Betriebssystem 5.1

Paging 分页的内存管理方法

- Ziel: Zusammenhängende Zuweisung von Speicherbereichen vermeiden
  - externe Fragmentierung verhindern
  - Segmente nach Bedarf vergrößern
- Idee: Adressraum und physischen Speicher in Bereiche fester Größe aufteilen(z.B 4KB) aufteilen
  - Adressraum: Seite(Page)
  - Physischer Speicher: Rahmen(Page Frame)
  - Übersetzung von Seiten-Adressen
    - Höcherwertige Bits: Seitennummer
    - Niederwertige Bits: Offset innderhalb der Seite

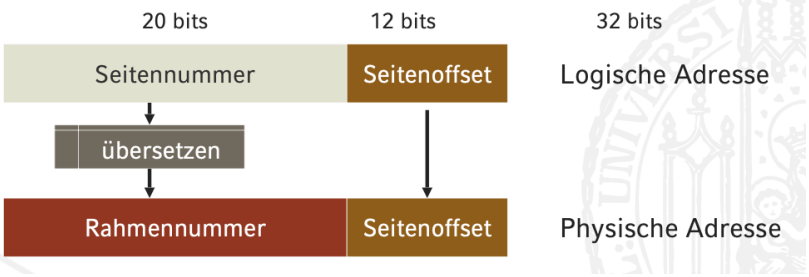

Figure 1: Example of page adress translation into physical adress

- 计算内存划分

```
假设虚拟空间大小 4GB(2^32 字节)，页面大小 4kB(2^12 字节)， 物理内存大小 16kB(2^14)
1. 虚拟空间地址： 2^32 / 2^12 = 2^20 个虚拟页面
2. 物理内存：2^14 / 2^12 = 4 个物理页框
```

- 计算页号和页内偏移量

```
假设页面大小 4kB(0x1000)，虚拟地址为 0x12345.
1. 页号:  0x12345 / 0x1000 = 0x12
2. 偏移量 ： 0x12345 % 0x1000 = 0x345
由此得出，该地址在第 18 页
```

- 分页的基本概念

```
1. Page：内存被划分为固定大小的块，通常为 4KB（也可以是其他大小）。这些块被称为页
2. Page Table：操作系统维护一个页表，用于将虚拟地址转换为物理地址。每个虚拟页面都在页表中有对应的物理页面
3. Page Frame：物理内存也被分成与虚拟页大小相同的块，称为页框，操作系统通过页表将虚拟页映射到物理页框
```

- Adresse von Segmentation vs. Paging
  - Segmentation: Phy. Adresse = virt. Offset + Basisregister
  - Paging: individuelle Zuordung von Seiten zu Rahmen
- Speicherort für Seitentabellen

```
z.B 32 Bit Adressraum, 4KB Seiten, 4 Byte Einträge
1. 32 - log(4096)= 32 - 12 = 20
2. 2^20
3. 2^20 * 4 bytes = 4 MiB
```

- Größe eienr Seitentabelle
  - Bits für Seitennummer = 32 - Bits für Seitenoffset

  - Anzahl Einträge = Anzahl Seiten = 2 ^(Bits für Seitennummer)
  - Größe Seitentabelle = mul (Anzahl Einträge) (Größe eines Eintrags)

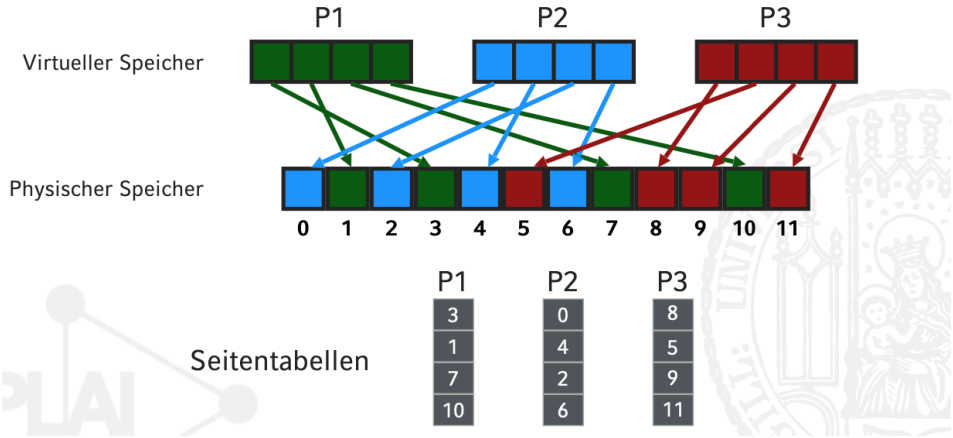

Figure 2: Example of page table

- Informationen in der Seitentabelle(in Bits)
  - Zuordnung
  - Gültigkeit
  - Schutz
  - Präsenz
  - Verwendung
  - Veränderung
- Einträge der Seitentabelle liegt im Hauptspeicher
  - Hardware and BS verwenden gemeinsames Format für Seitentabelle

- Speicherzugriffe mit Paging
  - 访问内存数量翻倍：一次需要访问页表（这本来就处于内存当中），然后再访问真实存在于物理内存中的数据

- Vorteile von Paging
  - Keine externe Fragmentierung(Buddy Allocator)
    - Jede Seite kann in jedem Seitenrahmen im physischen Speicher platziet werden
  - Schnelle Allokation und Freigabe
    - Allokation: keine Suche nach geeignetem freien Platz notwendig
    - Freigabe: kein Verschmelzen notwendig
    - Freie/belegte Rahmen werden mit einer Bitmap angezeigt
  - Swapping
    - Teile des Hauptspeichers lassen sich bei Bedarf leicht in den Festspeicher auslagern
      - Seitengröße passt zu Blockgröße in Festspeicher
      - Prozess kann laufen selbst wenn einzelne Seiten im Festspeicher sind
      - Präsenzbit im Eintrag in der Seitentabelle

- Nachteile von Paging
  - Interne Fragmentierung. Je größe die Seiten, desto mehr interne Fragmentierung.
    - 由于程序的内存需求通常不会恰好等于页面大小（4KB），这就导致了未用空间的浪费，从而产生内部碎片
  - Hohe Kosten durch zusätzlichen Speicherzugriff um Seitentabelle auszulesen
    - Grund:
      - Seitentabelle liegt im Hauptspeicher

    - MMU speichert nur die Startadresse der Seitentabelle
    - Lösung: TLB(translation lookaside buffer)
- Platzbedarf für Seitentabellen
  - Grund:
    - Eintrag für jede Seite, seblst wenn die Seite nicht verwendet wird
    - Problematisch mit dynamischem Stack und Heap im Adressraum
      - 频繁的页面映射和交换: 栈和堆的动态变化使得分页系统需要频繁地更新页表并进行页面交换，这增加了内存管理的复杂性和开销
      - 内存碎片化: 栈和堆的动态分配和释放会导致内存碎片，尤其是在分页系统中，可能无法充分利用已经释放的内存，影响系统性能
    - Seitentabelle muss zusammenhängend gespeichert werden
      - 连续储存（为了访问效率和提高缓存命中率）
        - 页表需要连续的物理内存，但操作系统中的物理内存资源通常是有限的，并且是动态分配的。随着程序运行时，物理内存被不同的进程占用，这会导致物理内存中的空闲区域变得零散。即使空闲内存的总量足够，也可能因为无法找到足够大的连续空间来存储页表而导致内存分配失败。
        - 大量页表性能开销大
    - Lösung: Mehrstufige Seitentabellen