FHV
Vorarlberg University
of Applied Sciences

# Dokumentation des OS

eine Arbeit von

**Nicolaj Höss, Marko Petrović, Kevin Wallis**

**Master Informatik (ITM2)**

für die Lehrveranstaltung

**S1: Softwarelösungen für ressourcenbeschränkte Systeme**

Fachhochschule Vorarlberg

8. Mai 2015, Dornbirn

## Abstract

# Inhaltsverzeichnis



# Abbildungsverzeichnis

# 1 Allgemein

bla

## 1.1 Aufbau

bla

# 2 Architektur

bla

## 2.1 Aufbau

bla

# 3 HAL

bla

## 3.1 Aufbau

bla

# 4 Treiber

bla

## 4.1 Aufbau

bla

# 5 Virtual Memory Management

Bei der virtuellen Speicherverwaltung erfolgt die Umwandlung von vom ARM Prozessor generierten, virtuellen Adressen in physikalische Adressen durch die *Memory Management Unit* (MMU). Dieses Kapitel enthält die Beschreibung des Designs und der Implementierung der virtuellen Speicherverwaltung des Betriebssystems sowie der Einstellungen der MMU.

## 5.1 Aufteilung des virtuellen Speichers und Mapping

Die *VSMAv7* definiert zwei unabhängige Formate für translation tables [1, S. B3-1318]:

- *Short-descriptor format*:
  - zweistufige Seitentabelle
  - 32-bit Deskriptoren (PTE)
  - 32-bit virtuelle Eingangsadresse
  - bis zu 40-bit große physikalische Ausgangsaddresse
- *Long-descriptor format*:
  - dreistufige Seitentabelle
  - 64-bit Deskriptoren (PTE)
  - verwendet *Large Physical Address Extension* (LPAE)
  - bis zu 40-bit große virtuelle Eingangsadresse
  - bis zu 40-bit große physikalische Ausgangsaddresse

Um die Anforderungen an das Betriebssystem zu erfüllen, reicht das zweistufige Seitentabellensystem vollkommen aus. Tabelle 1 fasst die wichtigsten gegebenen Eigenschaften unter Verwendung des Short-descriptor format zusammen.

| **Eigenschaft** | **Beschreibung** |
|---|---|
| Virtueller Speicher | 4 GB |
| Größe eines Page Table Entry (PTE) | 4 Byte |
| Einträge L1 Page Table | 4096 |
| Einträge L2 Page Table | 256 |
| Speicherbedarf L1 Page Table | 4 Byte * 4096 = 16kB |
| Speicherbedarf L2 Page Table | 4 Byte * 256 = 1kB |
| Unterstützte Pagegrößen: | *small page* (4 kB), *large page* (64 kB) |
| Unterstützte Sectiongrößen: | *section* (1 MB), *supersection* (16 MB) |

Tabelle 1: Eigenschaften der virtuellen Speicherverwaltung der ARMv7-Architektur

## Seitentabellen

Der verwendete ARM Prozessor verfügt über zwei Register (*Translation Table Base Register*, *TTBR0* und *TTBR1*), welche Startadressen von Seitentabellen enthalten können [1, S. B3-1320]. Diese Register übernehmen die folgende Funktion:

- TTBR0: Wird für prozessspezifische Adressen verwendet. Jeder Prozess enthält eine eigene L1-Seitentabelle. Bei einem Kontextwechsel erhält das TTBR0 eine Referenz auf L1-Seitentabelle des neuen Kontextes/Prozesses.
- TTBR1: Wird für das Betriebssystem selbst und für memory-mapped I/O verwendet. Diese ändern sich bei einem Kontextwechsel nicht.

Abbildung 1 zeigt die Speicherverwaltung des Betriebssystems. Die rechte Seite stellt das physikalische Speichermapping dar und wurde dem Datenblatt des ARM [2, S. 155] entnommen. Die linke Seite zeigt die Aufteilung des virtuellen Speichers.

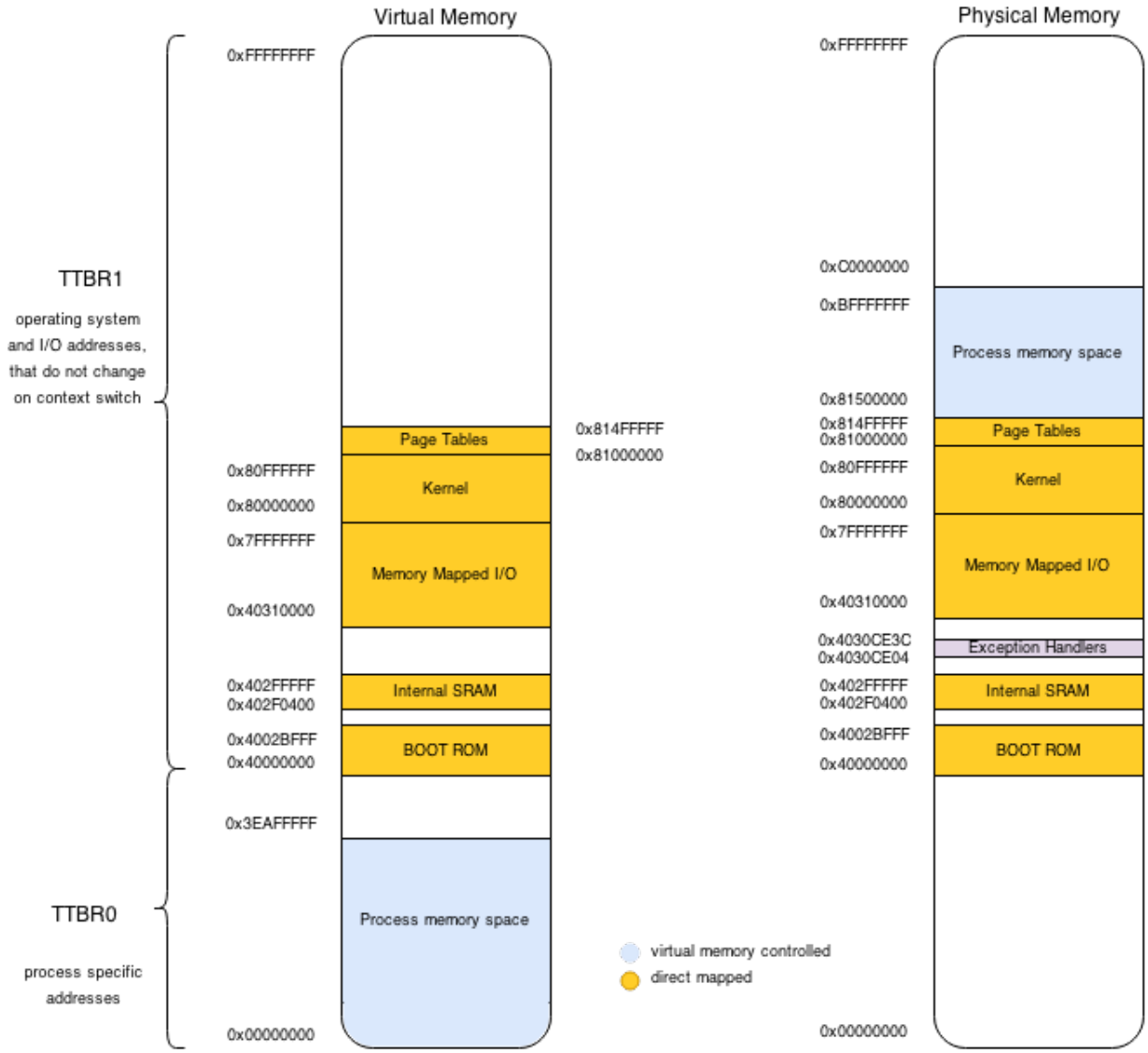


Abbildung 1: Memory Map des Betriebssystems

| Eigenschaft | Beschreibung |
|---|---|
| Größe der Pages | 4 kB |
| Speicherbedarf Kernel | 16 MB |
| Virtueller Speicher für Prozesse | 1003 MB |
| Physikalischer Speicher für Page Tables | 5 MB |
| Max. Anzahl von L1 und L2 Page Tables | 320 L1 Page Tables oder 1 L1 Page Table + 1276 L2 Page Table |
| Theoretisch Max. Anzahl von Prozessen | 320 |

Tabelle 2: Eigenschaften der virtuellen Speicherverwaltung des OS

```
1  typedef struct region
2  {
3      unsigned int startAddress;
4      unsigned int endAddress;
5      unsigned int pageSize;
6      unsigned int accessPermission;
7      unsigned int cacheBufferAttributes;
8      unsigned int reservedPages;
9      pageStatusPointer_t pageStatus;
10 } memoryRegion_t;
```

## 5.2 Umwandlung virtueller Adressen zu physikalische Adressen

Abbildung 2 zeigt die Umwandlung einer vom ARM Prozessor erzeugten virtuellen Adresse in eine physikalische Speicheradresse. Die Umwandlung wird vollständig durch die Prozessor-Hardware durchgeführt.

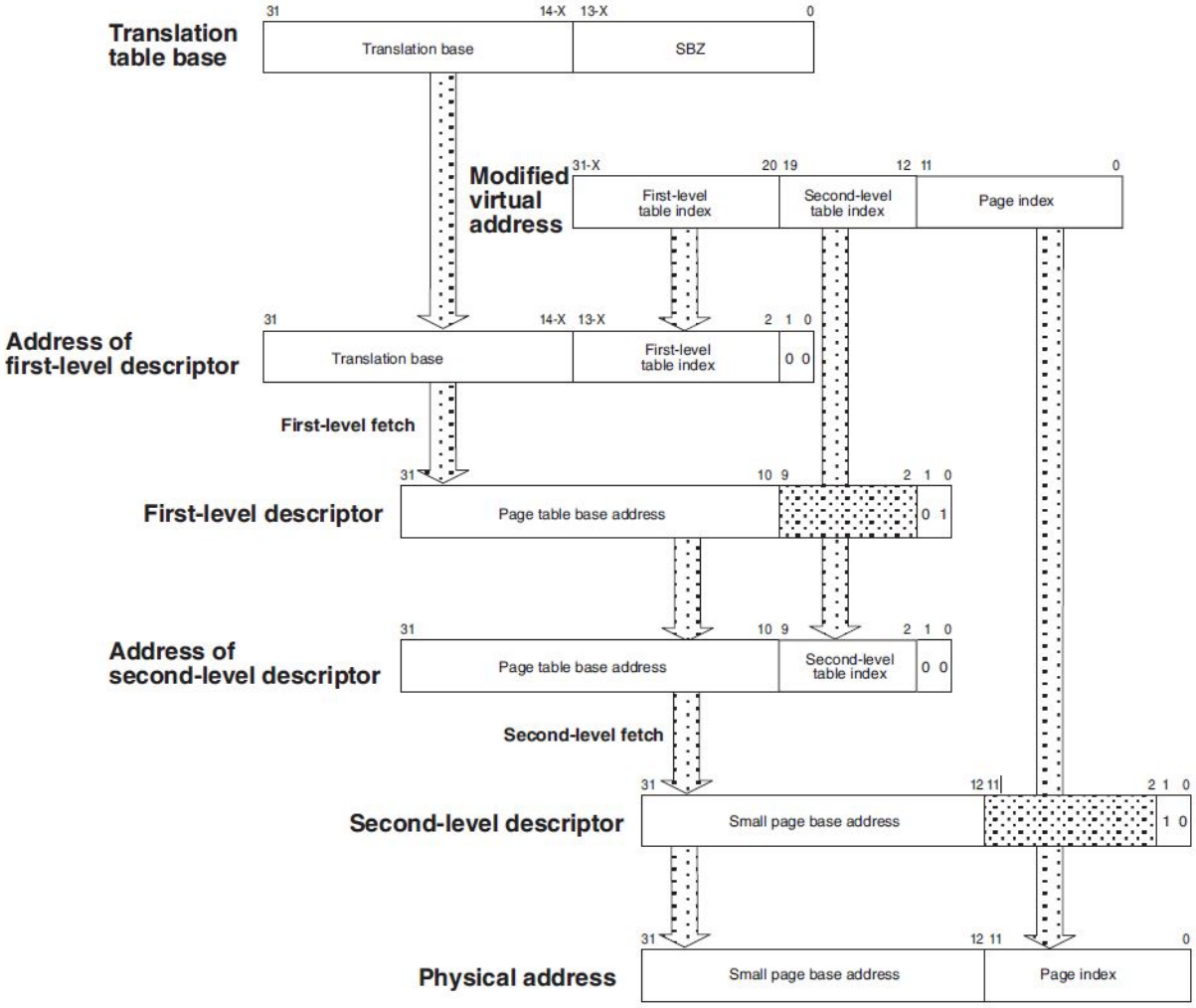


Abbildung 2: Umwandlung einer virtuellen Adresse durch die ARM CPU [1, S. B3-1337]

## 5.3 Allokierung der Page Frames

Für die Verwaltung der page frames wurde eine Bitsmap verwendet. Abbildung 3 zeigt das Prinzip. Die Bitsmap wird durch ein Array der Länge N/8 Bytes realisiert. N steht hier für die Anzahl der page frames. Das i-te Bit im n-ten Byte der Bitsmap definiert den Verwendungsstatus des (n*8 + i) –ten page frame.

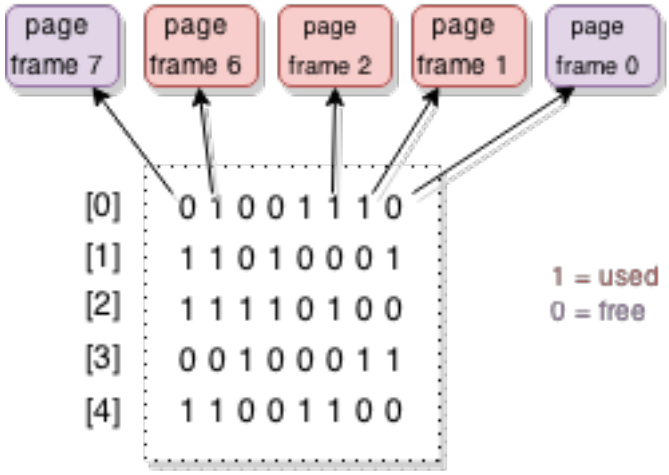


Abbildung 3: Beispiel einer Bitsmap zur Verwaltung der Page Frames

# 6 Zusammenfassung

bla
[2]
[1]

## 6.1 xxx

bla

# Literatur

[1] ARM Limited. *ARM Architecture Reference Manual ARMv7-A and ARMv7-R edition*, 2012. ARM DDI 0406C.b.

[2] Texas Instruments. *AM335x ARM Cortex-A8 Microprocessors (MPUs) Technical Reference Manual*, 2011. Revised April 2013.