B4B35OSY: Operační systémy

Lekce 6. Systém NOVA a stránkování

Petr Štěpán
stepan@fel.cvut.cz

19. října, 2022

Outline

Obsah

Názvosloví

- FAP
  - fyzický adresní prostor
  - skutečná paměť počítače – RAM
  - velikost závisí na možnostech základní desky a na osazených paměťových modulech
- LAP
  - logický adresní prostor
  - někdy také virtuální paměť
  - velikost záleží na architektuře CPU
    - 16 bitová adresace – 64 KiB
    - 20 bitová adresace – 1 MiB
    - 32 bitová adresace – 4 GiB
    - 64(48) bitová adresace – 16 EiB (256TiB)

Počítače bez správy paměti

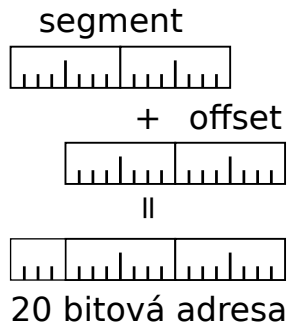
- Výhody systému bez správy paměti:
  - rychlost přístupu do paměti
  - jednoduchost implementace
  - lze používat i bez operačního systému – robustnost
- Nevýhody systému bez správy paměti
  - Nelze kontrolovat přístup do paměti (kdokoli může cokoli v paměti přepsat)
  - Omezení paměti vlastnostmi HW
- Použití
  - Historické počítače
    - Osmibitové počítače (procesory Intel 8080, Z80, apod.)
    - 8bitová datová sběrnice, 16bitová adresová sběrnice, možnost využít maximálně 64 kB paměti
  - Programovatelné mikrokontrolery
  - Řídicí počítače – embedded
  - V současné době již jen ty nejjednodušší řídicí počítače 8/16-bitové(např. Atmel Xomega)

Jednoduché segmenty

Jednoduché segmenty – Intel 8086

- Procesor 8086 má 16bitovou datovou sběrnici a registry
- Procesor má 20 bitů adresové sběrnice.
- 20 bitů je ale problém. Co s tím?
- Řešením jsou jednoduché segmenty:

- Procesor 8086 má 4 tzv. segmentové registry
- Adresa je tvořena adresou segmentu 16 bitů a
- adresou uvnitř segmentu (offset) 16 bitů.
- Výsledná FA se tvoří podle pevného pravidla:
  - $adr = (segment << 4) + offset$

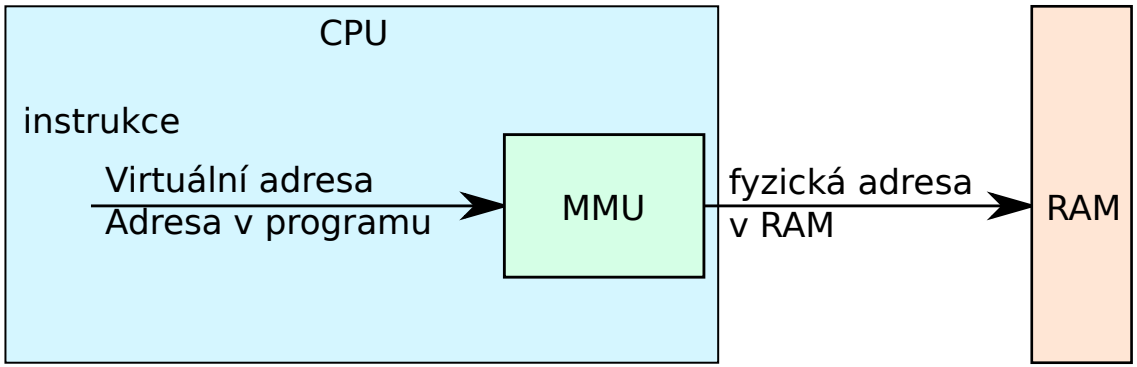


- Prostředek, jak používat větší paměť než dovoluje systém
- Využívá se i v současnosti u jednoduchých 16-bitových procesorů (např. Infineon xc167)
  - někdy se hovoří o mapování
- zavádí dva druhy adres:
  - near – uvnitř segmentu, pouze 16bitový ukazatel
  - far – mezi segmenty, 16bitový ukazatel a 16 bitů číslo segmentu

Segmentace

Skutečné segmenty – Intel 80286

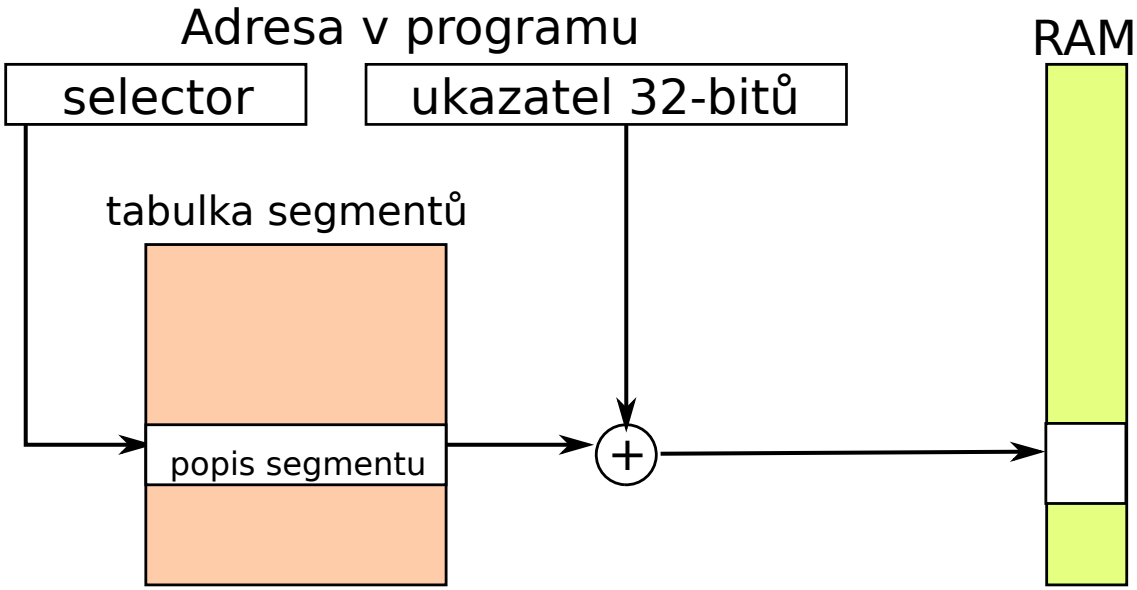
- První procesor s MMU na čipu
  - MMU – memory management unit – převádí adresu (16 bit selektor, 16 bit offset) na 24 bitů adresa ve fyzické paměti



- Program je kolekce segmentů
- Každý segment má svůj logický význam:
  - hlavní program, procedura, funkce,
  - objekt a jeho metoda, proměnné, pole, ...

Segmenty – Intel 80286

- Základní úkol – převést adresu typu (segment selector, offset) na adresu FAP
  - CS, DS, SS, ES – code, data, stack, extra segment selector 16 bitů
    - bit 0–1 RPL – request privilege level – úroveň ochrany
    - bit 2 TI – 0 – global descriptor (patří všem procesům), 1 – local descriptor (jen pro jeden proces)
    - bit 3–15 – index v tabulce



Segmenty – Intel 80286

- Tabulka segmentů – Segment table (ST) – Zobrazení 2-D (segment, offset) LAP do 1-D (adresa) FAP
- ST je uložena v normální paměti RAM, informace vybraných segmentech CS, DS, ES, SS je uložena uvnitř CPU
  - Registr L/GDT – Local/global descriptor table 24-bitů – umístění tabulky segmentů v pměti
  - Registr LL/GDT – limit Local/global descriptor table16 bitů – počet segmentů procesu, velikost tabulky
- Položka ST (64 bitů):
  - base – 24 bitů – počáteční adresa umístění segmentu ve FAP,
  - limit – 16 bitů – délka segmentu (max. 64 KiB)
  - práva – 8 bitů – P – presnet, DPL – descriptor privilege level, S – Segment descriptor (system/user) oprávnění ke zápisu, E- executable, ED Expansion direction (>limit, <limit), w- Writeable, A =accessed
  - rezerva – 8 bitů – pro 386 rozšíření
    - 386 modifikuje segmenty na 20 bitů limit, 32 bitů adresa
    - G – velikost segmentu je v bajtech, nebo v 2^{12} tj. 4KiB – maximální velikost 2^{32}
- vzdálené skoky – lcall – long call, int – i změna segmentu, lret, iret – vzdálený návrat

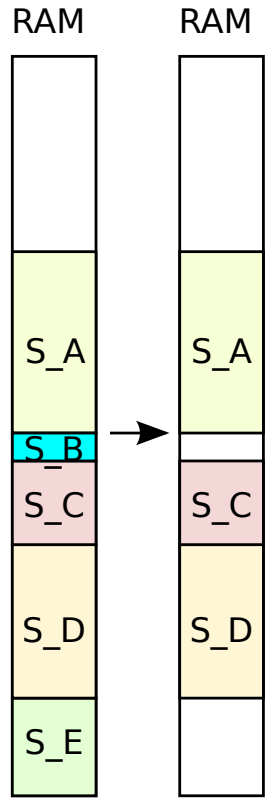
Segmentace – vlastnosti

- Výhody segmentace
  - Segment má délku uzpůsobenou skutečné potřebě
    - minimum vnitřní fragmentace
    - Lze detekovat přístup mimo segment, který způsobí chybu segmentace – výjimku typu „segmentation fault“
  - Lze nastavovat práva k přístupu do segmentu
    - Operační systém požívá větší ochrany než aplikační proces
    - Uživatel nemůže ohrozit operační systém
  - Lze pohybovat s daty i programem v fyzické paměti
    - posun počátku segmentu je pro aplikační proces neviditelný a nedetekovatelný
- Nevýhody segmentace
  - Alokace segmentů v paměti je netriviální úloha
    - Segmenty mají různé délky. Při běhu více procesů se segmenty ruší a vznikají nové.
    - Problém s externí fragmentací
  - Režie při přístupu do paměti
    - Převod na lineární adresu se opírá o tabulku segmentů a ta je také v paměti
    - Při změně segmentového registru – nutné načíst položku z tabulky
    - Častá změna segmentů (po pár instrukcích) – časově náročná

Alokace segmentů

Obecný problém – alokace bloků paměti – umístění segmentu v RAM

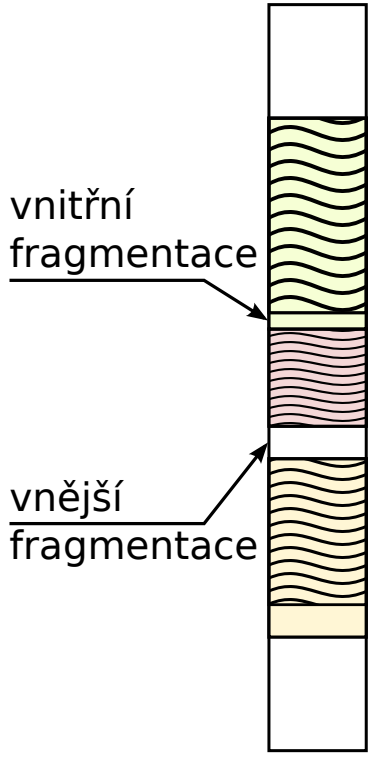
- ”Díra-blok neobsazené paměti
- Segmentu se přiděluje díra, která jeho požadavek uspokojí
  - tím může vzniknout další malá díra
- Díry jsou roztroušeny po FAP
- Evidenci o dírách a obsazených místech udržuje jádro OS
- Kde přidělit oblast délky *n*, když je volná paměť rozmístěna ve více souvislých nesousedních sekcích?
  - First fit – první volná oblast dostatečné velikosti – rychlé, nejčastější
  - Best fit – nejmenší volná oblast dostatečné velikosti – neplýtvá velkými děrami, mohou vznikat mini-díry
  - Worst fit – největší volná oblast – zanechává velké volné díry vhodné pro další alokaci



Fragmentace

Obecný problém nevyužitelného prostoru

- Externí (vnější) fragmentace
  - Celkové množství volné paměti je sice dostatečné, aby uspokojilo požadavek procesu, avšak prostor není souvislý, takže ho nelze přidělit
  - Existence mnoha malých děr
- Interní (vnitřní) fragmentace
  - Přidělená díra v paměti je o málo větší než potřebná, avšak zbytek je tak malý, že ho nelze využít
- Redukce externí fragmentace pomocí setřásání
- Přesouvají se obsahy úseků paměti s cílem vytvořit (jeden) velký souvislý volný blok
- Použitelné pouze při dynamické relokaci
- Při absolutních adresách v paměti by bylo nutno přepočítat a upravit všechny adresy v instrukcích
- Problém s I/O: S vyrovnávacími pamětmi plněnými z periferií (zejména přes DMA) nelze kdykoli hýbat, umisťují se proto do prostoru JOS



Stránkování

Stránkování – Intel 80386

- Procesor 386 – 32 bitový procesor – přidal k segmentům stránkování:
  - Souvislý LAP procesu není zobrazován jako jediná souvislá oblast FAP
- FAP se dělí na úseky zvané rámce
  - Pevná délka, zpravidla v celistvých mocninách 2
    - (512 až 8.192 B, nejčastěji 4KiB, ale někdy i 4 MiB)
- LAP se dělí na úseky zvané stránky
  - Pevná délka, shodná s délkou rámců
- Proces o délce *n* stránek se umístí do *n* rámců
  - rámce ale nemusí v paměti bezprostředně sousedit
- Mechanismus překladu logická adresa $\rightarrow$ fyzická adresa
  - pomocí tabulky stránek (PT = Page Table)
- Může vznikat vnitřní fragmentace
  - stránky nemusí být zcela zaplněny

Kvíz – velikost stránek a rámců

Na čem závisí velikost stránek a rámců?

A - Na velikosti paměti.

B - Na velikosti disku.

C - Na architektuře procesoru.

D - Na velikosti adresové sběrnice paměti.

Stránkování – překlad adres

- Logická adresa použitá v programu se dělí na:
  - číslo stránky, p (index do tabulky stránek)
    - tabulka stránek obsahuje počáteční adresy rámců přidělených stránkám
  - posunutí (offset) ve stránce, d
    - relativní adresa (posunutí = offset, displacement) ve stránce/v rámci
- Protože velikost stránky je mocnina 2, je rozklad na číslo stránky a posunutí jednoduchý
  - číslo stránky $p = \frac{addr}{2^k} = addr >> k$
  - posunutí $off = addr \% 2^k = addr \& (2^k - 1)$
- V jazyce C je rozklad na číslo stránky pro 32-bitový systém s 4KiB stránkami
  - číslo stránky `p = addr >> 12`
  - posunutí `off = addr & 0x00000FFF`

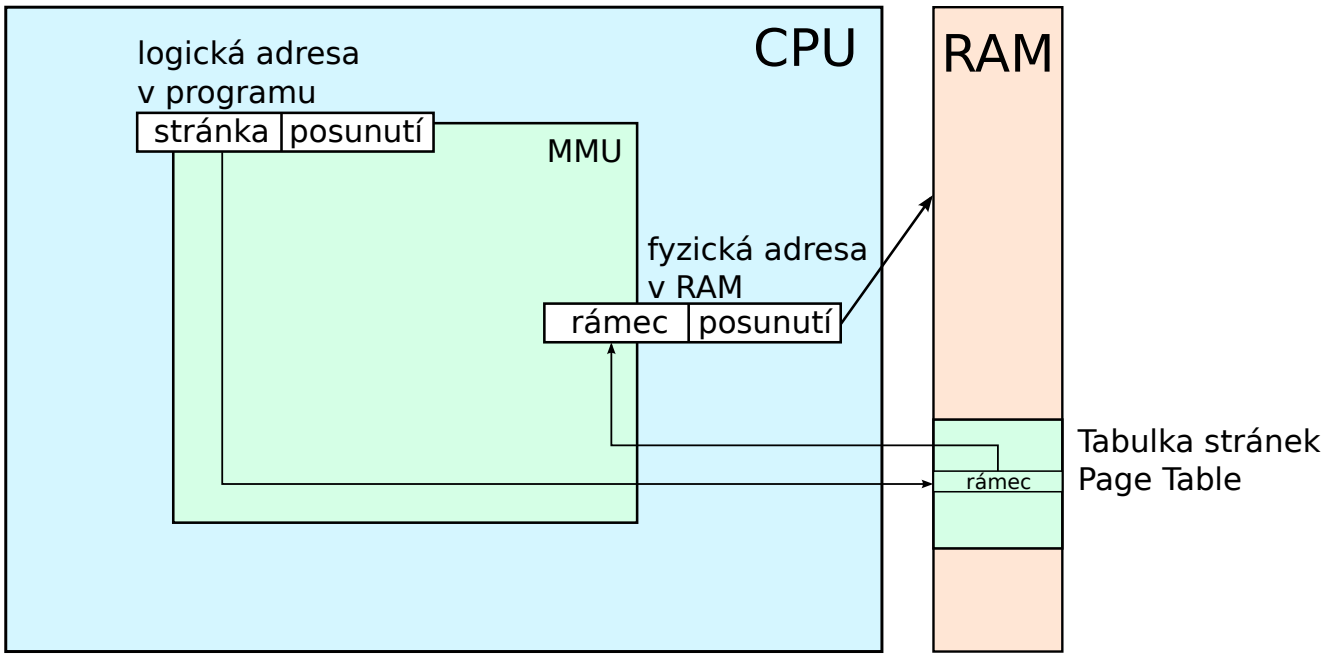
Obsah tabulky stránek

Položky tabulky stránek:

- Číslo rámce
  - umístění stránky v reálné paměti počítače
- Atributy stránek
  - Základní příznaky
    - p present – stránka je v paměti, číslo rámce je platné
    - ps page size – velikost stránky
    - g global – stránka je globální, nepatří jednomu procesu
  - Řízení přístupu
    - r/w read/write – povolení zápisu do stránky
    - u/s user/supervisor – povolení přístupu pro uživatele
  - Optimalizace
    - pwt page-level write through – nastavení cache
    - pcd page-level cache disable – zákaz použití cache
  - Statistika
    - a accessed – stránka použita pro čtení
    - d dirty – stránka použita pro zápis
  - Virtualizační příznaky
    - v/i = valid/invalid indikuje přítomnost stránky ve FAP
    - a = accessed značí, že stránka byla použita
    - d dirty indikuje, že obsah stánky byl modifikován

Efektivita tabulky stránek

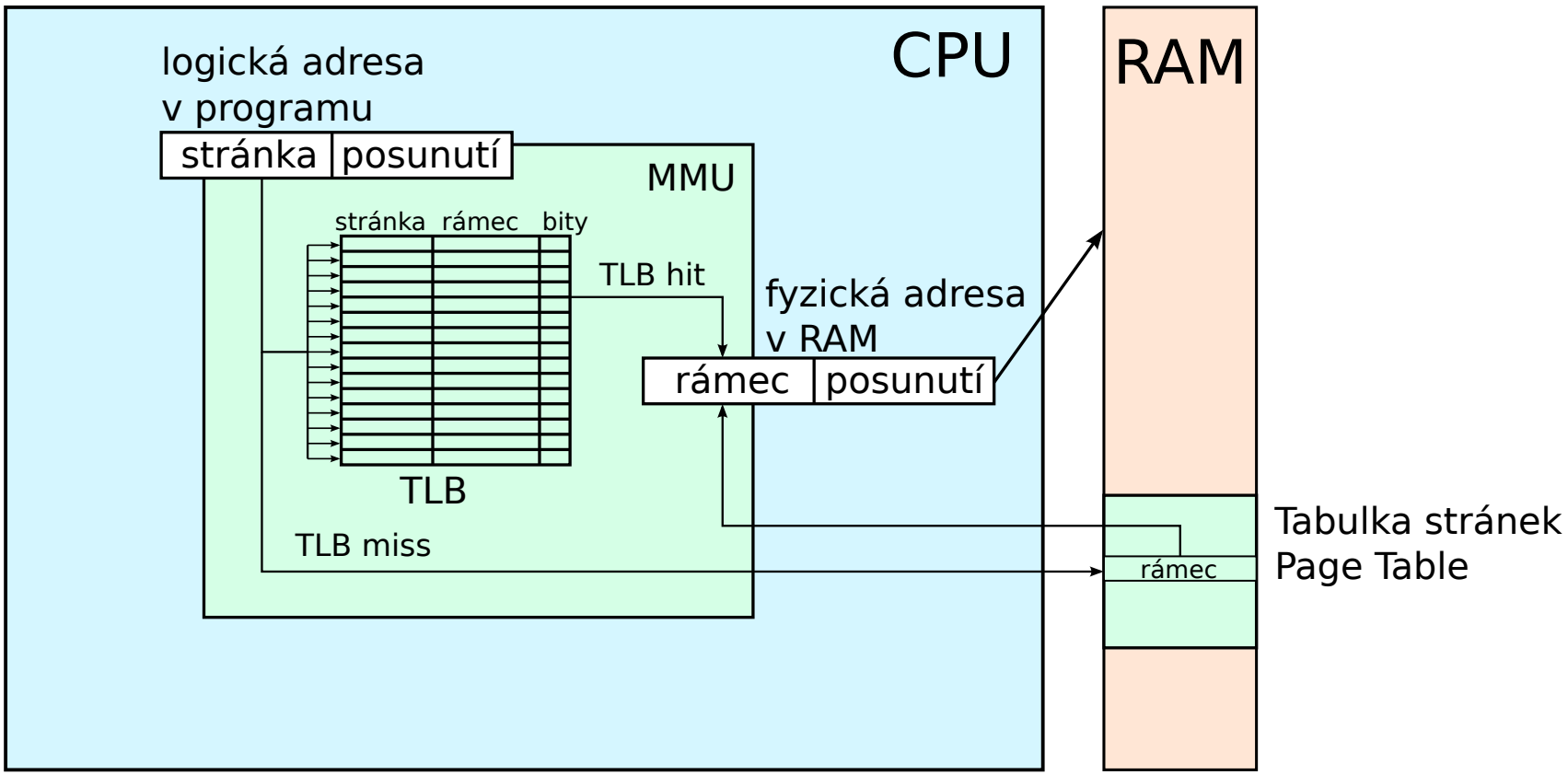
- každý přístup do paměti znamená, že je potřeba převést číslo stránky na číslo rámce, to znamená číst z RAM
- jedna instrukce může číst i dvakrát z paměti z různých stránek
- jedna instrukce tedy potřebuje 4 přístupy do paměti
- přístup do pamět velmi zdržuje



TLB

- Řešení – speciální rychlá cache paměť pro čísla rámců a čísel stránek – Translation Lookaside Buffer
- asociativní paměť – paměť adresovaná obsahem
  - oproti normální paměti, ptám se, kde v paměti je hodnota 15?
- TLB je asociativní paměť
- relativně malá kapacita, vysoká efektivita a zrychlení přístupu do paměti
- nevýhoda – obvodová složitost implementace TLB
- MMU se zeptá TLB znáš hodnotu rámce pro číslo stránky p? Odpověď buď ano je to *r* (TLB hit), nebo neznám (TLB miss)

Tabulka stránek



Význam TLB

- Skutečná přístupová doba – Effective Access Time (EAT) – 10–100 cyklů procesoru
- Přístupová doba TLB – 0.5 – 1 cyklus procesoru
- Neúspěšnost TLB, TLB miss – 0.01 % – 1 %
- Příklad:
  - Přístupová doba do fyzické paměti $t = 30 cyklu$, do TLB $t_{TLB} = 1 cyklus$
  - Neúspěšnost TLB $\alpha = 0.01$ (jedno procento)
  - Stránkování bez TLB $t_{celkem} = 2 \cdot t = 60 cyklu$
  - Průměrná doba přístupu do paměti s TLB $t_{celkem} = \alpha \cdot (t_{TLB} + 2 \cdot t) + (1 - \alpha) \cdot (t_{TLB} + t) = 31.3 cyklu$

Vliv TLB

- Velikost TLB 8–4096 položek
- Moderní procesory mají více-úrovní TLB – podobně jako úrovně cache
- Intel Core i7 má 64 TLB položek L1 první úrovně a 1536 TLB položek L2 úrovně
- I pro malé TLB je úspěšnost nalezení položky 99–99.99% (souvisí s principem lokality tj. prostorovou závislostí programů)
- Problém TLB je při změně procesu a tím i změně tabulky stránek
- Intel umožňuje speciálními instrukcemi ponechat stránku v TLB

Kvíz – velikost tabulky stránek

Kolik položek má tabulka stránek?

A - Závisí na velikosti RAM.

B - Závisí na velikosti virtuálního prostoru.

C - Závisí na velikosti programu a dat.

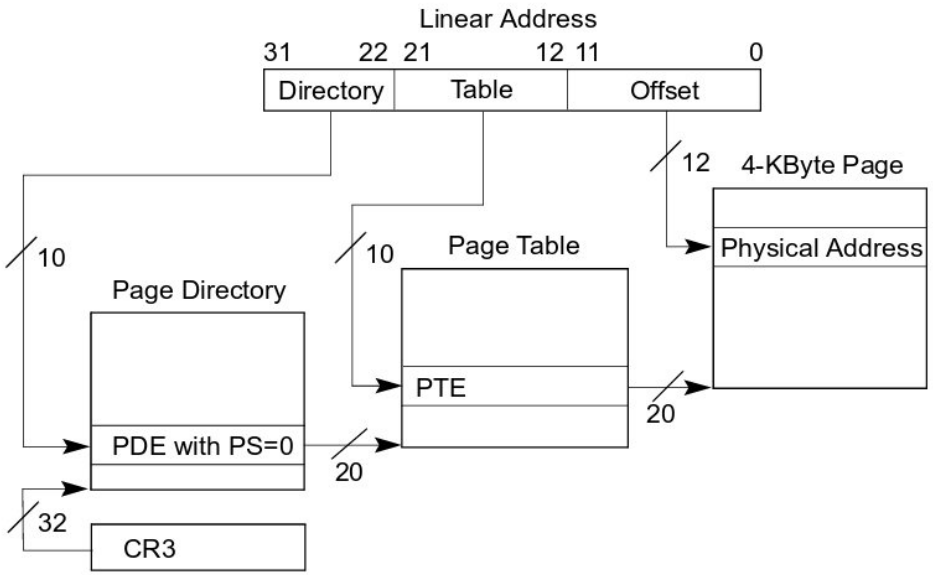
D - Závisí na všem výše uvedeném.

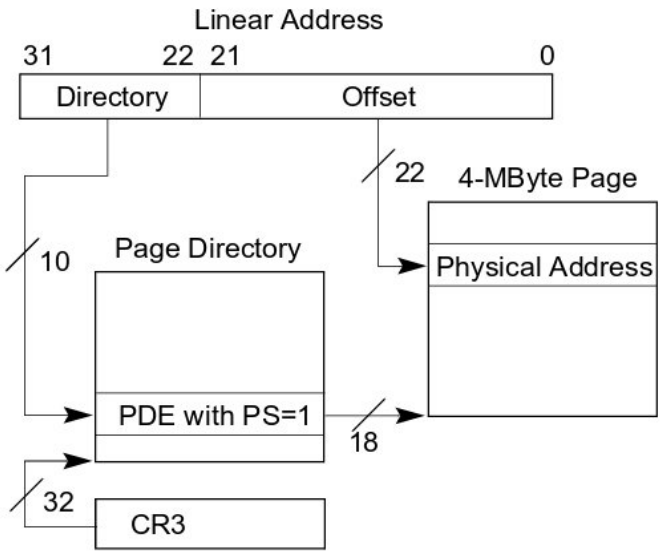
Velikost tabulky stránek

- Otázka velikosti PT
  - Každý proces má svoji PT
  - 32-bitový LAP, 4 KiB stránky – PT má 1 Mi položek, tj. velikost 4 MiB
  - 64-bitový LAP, 4 KiB stránky – PT má 4 Pi (peta) položek, tj. velikost 32 PiB
  - musí být stále v hlavní paměti
- Hierarchické stránkování
  - Zobrazování LAP se dělí mezi více úrovní PT
  - Pro 32-bitový LAP typicky dvouúrovňové PT
  - PT 0 obsahuje definice (odkazy) vlastních tabulek PT 1
  - Tabulky stránek nižších úrovní mohou být odkládány na disk
  - v RAM lze zobrazovat jen skutečně využité stránky s vlastními PT
- Hašovaná PT
  - Náhrada přímého indexování číslem p v PT hašovací funkcí hash(p)
- Invertovaná PT
  - Jediná PT pro všechny koexistující procesy
  - Počet položek je dán počtem fyzických rámců
  - Vyhledávání pomocí hašovací funkce hash(pid, p)

Dvouúrovňové stránkování 32-bitů

- 32-bitový procesor se stránkou o velikosti 4 KiB – 12 bitů posunutí (offset)
- 10 bitů index v tabulce tabulek (page directory – PT_0)
- 10 bitů index v tabulce stránek (page table – PT_1)
  - při nastaveném bitu PS – velikost stránky 4 MiB, nepoužije se tabulka PT_0





Kvíz

Kdo provádí převod virtuální adresy na fyzickou?

A - operační systém v rámci přerušení.

B - operační systém jako svoji službu.

C - CPU konkrétně MMU.

D - sběrnice propojující CPU a pamětí.

Více úrovňové stránkování (32bitů) – bitová aritmetika

Tabulka tabulek - vrchních 10 bitů adresy

```
pdir[virt >> 22]
```

Test přítomnosti tabulky stránek v paměti

```
if ((pdir[virt >> 22] & PRESENT) == 0)
// tabulka stranek neni v pameti
```

Pozice tabulky stránek v paměti

```
ptab = pdir[virt >> 22] & ~PAGE_MASK;
```

Tabulka stránek - prostředních 10 bitů adresy

```
ptab[(virt >> PAGE_BITS) & 0x3ff]
```

NOVA stránkování 32-bitů

```
#define PAGE_BITS       12
#define PAGE_SIZE       (1 << PAGE_BITS)
#define PAGE_MASK       (PAGE_SIZE - 1)
class Ptab {
    public:
    enum {
        PRESENT = 1<<0,
        RW      = 1<<1,
        USER    = 1<<2,
        ACCESS  = 1<<5,
        DIRTY   = 1<<6, };
        static void insert_mapping (mword virt, mword phys, mword attr);
        static void * remap (mword addr);
};
void Ptab::insert_mapping (mword virt, mword phys, mword attr) {
    mword* pdir = static_cast<mword*>(Kalloc::phys2virt(Cpu::cr3()));
    mword* ptab;
    if ((pdir[virt >> 22] & PRESENT) == 0) {  // add ptab
        ptab = static_cast<mword*>(Kalloc::allocator.alloc_page(1, Kalloc::FILL_0));
        mword p = Kalloc::virt2phys (ptab);
        pdir[virt >> 22] = p | ACCESS | RW | PRESENT | USER;
    } else {                                  // find ptab
        ptab = static_cast<mword*>(Kalloc::phys2virt (pdir[virt >> 22] & ~PAGE_MASK));
    }
    ptab[(virt >> PAGE_BITS) & 0x3ff] = (phys & ~PAGE_MASK) | (attr & PAGE_MASK);
}
```

Kvíz – obsah tabulky stránek

V tabulce stránek je uloženo:

A - číslo stránky, číslo rámce, příznakové bity.

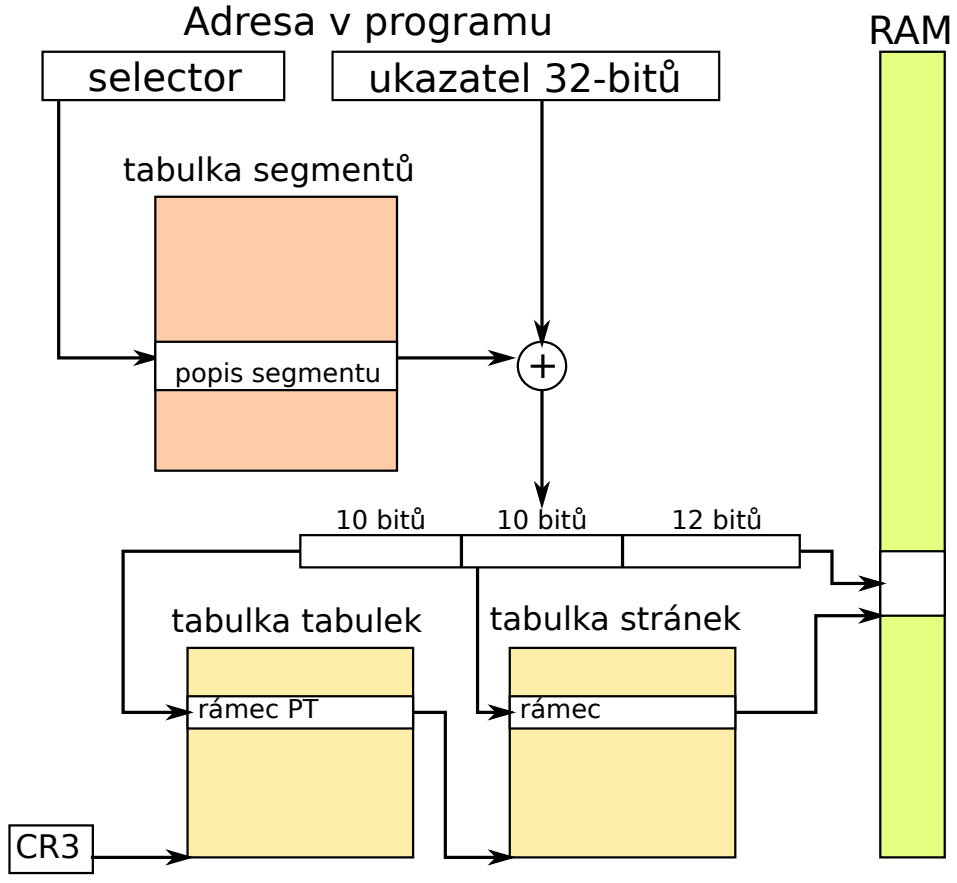
B - číslo rámce, příznakové bity.

C - pouze číslo stránky.

D - pouze příznakové bity.

Segmentace se stránkování IA-32

- V 32 bitovém módu nelze zrušit používání segmentů
- LAP: 2x8 Ki segmentů s délkou až 4 GiB každý
- Logická adresa = (popisovač segmentu, offset), offset = 32-bitová adresa v segmentu
- Lineární adresní prostor všech segmentů se stránkuje s použitím dvouúrovňového mechanismu stránkování
- Délka stránky 4 KiB, offset ve stránce 12 bitů, číslo stránky 2x10 bitů
- OS to řeší nafouknutím segmentů na 4GiB – což ve skutečnosti ruší segmenty



Kvíz

Více-úrovňové stránkování:

A - **zrychluje** nalezení odpovídající paměti, ale **zvyšuje** velikost paměti pro uložní tabulky.

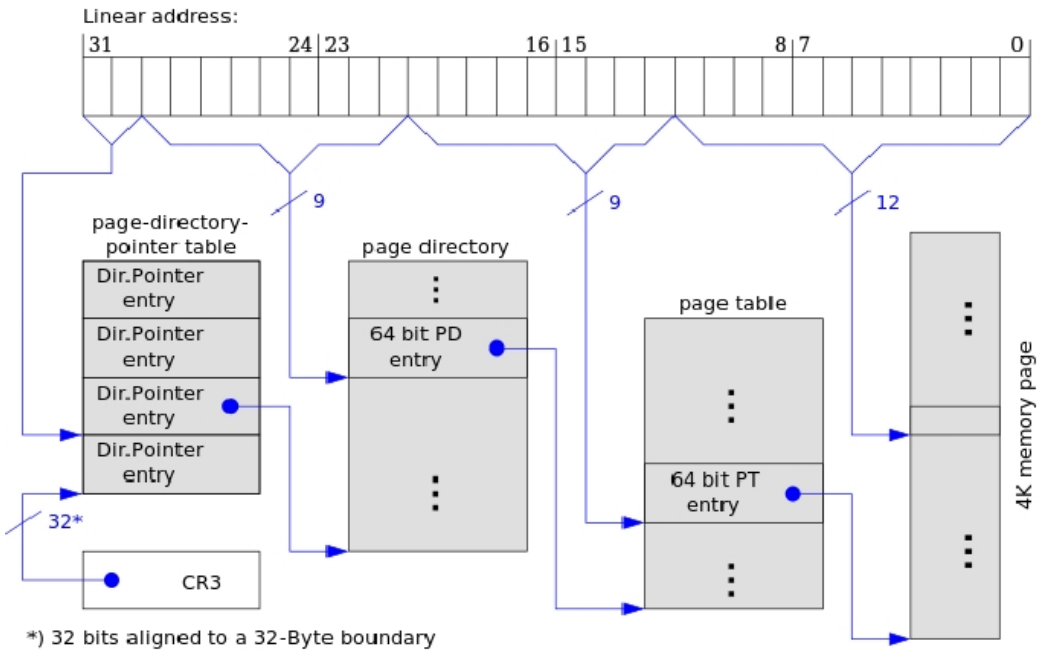
B - **zrcyhluje** nalezení odpovídající paměti a **zmenšuje** velikost paměti pro uložní tabulky.

C - **zpomaluje** nalezení odpovídající paměti, ale **zmenšuje** velikost paměti pro uložní tabulky.

D - **zpomaluje** nalezení odpovídající paměti i **zvyšuje** velikost paměti pro uložní tabulky.

Tříúrovňové stránkování 32-bitů s PAE

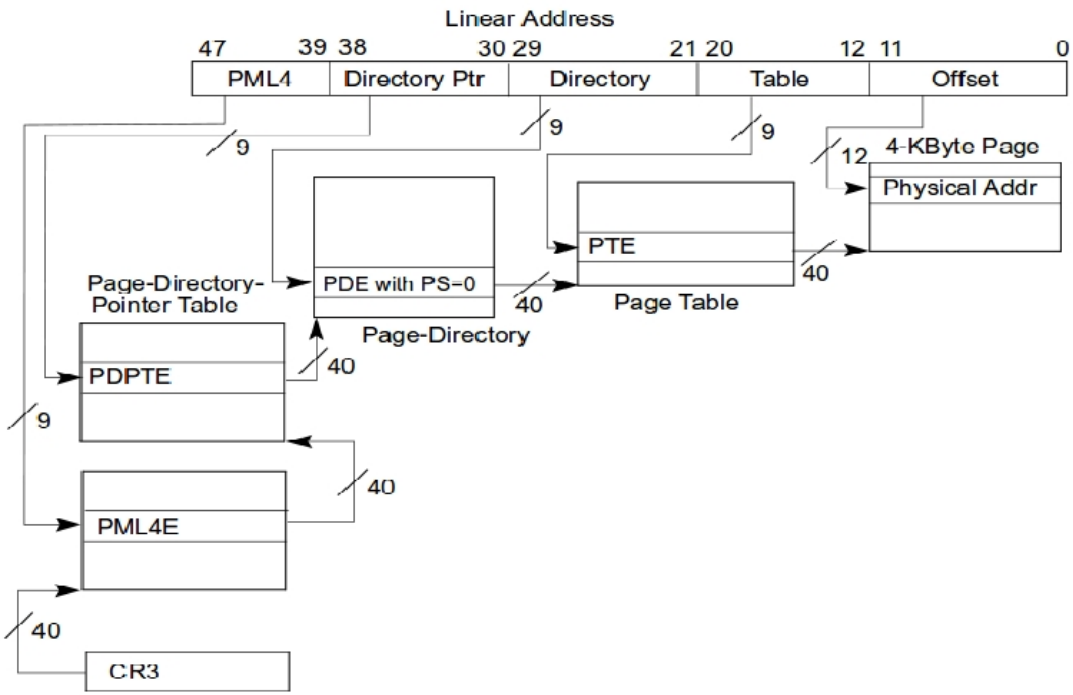
- PAE – fyzická paměť až 4 PiB – 52 bitů pro adresaci, tedy číslo rámce má 40 bitů
- potřebujeme více bitů pro uložení čísla rámce → položka v tabulce stránek bude mít 64 bitů
- do 4KiB se vejde jen 512 položek, tzn. index do této tabulky má 9 bitů
- 32-bitový procesor se stránkou o velikosti 4 KiB – 12 bitů posunutí (offset)
- 9 bitů index v tabulce tabulek
- 9 bitů index v tabulce stránek
- 2 bitů index v tabulce stránek druhé úrovně



Stránkování IA-32e

$4\times$ pomalejší pokud není nalezen překlad v TLB.

- Lineární adresa 48 bitů – Virtuální prostor o velikosti 256 TiB
- Fyzická adresa 52 bitů – což je 4 PiB RAM
- Varianty s 4KiB, 2MiB nebo 1GiB stránkami
- Posunutí 12 bitů, 21 bitů, nebo 30 bitů
- 9 bitů indexy do tabulek tabulek/stránek



Kvíz – ukazatele

Pokud v programu je ukaztel `int *p = &a;` Pak ukazatel `p` obsahuje:

A - virtuální adresu.

B - fyzickou adresu v RAM.

C - ukazatel do stránkovací tabulky.

D - záleží na typu překladače.