**第一章**

**1． 嵌入式系统含义**

嵌入式系统是以应用为中心、以计算机技术为基础、软件硬件可裁剪、适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。

嵌入式系统是将先进的计算机技术、半导体技术和电子技术与各个行业的具体应用相结合后的产物。包含有计算机，但又不是通用计算机的计算机应用系统。

第二章 处理器设计导论

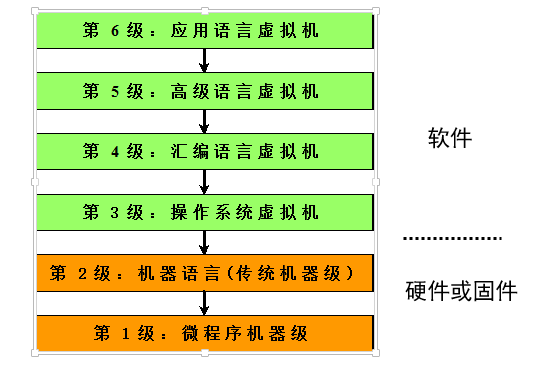
1. 计算机系统中的层次概念

计算机系统＝软件＋硬件/固件

计算机语言由低级向高级发展--高一级语言的语句相对于低级语言功能更强，更便于应用，但又都以低级语言为基础。

从计算机语言的角度，把计算机系统按功能划分成多级层次结构。

2. 计算机体系结构



经典计算机体系结构概念的实质：计算机系统中软硬件界面的确定，其界面之上的是软件的功能，界面之下的是硬件和固件的功能。

3. 组织-体系结构实现方式：

体系结构描述从用户角度看到的计算机，即概念性结构与功能特性。按照计算机系统的多级层次结构，不同级程序员所看到的计算机具有不同的属性。如：指令集结构(Instruction Set Architecture)计算机体系结构的核心，硬件与低层软件之间的接口层次

计算机组织描述从用户角度不能看到的体系结构的实现方式:流水线结构（Pipeline）高速缓存（Cache）步行表硬件（table-walking）转换后备缓冲（TLB）

4. MU0的扩展：扩展地址空间；增加更多的寻址方式；能够保存PC,支持子程序调用；增加更多的寄存器，支持中断等

5.RISC与CISC区别：

RISC体系结构：固定的指令长度（32位），指令类型很少；Load-Store结构，数据处理只访问寄存器，与访问存储器的指令分开；大量的寄存器，不少于32个。可以用于任何用途，极大简化了处理器的设计



第三章 ARM体系结构

1. 继承与抛弃

ARM采用的技术特征: Load/Store体系结构，固定的32位指令，3地址指令格式

ARM未采用的技术特征：寄存器窗口，延迟转移，所有指令单周期执行

2.V4-V7:增加不断改善的Thumb指令集以提高代码密度并减小功耗，平衡代码空间和性能，扩展DSP指令集，提高DSP能力；支持Java加速功能，降低功耗；内存系统，异常处理改善，较好地支持多处理器；提供浮点支持，满足下一代3D图形，游戏以及传统嵌入式控制的需要

ARM7-ARM10如何改善性能：



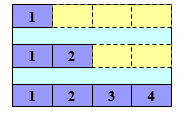
Cortex系列改善性能：



第四章 ARM编程模型

1. 体系结构直接支持的数据类型

ARM处理器支持下列数据类型：字节8位，半字16位（必须分配为占用两个字节），字 32位（必须分配为占用4个字节）



2. ARM微处理器的工作状态一般有两种:

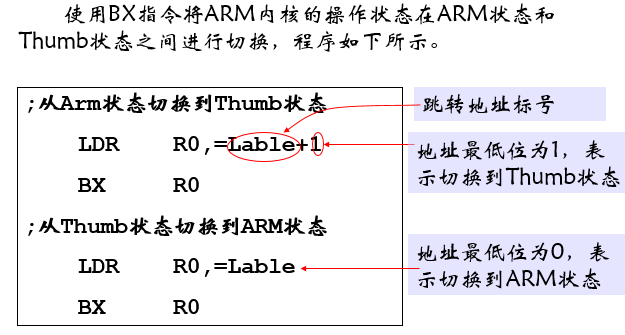
ARM状态－处理器执行32位的字对齐的 ARM指令；

Thumb状态－ 处理器执行16位的、半字对齐的Thumb指令。

状态切换：

进入Thumb状态：当操作数寄存器的状态位(位[0])为1时，执行BX指令进入Thumb状态。如果处理器在Thumb状态进入异常，则当异常处理返回时，自动转换到Thumb状态。

进入ARM状态：当操作数寄存器的状态位(位[0])为0时执行BX指令进入ARM状态。当处理器进行异常处理时，进入ARM状态从异常向量地址处开始执行。



3.存储器及存储器映射I/O

a. ARM处理器将存储器看作是一个从0开始的线性递增的字节集合：字节0到3保存第1个存储的字，字节4到7保存第2个存储的字，字节8到11保存第3个存储的字，依此类推

b. 地址空间：ARM结构使用单个平面的232个8位字节地址空间。字节地址按照无符号数排列，从0到2的32次方－1。如果地址向上或向下溢出地址空间，通常会发生翻转。注意：如果在取指操作时地址发生溢出，只要没有执行预取的无效指令，就不会导致异常。

c. 跳转目标的计算方法：(当前指令的地址) + 8 + 偏移量

下一条指令位置的计算方法：(当前指令的地址) + 4

d. 地址空间的规则：

位于地址A的字包含的字节位于地址A,A+1,A+2和A+3；

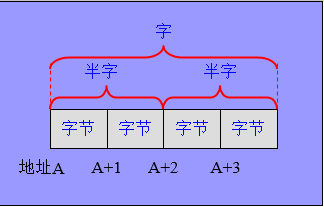
位于地址A的半字包含的字节位于地址A和A+1；

位于地址A+2的半字包含的字节位于地址A+2和A+3；

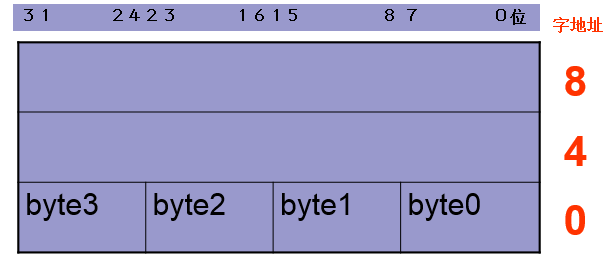
位于地址A的字包含的半字位于地址A和A+2；

存储器系统有两种映射机制：

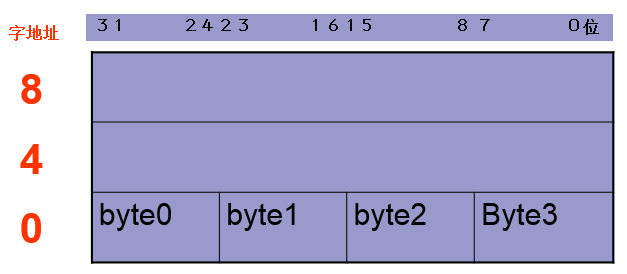
小端存储器系统：在小端格式中，高位数字存放在高位字节中。因此存储器系统字节0连接到数据线7～0。 大端存储器系统：在大端格式中，高位数字存放在低位字节中。因此存储器系统字节0连接到数据线31～24。



小端模式：

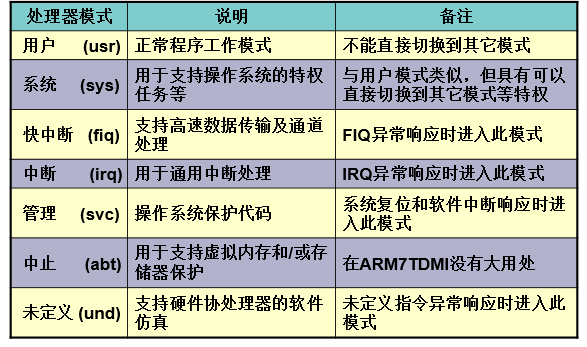


大端模式：



4. 处理器模式：

a. ARM体系结构支持7种处理器模式，分别为：用户模式、快中断模式、中断模式、管理模式、中止模式、未定义模式和系统模式。这样的好处是可以更好的支持操作系统并提高工作效率。



b. 特权模式：除用户模式外，其它模式均为特权模式。ARM内部寄存器和一些片内外设在硬件设计上只允许（或者可选为只允许）特权模式下访问。此外，特权模式可以自由的切换处理器模式，而用户模式不能直接切换到别的模式。

c. 异常模式：后五种模式称为异常模式。它们除了可以通过程序切换进入外，也可以由特定的异常进入。当特定的异常出现时，处理器进入相应的模式。每种异常模式都有一些独立的寄存器，以避免异常退出时用户模式的状态不可靠。

d. 用户和系统模式：这两种模式都不能由异常进入，而且它们使用完全相同的寄存器组。系统模式是特权模式，不受用户模式的限制。操作系统在该模式下访问用户模式的寄存器就比较方便，而且操作系统的一些特权任务可以使用这个模式访问一些受控的资源。

5.寄存器

ARM处理器有37(31+6)个物理寄存器,有18个可编程访问的寄存器。寄存器被安排成部分重叠的组。在不同的工作模式和处理器状态下，程序员可以访问的寄存器也不尽相同。分组的寄存器在处理处理器异常和特权操作时可得到快速的上下文切换。

在汇编语言中寄存器R0～R13为保存数据或地址值的通用寄存器。它们是完全通用的寄存器，不会被体系结构作为特殊用途，并且可用于任何使用通用寄存器的指令。其中R0～R7为未分组的寄存器，也就是说对于任何处理器模式，这些寄存器都对应于相同的32位物理寄存器。寄存器R8～R14为分组寄存器。它们所对应的物理寄存器取决于当前的处理器模式，几乎所有允许使用通用寄存器的指令都允许使用分组寄存器。寄存器R8～R12有两个分组的物理寄存器。**一个用于除FIQ模式之外的所有寄存器模式，另一个用于FIQ模式。这样在发生FIQ中断后，可以加速FIQ的处理速度。**寄存器R13、R14分别有6个分组的物理寄存器。一个用于用户和系统模式，其余5个分别用于5种异常模式。寄存器R13常作为堆栈指针（SP）。在ARM指令集当中，没有以特殊方式使用R13的指令或其它功能，只是习惯上都这样使用。但是在Thumb指令集中存在使用R13的指令。 R14为链接寄存器（LR），在结构上有两个特殊功能：在每种模式下，模式自身的R14版本用于保存子程序返回地址；

当发生异常时，将R14对应的异常模式版本设置为异常返回地址（有些异常有一个小的固定偏移量）。

R14: .

1.执行用户模式下的程序；2.发生IRQ中断，硬件将某个地址存入IRQ模式下的R14\_irq寄存器，用户模式下的R14没有被破坏；3. IRQ服务程序A执行完毕，将R14\_irq寄存器的内容减去某个常量后存入PC，返回之前被中断的程序；4. 如果在IRQ处理程序中打开IRQ中断，并且再次发生IRQ中断，或者调用子程序；5. 硬件将返回地址保存在R14\_irq寄存器中，原来保存的返回地址将被覆盖，造成错误；6. 在程序B返回到程序A，然后在返回到用户模式下被中断的程序时，发生错误，将不能正确返回；

解决办法是确保R14的对应版本在发生中断嵌套时不再保存任何有意义的值（将R14入栈），或者切换到其它处理器模式下。

寄存器R15为程序计数器（PC），它指向正在取指的地址。可以认为它是一个通用寄存器，但是对于它的使用有许多与指令相关的限制或特殊情况。如果R15使用的方式超出了这些限制，那么结果将是不可预测的。

6. 程序状态寄存器—CPSR(1)+SPSR(5)

CPSR反映了当前处理器的状态：４个条件代码标志；2个中断控制位；5个对当前处理器模式进行编码的位；1个指示当前执行指令的工作状态位；保留位。

SPSR：备份程序状态字，保存异常事件发生之前的CPSR．

每个异常模式带有一个备份程序状态寄存器，用于保存在异常事件发生之前的CPSR；CPSR和SPSR通过特殊指令进行访问。

7.异常：

只要正常的程序流被暂时中止，处理器就进入异常模式。例如响应一个来自外设的中断。在处理异常之前，ARM内核保存当前的处理器状态，这样当处理程序结束时可以恢复执行原来的程序。如果同时发生两个或更多异常，那么将按照固定的顺序来处理异常，详见“异常优先级”部分。



异常的入口和出口管理：如果异常处理程序已经把返回地址拷贝到堆栈，那么可以使用一条多寄存器传送指令来恢复用户寄存器并实现返回。

进入异常：

a. 在适当的LR中保存下一条指令的地址，当异常入口来自：

ARM状态，那么ARM内核将当前指令地址加4或加8 （取决于异常的类型）复制到LR中；

Thumb状态，那么ARM内核将当前指令地址加2或加4 （取决于异常的类型）复制到LR中；异常处理程序不必确定状态。

b. 将CPSR复制到适当的SPSR中；

c. 将CPSR模式位强制设置为与异常类型相对应的值；

d. 强制PC从相关的异常向量处取指。

注意：异常总是在ARM状态中进行处理。当处理器处于Thumb状态时发生了异常，在异常向量地址装入PC时，会自动切换到ARM状态。

退出异常：

1.将LR中的值减去偏移量后存入PC，偏移量根据异常的类型而有所不同；

2.将SPSR的值复制回CPSR；

3.清零在入口置位的中断禁止标志。

注：恢复CPSR的动作会将T、F和I位自动恢复为异常发生前的值。

8.快速中断请求

快速中断请求(FIQ)适用于对一个突发事件的快速响应，这得益于在ARM状态中，快中断模式有8个专用的寄存器可用来满足寄存器保护的需要（这可以加速上下文切换的速度）。

不管异常入口是来自ARM状态还是Thumb状态，FIQ处理程序都会通过执行下面的指令从中断返回：SUBS PC,R14\_fiq,#4

在一个特权模式中，可以通过置位CPSR中的F位来禁止FIQ异常。

9.中断请求：

中断请求（IRQ）异常是一个由nIRQ输入端的低电平所产生的正常中断（在具体的芯片中，nIRQ由片内外设拉低，nIRQ是内核的一个信号，对用户不可见）。IRQ的优先级低于FIQ。对于FIQ序列它是被屏蔽的。任何时候在一个特权模式下，都可通过置位CPSR中的I 位来禁止IRQ。

不管异常入口是来自ARM状态还是Thumb状态，FIQ处理程序都会通过执行下面的指令从中断返回：SUBS PC,R14\_irq,#4

10. 中止

中止发生在对存储器的访问不能完成时，中止包含两种类型：

预取中止 发生在指令预取过程中

当发生预取中止时，ARM内核将预取的指令标记为无效，但在指令到达流水线的执行阶段时才进入异常。如果指令在流水线中因为发生分支而没有被执行，中止将不会发生。

在处理中止的原因之后，不管处于哪种处理器操作状态，处理程序都会执行下面的指令恢复PC和CPSR并重试被中止的指令：SUBS PC,R14\_abt,#4

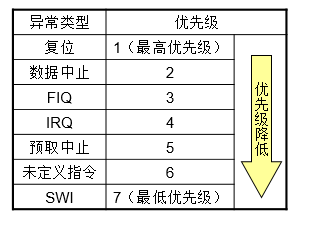
数据中止 发生在对数据访问时

11.软件中断指令：

使用软件中断(SWI)指令可以进入管理模式，通常用于请求一个特定的管理函数。SWI处理程序通过执行下面的指令返回：MOVS PC,R14\_svc

这个动作恢复了PC和CPSR并返回到SWI之后的指令。SWI处理程序读取操作码以提取SWI函数编号。

12.异常优先级：



第五章 汇编语言编程

第六章 ARM流水线组织

1. ARM7系列采用3级流水线结构，流水线级分别为：

取指：从存储器取指，放入指令流水线；

译码：指令被译码，并为下一个周期准备数据通路的控制信号；

执行：占有数据通路，寄存器堆被读取，操作数被移位，ALU产生结果并回写到目的寄存器。

2. ARM9系列采用5级流水线结构，流水线级分别为：

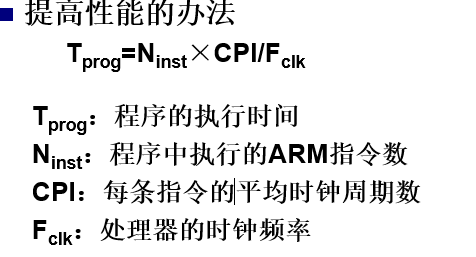
取指：从存储器取指，放入指令流水线；

译码：指令被译码，从寄存器堆中读取操作数

执行：把一个操作数移位，产生ALU结果

缓冲/数据：访问数据存储器或者缓冲1个周期

回写：将结果回写到寄存器堆



提高性能的办法

减小执行指令的数量（通过编译器优化）

提高时钟频率Fclk--简化流水线每一级的逻辑，增加级数

减小CPI--重新考虑3级流水线多于1个流水线槽的指令实现方法，使它占有较小的槽；减小因指令相关造成的流水线停顿

3. 减小CPI根本问题－存储器瓶颈

冯.诺伊曼结构：指令和数据放在同一个存储器，性能受到存储器带宽限制

指令在每个周期都要访问存储器

改进的方法：

在一个时钟周期给出多于32位数据

指令、数据分别设置存储器（哈佛结构）

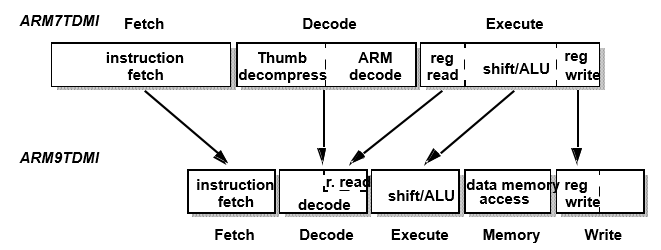
减小CPI

解决数据相关而不使流水线停顿的惟一方法：增加前推通路

前推通路使结果产生后可以立即在级间传送

5级流水线要求3个源操作数都能从任何3个中间结果寄存器中前推

4. 三级与五级流水线比较：



第七章 ARM指令集

寻址方式是根据指令中给出的地址码字段来实现寻找真实操作数地址的方式。ARM处理器具有9种基本寻址方式。

(1).寄存器寻址； (2).立即寻址；(3).寄存器偏移寻址； (4).寄存器间接寻址；

(5).基址偏移寻址； (6).多寄存器寻址；(7).堆栈寻址； (8).块拷贝寻址；

(9).相对寻址。

第十章ARM存储器接口和AMBA总线

1. 总线周期类型与特点

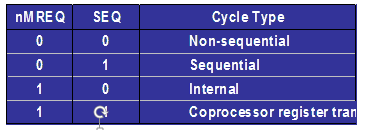
非连续 (N):在接下来的周期中的地址与前一个地址无关

连续 (S):在接下来的周期中的地址与前一个地址一样或大一个操作数（字或半字）

内部 (I):处理器正在执行一个内部操作，同时，没有有用的预取执行

协处理器寄存器传送 (C):处理器和协处理器之间通讯，不涉及存储器访问，但 D[31:0] 用于传送数据

合并的内部连续 (IS):I和S周期的特殊组合，容许优化存储器访问



2.AMBA总线

AMBA（Advanced Microcontroller Bus Architecture）是ARM 公司研发的一种总线规范，最新为4.0 版本，在AMBA总线规范中，定义了3种总线：

AHB(Advanced High-performance Bus)：用于高性能系统模块的连接，支持突发模式数据传输和事务分割；可以有效地连接处理器、片上和片外存储器，支持流水线操作

ASB（Advanced System Bus）：也用于高性能系统模块的连接，由AHB总线替代

APB（Advanced Peripheral Bus）：用于较低性能外设的简单连接，一般是接在AHB或ASB系统总线上的第二级总线。

3.总线测试：

第十三章 ARM JTAG测试与调试结构

1. 含义

JTAG－Joint Test Action Group

联合测试行动组开发了用于印制板测试标准，即IEEE1149

IEEE1149.1 －JTAG & Boundary Scan

JTAG边界扫描结构标准描述了一个用于数字电路引脚信号电平访问和控制的5引脚串行协议，并扩展到测试芯片上的电路

1. 最小集的公开指令：

BYPASS：器件将TDI经过1个时钟延时连接到TDO，用于同一个测试环中其他器件的测试

EXTEST：将边界扫描寄存器连接到TDI和TDO之间，能够捕获和控制引脚状态。这条指令用于支持板级连接测试

IDCODE：将ID寄存器连接到TDI和TDO之间

INTEST：将边界扫描寄存器连接到TDI和TDO之间，能够捕获和控制核逻辑的输入及输出状态。这条指令用于内部逻辑核的测试

1. ARM调试标准

桌面调试与嵌入式调试

桌面调试：调试器与要调试的系统都是运行在同一台PC上的不同程序，所有用户接口准备好，采用断点和调试符号表，普遍缺乏观察点工具

嵌入式调试：调试工具必须在远程机上运行，并通过某种通信方式与目标机器连接。通常采取在线仿真器（ICE）替代目标系统中的处理器，并包含缓冲器以实现硬件跟踪

1. ARM7TDMI的JTAG包含3条主要的扫描链：

Scan Chain 0 －通过扫描链0可以访问ARM7TDMI内核的外围电路，包括数据总线。通过该扫描链可以进行芯片间的测试（EXTEST）和芯片的内部测试（INTEST）。该扫描链长度为113位，具体包括：数据总线的0－31位，内核控制信号，地址总线的31－0位，EmbeddedICE的控制信号。

Scan Chain 1－扫描链1是扫描链0的子集，长度为33位，具体包括：数据总线的0－31位、BREAKPT信号。扫描链1比扫描链0的长度短了很多，通过扫描链1可以更快的插入指令或者是数据到ARM7TDMI的内部。 􀂃

Scan Chain 2－扫描链2长度为38位，该扫描链是专门用来访问EmbeddedICE内部的寄存器。通过访问EmbeddedICE的内部寄存器，可以让ARM7TDMI进入调试状态、设置断点、观察点。

1. EmbeddedICE

模块提供了

断点和观察点事件

检查并修改处理器和系统的状态

观察处理器在感兴趣点活动的轨迹

一些外部输入以便当系统级事件发生时使处理器停止

通过扩展JTAG测试端口进行控制和访问

EmbeddedICE模块包括：

1. 两个观察点单元

可以通过监控地址总线，数据总线和控制信号来探测观察点（watchpoint）和断点

每个单元可以用来提供

1 观察点, 或

1个 ROM或RAM里的硬件断点，或RAM里的多个软件断点

1. 调试控制和状态寄存器
2. 调试通讯通道（DCC）

第十五章 ARM处理器核系列介绍

1. ARM7TDMI

目前低端的ARM核，应用广泛，最适合用于对价位和功耗敏感的消费类应用

ARM7TDMI使用了3级流水线，实现ARM体系结构版本4T，支持64位结果的乘法，半字、有符号字节Load和Store以及Thumb指令集，包含了EmbeddedICE模块以支持嵌入式系统调试

ARM7TDMI典型的应用是移动电话手机（包含了一些复杂的数字信号处理硬件和相关的存储器）

进一步性能提高需要增加Cache，或者使用更复杂的更高性能水平的ARM核

1. ARM8

ARM8提供了比ARM7更加出色的性能

增加时钟速率：简化每级流水线的逻辑，增加级数

降低CPI：将ARM7占据一个以上流水线槽的指令重新实现，以占据较少的流水线槽，减小由指令间的相关性引起的流水线停止

ARM8使用5级流水线：指令预取，指令译码和寄存器读，执行（移位和ALU），访问数据存储器

回写结果

预取指单元为第1级，整数单元使用余下的4级

1. ARM9TDMI

ARM9TDMI核将ARM7TDMI的功能显著提高到更高性能的水平

ARM9TDMI支持Thumb指令集，含有EmbeddedICE模块以支持片上调试

采用5级流水线以增加最高时钟频率，使用分开的指令与数据存储器端口以改善CPI

协处理器支持：允许支持片上浮点协处理器、数字信号处理或其他专用的硬件加速要求

片上调试：EmbeddedICE功能提供了系统级的调试特性，并同时提供附加的特性

支持硬件单步调试，除了ARM7TDMI支持的地址/数据/控制条件之外，还可以在异常时设置断点

1. ARM10TDMI

ARM10TDMI核达到ARM9TDMI两倍的性能

ARM10TDMI同样在增加时钟速率与降低CPI两个方面来充分优化性能

1. 提高时钟频率

ARM9TDMI的5级流水线已经平衡得很好，如再增加级数，提高的好处可能会因为增加流水线相关性而恶化了的CPI所抵消

ARM10TDMI保留了与ARM9TDMI非常类似的流水线，采用了特别的方式优化每一级，从而支持更高的时钟频率，优化措施包括：

提前提供下一周期所需的地址，取指和存储器级有效的由一个时钟周期增加到一个半时钟周期

执行级使用改善了的电路技术和结构以缩短关键路径

指令译码级是处理器逻辑中唯一不能充分流水化以支持高速时钟的部分，所以增加了发射这一级

1. 降低CPI

改善CPI必须从考虑存储器带宽开始，ARM10TDMI采用的途径是使用64位存储器，增加指令带宽：

转移预测逻辑功能更强

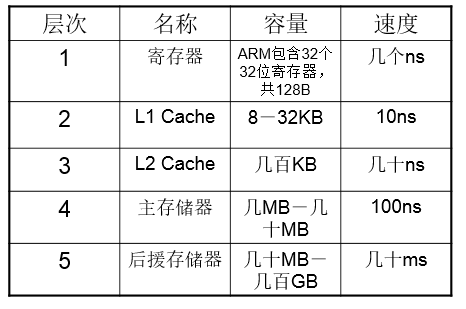
ARM10TDMI采用静态转移预测机制

非阻塞的Load和Store执行

64位的数据存储器使多寄存器Load和Store指令能够在每个时钟周期传送两个寄存器

第十五章 存储器层次与高速缓存

1. 存储器层次



主存储器与后援存储器之间的性能差别远大于其他相邻级之间的差别

寄存器中的数据可以由编译器或者汇编语言直接控制，其他层次中的内容通常为自动管理

Cache对于应用程序往往是不可见的，指令和数据以块或页的形式移动

主存储器与后援存储器的映射由操作系统控制，对于应用程序是透明的

1. 许多嵌入式系统采用简单的片上RAM，而不是Cache

简单、便宜且功耗低

有更确定的行为

使程序员能够根据对将来处理工作量的了解来划分RAM空间

缺点是需要程序员直接管理，不是透明的，不易于程序的混合不可预知的情况下

1. 高速缓存
2. Cache

高速缓冲存储器中存放的是当前使用得最多的程序代码和数据，即主存中部分内容的副本

在嵌入式系统中Cache全部都集成在嵌入式微处理器内

可分为数据Cache、指令Cache或统一Cache

不同的处理器其Cache的大小不一样

一般32位的嵌入式微处理器都内置Cache

1. 构造方法

统一的Cache，指令和数据用同一个Cache。可以根据当前程序的需要自动调整指令在Cache里的比例，比固定划分有更好的性能

指令和数据分开的Cache，称为改进的哈佛结构。速度更快，使Load/Store指令单周期执行

1. 三种组织

--直接映射Cache

最简单的Cache组织，数据行（Cache Line）连同存储器地址Tag一起保存，Tag由存储器地址中的一部分（Index）来寻址

用地址的index位访问Cache的标志项，将高地址位与存储的tag进行比较，相等说明内容在Cache中，低地址位用于访问行中相应内容

直接映射Cache的特点：

特定存储器项只能保存到cache的唯一位置

Tag存储器保存除行内寻址和cache RAM寻址所需要的位之外的其他位

Tag和数据访问可以同时进行，速度最快

Tag RAM通常远小于数据RAM，因此其访问时间短，使Tag的比较能够在数据访问时间内完成

--组相联映射Cache

组相联Cache使存储器块可以保存到多个位置以减小竞争问题，复杂性较高

一个两组相联Cache由两个可以有效并行工作的直接映射Cache构成

存储器地址可以保存到两者中的任何一个，使Cache两者都命中

当一个新数据项要放到Cache时，必须决定放到哪一半。选择方式如下：

随机存放——存放取决于一个随机或伪随机数

最近未使用（LRU）——Cache记录两个位置中哪一个是最后访问的，并将新数据放到另外一个

循环使用——Cache记录两个位置中哪一个是最近分配的，并将新数据放到另外一个

--全相联Cache组织

不是将直接映射Cache继续分为更小的元件，而是使用内容寻址存储器CAM来设计Tag RAM

一个CAM单元是具有内建比较器的RAM单元，基于CAM的Tag存储器能够并行查寻并定位任何位置地址

特点是减小了存储器之间的竞争问题，Cache利用率很高，但需要访问速度很高的CAM，其代价很高

由于没有任何地址位隐藏Cache中数据的位置，Tag存储器必须保存除用于行内字节寻址的地址位以外的其他所有地址位

1. 写策略

写策略按照复杂程度由低到高可以有：

写直达（write through）

所有的写操作直接写入主存储器。如果所寻址的数据正好保存在Cache中，则Cache更新以保存新数据。在写操作时处理器必须降到主存储器的速度

带缓冲的写直达

与写直达操作基本一致，只是在处理器与主存储器之间增加了高速接收写信息的写缓冲器（write buffer），不必降低处理器执行速度

写回法（write back）

写操作只更新Cache，并不更新主存储器。因此，Cache行必须记录是否被修改过（通过设置dirty位实现），如果新数据要调入到一个标记dirty的Cache行中，那么这个行必须先写回到主存中

Cache写策略的特点

写直达Cache实现最简单，优点是主存随时更新，缺点是每个写操作过程中处理器都降低到主存速度

加上写缓冲的写直达可以使处理器继续高速工作直至写速度超过外部写带宽

写回式Cache在最终值写入主存之前，一个位置可以被多次写入，降低了对外部写带宽的需求，但是实现上更为复杂，缺乏一致性而难于管理

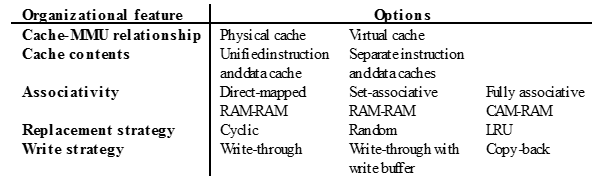
1. 物理与虚拟Cache

当系统同时实现了MMU和Cache时，Cache可以工作于虚拟（MMU之前）地址或物理（MMU之后）地址

虚拟Cache优点是处理器产生一个地址之后可以立即开始一个Cache访问，缺点是可能包含同义项（synonyms），同一主存数据项在Cache中有重复拷贝，导致Cache不一致性

物理Cache由于地址与数据项的唯一相关性，避免了同义项问题，但是MMU必须在每个Cache访问时激活，导致Cache延迟增加

Cache特点



第十六章 存储器管理单元及保护单元

用于一般应用的ARM CPU，其应用程序的范围及数量在设计时是未知的，因此采取具有地址转换的、完整的存储器管理单元

MMU完成两个基本功能：将虚拟地址转换为物理地址；控制存储器访问权限，中止非法访问

ARM MMU使用具有步行表（table-walking）硬件的两级页表和一个保存最近使用的页转换的TLB

对于指令和数据分开的微处理器，也可能包含分开的指令和数据TLB