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

11. 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 Grundlegende Funktionsweise

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

Generiert die ARM CPU einen Speicherzugriff, wird von der MMU ein Suchlauf durchgeführt. Dieser Suchlauf wird *translation table lookup* genannt. Dabei wird zuerst im Translation Lookaside Buffer (*TLB*) nachgesehen, ob einer der 64 Einträge des TLB die zur virtuellen Adresse korrespondierende physikalische Adresse enthält. Ist dies der Fall (so genannter *TLB hit*), wird der Suchlauf an dieser Stelle erfolgreich beendet.

Ist die angeforderte virtuelle Adresse nicht im TLB enthalten (TLB miss), wird ein page table walk durchgeführt. Das Funktionsprinzip des zweistufigen Seitentabellensystems zeigt Abbildung 1. Aus einem der zwei Seitentabellenregister wird die Basisadresse der darin zuvor abgelegten L1-Seitentabelle geholt. Das Format der *page table entries (PTE)* bestimmt dann, um welchen Typ von Verweis es sich handelt. Seitentabellen und ihre Einträge werden im nachfolgenden Abschnitt 5.3 genauer beschrieben.

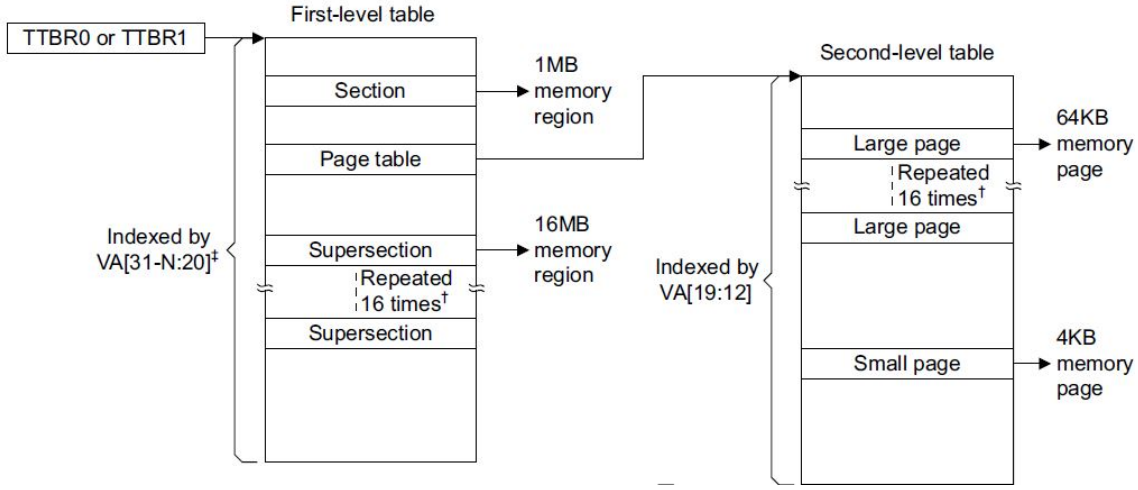

Abbildung 1: Zweistufiges Seitentabellensystem [1, S. B3-1325]

DATA ABORT HANDLER BESCHREIBEN

5.2 Umwandlung virtueller Adressen zu physikalische Adressen

Der genaue Vorgang der Umwandlung einer vom ARM Prozessor erzeugten virtuellen Adresse in eine physikalische Speicheradresse zeigen die nachfolgenden beiden Abbildungen. Abbildung 2 zeigt die Umwandlung einer virtuellen Adresse in die physikalische Adresse einer 1 MB Section ohne Verwendung einer L2-Seitentabelle, Abbildung 3 diejenige einer virtuellen Adresse in ein 4 kB page frame unter Verwendung einer L2-Seitentabelle. Die Umwandlung wird vollständig durch die Prozessor-Hardware durchgeführt.

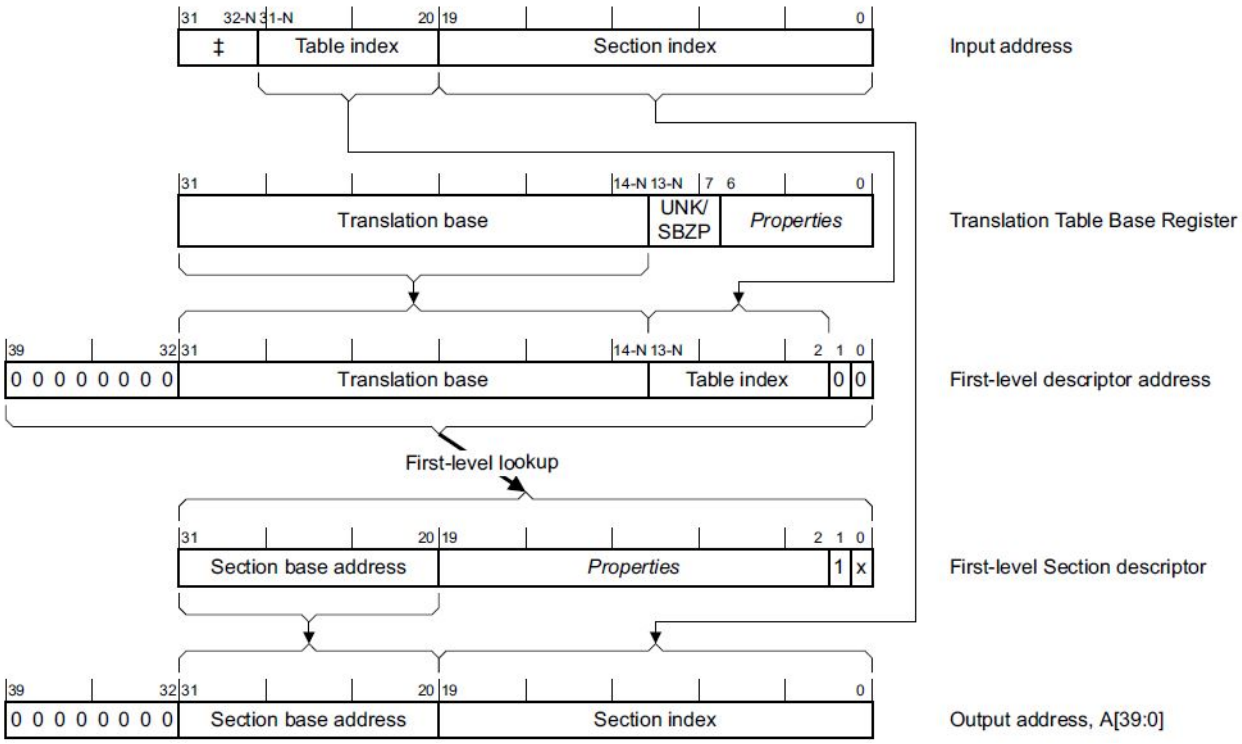

Abbildung 2: 1 MB Section Translation durch die ARM CPU [1, S. B3-1335]

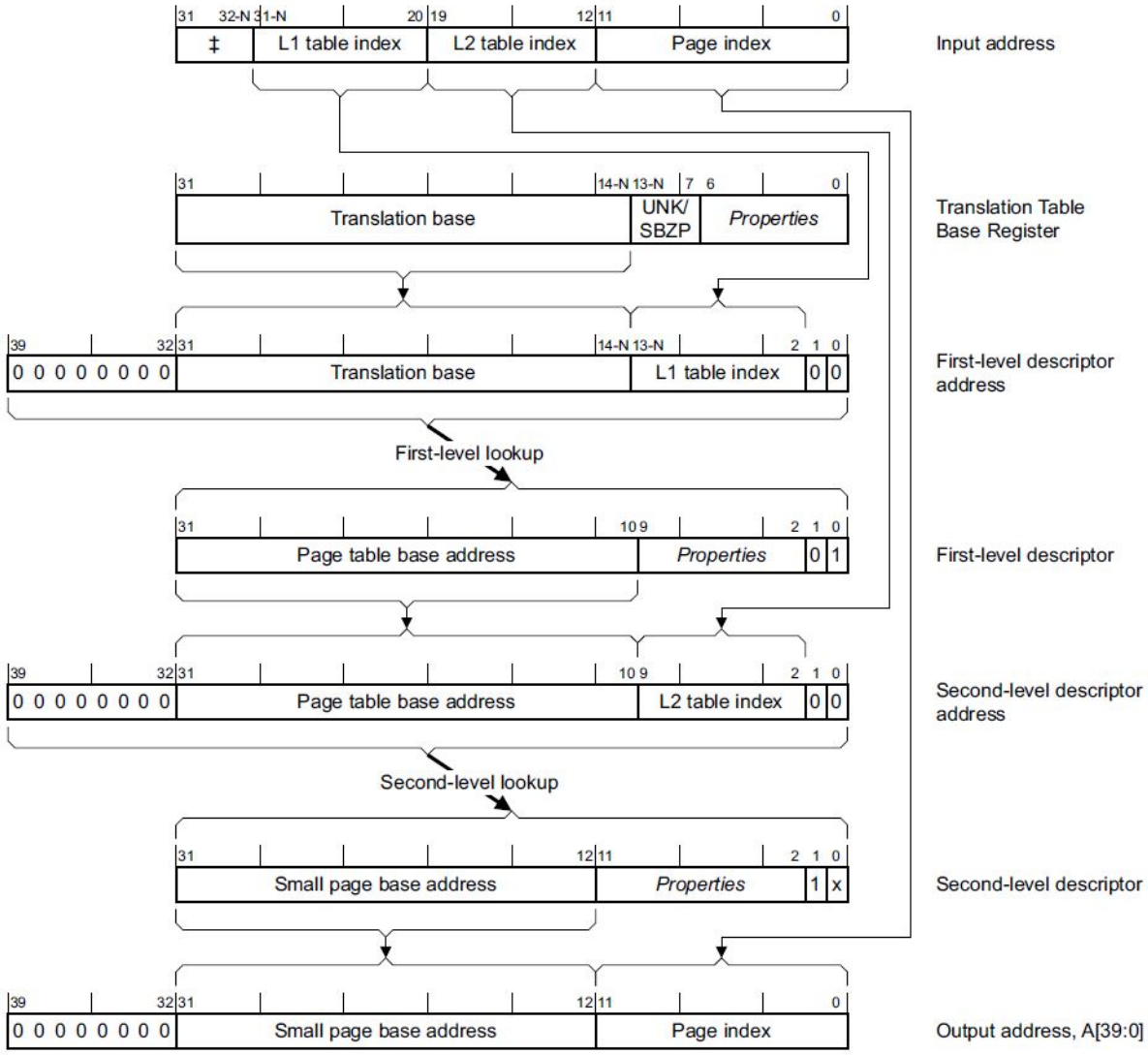

Abbildung 3: Small Page Translation durch die ARM CPU [1, S. B3-1337]

5.3 Seitentabellen und Seitentabelleneinträge

Der verwendete ARM Prozessor verfügt über zwei Register (*Translation Table Base Register*, *TTBR0* und *TTBR1*), welche Startadressen von Seitentabellen enthalten [1, S. B3-1320]. Ihre Formate sind nahezu identisch und in den Abbildungen 4 und 5 zu sehen. Diese Register übernehmen im Betriebssystem die folgende Funktion:

- TTBR0: Wird für prozessspezifische Adressen verwendet. Jeder Prozess enthält bei seiner Initialisierung 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.

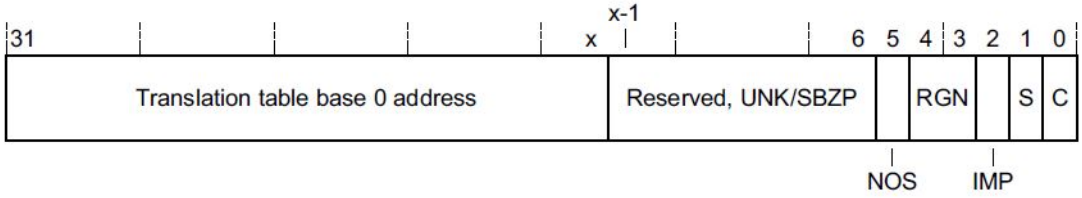

Abbildung 4: TTBR0 Format [1, S. B4-1726]

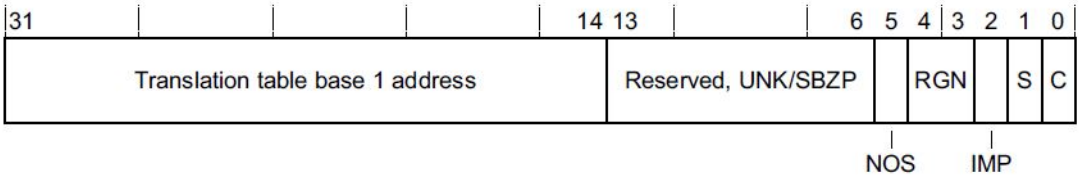

Abbildung 5: TTBR1 Format [1, S. B4-1730]

Das Beschreiben der Seitentabellenregister erfolgt, wie bei nahezu jeder MMU-Funktionalität, mittels Assemblerbefehlen, die auf die CP15 Coprozessor Register zugreifen.

Beim Füllen der Seitentabllen sind vorgegebene Formate für die beiden Typen von Deskriptoren unbedingt zu beachten. Die Abbildungen 6 und 7 fassen die Formate für first-level und second-level Deskriptoren zusammen. Beiden Deskriptortypen gleich ist der Fehlereintrag, der aus nullen besteht.

First-level Deskriptoren

Die First-Level Deskriptortypen werden auf folgende Weise verwendet:

- sections für die master page table (siehe Abschnitt 5.4)
- page table für L1-Seitentabellen von Prozessen (siehe Abschnitt 5.4)

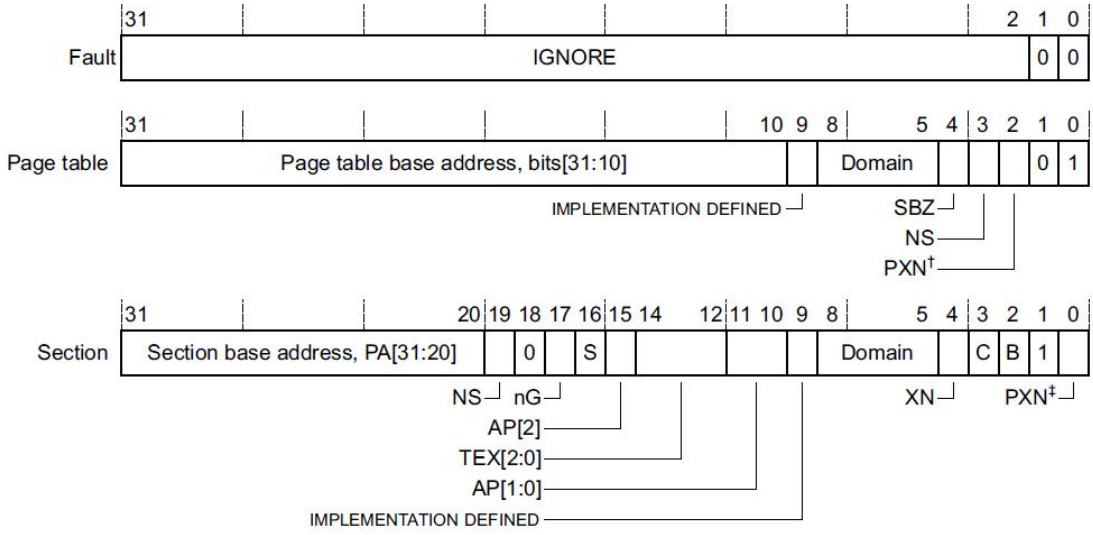

Abbildung 6: First-Level Deskriptorformate [1, S. B3-1326]

Second-level Deskriptoren

In der Speicherverwaltung des Betriebssystems werden ausschließlich small pages verwendet. Ausschlaggebende Gründe, warum small pages den Vorzug gegenüber large pages erhielten, sind die folgenden:

- small pages müssen nur einmal in die L2-Seitentabelle eingetragen werden, large pages hingegen 16 mal
- L1- und L2-Seitentabellen, die 16 kB bzw. 1 kB Speicher benötigen, belegen bei ihrer Erzeugung nur vier volle page frames bzw. ein page frame physikalischen Speichers zu einem Viertel. Dadurch wird die Speicherfragmentierung vergliechen mit large pages stark verringert

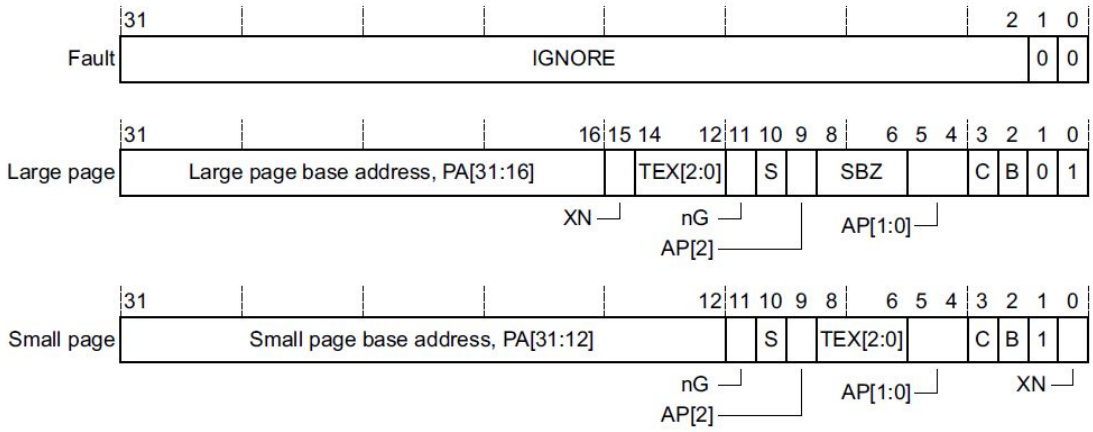

Abbildung 7: Secondt-Level Deskriptorformate [1, S. B3-1327]

5.4 Aufteilung des virtuellen Speichers und Mapping

Die Speicherverwaltung des Betriebssystems kann Abbildung 8 entnommen werden. 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.

Organisiert ist der virtuelle Speicher in Speicherregionen. Eine zusätzliche Aufteilung betrifft die Zuständigkeitsbereiche für die Seitentabellenregister TTBR0 und TTBR1. Im Seitentabellenkontrollregister TTBCR (Translation Table Base Control Register, [1, S. B4-1721]) können den Seitentabellenregistern Adressbereiche zugewiesen werden.

Physikalisch steht 1 GB Speicher für die page frames zur Verfügung.

Komponenten, die sich bei einem Kontextwechsel nicht ändern, werden statisch in master page table gemappt. Dazu zählen memory mapped I/O, Kernel des Betriebssystems, die Adressen der Exceptionhandler sowie die page table region, in welcher die L1- und L2-Seitentabellen abgelegt werden. Diese Komponenten befinden sich an den Adressen von 0x40000000 bis 0x81500000 und werden in die master page table direkt, d.h. Eins-zu-eins, gemapped. Ein Zugriff auf eine virtuelle Adresse ergibt damit einen Zugriff auf dieselbe Adresse in physikalischen Speicher.

Der Adressbereich von 0x00000000 bis 0x40000000 entspricht genau 1 GB. In ihn wird der virtuelle Prozessbereich an diese Adressen gemapped und somit 0x40000000 als Selektionsgrenze zwischen TTBR0 (für Prozesse) und TTBR1 (für Kernelfunktionen) festgelegt.

Die master page table wird bei der Initialisierung der MMU im TTBR1 abgelegt. Danach wird der Inhalt des TTBR1 während der Laufzeit des Betriebssystems nie wieder aktualisiert bzw. geändert.

Bei der Initialisierung eines Prozesses wird für den Prozess eine L1-Seitentabelle, die mit Fehlereinträgen initialisiert ist, angelegt. Soll ein Prozess zur Ausführung gebracht werden, muss seine L1-Seitentabelle in das TTBR0 geschrieben werden. Das TTBR0 muss zur Laufzeit

des Betriebssystems bei Kontextwechseln von Prozessen aktualisiert werden.

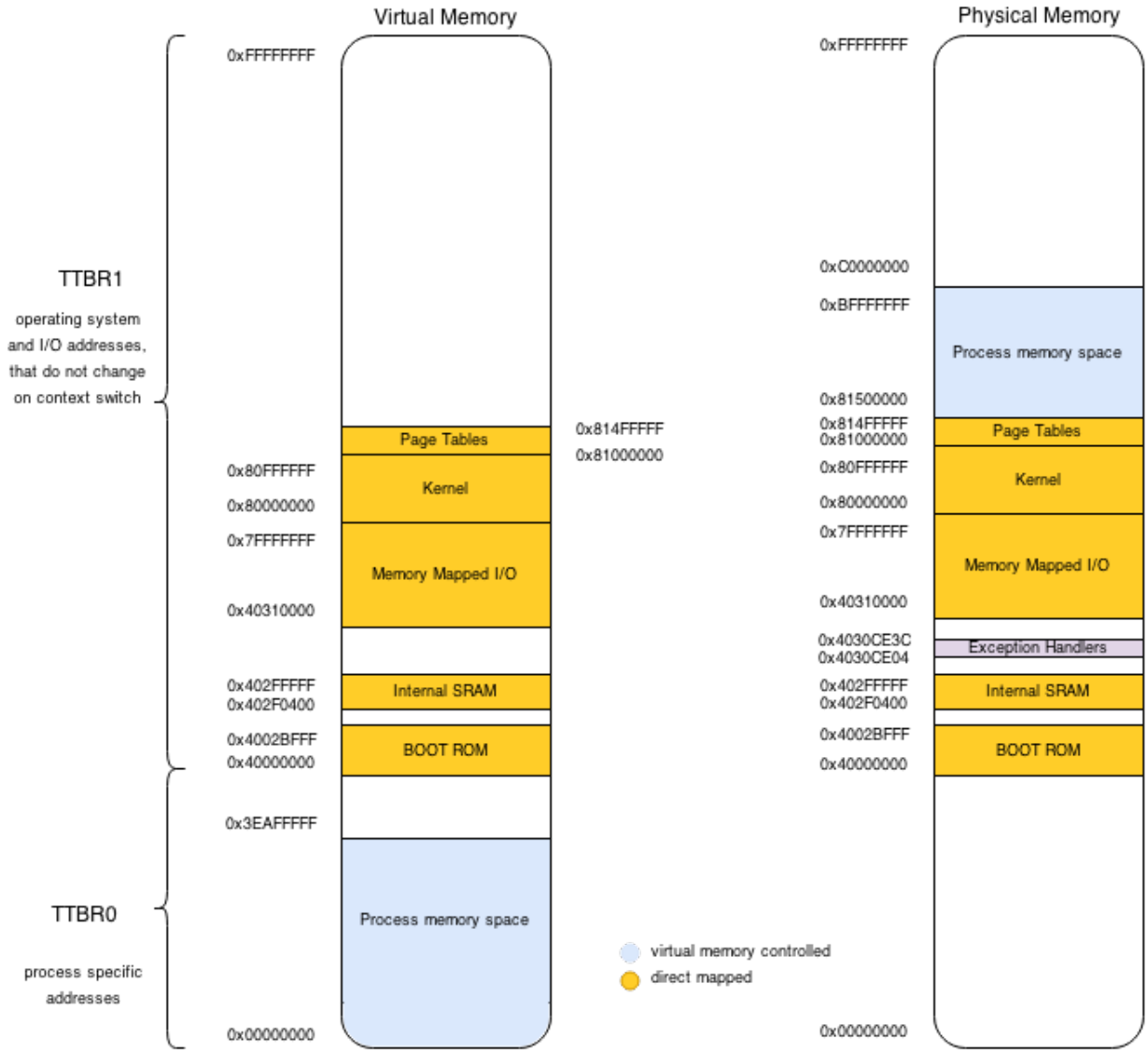

Abbildung 8: Memory Map des Betriebssystems

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;
```

5.5 Allokierung der Page Frames

Für die Verwaltung der page frames wurde eine Bitmap verwendet. Abbildung 9 zeigt das Prinzip. Die Bitmap 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 Bitmap definiert den Verwendungsstatus des (n*8 + i) –ten page frame.

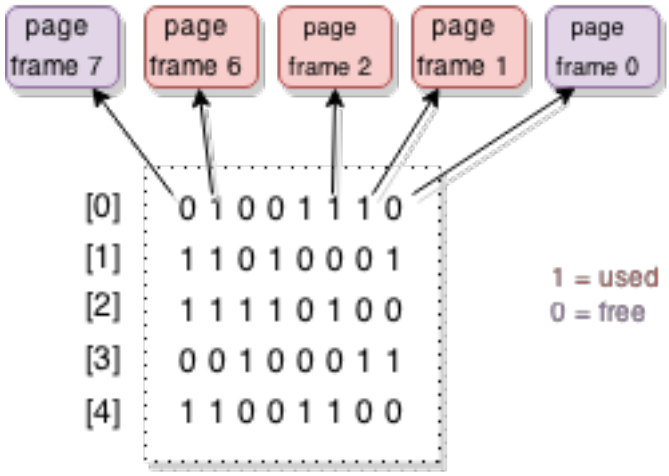

Abbildung 9: Beispiel einer Bitmap zur Verwaltung der Page Frames

5.6 Initialisierung der MMU

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.