南 京 工 程 学 院

毕业设计说明书(论文)

作 者：  **朱宇**  学 号： **202150340**

院 系： **计算机工程学院**

专 业： **计算机科学与技术**

题 目： **基于Cortex-A9多核处理器的**

**操作系统内核设计和实现**

指导者： **施 展** **讲 师**

评阅者：

2019年5月 南 京

**毕 业 论 文 中 文 摘 要**

|  |
| --- |
| **题目：** 基于Cortex-A9多核处理器的  操作系统内核设计和实现  摘要：  在科技日益发展的今天，新兴的技术层出不穷，我国的计算机行业也发展迅猛，然而，不可否认的是在基础软件领域任然落后于发达国家。特别像操作系统领域，还缺少一个认可度较高，通用性强，稳定的操作系统内核软件项目。随着新型处理器架构和物联网的兴起，势必需要一个新的操作系统内核，因此内核在未来具有深刻的研究价值。本课题基于Cortex-A9多核处理器，实现了一个支持多任务和文件系统的操作系统内核。内核的具体功能模块包括内存管理、进程管理（添加，删除，切换）、中断管理、设备驱动、文件系统和系统调用等。  本文以Linux内核作为主要参考对象，来编写一个操作系统雏形，并运行在s5p4418开发板上。内核的性能并没有专门优化，单纯展示内核是如何设计和实现。同时内核尽可能遵循POSIX标准，实现通用的编程接口。  关键词：Cortex-A9　ARM　操作系统　内核　POSIX |

**毕 业 论 文 外 文 摘 要**

|  |
| --- |
| **Title** Design and Implementation of Operating System Kernel Based on Cortex-A9 Multi-core Processor  **Abstract:**  Today, with the development of science and technology, emerging technologies are emerging one after another. China's computer industry is also developing rapidly. However, it is undeniable that it still lags behind developed countries in the field of basic software. Especially like the operating system field, there is still a lack of a highly recognized, versatile, and stable operating system kernel software project. With the rise of new processor architectures and the Internet of Things, a new operating system kernel is bound to be required, so the kernel has profound research value in the future. Based on the Cortex-A9 multi-core processor, this project implements an operating system kernel that supports multitasking and file systems. Specific functional modules of the kernel include memory management, process management (add, delete, switch), interrupt management, device drivers, file systems, and system calls.  This article uses the Linux kernel as the main reference object to write a prototype of the operating system and runs on the s5p4418 development board. The performance of the kernel is not specifically optimized, simply showing how the kernel is designed and implemented. At the same time, the kernel follows the POSIX standard as much as possible to implement a common programming interface.  **Keywords:** Cortex-A9 ARM operating system kernel POSIX |

目　　录

[前言 1](#_Toc9083379)

[第一章 绪论 2](#_Toc9083380)

[1.1 研究背景和意义 2](#_Toc9083381)

[1.2 设计目标 2](#_Toc9083382)

[第二章 硬件与软件 3](#_Toc9083383)

[2.1 硬件平台 3](#_Toc9083384)

[2.2 软件框架 3](#_Toc9083385)

[第三章 开发环境搭建 4](#_Toc9083386)

[3.1 启动镜像制作 4](#_Toc9083387)

[3.2 USB下载驱动 4](#_Toc9083388)

[3.3 启动代码 4](#_Toc9083389)

[第四章 功能模块设计 5](#_Toc9083390)

[4.1 内存管理 5](#_Toc9083391)

[4.2 中断管理 5](#_Toc9083392)

[4.3 设备管理 5](#_Toc9083393)

[4.4 文件系统 5](#_Toc9083394)

[4.5 系统调用 5](#_Toc9083395)

[4.6 任务管理 5](#_Toc9083396)

[第五章 接口和性能测试 6](#_Toc9083397)

[5.1 接口测试 6](#_Toc9083398)

[5.2 性能测试 6](#_Toc9083399)

[参考文献 7](#_Toc9083400)

[附录：英文技术资料翻译 8](#_Toc9083401)

# 前言

# 第一章 绪论

## 1.1 研究背景和意义

## 1.2 设计目标

# 第二章 硬件与软件

## 2.1 硬件平台

## 2.2 软件框架

# 第三章 开发环境搭建

## 3.1 启动镜像制作

## 3.2 USB下载驱动

## 3.3 启动代码

## 

# 第四章 功能模块设计

## 4.1 内存管理

## 4.2 中断管理

## 4.3 设备管理

## 4.4 文件系统

## 4.5 系统调用

## 4.6 任务管理

# 第五章 接口和性能测试

## 5.1 接口测试

## 5.2 性能测试

# 参考文献

[1] 田宇．一个64位操作系统的设计与实现[M]．北京：人民邮电出版社，2018 [02] 左飞，万晋森，刘航．Visual C++数字图像处理开发入门与编程实践．电子工业出版社，2008

[2] (荷)安德鲁 S. 塔嫩鲍姆(Andrew S. Tanenbaum)，(荷)赫伯特·博斯(Herbert Bos)．现代操作系统[M]．北京：机械工业出版社，2017

[3] (美)Abraham Silberschatz．操作系统概念[M]．北京：高等教育出版社，2004

[4] 彭东．深度探索嵌入式操作系统[M]．北京：机械工业出版社，2015

[5] (美) William Stallings．操作系统：精髓与设计原理[M]．北京：电子工业出版社，2006

[6] Maurice J.Bach．UNIX操作系统设计[M]．北京：机械工业出版社，2000

[7] (美)塔嫩鲍姆．操作系统设计与实现[M]．北京：电子工业出版社，2007

[8] 于渊．Orange'S一个操作系统的实现[M]．北京：电子工业出版社，2006

[9] 郑钢．操作系统真象还原[M]，北京：人民邮电出版社，2016

[10] 李无言．一步步写嵌入式操作系统[M]．北京：电子工业出版社，2011

# 附录：英文技术资料翻译

**英文原文：**

**Chapter 6 Memory Management Unit**

**6.1 About the MMU**

The MMU works with the L1 and L2 memory system to translate virtual addresses to physical addresses. It also controls accesses to and from external memory.

The Virtual Memory System Architecture version 7 (VMSAv7) features include the following:

• Page table entries that support 4KB, 64KB, 1MB, and 16MB.

• 16 domains.

• Global and address space identifiers to remove the requirement for context switch TLB flushes.

• Extended permissions check capability.

See the ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition for a full architectural description of the VMSAv7.

The processor implements the ARMv7-A MMU enhanced with Security Extensions and multiprocessor extensions to provide address translation and access permission checks. The MMU controls table walk hardware that accesses translation tables in main memory. The MMU enables fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes.

Note:

In VMSAv7 first level descriptor formats page table base address bit [9] is implementation-defined. In Cortex-A9 processor designs this bit is unused.

The MMU features include the following:

• Instruction side micro TLB.

— Hardware configurable 32 or 64 fully associative entries.

• Data side micro TLB.

— 32 fully associative entries.

• Unified main TLB.

— 2-way associative:

2x32 entry TLB for the 64-entry TLB.

2x64 entry TLB for the 128-entry TLB.

2x128 entry TLB for the 256-entry TLB.

2x256 entry TLB for the 512-entry TLB.

— 4 lockable entries using the lock-by-entry model.

— Supports hardware page table walks to perform lookups in the L1 data cache.

This section contains the following subsections:

• 6.1.1 Memory Management Unit on page 6-110.

**6.1.1 Memory Management Unit**

The MMU checks the Virtual Address and ASID, domain access permissions, and memory attributes.

The MMU also performs the following operations:

• Virtual-to-physical address translation.

• Support for four page (region) sizes.

• Mapping of accesses to cache, or external memory.

• TLB loading for hardware and software.

Domains

The Cortex-A9 processor supports 16 access domains.

TLB

The Cortex-A9 processor implements a 2-level TLB structure. Four entries in the main TLB are lockable.

ASIDs

Main TLB entries can be global, or can be associated with particular processes or applications using Address Space Identifiers (ASIDs). ASIDs enable TLB entries to remain resident during context switches, avoiding the requirement of reloading them subsequently.

Related concepts

Invalidate TLB Entries on ASID Match on page 4-100.

System control coprocessor

TLB maintenance and configuration operations are controlled through a dedicated coprocessor, CP15, integrated within the processor. This coprocessor provides a standard mechanism for configuring the level one memory system.

**6.2 TLB Organization**

The TLB is organized as a micro TLB and a main TLB.

This section contains the following subsections:

• 6.2.1 Micro TLB on page 6-112.

• 6.2.2 Main TLB on page 6-112.

**6.2.1 Micro TLB**

The first level of caching for the page table information is a micro TLB of 32 entries on the data side, and configurable 32 or 64 entries on the instruction side. These blocks provide a fully associative lookup of the virtual addresses in a single CLK signal cycle.

The micro TLB returns the physical address to the cache for the address comparison, and also checks the protection attributes to signal either a Prefetch Abort or a Data Abort.

All main TLB related operations affect both the instruction and data micro TLBs, causing them to be flushed. In the same way, any change of the Context ID Register causes the micro TLBs to be flushed.

**6.2.2 Main TLB**

The main TLB catches the misses from the micro TLBs. It also provides a centralized source for lockable translation entries.

Accesses to the main TLB take a variable number of cycles, according to competing requests from each of the micro TLBs and other implementation-dependent factors. Entries in the lockable region of the main TLB are lockable at the granularity of a single entry. As long as the lockable region does not contain any locked entries, it can be allocated with non-locked entries to increase overall main TLB storage size.

The main TLB is implemented as a combination of:

• A fully-associative, lockable array of four elements.

• A 2-way associative structure of 2x32, 2x64,2x128 or 2x256 entries.

TLB match process

Each TLB entry contains a virtual address, a page size, a physical address, and a set of memory properties. Each is marked as being associated with a particular application space, or as global for all application spaces. CONTEXIDR determines the selected application space.

A TLB entry matches if bits [31:N] of the modified virtual address match, where N is log2 of the page size for the TLB entry. It is either marked as global, or the ASID matched the current ASID.

A TLB entry matches when these conditions are true:

• Its virtual address matches that of the requested address.

• Its Non-secure TLB ID (NSTID) matches the Secure or Non-secure state of the MMU request.

• Its ASID matches the current ASID or is global.

The operating system must ensure that, at most, one TLB entry matches at any time.

Supersections, sections, and large pages are supported to permit mapping of a large region of memory while using only a single entry in a TLB. If no mapping for an address is found in the TLB, then the translation table is automatically read by hardware and a mapping is placed in the TLB.

TLB lockdown

The TLB supports the TLB lock-by-entry model as described in the ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition.

Related references

4.3.26 TLB lockdown operations on page 4-97.

**6.3 Memory access sequence**

When the processor generates a memory access, the MMU performs a lookup for the requested virtual address and current ASID and security state in the relevant instruction or data micro TLB.

If there is a miss in the micro TLB, the MMU performs a lookup for the requested virtual address and current ASID and security state in the main TLB. If there is a miss in the main TLB, the MMU performs a hardware translation table walk.

You can configure the MMU to perform hardware translation table walks in cacheable regions by setting the IRGN bits in the Translation Table Base Registers. If the encoding of the IRGN bits is write-back, then an L1 data cache lookup is performed and data is read from the data cache. If the encoding of the IRGN bits is write-through or non-cacheable then an access to external memory is performed.

The MMU might not find a global mapping, or a mapping for the selected ASID, with a matching Non-secure TLB ID (NSTID) for the virtual address in the TLB. In this case, the hardware does a translation table walk if the translation table walk is enabled by the PD0 or PD1 bit in the TTB Control Register. If translation table walks are disabled, the processor returns a Section Translation fault.

If the MMU finds a matching TLB entry, it uses the information in the entry as follows:

1. The access permission bits and the domain determine if the access is enabled. If the matching entry does not pass the permission checks, the MMU signals a memory abort. See the ARM® Architecture

Reference Manual, ARMv7-A and ARMv7-R edition for a description of access permission bits, abort types and priorities, and for a description of the IFSR and Data Fault Status Register (DFSR).

2. The memory region attributes specified in both the TLB entry and the CP15 c10 remap registers control the cache and write buffer, and determine if the access is

• Secure or Non-secure.

• Shared or not.

• Normal memory, Device, or Strongly-ordered.

3. The MMU translates the virtual address to a physical address for the memory access.

If the MMU does not find a matching entry, a hardware table walk occurs.

**6.4 MMU enabling or disabling**

You can enable or disable the MMU as described in the ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition.

**6.5 External aborts**

External memory errors are defined as those that occur in the memory system rather than those that are detected by the MMU. External memory errors are expected to be extremely rare. External aborts are caused by errors flagged by the AXI interfaces when the request goes external to the processor. External aborts can be configured to trap to Monitor mode by setting the EA bit in the Secure Configuration Register.

This section contains the following subsections:

• 6.5.1 External aborts on data read or write on page 6-116.

• 6.5.2 Synchronous and asynchronous aborts on page 6-116.

**6.5.1 External aborts on data read or write**

Externally generated errors during a data read or write can be asynchronous. This means that the r14\_abt on entry into the abort handler on such an abort might not hold the address of the instruction that caused the exception.

The DFAR is UNPREDICTABLE when an asynchronous abort occurs.

In the case of a load multiple or store multiple operation, the address captured in the DFAR is that of the address that generated the synchronous external abort.

**6.5.2 Synchronous and asynchronous aborts**

To determine a fault type, read the DFSR for a data abort or the IFSR for an instruction abort.

The processor supports an Auxiliary Fault Status Register for software compatibility reasons only. The processor does not modify this register because of any generated abort.

**中文译文：**

**第六章内存管理单元**

**6.1关于MMU**

MMU与L1和L2内存系统一起工作，将虚拟地址转换为物理地址。它还控制对外部内存的访问和来自外部内存的访问。

虚拟内存系统体系结构版本7 (VMSAv7)的特性包括:

• 支持4KB、64KB、1MB和16MB的页表条目。

• 16域。

• 全局和地址空间标识符消除了上下文切换TLB刷新的要求。

• 扩展权限检查功能。

查看ARM®体系结构参考手册,ARMv7-A和ARMv7-R版以获取VMSAv7h的完整架构描述。

处理器通过安全扩展和多处理器扩展来实现ARMv7-A MMU，以提供地址转换和访问权限检查。MMU控制访问主存中的转换表的表遍历硬件。MMU通过一组虚拟地址到物理地址的映射和内存属性，进行细粒度的内存系统控制。

注意:

在VMSAv7一级描述符格式中，页表基本地址位[9]有具体定义的。在Cortex-A9处理器设计中，这个位是却未使用的。

MMU的特点包括:

• 指令微TLB。

- 硬件可配置32或64个全相联条目。

• 数据微TLB。

- 32个全相联条目。

• 统一主TLB。

- 双路相联:

2x32 entry TLB 对于 64-entry TLB.

2x64 entry TLB 对于 128-entry TLB.

2x128 entry TLB 对于 256-entry TLB.

2x256 entry TLB 对于 512-entry TLB.

- 4个使用逐项锁定模型的可锁定条目。

-支持硬件页表遍历来进行L1数据缓存查找。

本节包括以下各节:

• 6.1.1内存管理单元，见第6-110页。

**6.1.1内存管理单元**

MMU检查虚拟地址和ASID、域访问权限和内存属性。

MMU还执行以下操作:

• 虚拟地址到物理地址的转换。

• 支持四页(区域)大小。

• 缓存或外部内存访问的映射。

• 硬件和软件的TLB加载。

域

Cortex-A9处理器支持16个访问域。

TLB

Cortex-A9处理器实现了一个2级的TLB结构。主TLB中的4个条目是可锁定的。

ASIDs

主TLB条目可以是全局的，也可以与使用地址空间标识符(ASIDs)的特定进程或应用程序相关联。ASIDs允许TLB条目在上下文切换期间保持驻留，从而避免随后重新加载它们的需求。

相关的概念

使ASID匹配的TLB条目失效，见第4-100页。

系统控制协处理器

TLB的维护和配置操作通过集成在处理器中的专用协处理器CP15进行控制。这个协处理器提供了配置一级内存系统的标准机制。

**6.2 TLB组织**

TLB组织为微TLB和主TLB。

本节包括以下各节:

• 6.2.1 微 TLB，见第6-112页。

• 6.2.2 主TLB，见第6-112页。

**6.2.1微TLB**

页表信息的第一级缓存是32个条目的数据微TLB，32或64个指令条目。这些块在单个CLK信号周期中提供虚拟地址的全相联查找。

微TLB将物理地址返回缓存以进行地址比较，并检查保护属性以发出预取中止或数据中止的信号。

所有与TLB相关的主要操作都会影响指令和数据微TLBs，导致它们被刷新。同样，上下文ID寄存器的任何更改都会导致微TLBs被刷新。

**6.2.2主TLB**

主TLB从微TLBs中捕捉缺失。它还为可锁定的转换条目提供了一个集中的源。

对主TLB的访问周期是不固定的，具体有来自每个微TLBs和其他依赖实现的因素的竞争性请求。，主TLB的可锁定区域的中的每个条目是可在单个条目的粒度上锁定的。只要可锁定区域不包含任何已锁定项，就可以使用非锁定项分配该区域，以增加主TLB总体存储大小。

主要的TLB实现为:

• 四个元素组成的全相联可锁定数组。

• 2x32、2x64、2x128或2x256双路相联结构。

TLB匹配过程

每个TLB条目包含一个虚拟地址、一个页面大小、一个物理地址和一组内存属性。每个都标记为与特定的应用程序空间相关联，或者作为所有应用程序全局空间。CONTEXIDR决定所选的应用程序空间。

如果修改后的虚拟地址匹配位[31:N]，则TLB条目匹配，其中N为TLB条目页面大小的log2。它要么标记为全局的，要么与当前ASID匹配。

当这些条件为真时，TLB条目匹配:

• 它的虚拟地址与请求地址匹配。

• 其非安全TLB ID (NSTID)匹配MMU请求的安全或非安全状态。

• 它的ASID匹配当前ASID或是全局的。

操作系统必须确保在任何时候最多匹配一个TLB条目。

可以使用单个条目在TLB中映射大内存区域比如超节、节和大页。如果在TLB中没有找到地址的映射，那么硬件将自动读取转换表，并在TLB中设置映射。

TLB封锁

TLB支持TLB lock-by-entry模型，具体描述在ARMv7-A 和ARMv7-R版的ARM®体系结构参考手册。

相关的参考文献

4.3.26 TLB锁定操作，见第4-97页。

**6.3内存访问顺序**

当处理器生成内存访问时，MMU在相关指令或数据微TLB中查找请求的虚拟地址、当前ASID和安全状态。

如果在微TLB中出现遗漏，MMU将对请求的虚拟地址、当前ASID和主TLB中的安全状态执行查找操作。如果在主TLB中出现缺失，MMU将执行硬件转换表遍历。

你可以通过设置转换表基地址寄存器的IRGN位，将MMU配置为在可缓存区域中执行硬件转换表遍历。如果IRGN位的编码是写回的，则执行L1数据缓存查找并从数据缓存中读取数据。如果IRGN位的编码是写通的或不可缓存的，则执行对外部内存的访问。

MMU可能无法在TLB中找到一个全局映射或特定ASID的映射的虚拟地址，这个映射匹配非安全TLB ID (NSTID)。在这种情况下，如果转换表遍历由TTB控制寄存器中的PD0或PD1位启用，硬件将执行转换表遍历。如果转换表的遍历被禁用，处理器将返回一个段转换错误。

如果MMU找到一个匹配的TLB条目，它将使用如下条目信息:

1.访问权限位和域决定是否启用访问。如果匹配的条目没有通过权限检查，MMU将发出内存中止的信号。查看ARMv7-A和ARMv7-R版本的ARM®体系结构参考手册中关于访问权限位、中止类型和优先级的描述，以及IFSR和数据故障状态寄存器(DFSR)的描述。

2.在TLB条目和CP15 c10重映射寄存器中指定的内存区域属性控制缓存和写缓冲区，并确定访问是否是

• 安全或不安全。

• 是否共享。

• 正常内存、设备或强顺序。

3.MMU将虚拟地址转换为物理地址以便进行内存访问。

如果MMU没有找到匹配的条目，就会出现硬件表遍历。

**6.4 MMU启用或禁用**

您可以启用或禁用MMU，具体描述在ARMv7-A ARMv7-R版的ARM®架构参考手册中描。

**6.5外部中止**

外部内存错误定义为发生在内存系统中的错误，而不是MMU检测到的错误。外部内存错误是非常罕见的。外部中止是由请求到处理器外部时AXI接口标记的错误引起的。通过在安全配置寄存器中设置EA位，可以将外部中止配置为陷阱至监视模式。

本节包括以下各节:

· 6.5.1数据读写的外部中止，见第6-116页。

· 6.5.2同步和异步中止，见第6-116页。

**6.5.1数据读写的外部中止**

数据读写过程中外部生成的错误可以是异步的。这意味着在进入中止处理程序时r14\_abt可能不包含导致异常的指令的地址。

当发生异步中止时，DFAR是不可预测的。

在加载多个或存储多个操作的情况下，DFAR中捕获的地址是产生同步外部中止的地址。

**6.5.2同步和异步中止**

若要确定故障类型，请读取数据中止的DFSR或指令中止的IFSR。

由于软件兼容性的原因，处理器支持辅助故障状态寄存器。处理器不会因为任何生成的中止而修改这个寄存器。