládá

Překlad vytváří t.zv. **OBJEKT PROGRAM** (modul) v cílové úrovni

Při **realizaci** programu na **M4**:

1. Generace ekvivalentního programu cílové úrovně ( **M3** )
2. Exekuce nového programu na cílovém stroji ( **M3+M2+M1** )

**Typy:**

1. Zdroj. jazyk: **jazyk symbol.adres** - jazyk symbol.reprezentací stroj.jaz.  
cílový jazyk: strojový jazyk  
překladač: **assembler**
2. Zdroj. jazyk: **problémově orient. jazyk** (ALGOL, PASCAL, C a j.)  
cílový jazyk: strojový jazyk resp. jeho symbolická reprezentace  
překladač: **compiler**

# Úroveň assembleru

## M4 Úroveň assembleru

Program této úrovně je **překládán** do M3 (M2). Jazyk s přímou návazností na stroj. jazyk poč., využívá všech možností instrukcí cílové úrovně.

**Výhody:** symbolická jména (instr., proměn., konst.), symbol. adr. (návěští)

## Formát assembleru

NÁVĚŠTÍ	OPER. KÓD	OPERANDY	KOMENTÁŘ
START :	MOV _	A,B	; přesuň reg B do reg A
...	...	...	...

Mezi jednotlivými poli jsou oddělovače: : \_ , ;

V poli **OPER.KÓD** může být:

- **instrukce** strojového jazyka v symbolickém tvaru
- **pseudoinstrukce** assembleru - příkazy pro řízení překladu
- **makroinstrukce** - náhrada procedury symb. jm. (volání/expanze m.)

V poli **OPERANDY** může být:

- názvy registrů
- symbolická jména, symbolické adresy
- jsou dovoleny aritmetické a log. operace s operandy

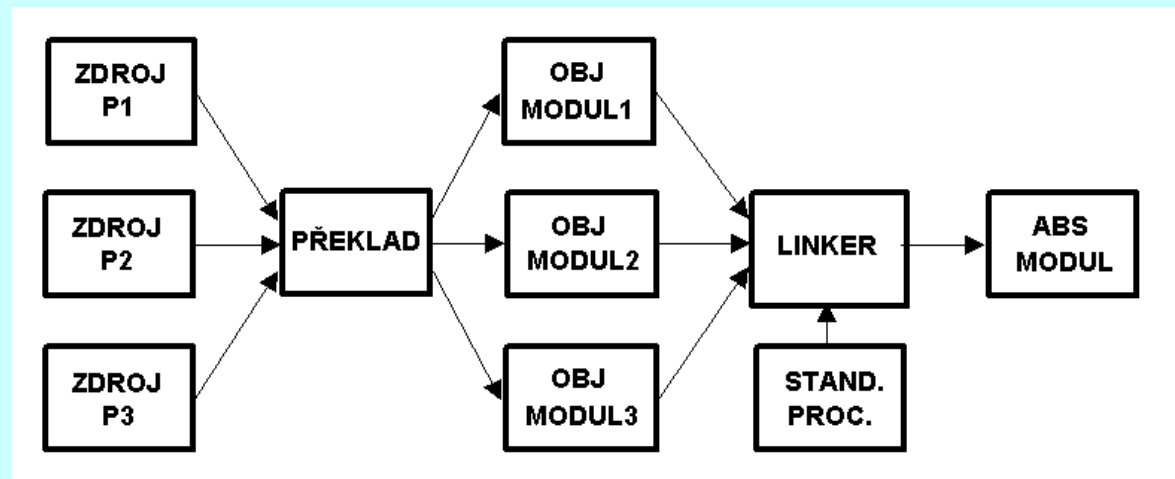


# Úroveň assembleru

## Průběh překladač assembleru (dvouprůchodový překladač)

- 1. průchod:** vytváří **tab. symbolů prgm** (návěští, prom., konst.), kontroluje **syntaxi** instr., přiřazuje řádkům hodnotu instr. čit..
- 2. průchod:** generuje **objekt prgm** - hledá OPCODE, přiřazuje symb. jm. adr. a hodnoty PC., generuje **informační blok** pro LINKER.

Přeložený **OBJ prgm**  
je **relokovatelný**  
bez procedur  
(v knihovně)



**Sestavení** OBJ programů s procedurami a umístění na pevnou adr. provádí **LINKER (+ LOADER)**

LINK	- spojování -> relativní modul
LOAD	- umíst'ování -> absolutní modul

# Úroveň assembleru

## Struktura OBJ modulu

<b>IDENTIFIKACE</b>	— jméno modulu, délka jednotlivých částí
<b>TAB. VSTUP. BODU</b>	— list symbolů v modulu a jejich přiřazení k adresám ( PUBLIC )
<b>TAB. EXTERN. REF.</b>	— list externích symbolů a jejich výskyt v programu ( EXTR )
<b>STROJOVÉ INSTRUKCE</b>	— program ve strojních instrukcích
<b>ADRESÁŘ RELOKACE</b>	— soubor všech adres, které je třeba při relokaci změnit ( bitová mapa )
<b>KONEC MODULU</b>	— identifikace konce a kontrolní suma

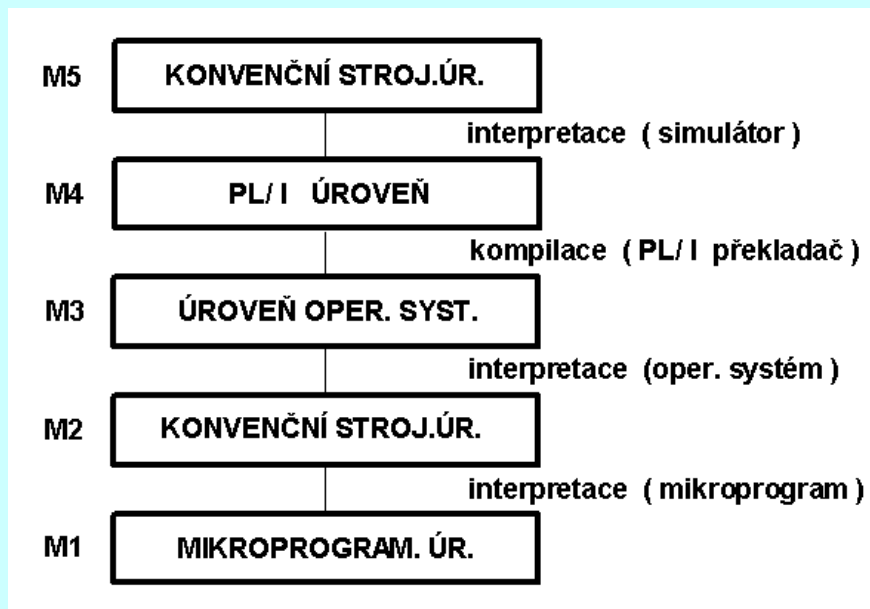
## LINKER

Většina LINKERU má 2 průchody:

1. **průchod** - čte všechny moduly a konstruuje tab. jmen modulů, jejich délky, přiřazuje globální symboly (interní a externí reference)
2. **průchod** - čte moduly, relokuje a linkuje.

# Víceúrovňové stroje

Stroje s **hierarchickou úrovní jazyků**, další úr. mohou být **interpretovány**, **překládány** nebo pracovat s **procedurálním rozšířením**.



**Interpretace** - program je interpret. jako data

**Překlad** - program je překládán do jazyka nižší úrovně

- **assembler** - do cíl. stroj. úr.
- **kompiler** - pro násl. interpret nebo assembler

## Návrh víceúrovňových strojů

**TOP - DOWN** - zhora, navrhují se úr. nižší až po interpreter na  $\mu$ P úr. - **SW**

**BOTTOM - UP** - zdola, - začíná se u  $\mu$ P úr., navrhuje se optim. SW - **HW**

**MIDDLE - OUT** - ze středu, oba směry se navrhují souč., základem je **OS (M2)**