第一章 嵌入式系统导论

按嵌入式系统软件复杂程度来分类

循环轮询系统

有限状态机系统

前后台系统

单处理器多任务系统

多处理器多任务系统

第三章 处理器设计导论

计算机体系结构

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。

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：

ARM 指令寻址模式

- 立即寻址
- 寄存器寻址
- 寄存器间接寻址
- 基址偏移寻址
- 基址变址寻址
- 堆栈寻址

指令类型：

- 数据处理指令
- 数据传送指令
- 控制流指令

RISC VS CISC

	CISC	RISC
价格	由硬件完成部分软件功能，硬件复杂性增加，芯片成本高	由软件完成部分硬件功能，软件复杂性增加，芯片成本低
性能	减少代码尺寸，增加指令的执行周期数	使用流水线降低指令的执行周期数，增加代码尺寸
指令集	大量的混杂型指令集，有简单快速的指令，也有复杂的多周期指令，符合HLL（high level language）	简单的单周期指令，在汇编指令方面有相应的CISC微代码指令
高级语言支持	硬件完成	软件完成
寻址模式	复杂的寻址模式，支持内存到内存寻址	简单的寻址模式，仅允许LOAD和STORE指令存取内存，其它所有的操作都基于寄存器到寄存器
控制单元	微码	直接执行
寄存器数目	寄存器较少	寄存器较多

数据处理指令

数据处理不涉及memory

- 功能：完成寄存器数据的算术和逻辑操作
- 原则：
  - 操作数32位，来自寄存器或者指令中的立即数
  - 如果有结果，结果也32位宽，存放于寄存器中
  - 每个操作数寄存器与结果寄存器在指令中单独指定，采用3地址模式

操作数1、操作数2、目的地

立即数通过“#”表示，且在32位指令中编码，编码方式为：

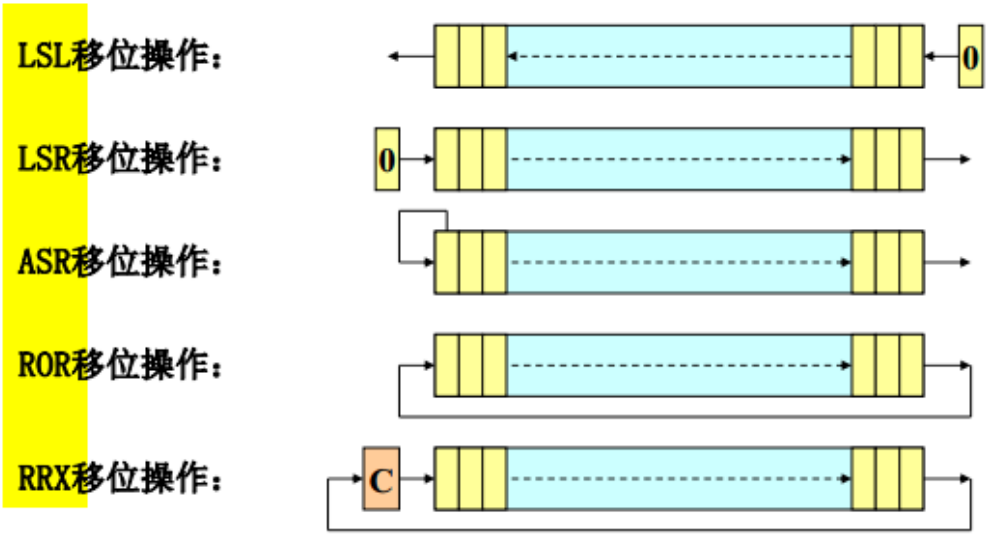
$$立即数=（0-255）\times 2^{2n}$$

$0 \leqslant n \leqslant 12$　单字节数左移2n位

数据处理指令

6种移位指令，
循环移位和带扩展循环移位都只能右移

ASL与LSL等价（取最后一位来补位是没有意义的）



数据传送指令

- 功能：完成寄存器数据与存储器数据的传送
- 包括三种基本的数据传送指令：
  - 单寄存器Load/Store指令
  - 多寄存器Load/Store指令
  - 单寄存器交换（SWP）指令

多寄存器Load/Store指令

```
LDMIA r1, {r0,r2,r5}      ; r0:=mem32[r1]
                          ; r2:=mem32[r1+4]
                          ; r5:=mem32[r1+8]
```

无论指令中寄存器如何排列，
存取顺序都按编号从小到大来

STM 批量存储，LDM 批量加载；
对于堆栈寻址，
　　第四个字母表示指针指向的项类型，
　　　E 为空栈，F 为满栈
　　第五个字母表示增长方向，
　　　A 为向上，D 为向下
　　　（向上增长的堆栈，加变址存储，减变址加载）
对于块拷贝寻址，
　　第四个字母表示变址方向，
　　　I 为加变址，D 为减变址
　　　（加变址存储，减变址加载为向上增长的块）
　　第五个字母表示变址时机，
　　　A 为后变址，B 为前变址

控制流指令

- 功能：使指令执行切换到不同的地址
- 按照切换方式可以分为：
  - 永久转移—转移指令（B）
  - 保存返回地址以恢复原来的执行顺序（BL）
  - 陷入系统代码（SWI）

ARM伪指令有四条：

- **ADR伪指令** 小范围地址读取
- **ADRL伪指令** 中等范围地址读取
- **LDR伪指令** 大范围地址读取
- **NOP伪指令**

处理器状态切换

1、设置状态为（位[0]）后用BX跳转进入
2、Thumb状态下进入异常切换到ARM状态；
异常处理完成后返回到Thumb状态

- 将一个非字（半字）对齐的地址写入ARM（Thumb）状态的R15寄存器，将引起非对齐的指令取指。所以不能直接通过忘PC写入半字对齐的地址来切换到Thumb状态

大端格式（Big-endian） 高字节低地址

小端格式（Little-endian） 低字节低地址

处理器模式

处理器模式	说明	备注
用户 (usr)	正常程序工作模式	不能直接切换到其它模式
系统 (sys)	用于支持操作系统的特权任务等	与用户模式类似，但具有可以直接切换到其它模式等特权
快中断 (fiq)	支持高速数据传输及通道处理	FIQ异常响应时进入此模式
中断 (irq)	用于通用中断处理	IRQ异常响应时进入此模式
管理 (svc)	操作系统保护代码	系统复位和软件中断响应时进入此模式
中止 (abt)	用于支持虚拟内存和/或存储器保护	在ARM7TDMI没有大用处
未定义 (und)	支持硬件协处理器的软件仿真	未定义指令异常响应时进入此模式

ARM状态各模式下的寄存器

寄存器类别	寄存器在汇编中的名称	各模式下实际访问的寄存器						
		用户	系统	管理	中止	未定义	中断	快中断
通用寄存器和程序计数器	R0(a1)	R0						
	R1(a2)	R1						
	R2(a3)	R2						
	R3(a4)	R3						
	R4(v1)	R4						
	R5(v2)	R5						
	R6(v3)	R6						
	R7(v4)	R7						
	R8(v5)	R8						R8_fiq
	R9(SB,v6)	R9						R9_fiq
	R10(SL,v7)	R10						R10_fiq
	R11(FP,v8)	R11						R11_fiq
	R12(IP)	R12						R12_fiq
	R13(SP)	R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
	R14(LR)	R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
	R15(PC)	R15						
状态寄存器	CPSR	CPSR						
	SPSR	无		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

异常处理器模式

七个异常类型，五个异常模式

异常类型	模式	正常地址
复位	管理	0x00000000
未定义指令	未定义	0x00000004
软件中断（SWI）	管理	0x00000008
预取中止（取指令存储器中止）	中止	0x0000000C
数据中止（数据访问存储器中止）	中止	0x00000010
IRQ（中断）	IRQ	0x00000018
FIQ（快速中断）	FIQ	0x0000001C

进入异常

1. 保存当前 PC 到 LR（可能是当前指令的下一条指令也可能是下下条指令）

2. 保存 CPSR 到 SPSR
3. 设置 CPSR 的模式位
4. 设置 PC 从异常向量处取指
5. 设置中断禁止位

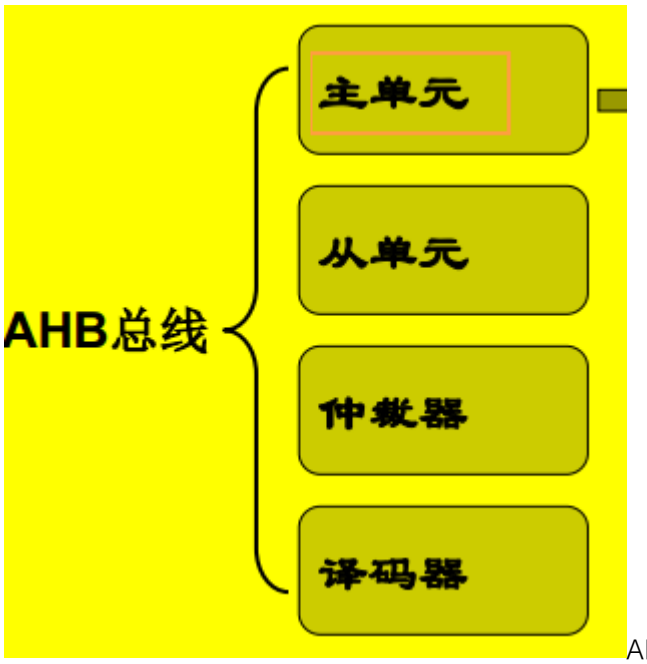
退出异常

1. LR 减去偏移量后恢复到 PC
2. 恢复 SPSR 到 CPSR
3. 清零中断禁止位

第六章 ARM 外设资源

AMBA规范

- **AHB（the Advanced High-performance Bus）** 高级高性能总线
  - 应用于高性能、高时钟频率的系统模块，它构成了高性能的系统骨干总线（ back-bone bus ）。
- **ASB（the Advanced System Bus）** 高级系统总线
  - 是第一代AMBA系统总线，同AHB相比，它数据宽度要小一些，它支持的典型数据宽度为8位、16位、32位。
  - 后来由AHB取代。
- **APB（the Advanced Peripheral Bus）** 高级外设总线
  - 是本地二级总线（local secondary bus ），通过桥和AHB/ASB相连。它主要是为了满足不需要高性能流水线接口或不需要高带宽接口的设备的互连。



AHB 一个或多个主单元

- APB主要由下面2部分组成：
  - APB桥
  - APB从单元（Slave）
- APB桥是APB中唯一的主单元，是AHB/ASB的从单元

第七章 体系结构对高级语言的支持
简单的条件执行——一个条件、一条执行语句，
可以无需跳转语句，
直接由 CMP 和两个带条件的执行语句完成，
产生很简短高效的汇编代码

复杂的条件语句——多个条件或有多条执行语句

往往需要借助条件跳转语句来完成，
执行效率就比较低

Switch 语句：
1、转换为若干组 if-else
2、基于基址的偏移的跳转表（case 的值为简单的 0、1、2……）

```
        ；r0包含表达式的值
        ADR r1，JUMPTABLE ；得到跳转表的基址
        CMP r0，＃TABLEMAX          ；检查是否超限
        LDRLS pc，[r1，r0，LSL ＃2]  ；否则得到pc
              …每条ARM指令有四个字节；statementD
        B EXIT                      ；中断
L1            …                     ；statement1
        B EXIT                      ；中断
              …
LN            …                     ；statementN
EXIT          …
```

3、基于 PC 的偏移的跳转表（case 的值为简单的 0、1、2……）

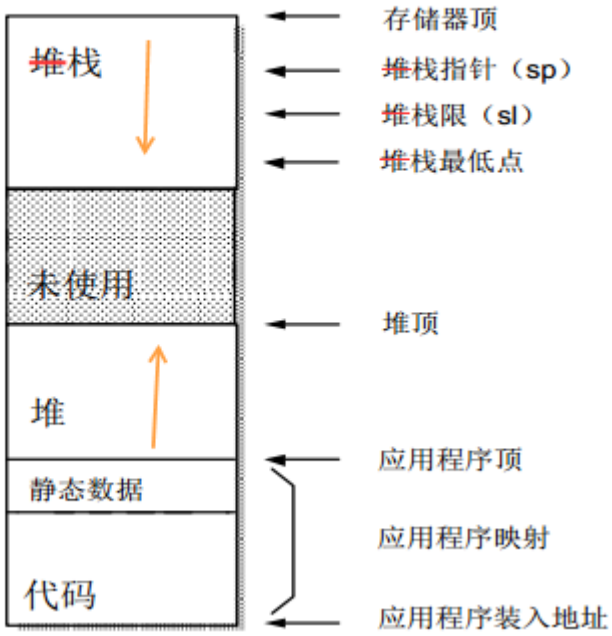
```
        ；在入口，a1=0,a2=3,v1=b,v2=表达式
        CMP v2,＃4                  ;检查是否超限
PC      ADDLS pc,pc,v2,LSL #2       ;不超限，则加到pc  PC指向下下条指令
        LDMDB fp, {v1,v2,fp,sp,pc}  ;如果超限，则返回
        B L0                        ;case 0     跳转指令和跳转表之间有且
        B L1                        ;case 1     只能有一条指令；
        B L2                        ;case 2     否则需要计算准确的偏移量
        LDMDB fp, {v1,v2,fp,sp,pc}  ;case 3
        MOV a1,#2                   ;case 4
        STR a1,[v1]
        LDMDB fp, {v1,v2,fp,sp,pc}  ;返回
L0      STR a1,[v1]
        LDMDB fp, {v1,v2,fp,sp,pc}  ;返回
L1      LDR a3,c_ADDR               ;得到c的地址
        LDR a3,[a3]                 ;得到c
        CMP a3,#&64                 ;c>100? …
        STRLE a2,[v1]               ;否：*b=3
        STRGE a1,[v1]               ;是：*b=0
        LDMDB fp, {v1,v2,fp,sp,pc}  ;返回
c_ADDR  DCD <address of c>
L2      MOV a1,#1
        STR a1,[v1]                 ;*b=1
        LDMDB fp, {v1,v2,fp,sp,pc}  ;返回
```

For 循环
初始化-(测试-跳转-执行-跳转)-(测试-跳转-执行-跳转)-…-(测试-跳转)

While 循环
1. (测试-跳转-执行-跳转)- (测试-跳转-执行-跳转)-…-(测试-跳转)
2. 跳转-(测试-跳转-执行)-(测试-跳转-执行)-…-(测试-跳转)-结束
3. 测试-跳转-(执行-测试-跳转)-(执行-测试-跳转)-…-(执行-测试-跳转)-结束

ATPCS：
R0~R3(a1~a4)：传递参数或返回值，无需保护
R4~R11(v1~v8)：保存局部变量，需要保护
【Thumb 只有 R4~R7】
R12(ip)：子程序临时寄存器，用于保护 sp
R13(sp)：数据栈指针，进入子程序和退出子程序必须相等
R14(lr)：链接寄存器，保存子程序的返回地址
R15(pc)：程序计数器
数据栈为字节对齐的 FD 栈
参数个数可变 & 参数个数固定
参数、返回值优先使用 R0-R3，如果不够则使用数据栈（优先使用编号小的寄存器）
整数用 R0、R1 返回，
浮点数或复合浮点数用浮点运算部件的寄存器返回；
如果需要更多位数，则通过内存（数据栈）传递；

地址空间模型

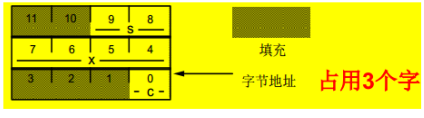


存储器的使用

- 对齐的数据访问效率最高，非对齐的数据访问则要低效很多，如向非字对齐的地址存储一个字将使用多达7条ARM指令，并需要临时工作寄存器
- 如果同时声明几种不同类型的数据项，编译器需要进行填充操作

struct的内存分配顺序是固定的（先定义的先分配）

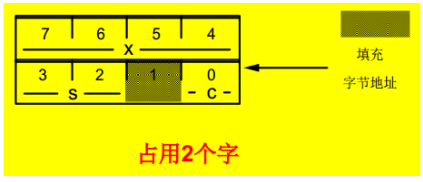
struct S1{char c；int x；short s；} example1；



存储器的使用

- 可以通过重新组织结构体来帮助编译器减小存储器的浪费

struct S1{char c；short s；int x；} example2；



第九章 ARM 体系结构

ARM采用的技术特征：

- Load/Store体系结构 RISC一个重要的特点
- 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

寄存器窗口：
处理器中设置一个数量较大的寄存器堆，每个过程使用一组寄存器，并且部分寄存器与上一个过程、下一个过程重叠，以此来传递过程与过程间的参数；

延迟转移：
RISC流水线机制，使得取指、译指、执行并行进行，当执行转移指令时可能会产生断流，因此在转移指令之后插入一条有效指令，使转移看起来像是被延迟了

ARM体系结构的主要特点：

- Load/Store结构
- 3地址的数据处理指令
- 所有指令条件执行
- 强大的多寄存器Load/Store指令
- 单时钟周期，单条指令完成一项普通的移位操作和一项普通的ALU操作 （不是指所有指令都是单周期）

- 通过协处理器指令集扩展ARM指令集，在编程模型中增加了新的寄存器和数据类型
- 在THUMB体系结构中以高密度的16位压缩形式表示的指令集

ARM体系结构的各种版本

- ARM体系结构V1-V3没有用于商业授权
- ARM体系结构V4-V7正在使用
- ARM体系结构最新版本—V8

V4增加了Thumb
V5增加了Jazelle
V6
V7将版本分为A、R、M
V8引入64位架构

- ARMv7定义了3种不同的处理器配置（processor profiles）：
  - **Profile A是面向复杂、基于虚拟内存的OS和应用的**
  - **Profile R是针对实时系统的**
  - **Profile M是针对低成本应用的优化的微控制器的**

Application
Real-time
Micro controler

MIPS（Million Instrunction Per Second）：
每秒执行多少百万条单字长定点指令

第十章 ARM 存储器接口及 AMBA 总线

周期类型

nMREQ	SEQ	Cycle Type
0	0	Non-sequential
0	1	Sequential
1	0	Internal
1	1	Coprocessor register tran

- 非连续 (N) 等待时间较长
  - 在接下来的周期中的地址与前一个地址无关
- 连续 (S) 等待时间较短
  - 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

IS：下条指令取指地址出现在地址总线上，容许提前译码

第十一章 ARM 流水线组织

3级流水线组织

- 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。

5级流水线组织

- 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
    3级里该操作是在执行中
  - 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
    比3级多出来的操作，专门用于数据访问
  - 回写：将结果回写到寄存器堆
    3级里该操作在执行中

$$T_{prog}=N_{inst}\times CPI/F_{clk}$$

T_{prog}：程序的执行时间
N_{inst}：程序中执行的ARM指令数
CPI：每条指令的平均时钟周期数 Clock Per Instrument
F_{clk}：处理器的时钟频率

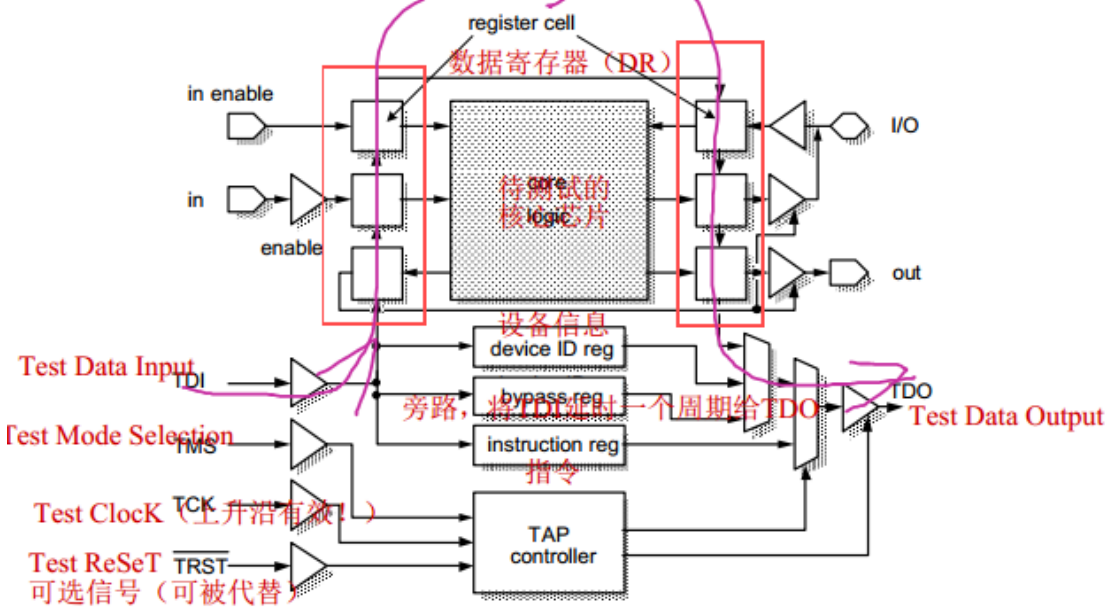
提高性能的办法

- 减小执行指令的数量（通过编译器优化）
- 提高时钟频率F_{clk}
  简化流水线每一级的逻辑，增加级数
- 减小CPI

数据处理：单周期（包括移位、偏移等）
数据传送：STR 两周期（写信号和数据，实际写入）
　　LDR 三周期（读信号，取数据，实际读入）
控制流：三周期（计算目标地址，保存返回地址，修正返回地址）

第十二章 ARM JTAG 测试和调试结构

JTAG边界扫描测试结构



JTAG边界扫描测试结构

- 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

宏单元测试

- 系统芯片使用大量的复杂、已设计好的宏单元
- 宏单元的产品测试向量主要依赖于宏单元供应商
- 将测试向量加到宏单元的方法：
  - 通过多路器使每个宏单元的信号依次连接到系统芯片的引脚上的测试模式
  - 片上总线（AMBA总线）可以支持每个连接到总线上的宏单元的直接测试访问
  - 每个宏单元可以有一个边界扫描路径，使用扩展的JTAG结构，测试向量可以通过扫描路径加到宏单元上

几种常见的调试方法：

- 指令集模拟器 用的不多

  一种利用PC机端的仿真开发软件模拟调试的方法。
- 驻留监控软件

  驻留监控程序运行在目标板上，PC机端调试软件可通过并口、串口、网口与之交互，以完成程序执行、存储器及寄存器读写、断点设置等任务
- JTAG仿真器

  通过ARM芯片的JTAG边界扫描口与ARM核进行通信，不占用目标板的资源，是目前使用最广泛的调试手段
- 在线仿真器

  使用仿真头代替目标板上的CPU，可以完全仿真ARM芯片的行为。但结构较复杂，价格昂贵，通常用于ARM硬件开发中

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

ARM调试结构

观察点：任意方式访问特定地址
硬件断点：执行特定地址的任意指令
软件断点：执行任意地址的特定指令

EmbeddedICE模块包括：

- 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
- 调试控制和状态寄存器
- 调试通讯通道（DCC）

第十三章 ARM 处理器核系列
体系结构版本：v1~v8
核系列：ARM7~ARM10

ARM7TDMI：
T：Thumb
D：Debug
M：Multiplier
I：Embedded ICE
三级流水线
冯诺依曼结构
Thumb 指令译码需要解压
ARM8
冯诺依曼结构
顺序访问存储器可以获得双倍带宽
五级流水线
分为预取指单元（第一级）和整数单元（后四级）
ARM9TDMI
哈佛结构
Thumb 指令硬件译码
ARM10TDMI

六级流水线（从译码分离出发射）
提前提供下一周期所需地址
改善电路技术与结构
使用 64 位存储器增加带宽
转移预测机制
非阻塞 LOAD/STORE

第十四章 存储器层次与高速缓存

Cache的有效性依赖于程序具有空间局部性和时间局部性的特性

主存：处理器可以直接访问
外村：处理器不能直接访问

NandFlash具有容量大、回写速度快、芯片面积小等特点，主要用于外存。

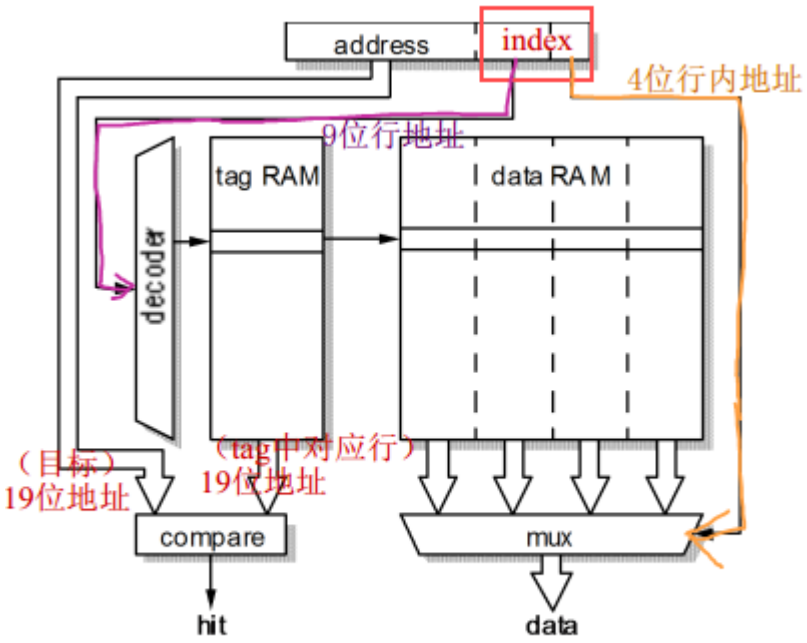
NOR Flash具有随机存储速度快、电压低、功耗低、稳定性高等特点，主要用于主存。

类似哈佛结构和冯诺依曼结构

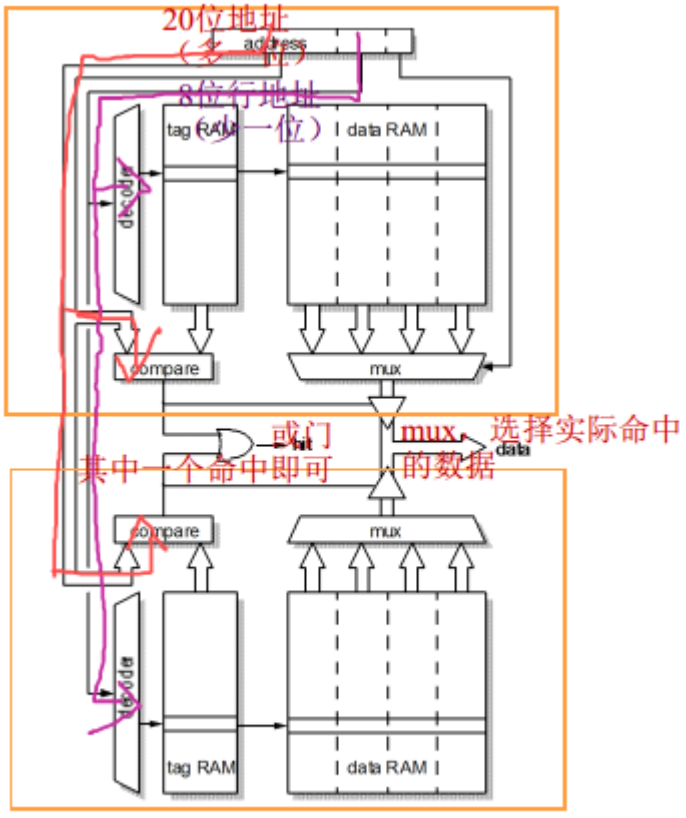
可分为数据Cache、指令Cache或统一Cache

CPU是不会直接对主存储器进行操作的

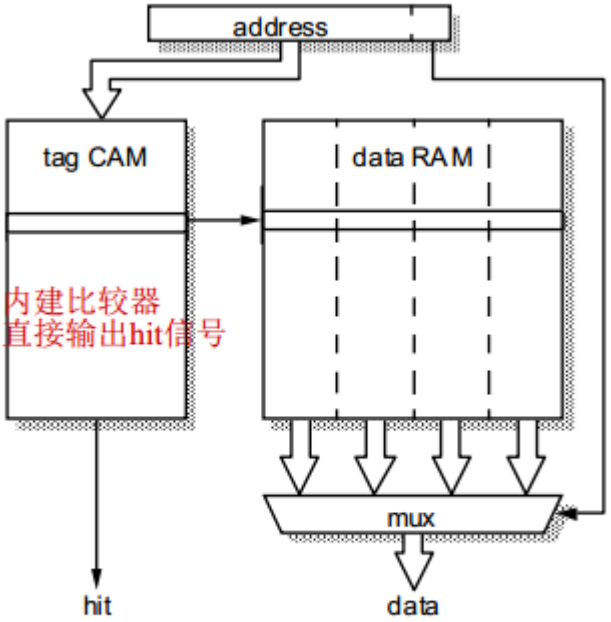
Cache 组织



- 直接映射Cache组织



- 组相联Cache组织



- 全相联Cache组织

- 组相联Cache组织
  - 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

- Cache写策略 处理器写入主存储器
  - 写策略按照复杂程度由低到高可以有：
    - 写直达（write through） 直接写入主存储器，检查是否在Cache并更新；主存随时保持最新，但处理器需要降速
      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 处理器和主存储器之间加缓冲器；处理器无需降速，只受缓冲器外部写速度的限制
      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） 只写入Cache，等到该行要被换出时再写入主存储器；没有外部写速度的限制，但复杂缺乏一致性难以管理
      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

复杂度由低到高

MMU和Cache的联系

- 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 速度快，可能包含同义项
  - 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 速度慢，没有同义项

第十五章 存储器管理单元和保护单元

程序间切换是由操作系统管理的，

程序切换是通过存储器管理单元支持的，

存储器管理的两种基本方法：

- 段式管理（segmentation）
- 页式管理（paging）

ARM 体系结构只支持页式

- 采用页方式访问存储器都会带来两次额外的存储器访问，即在访问需要数据之前要进行1次页目录和1从页表访问

- TLB可以有效改善这些开销。TLB是最近用过的页转换的Cache，同样具有与相联度和替换策略有关的组织选项

即专门用于加速页目录和页表访问的Cache

存储器管理单元（MMU）

- 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

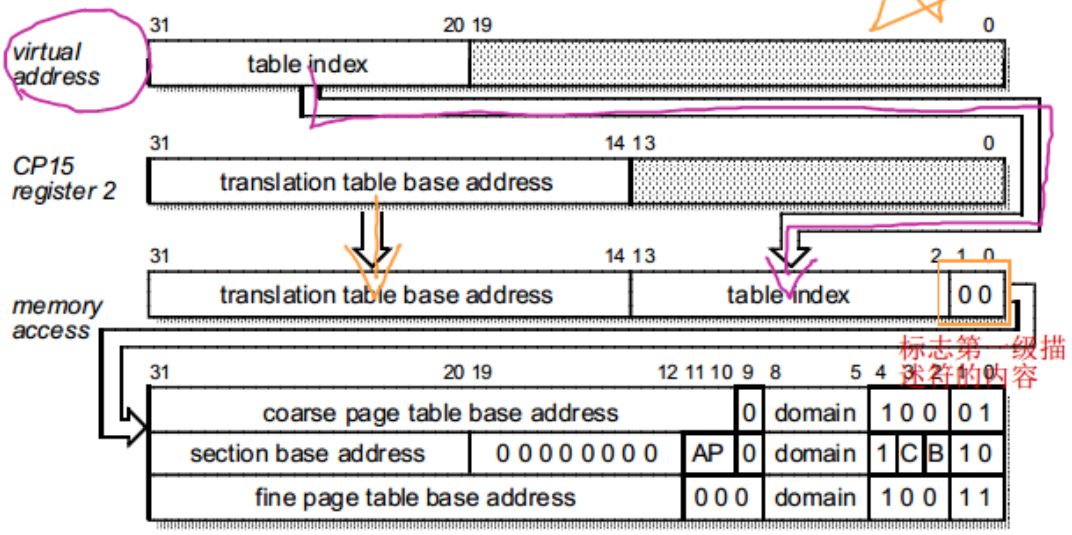
保护单元（Protect Unit）

- 适合应用于嵌入式系统的ARM CPU执行固定或可控制的应用程序，不需要完整的MMU，具有简单的保护单元就足够，相应的开销会小许多

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

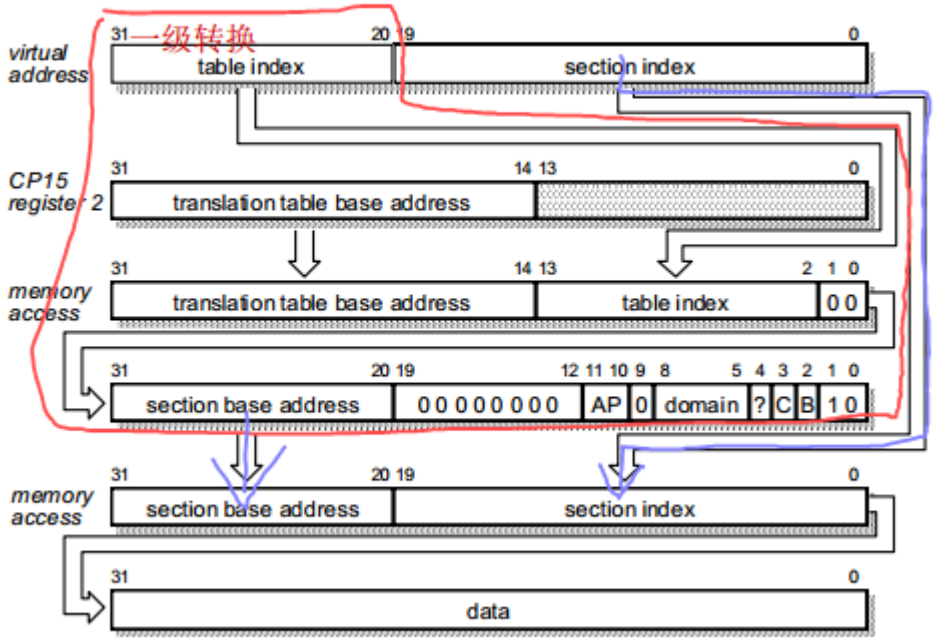
MMU转换过程



■ 第一级转换

段转换 二级转换

虚拟地址：一级页目录索引 + 段内索引

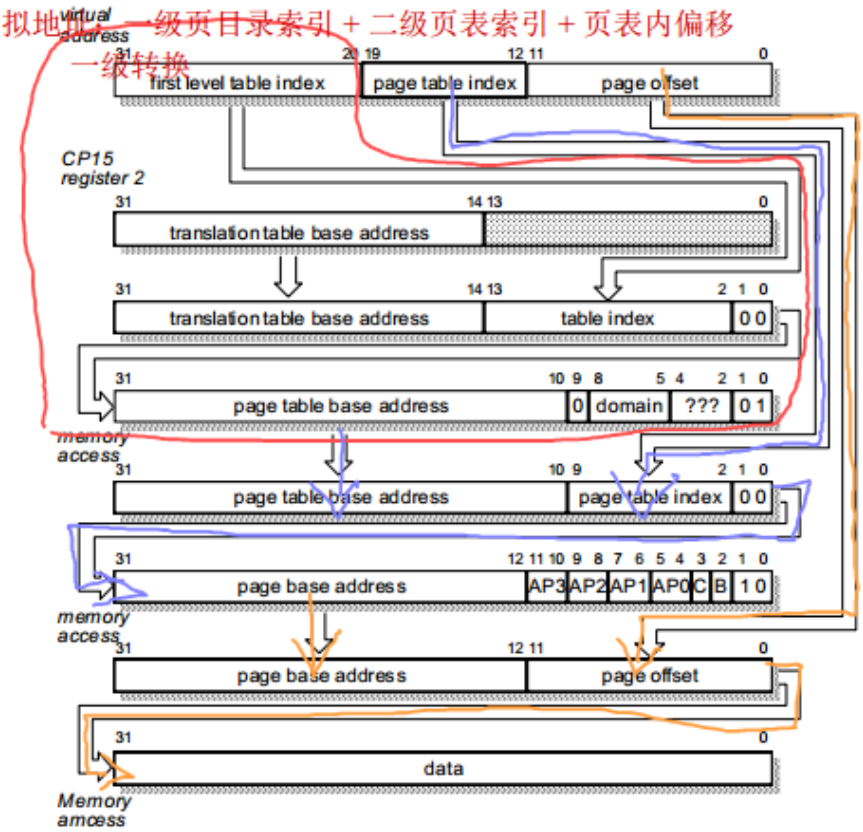


■ 段转换过程

小页转换

虚拟地址：一级页目录索引 + 二级页表索引 + 页表内偏移

- 小页转换过程



访问权限

1.地址对齐检查
2.查找第一级描述符指定的页域
3.检查当前过程对该页域的身份（用户？管理？都不是？）
4.若是用户程序，还需检查对该页域的访问权限