Virtuální paměť

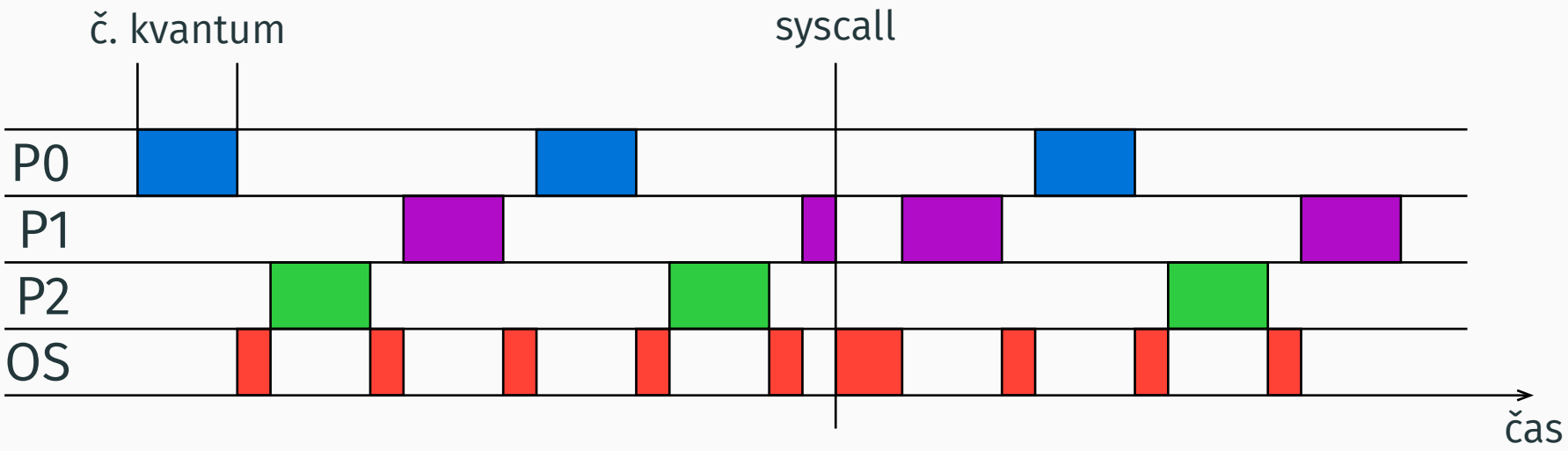
Adresový prostor, base/bounds registry

Milan Radojčić

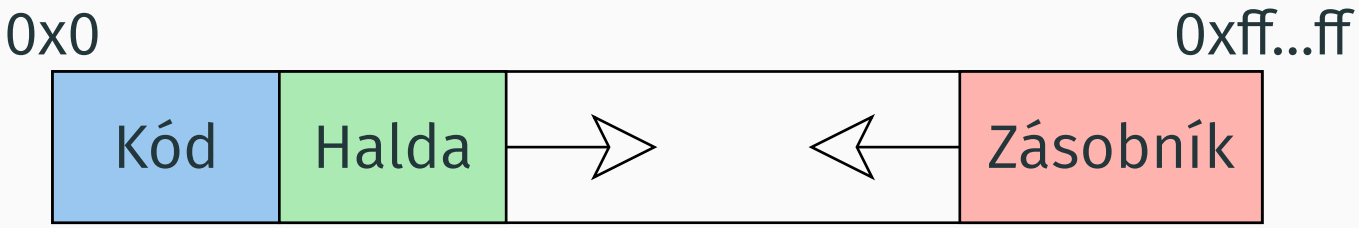
SSPŠaG – Operační systémy

2. 1. 2026

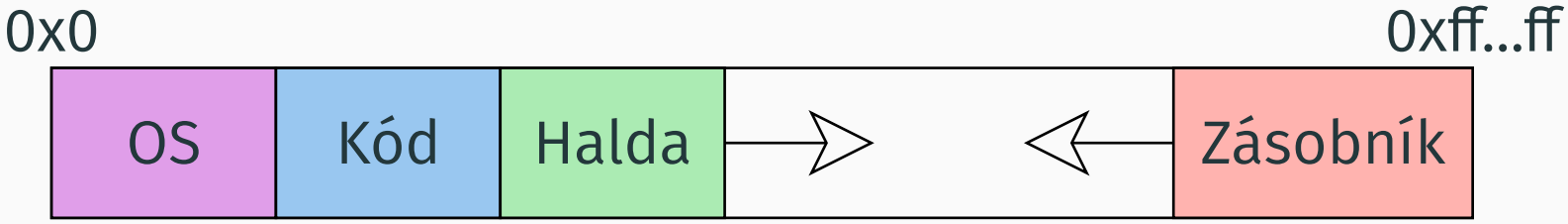
Shrnutí plánování



Rozdělení paměti procesů



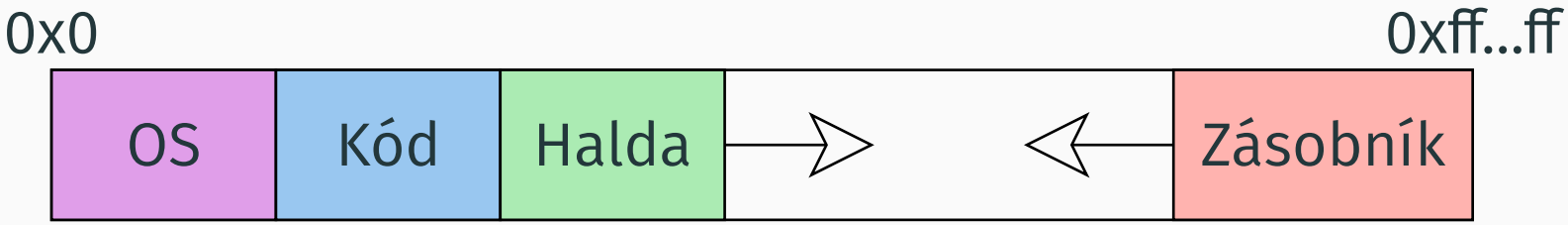
Příklad hl. paměti s 1 procesem



Příklad hl. paměti s 1 procesem a OS

Náš cíl je přijít na to, jak efektivně poskytovat 1 hlavní paměť více procesům.

1. pokus

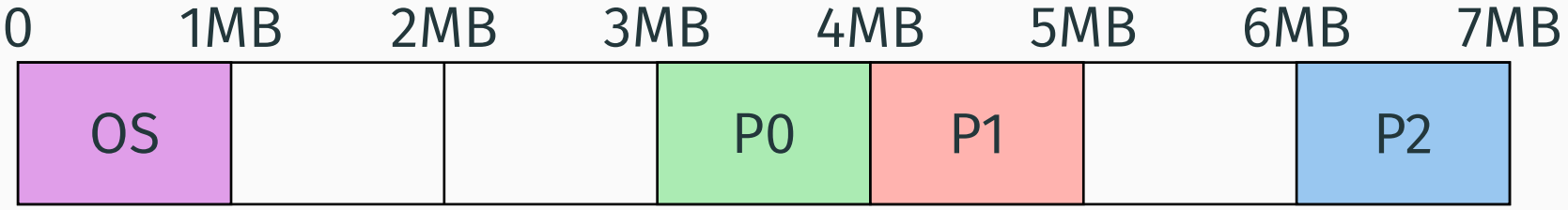


- Do hl. paměti nahrajeme **vždy jen 1 proces**.
- Když ho budeme chtít vyměnit s jiným, tak ho **odložíme na disk a načteme jiný**.

Jaké jsou problémy s tímto přístupem?

2. pokus

- Zkusíme nějakým způsobem udržet víc procesů v paměti.
- Paměť si **rozdělíme na „chlívečky“** o pevně dané velikosti.
- Každý proces dostane chlíveček.

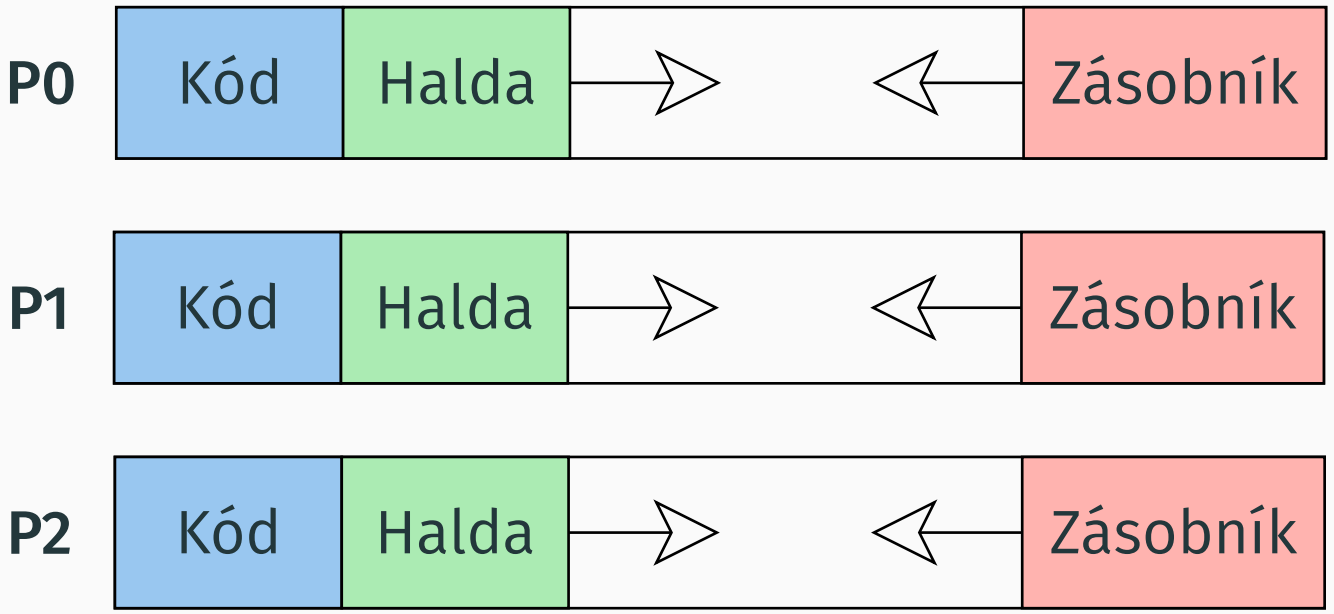


Jaké můžou být problémy tady?

Neposkytujeme **žádnou protekci paměti** — P0 může např. klidně číst paměť P1 nebo klidně i OS.

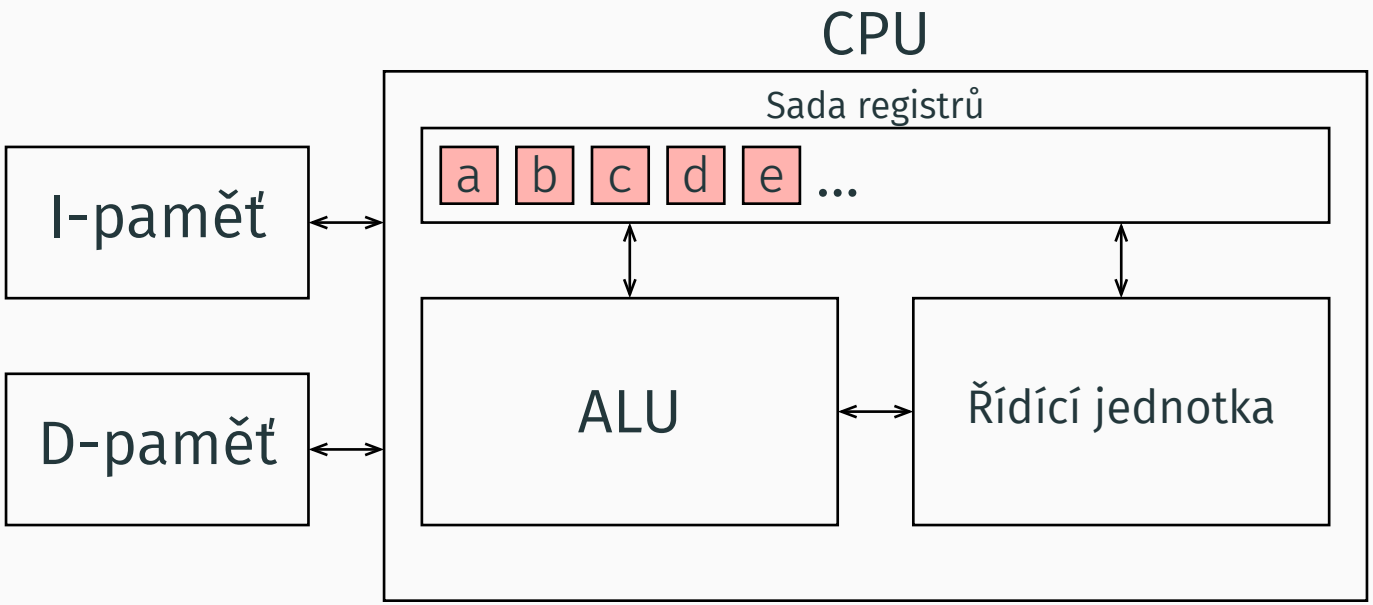
Adresový prostor

- Chceme každému procesu prezentovat vlastní **adresový prostor**.[1]
- Procesy mohou přistupovat **pouze k jejich** adresovému prostoru.
- Virtuální adresový prostor se bude nějak mapovat do **hlavní paměti**.



[1] angl. address space; Prohlédnout si adresový prostor nějakého procesu v Linuxu můžeme pomocí `cat /proc/<pid>/maps`.

Jednoduchá architektura CPU



Base/bounds

- Přidáme dva nové registry: **base** a **bounds**.
  - **base**: offset adresového prostoru od 0
  - **bounds**: velikost adresového prostoru („limit“)
- HW bude
  1. **přičítat k adrese base** a
  2. kontrolovat, že **adresa není větší než bounds**.

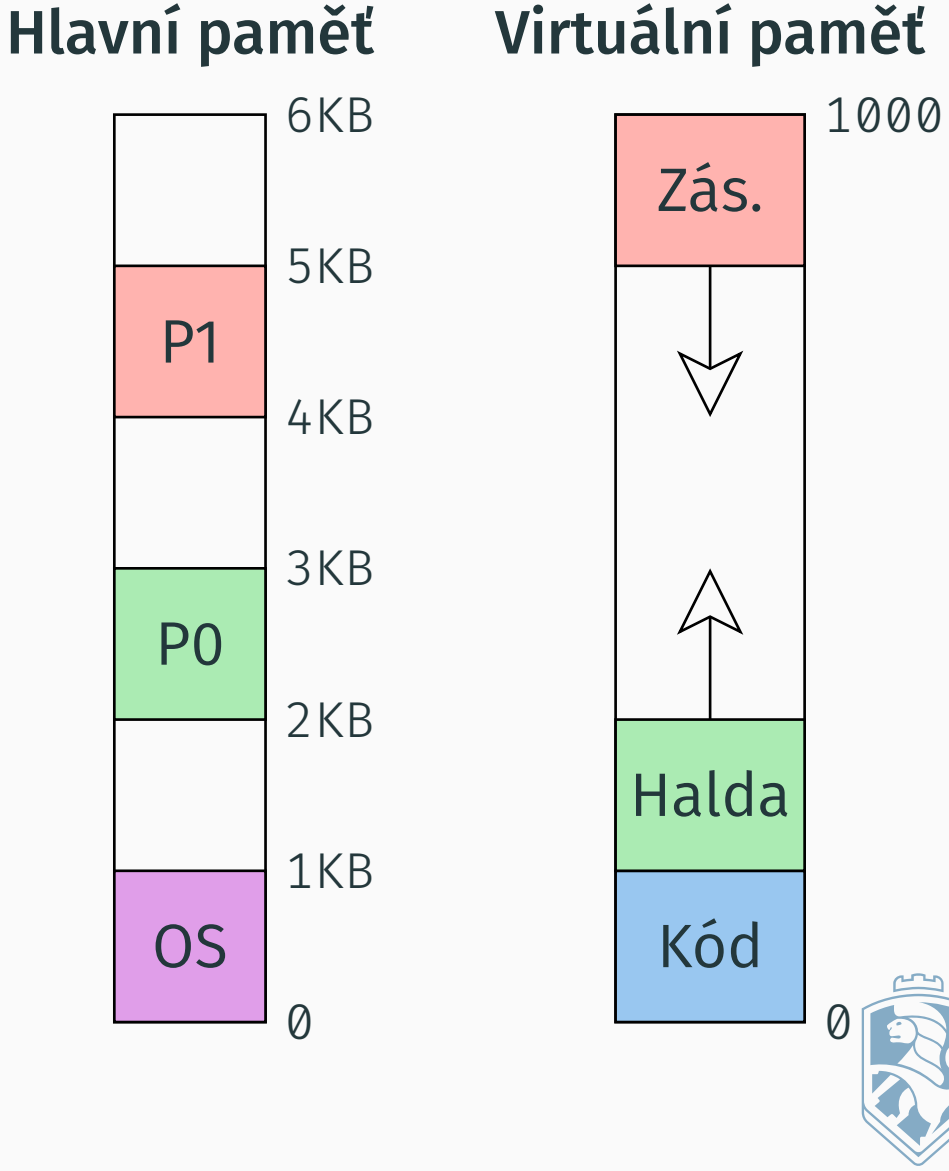
Base/bounds – příklad 1

- Je spuštěn **P0**.
- **Base**: `2000`
- **Bounds**: `1000`
- Provádíme instrukci:

```
1    ld 16, $a
```

- Co se stane?

1. Kontrola: `16 < 1000`
2. Přepočet adresy: `16 + 2000 = 2016`



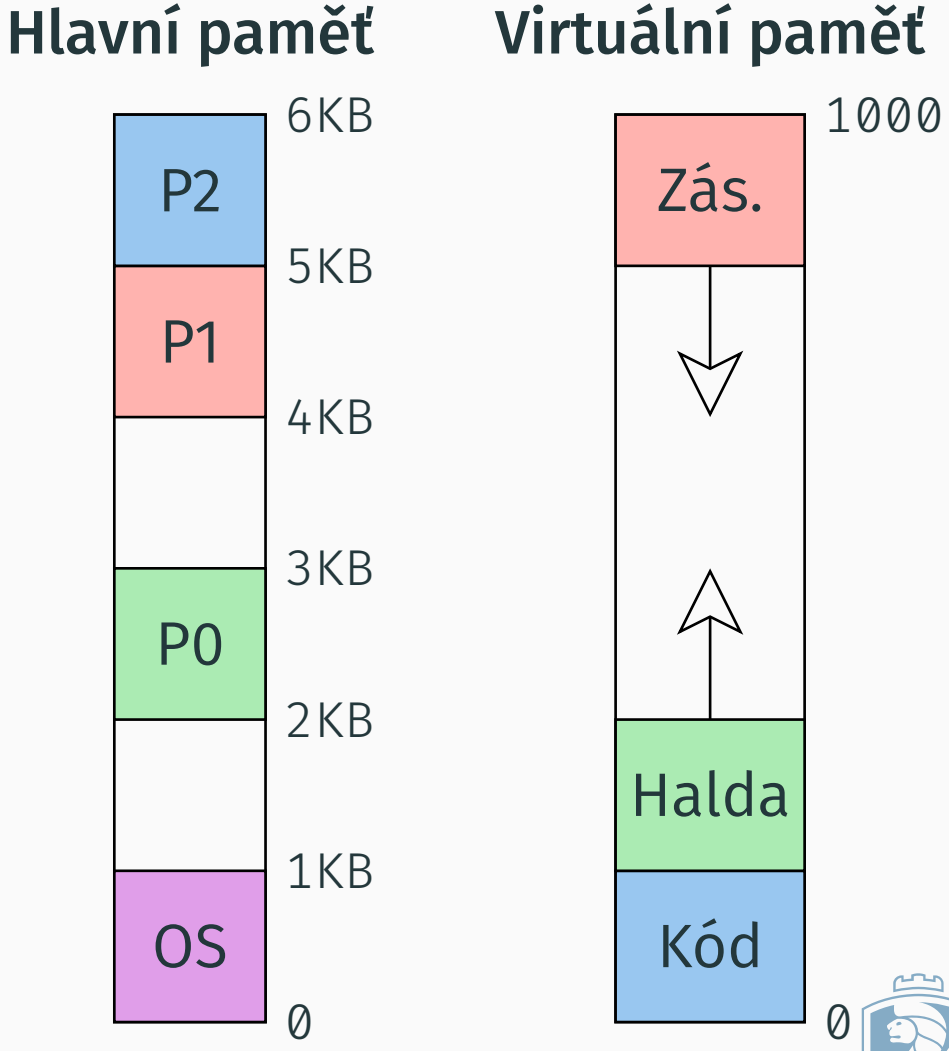
Base/bounds – příklad 2

- Je spuštěn **P1**.
- **Base**: `4000`
- **Bounds**: `1000`
- Provádíme instrukci:

```
1    ld 1000, $a
```

- Co se stane?

1. Kontrola `1000 < 1000`
2. **Přístup není validní!**
   - Vyvolá se výjimka
   - Proces je ukončen.



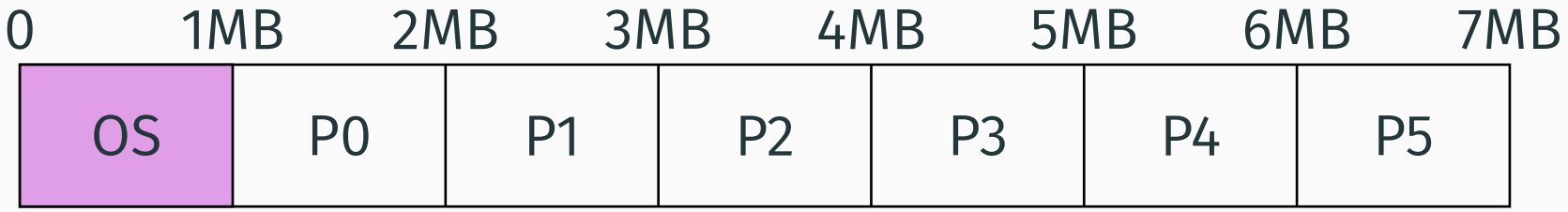
Problémy base/bounds

Jaké můžou být problémy?

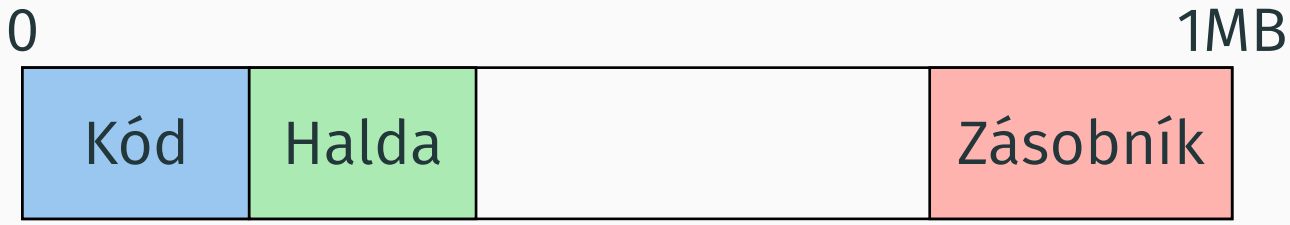
1. **Interní fragmentace**
2. Co když velikost adresového prostoru > velikost hlavní paměti?

Interní fragmentace

- V paměti je každý „chlíveček“ obsazený.



- Co když ale většina adresových prostorů vypadá takhle?



- Paměť **vypadá plně, i když plná není.**

Princip segmentace

- Pro každý segment (kód, halda, zásobník, ...) můžeme vytvořit **vlastní base/bounds**.
  - Každý segment budeme do paměti dávat zvlášť.

Segmentation violation/fault

Odsud pochází termín **segmentation violation** (fault). Ta nastane, když se přistupuje k nevalidní adrese.

- Segmentace ani base/bounds se na moderních počítačích nepoužívá.

Shrnutí

- Virtualizace paměti, adresový prostor
- Base/bounds registry, překlad adres
- Princip segmentace
- Příště si ukážeme **stránkování**.
  - Moderní přístup, který se používá.

Bonusové cvičení 1

- Udělat jednoduchý obvod, který bude provádět překlad adres.
- 2 bonusové body

- Vstupy: *base* (8b), *bounds* (8b), *virt. adresa* (8b)
- Výstupy: *fyzická adresa* (8b), *valid* (1b)
- *valid* = *virt. adresa* < *bounds* a překlad je validní
- *fyzická adresa* = *virt. adresa* + *base*

- Všechny čísla ve vstupech a výstupech jsou unsigned.[2]
- Úloha je jako bonus i v APS.[3]
- Odevzdání bude na Submitty.

[2] Co by jako měla znamenat záporná adresa?

[3] Jenom při použití základních hradel — vlastní sčítačka a vlastní komparátor.

Bonusové cvičení 2

- Zkuste si prográmek `pmap`.
- 3 bonusové body

1. Co dělá? Přečtěte si shrnutí v manuálu.
2. Zkuste si ho na nějakém procesu.
3. V čem se liší adresový prostor, který vidíte, od našeho zjednodušeného?
4. Co asi znamená předposlední sloupec? Proč bychom chtěli mít možnost tohle nastavit?

- Odevzdání buď ukázání na hodině nebo mailem krátký writeup.