# Proseminar
# **Konzepte von Betriebssystem-Komponenten (KVBK)**

Vortrag zum Thema:

Speicheraddressierung, Segmentierung, Paging

# Speicheradressierung

- Grundlegende Bedeutung von Speicheradressierung:
  - Wie sind die Daten auf Dem Speicher abgelegt?
  - Wie kann ich effizient auf die Daten zugreifen?

# Speicheradressierung

- Wie sind die Daten auf dem Speicher abgelegt?
  - Linear
  - Segmentiert
  - Gekachelt

# Speicheradressierung

- Lineare Speicherung
  - Daten werden linear auf dem Speicher abgelegt
    - Bei Löschen oder Änderung der Größe evtl. Neusortierung des gesamten Speichers nötig

# Speicheradressierung

- Segmentierte Speicherung
  - Daten werden in Segmenten verschiedener Größe auf dem Speicher abgelegt
    - Enormer Verwaltungsaufwand bei zu geringer Segmentgröße
  - Reihenfolge der Segmente nicht zwingend vorgegeben
    - Bei Änderungen muss evtl. nur dem entsprechendem Segment ein neuer Speicherplatz zugeordnet werden
    - Verschwendung von Speicherplatz bei ineffizienter Sortierung der Segmente

# Speicheradressierung

- Gekachelte Speicherung
  - Speicher wird in festgesetzte gleiche Teile Partitioniert (Kacheln)
  - Daten werden als Seiten mit identischer Größe (4 KB) in den Kacheln auf dem Speicher abgelegt
  - Reihenfolge der Kacheln ist linear
    - Änderung der Daten erfordert keine Umsortierung des Speichers, lediglich die Verwaltungstabellen müssen aktualisiert werden
    - Verschwendung von Speicherplatz und Verwaltungsaufwand abhängig von Größe der Kacheln

# Speicheradressierung

- Wie kann ich effizient auf die Daten zugreifen?
  - Lineare Adressierung
  - Segmentierung
  - Seitenadressierung (Paging)

# Speicheradressierung

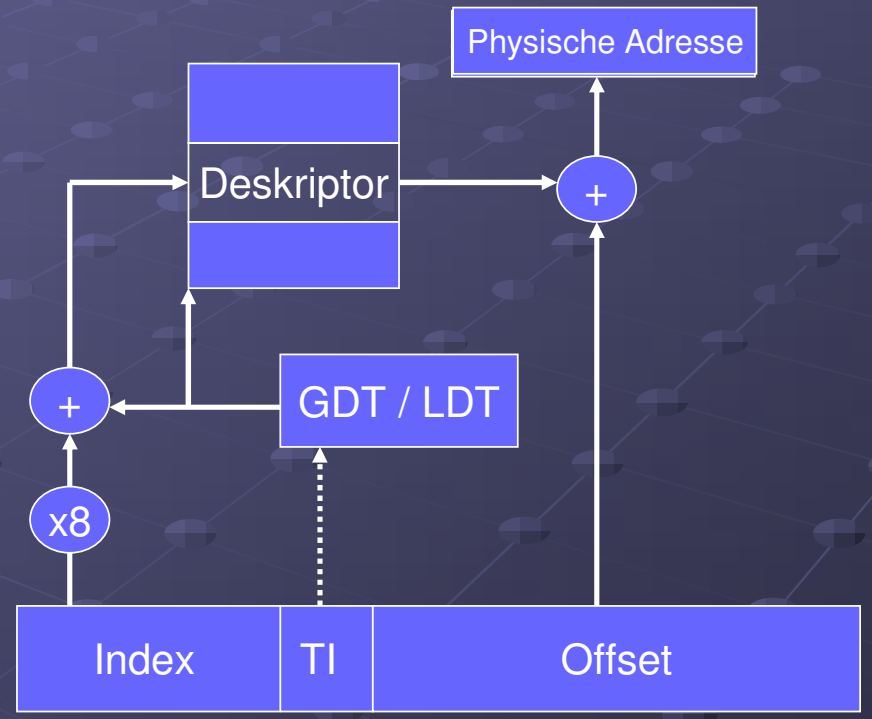
- Lineare Adressierung:
  - Eine 32-bit Integer, die die Adresse der Speicherzelle enthält (bis zu 4 GB Speicher ansprechbar)
  - Üblicherweise in hexadezimaler Schreibweise dargestellt
    - Enormer Verwaltungsaufwand
    - Unübersichtlich

# Speicheradressierung

- Segmentierung:
  - Daten werden über logische Adressen (Segmentselektoren) verwaltet
    - Werden in speziellen Registern verwaltet
    - Enthalten Adresse des Segmentdeskriptors
    - Enthalten Lage der Daten innerhalb des Segments (Offset)
  - Identifikation und Adressierung des Segments geschieht mittels eines Segmentdeskriptors
    - Werden in speziellen Tabellen verwaltet
    - Dienen auch der Kontrolle der Zugriffsrechte
  - Speicherzugriff wird von CPU-interner Segmentierungseinheit durchgeführt

# Speicheradressierung

- Segmentierungseinheit
  - TI-Feld des Segmentselektors gibt an in welcher Tabelle der Deskriptor liegt
  - mit der Basisadresse von GDT/LDT und Index von Segmentselektor wird Lage von Deskriptor innerhalb der Tabelle ermittelt
  - Adresse aus BASE-Feld des Deskriptors wird mit Offset des Selektors addiert um physische Adresse der Daten innerhalb des Segments zu ermitteln



# Speicheradressierung

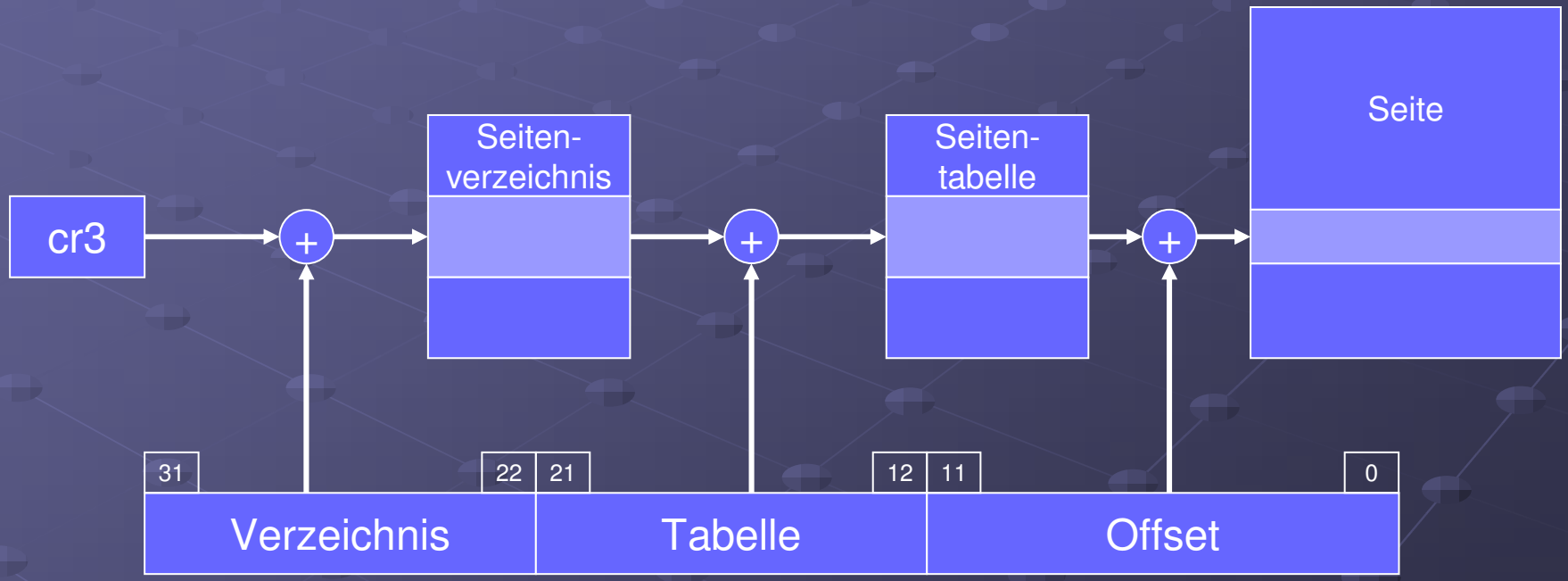
- Seitenadressierung (Paging):
  - Daten werden über logische Adressen verwaltet
  - Zugriff auf Seitenadressierungsstrukturen über Register
  - Einträge in Verzeichnissen und Tabellen sind identisch aufgebaut
  - Drei Arten von Seitenadressierung
    - Reguläre Seitenadressierung
    - Erweiterte Seitenadressierung
    - Drei-Stufen-Seitenadressierung

# Speicheradressierung

- Reguläre Seitenadressierung :
  - Seiten/Kacheln sind 4 KB groß
  - Logische Adresse ist in 3 Felder unterteilt:
    - Verzeichnis (10-Bit) verwaltet die Seitentabellen
    - Tabelle (10-Bit) verwaltet die Seiten
    - Offset (12-Bit) adressiert die Daten innerhalb der Seite
  - Seitenverzeichnis kann somit $1024 \times 1024 \times 4096 = 2^{32}$ Speicherzellen adressieren

# Speicheradressierung

- Reguläre Seitenadressierung
  - Die Seitenadressierungs-Einheit ermittelt physische Adresse der Daten innerhalb der Kachel

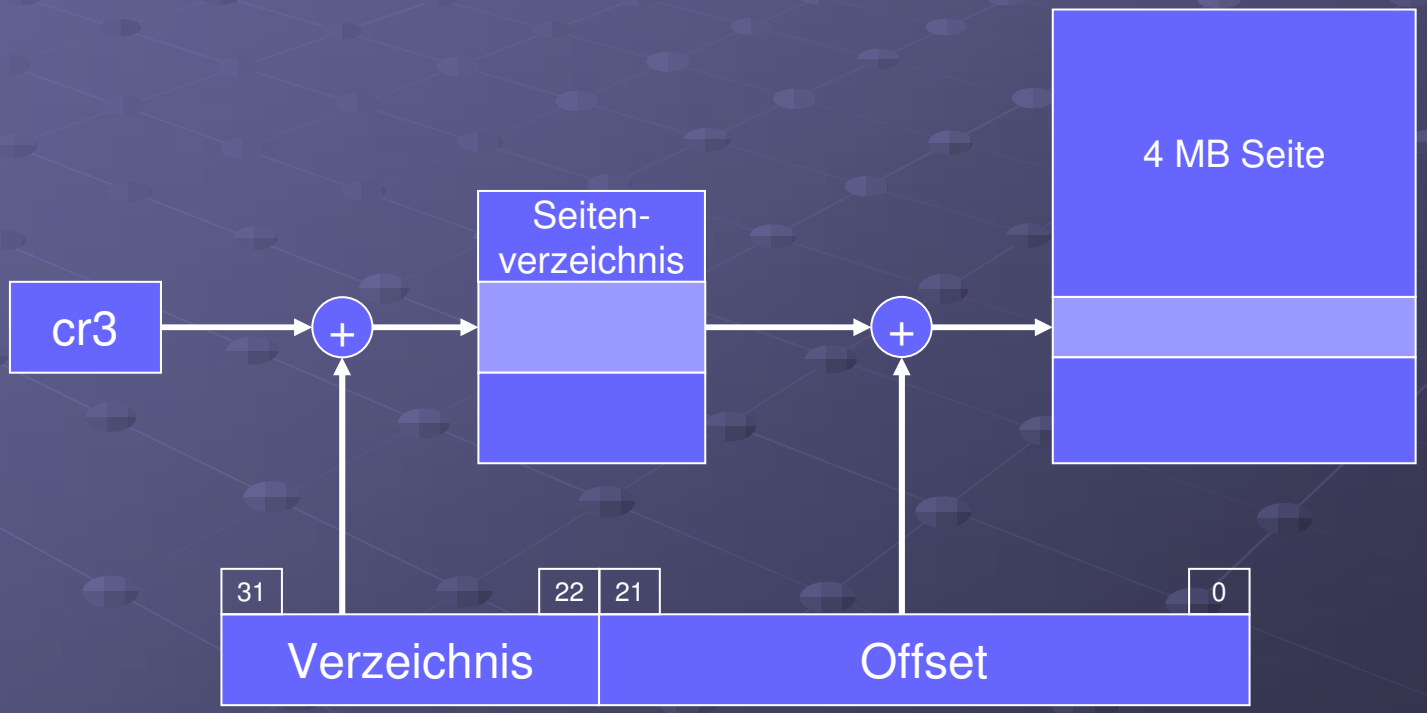


# Speicheradressierung

- Erweiterte Seitenadressierung :
  - Seiten/Kacheln sind 4 MB groß
    - Größere linear zusammenhängende Datenfelder können mit weniger Verwaltungsaufwand auch zusammenhängend auf Speicher gelegt werden
  - Logische Adresse ist in 2 Felder unterteilt:
    - Verzeichnis (10-Bit) verwaltet die Seiten
    - Offset (22-Bit) adressiert die Daten innerhalb der Seite
  - Seitenverzeichnis kann immer noch
    $1024 \times 2^{22} = 2^{32}$ Speicherzellen adressieren

# Speicheradressierung

- Erweitere Seitenadressierung



# Speicheradressierung

- Hardwareerweiterungen für effizientere Seitenadressierung
  - Probleme durch schnelle CPU und verhältnismäßig langsamerer Arbeitsspeicher
  - Mögliche Lösungen:
    - Hardware Cache
    - TLB (Translation Lookaside Buffer)

# Speicheradressierung

- Hardware Cache
  - Kleiner schneller Speicher zwischen CPU-Registern und Arbeitspeicherzellen
  - Zeilenorientierte Verwaltung von häufig genutzten Daten
  - Globale Aktivierung durch Register in der Seitenadressierungseinheit
  - Trotzdem individuelle Nutzung des Cache für jede Seite durch Deskriptorfunktionen (PCD/PWT-Flags) der Seitenverzeichnisse/-tabellen möglich

# Speicheradressierung

- TLB
  - Im Prinzip wie Cache
  - Speichert keine Daten sondern physische Adressen
    - Keine wiederholte Berechnung ein und derselben physischen Adresse nötig
  - Evtl. muss automatische Aktualisierung des TLB durch die Seitenadressierungseinheit durch das System unterbunden werden (z.B. bei Wechsel von Prozessen die auf die gleichen Tabellen zugreifen)

# Speicheradressierung in Linux

- Segmentierung in Linux
  - Segmentierung nur begrenzt angewendet
    - Überlegenheit von Seitenadressierung
    - Einschränkung der Segmentierung durch große Bandbreite an Architekturen die Linux bedient
  - Alle Prozesse nutzen die gleichen logischen Adressen
    - Gesamtanzahl an Segmenten ist begrenzt
    - Alle Segment-Deskriptoren können in der GDT gespeichert werden
    - LDTs können trotzdem von Prozessen erzeugt werden

# Speicheradressierung in Linux

- Segmentierung in Linux
  - Einträge des GDT in Linux:
    - Kernel Code Segment
    - Kernel Daten Segment
    - User Code Segment
    - User Daten Segment
    - Ein Prozess-Zustands-Segment (TSS) je Prozess
    - Ein Standard-LDT-Segment genutzt von allen Prozessen
    - 4 Segment-Deskriptoren für erweiterte Energieverwaltung
    - 4 ungenutzte Einträge des GDT

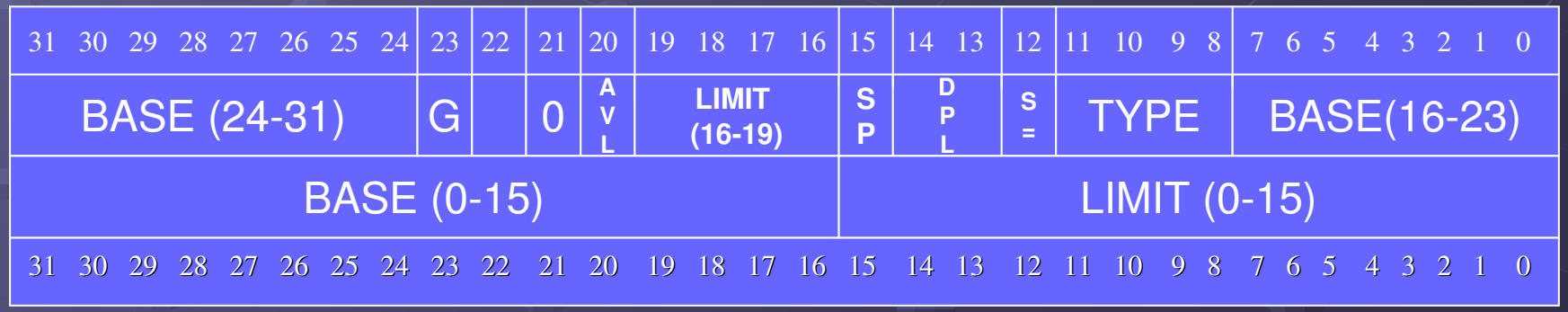
# Speicheradressierung in Linux

- Segmentierung in Linux
  - Wichtige Punkte im Deskriptor des Kernel-Code/Daten-Segments
    - G-Flag des Deskriptors auf 1 gesetzt, für Segmentgröße in Seiten
    - System-Flag auf 1 gesetzt, für Code-/Daten-Segment
    - DPL-Flag auf 0 gesetzt ,für Kernel Modus
    - D/B-Flag auf 1 gesetzt, für 32-Bit-Offset im Segmentselektor
    - Einziger Unterschied bei Type:
      - 0xa für Code-Segment
      - 2 für Daten-Segment

<table>
<tr><td>31 30 29 28 27 26 25 24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19 18 17 16</td><td>15</td><td>14 13</td><td>12</td><td>11 10 9 8</td><td>7 6 5 4 3 2 1 0</td></tr>
<tr><td colspan="2">BASE (24-31)</td><td>G</td><td></td><td>0</td><td>AVL</td><td colspan="2">LIMIT (16-19)</td><td>SP</td><td>DPL</td><td>S=</td><td>TYPE</td><td>BASE(16-23)</td></tr>
<tr><td colspan="6">BASE (0-15)</td><td colspan="6">LIMIT (0-15)</td></tr>
<tr><td colspan="11">31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td></tr>
</table>

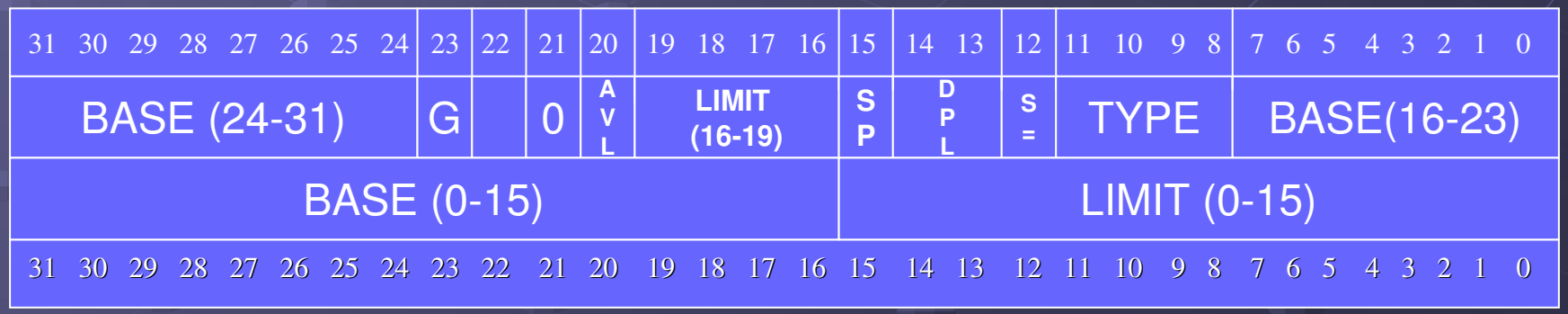
# Speicheradressierung in Linux

- Segmentierung in Linux
  - Wichtige Punkte im Deskriptor des User-Code/Daten-Segment
    - G-Flag des Deskriptors auf 1 gesetzt, für Segmentgröße in Seiten
    - System-Flag auf 1 gesetzt, für Code-/Daten-Segment
    - DPL-Flag auf 3 gesetzt, für User Modus
    - D/B-Flag auf 1 gesetzt, für 32-Bit-Offset im Segmentselektor
    - Einziger Unterschied bei Type:
      - 0xa für Code-Segment
      - 2 für Daten-Segment



# Speicheradressierung in Linux

- Segmentierung in Linux
  - Wichtige Punkte im Deskriptor des TSS
    - G-Flag des Deskriptors gelöscht, da Größe in Bytes
    - Das Limit-Feld wird auf 0xeb, da TSS 236 Bytes groß
    - DPL-Flag auf 0 gesetzt, da User keinen Zugriff auf TSS haben



# Speicheradressierung in Linux

- Segmentierung in Linux
  - Standard LDT-Segment
    - Beinhaltet lediglich Null-Segment-Deskriptor
    - Falls Prozess richtigen LDT benötigt, wird ein 4096-Byte-Segment erstellt
    - Standard-LDT-Segment-Deskriptor im GDT wird durch neuen spezifizierten Deskriptor ersetzt

# Speicheradressierung in Linux

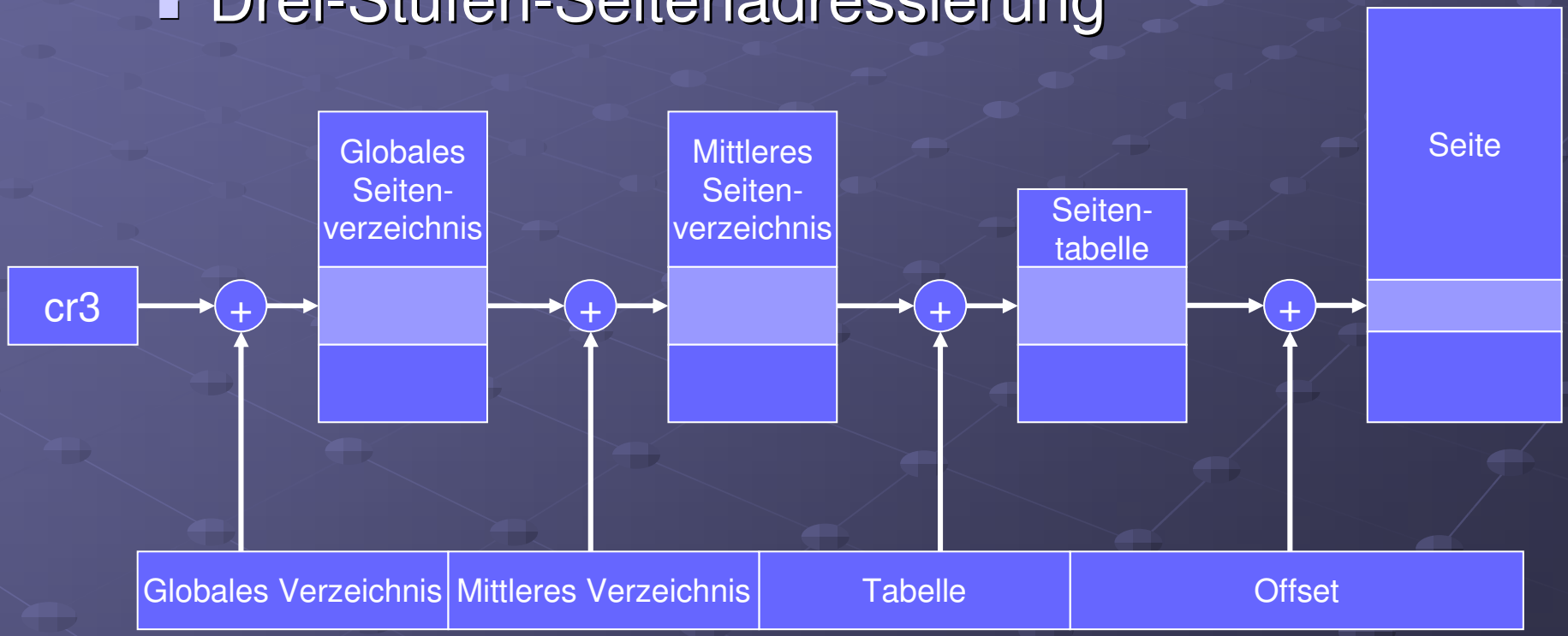
- Segmentierung in Linux
  - Weitere Einträge im GDT
    - GDT enthält für jeden existierenden Prozess je einen Segment-Deskriptor für das TSS-Segment und das LDT-Segment des jeweiligen Prozesses
    - GDT kann 4090 dieser Prozesseinträge verwalten
    - Für jeden Prozess existiert ein Prozess-Deskriptor im Kernel-Data-Segment, der sein eigenes TSS und einen Zeiger zu seinem, ebenfalls im KDS enthaltenen LDT-Segment

# Speicheradressierung in Linux

- Seitenadressierung in Linux
  - Seitenadressierung bevorzugt angewendet
    - Aufgrund der angestrebten Lauffähigkeit von Linux auf verschiedenen Architekturen
    - Nutzung von Drei-Stufen-Seitenadressierung auf 64-Bit-Basis
      - Noch kein Standard
      - Lediglich 43 Bits werden genutzt
        - 13-Bit-Feld Offset
        - Drei 10-Bit-Felder (Globales Seitenverzeichnis, Mittleres Seitenverzeichnis, Seitentabelle)
        - Mehr und größere Seiten adressierbar
    - Verwendung bei Multiprozessverwaltung

# Speicheradressierung in Linux

- Seitenadressierung in Linux
  - Drei-Stufen-Seitenadressierung



# Speicheradressierung in Linux

- Seitenadressierung in Linux
  - Multiprozessverwaltung
    - Jeder Prozess besitz eigenes Globales Seitenverzeichnis zur Verwaltung prozesseigener Daten und Codes
      - Trennung prozesseigener Adressräume, kein Mischen mit „fremden“ Daten
      - Einfache Ein- und Auslagerung aller wichtigen Elemente eines Prozesses, da in einem Adressblock enthalten
      - Einfache und schnelle Prozessumschaltung möglich ohne Umordnung von geteiltem Speicher

# Speicheradressierung in Linux

- Seitenadressierung in Linux
  - Multiprozessverwaltung
    - Sichere Verwaltung von verschiedenen Prozessen durch Aufteilung des Adressraumes des Arbeitsspeichers durch Prozess-Seitentabellen
      - User-Bereich
      - Kernel-Bereich, begrenzt auf 1 GB durch Linux
        - Implementierung von Physischer Adresserweiterung (PAE) in die Seitenadressierungseinheit (64 GB adressierbar durch Adresspinerweiterung von 32 auf 36 Pins)

# Speicheradressierung in Linux

- Seitenadressierung in Linux
  - Kernel Seitentabellen
    - Während Initialisierung des Systems erstellt
    - Analyse der CPU und Setzen entsprechender Startwerte, z.B.
      - Erweiterte Seitenadressierung aktiviert?
      - PAE aktiviert?
      - TLB oder Hardware Cache aktiviert?
    - Dienen als Referenz für weitere Verzeichnisse und Tabellen die von ihnen abgeleitet werden

# Speicheradressierung in Linux

- Seitenadressierung in Linux
  - Reservierte Seitenkacheln
    - Beginnen ab dem 2. MB des Arbeitsspeichers
    - Enthalten Kernel-Code und –Daten
      - Sind unverschachtelt für schnellen Zugriff
      - In der Regel bis zu 2 MB Gesamtgröße