# 第一章、导论

无处不在的计算机包括**通用计算机**和**嵌入式计算机系统**

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**。

|  |  |  |
| --- | --- | --- |
| 特征 | 通用计算机 | 嵌入式系统 |
| 形式和类型 | 看得见的计算机。  按其体系结构、运算速度和结构规模等因素分为大、中、小型机和微机。 | 看不见的计算机。  形式多样，应用领域广泛，按应用来分。 |
| 组成 | 通用处理器、标准总线和外设。  软件和硬件相对独立。 | 面向应用的嵌入式微处理器，总线和外部接口多集成在处理器内部。  软件与硬件是紧密集成在一起的。 |
| 开发方式 | 开发平台和运行平台都是通用计算机 | 采用交叉开发方式，开发平台一般是通用计算机，运行平台是嵌入式系统。 |
| 二次开发性 | 应用程序可重新编制 | 一般不能再编程 |

嵌入式系统一般由**嵌入式硬件和软件**组成

硬件以微处理器为核心集成存储器和系统专用的输入/输出设备

软件包括：初始化代码及驱动、嵌入式操作系统和应用程序等形成的特定的**一体化软件**。

嵌入式系统的特点

* 形式多样、**面向特定应用**的
* 得到多种类型的处理器和体系结构的支持
* 通常**极其关注成本**
* 有**实时性和可靠性**的要求（实时系统）
* 使用的操作系统一般是适应多种处理器、可剪裁、轻量型、实时可靠、可固化的嵌入式操作系统
* 开发需要**专门工具和特殊方法**

嵌入式处理器：

4、8、16位，大量使用，32位主流发展趋势，64位高复杂高速系统应用

**硬/软实时系统**：对实时性要求非常高，响应时间不满足即崩溃为硬，否则为软

**前后台系统：**后台是一个循环轮询系统一直在运行。前台是由一些中断处理过程组成的。当有一前台事件(外部事件)发生时，引起中断, 进行前台处理, 处理完成后又回到后台（需要考虑中断现场保护和恢复）

# 第二章、处理器设计导论

## Chapter1、处理器体系结构和组织

计算机系统＝软件＋硬件/**固件**

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。

**是否把指令和数据放在同一存储器中？**

优点：

不必**预先区分**指令和数据，易实现存储管理软件；

程序和指令在**执行过程中可以被修改**，因而可以编写出灵活的可修改的程序；

对于存取指令和数据仅需一套读/写和**寻址电路**，**硬件简单**；

数据可以分配于任何可用空间，从而可更**有效地利用存储空间**等。

缺点：

不利于进行程序**调试诊断**；

不利于实现程序的**可再入性和程序的递归**调用；

不利于**重叠和流水方式**的操作。

现在绝大多数计算机都规定，在执行进程中**不准修改程序**。

## Chapter2、简单处理器设计介绍-MU0

采用冯.诺伊曼结构，**指令存储在外部存储器**中，I/O模块提供对外接口，MPU、存储器、I/O模块之间通过处理器总线（数据、地址、控制）相连

MPU包括：

算术逻辑单元（ALU）；

控制逻辑（Control Logic）；

寄存器（Register）(PC,IR,ACC)

执行程序步骤：

取值（存储器）🡪译码（MPU）🡪取操作数（存储器）🡪执行

指令集：

12位地址空间、16位数据总线（指令长度可达16位 4操作+12地址）

每条指令占用的周期数由**访问存储器的次数**决定

指令一般包括：**操作码**与**操作数**

## Chapter3、指令集设计与流水线

指令地址（相加为例）：

四地址

三地址：下一条指令地址为隐含的（ARM指令集采用）；

二地址：一个源操作数用作目标寄存器；

一地址：将目的寄存器隐含

**零地址**：堆栈

寻址模式：

**立即寻址**、绝对寻址、间接寻址、**寄存器寻址、寄存器间接寻址、基址偏移寻址、基址变址寻址**、基址比例变址寻址、**堆栈寻址**

指令类型：

数据处理指令（使用率最高）、数据传送指令、控制流指令、控制处理器执行状态的特殊指令

让处理器运行更快：

**流水线技术、高速缓存**、超标量指令执行

**流水线典型步骤：**

取指🡪译码🡪取操作数🡪操作数进行组合🡪访问存储器🡪回写

流水线要求：

指令执行步骤一致、指令集长度固定、没有转移指令，**一般3—5级为好**

## Chapter4、RISC VS CISC

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；

RISC组织：

硬连线的指令译码逻辑；

**流水线**执行；

指令**单周期**执行；

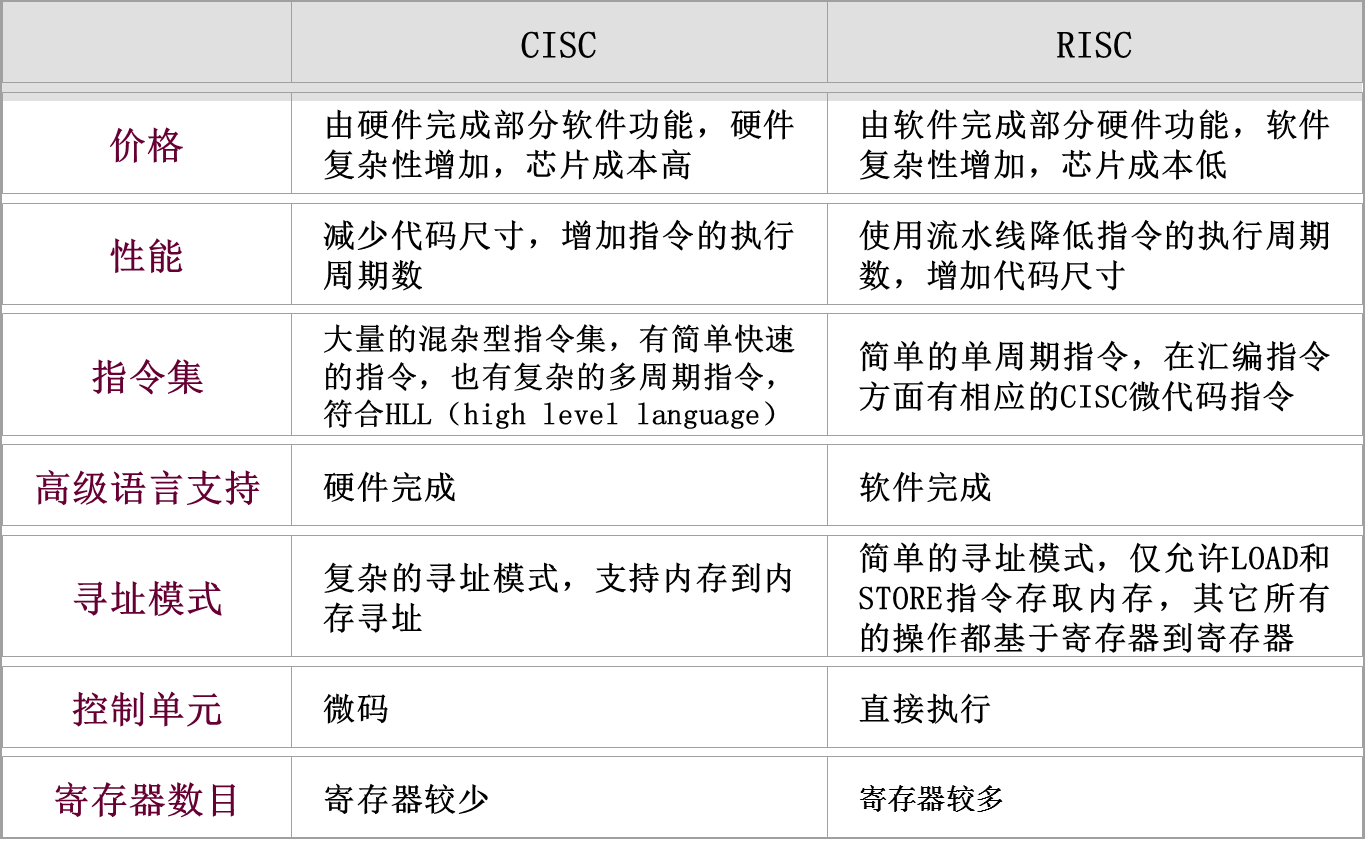
较**高的时钟频率；**

RISC优点：面积小、开发快、性能高

RISC缺点：**代码密度低**（指令长度固定，单周期导致代码无用部分多，在ARM中采用**16位Thumb代码**弥补）

CISC整体反之

昂贵、代码尺寸小（周期多）、大量混杂型指令集、寻址模式复杂等。



# 第三章、ARM体系结构

## Chapter1、ARM体系结构主要特点

**Load/Store结构**

**3地址的数据处理指令** 前三者为RISC的继承

**强大的多寄存器Load/Store指令**

32位指令

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

所有指令条件执行

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

## Chapter2、ARM体系结构的各种版本

V4（目前商用最老）：开始支持Thumb指令

V4T：增加了16位Thumb指令集

**Thumb指令在处理器中仍为32位，但用户采用16位，获得更高代码密度**

V5T**E**：增强Thumb体系，**更好平衡代码空间和性能**；

在ARM ISA上扩展了**增强的DSP指令集；**（命名中的E表示）

V6：多项改进，尤其在视频音频处理方面；

V7：最新的版本

## Chapter3、ARM处理器系列

按体系结构分：

V5TE：ARM7、ARM9

V6：ARM9、ARM10

V7：ARM11、ARM Cortex

按应用特征：

应用处理器、实时控制处理器、微控制器（性能上递减）

Cortex-A Cortex-R Cortex-M

目前最快的嵌入式处理器之一：Cortex-A9；最便宜的之一：Cortex-M3

# 第四章、ARM编程模型

## Chapter1、数据类型 处理器工作状态

支持的数据类型：

**字节：**8位 **半字：**16位 **字：**32位

工作状态：

ARM状态（字对齐的ARM指令）、Thumb状态（半字对齐的Thumb指令）

## Chapter2、存储器及存储器映射I/O

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（以字为基本单元）**

跳转目标的计算方法：

(当前指令的地址) **+ 8（两个字即跳转指令长度）**+ 偏移量

下一条指令位置的计算方法：

(当前指令的地址) **+ 4（四个字节）**

存储器格式：

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的字**包含的半字位**于地址A和**A+2**；

**小端模式：**在一个字中，高位数据存放在高位地址中，**大端模式**反之

未对齐的存储器访问：

不是字对齐的访问均为**未对齐的存储器访问**。

指令的预取：

在前一条指令的执行尚未完成时将指令从存储器中取出，可能最终不被执行，因为：**发生异常、发生跳转**

预取可能存在的问题：

在存储器中的指令可能在它**被预取之后，被执行之前发生改变**。如果发生这种情况，对存储器中的指令进行修改一般**不能阻止已取指的指令的执行。**

## Chapter3、处理器模式

ARM体系结构支持7种处理器模式，分别为：**用户模式、系统模式、快中断模式、中断模式、管理模式、中止模式、未定义模式**。

|  |  |  |
| --- | --- | --- |
| **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，与用户模式共用寄存器，此模式修改寄存器更方便 |
| 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 |

后**六**种为**特权模式**，可以访问一些片内外设和内部寄存器，特权模式下可以自由切换；后**五**种为**异常模式**，还可以通过特定的异常进入，都有独立的寄存器。

处理器启动时的模式转换

管理模式（复位后默认）🡪多种特权模式（主要设置各模式堆栈）🡪用户程序运行模式

## Chapter4、寄存器组织

ARM有**37（31+6）**个物理寄存器，有**18个可编程访问**的寄存器

31个通用32位寄存器，6个状态寄存器

18个可编程访问寄存器中

* **R0-R13**为保存数据或地址值的**通用寄存器**，不会被指定为特殊用途；
* **R0-R7**为**未分组寄存器**，即对于任何处理器模式，都对应于相同的32位物理寄存器
* **R8-R14**为**分组寄存器**，即对应的物理寄存器取决于当前的处理器模式：
* **R8-R12**为**两个分组**的物理寄存器。其中一个特殊用于**FIQ（快中断）**模式；
* **R13、R14**为**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。
* **R13**常作为**堆栈指针（SP）**（只是**习惯上**这样使用），指向各异常模式专用堆栈；
* **R14**为**链接寄存器（LR）**
* 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）。
* **注意要点**：当发生**异常嵌套**时，这些异常之间可能会发生冲突。（保存的地址可能被覆盖）
* 解决办法：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**。
* **R15**为**程序计数器（PC）**，指向**正在取指**的地址，可以认为是通用寄存器，但是使用受限
* 当前**正在执行**指令的地址**加上8个字节**（三级流水线）；
* 由于ARM指令总是以字为单位，所以R15寄存器的**最低两位总是为0**（4的倍数）；
* 写R15值相当于执行一次**无条件跳转**
* **CPSR**：**程序状态寄存器**
* ４个条件代码标志（最高四位）

负，零（前两主用于比较），进/借位，溢出（后两者主要用于加减）；

* 最低八位（0-7）为控制位，**发生异常时被硬件改变**或**特权模式下软件改变**：
* 2个中断控制位（IRQ中断禁止（7）、FIQ快中断禁止（6））；
* 5个对当前**七种处理器模式**进行编码的位（最低五位0-4）；
* 1个指示当前执行指令的工作状态位（ARM/Thumb）；
* 保留位（8-27），保留使用，提高程序可移植性。
* **SPSR**：**程序状态保存寄存器（异常模式下）**

每种异常均有自己对应的SPSR，保存异常发生之前的CPSR

Thumb状态寄存器是ARM状态集的**子集：**

R0-R7，PC（ R15），LR（R14），SP（R13），有条件访问CPSR

发生异常时，处理器**自动进入ARM状态**；

## Chapter4、异常

正常的程序流被暂时中止，处理器就进入异常模式

**进入异常**

1. 在对应模式的LR中保存下一条指令的地址

异常入口为**当前指令+8/4**（ARM状态）或**当前指令+2/4**（Thumb状态）

1. 将CPSR复制到对应SPSR中；
2. **强制设置CPSR处理器模式位**为与异常类型相对应的值；
3. 强制PC从相关的异常向量出取址（进入异常）。

IRQ和FIQ应**置位禁止标志**，防止不受控制的异常嵌套。

**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。

恢复CPSR的动作会将T（状态位）、F和I位（禁止标识）自动恢复为异常发生前的值。

快速中断请求

对异常的快速响应（得益于**八个**专用的寄存器），返回时R14\_fiq减4；

中断请求

优先级低于FIQ，返回时R14\_irq减4存入PC，SUBS PC,R14\_irq,#4；

中止：发生在对存储器的访问不能完成时

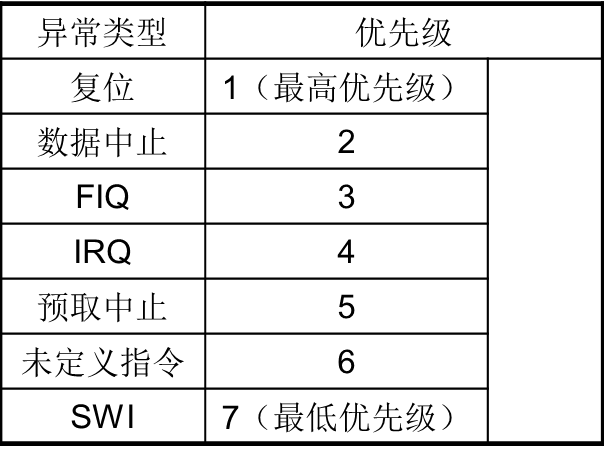
预取中止——指令预取过程中，但在**到达流水线执行阶段才进入异常**  SUBS PC,R14\_abt,#4

数据中止——数据访问时 SUBS PC,R14\_abt,**#8**

软件中断指令（SWI）：可以进入管理模式 **MOVS** PC,R14\_svc

未定义的指令：遇到无法处理的指令 **MOVS** PC,R14\_und

异常优先级：



未定义的指令和SWI异常互斥。因为同一条指令不能既是未定义的，又能产生有效的软件中断；

数据中止的优先级必须高于FIQ以确保数据转移错误不会被漏过。

当FIQ使能，并且FIQ和数据中止异常同时发生时

1. ARM内核进入数据中止处理程序
2. 立即跳转到FIQ向量。
3. 在FIQ处理结束后返回到数据中止处理程序。

# 第五章、ARM汇编语言编程

## Chapter1、数据处理指令

功能：完成寄存器数据的**算术和逻辑操作**

原则：操作数32位，来源寄存器/立即数，结果存放寄存器，三地址模式

* 算数操作：

ADD r0,r1,r2 ；r0:=r1+r2

ADC r0,r1,r2 ；r0:=r1+r2+C

SUB r0,r1,r2 ；r0:=r1-r2

SBC r0,r1,r2 ；r0:=r1-r2**+C-1**

RSB r0,r1,r2 ；r0:=r2-r1

RSC r0,r1,r2 ；r0:= r2-r1**+C-1**

* 按位逻辑操作

AND r0,r1,r2 ；r0:=r1 and r2

ORR r0,r1,r2 ；r0:=r1 or r2

**EOR** r0,r1,r2 ；r0:=r1 **xor** r2

**BIC** r0,r1,r2 ；r0:=r1 **and not** r2

* 寄存器传送操作

MOV r0,r2 ；r0:=r2

MV**N** r0,r2 ；r0:=**not** r2

* 比较操作 不产生输出，只影响CPSR标志位（条件码）

CMP r1,r2 ；根据**r1-r2**的结果设置cc

CMN r1,r2 ；根据**r1+r2**的结果设置cc

TST r1,r2 ；根据**r1 and r2**的结果设置cc

TEQ r1,r2 ；根据**r1 xor r2**的结果设置cc

* 立即数操作

通过“#”表示，编码方式，立即数＝（0~255）×22n (0≤n≤12)

* 移位操作

ASR #n 算术右移n位（**补最左边一位**）

LSL #n 逻辑左移n位（**补零**） LSR #n 逻辑右移n位（**补零**）

ROR #n 循环右移n位 RRX 带扩展的循环右移1位（带进位）

## Chapter2、数据传送指令

功能：完成**寄存器**数据与**存储器**数据的传送

* 单寄存器Load/Store指令

支持寻址方式：

寄存器间接寻址、基址偏移寻址、基址变址寻址

LDR r0,[r1] ；r0:=**r1存放的地址所指向的数据**（寄存器间接寻址）

STR r0,[r1] ；mem32[r1]:=r0

前变址寻址

LDR r0，[r1，＃4] ； r0:＝mem32**[r1＋4]**

LDR r0，[r1，＃4]**！** ； r0:＝mem32[r1＋4]，**r1:＝r1＋4**

感叹号实现基址寄存器**自动变址**，表示**将最后的地址写回到r1中。**

后变址寻址

LDR r0，[r1]，＃4 ； r0:＝**mem32[r1]**，r1:＝r1＋4

不能使用感叹号，**通过立即数来实现基址寄存器的自动变址**

* 多寄存器Load/Store指令

如：

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

**传输次序按照r0-r15增大顺序、**可以用一条指令同时存取多个寄存器**、**寻址模式更加有限

最适用于：

堆栈寻址(F: full; E: empty; A: ascend; D: download)空/满，递增/递减

* 满递增：堆栈**向上增长**，堆栈指针指向**内含有效数据项的最高地址**。如LDMFA、STMFA等；
* 空递增：堆栈**向上增长**，堆栈指针指向堆栈上的**第一个空位置**。指令如LDMEA、STMEA等；
* 满递减：堆栈**向下增长**，堆栈指针指向**内含有效数据项的最低地址**。如LDMFD、STMFD等；
* 空递减：堆栈**向下增长**，堆栈指针向堆栈下的**第一个空位置**。指令如LDMED、STMED等。

块拷贝寻址

多寄存器传送指令用于将一块数据从存储器的某一位置拷贝到另一位置

STM(I/D)(A/B)🡪(I: increase; D: download; A: after; B: before)向上/下增长，在之前/之后变址

## Chapter3、控制流指令

功能：使指令执行切换到不同的地址

* 永久转移——转移指令（B）
* 使处理器由顺序执行指令转移到Label处，利用的寻址方式是PC相对寻址。

B Label

…

Label …

* 转移可以有条件进行－条件转移：通过**CPSR条件码**判断转移条件是否满足。
* 保存返回地址以恢复原来的执行顺序（BL）

在转移到子程序的同时，**保存返回地址r14**

BL SUBR ；转移到SUBR

… **；返回到这里**

SUBR … ；子程序入口

MOV pc,r14 ；返回

注意：如果存在子程序嵌套，则必须先保存好r14！

* 软件中断指令（SWI）

处理器**进入管理（监控）模式**，从固定向量地址0x0000000**8**取指

通过特定的监控调用才能**访问系统级函数**，实现受控的保护机制

这些函数包括对外界硬件寄存器的访问，大量的I/O操作

## Chapter4、指示符与伪指令

* 指示符（ppt41页）
* 符号定义指示符

定义ARM汇编程序中的**变量、对变量赋值**以及**定义寄存器的别名**等操作。

* 数据定义指示符
* 汇编控制指示符
* 伪指令

不属于ARM指令集中的指令，是为了编程方便而定义的。伪指令可以像其它ARM指令一样使用，但在编译时这些指令将被等效的ARM指令代替。

* ADR**小范围地址读取**伪指令

将基于PC**相对偏移的地址值**或基于寄存器**相对偏移的地址值**读取到寄存器中。

如：LOOP MOV R1，#0xF0

ADR R2，LOOP ；将LOOP的地址放人R2

* ADRL**中等范围的地址读取**伪指令

比ADR有更大的取地址范围

* LDR**大范围的地址读取**伪指令

用于加载**32位的立即数**或**一个地址值**到指定寄存器

如：LDR R0，**=**0x12345678 ；加载32位立即数0x12345678

* NOP伪指令

## Chapter5、汇编语言的程序结构

注释用；打头，ENTRY伪指令标识程序的入口点，接下来为指令序列。结尾用到END伪代码，每一个汇编程序段都必须有一条END伪指令，指示代码段的结束。

以程序段为单位组织代码。段是相对独立，具有特定的名称。可以分为**代码段**和**数据段**，代码段的内容为执行代码，数据段存放代码运行时需要用到的数据。一个汇编程序**至少应该有一个代码段**。

# 第六章、ARM流水线组织

## Chapter1、3级流水线组织

ARM7系列（ARM v5TE版本）采用

取指（放入指令流水线）🡪译码🡪执行（占有数据通路）

一条指令有**3个周期**执行时间，但**吞吐率**是每个周期一条指令

流水线观察：

* 所有指令都占用数据通路一个或多个相邻周期
* 在指令占用数据通路的每一个周期，都在前面的相邻周期占有译码逻辑
* 在第一个**数据周期**，每条指令为**下下条指令**发出取指信号
* 转移指令**清空和重填指令流水线**
* 保存处理器状态的寄存器堆： 两个读端口一个写端口，访问寄存器
* 桶式移位器： 把一个**操作数移位**或循环移位
* ALU
* 地址寄存器和增值器 选择和保存所有的**存储器地址**，并在需要时产生顺序地址
* 数据寄存器 保存传送到存储器或从存储器取出的数据（相当于**缓存**）
* 指令译码器和控制逻辑

PC必须在当前指令**之前**计数，增加8个字节（为**下下条**指令取指），试图通过r15访问PC需要考虑流水线的情况

Chapter2、5级流水线组织

ARM9系列（ARM v6版本）

取指🡪译码🡪执行🡪**缓冲/数据**（访问数据存储器或缓冲一周期）🡪**回写**

程序的执行时间

**Tprog=Ninst×CPI/Fclk**

N：指令数 CPI：每条指令的平均时钟周期数 F：处理器的时钟频率

因此，提高性能的方法：

减小指令数量———🡪优化编译器

提高时钟频率表

增加流水线级数，从而简化每一级的逻辑（即减少每个时钟周期内必须完成的最大工作量）

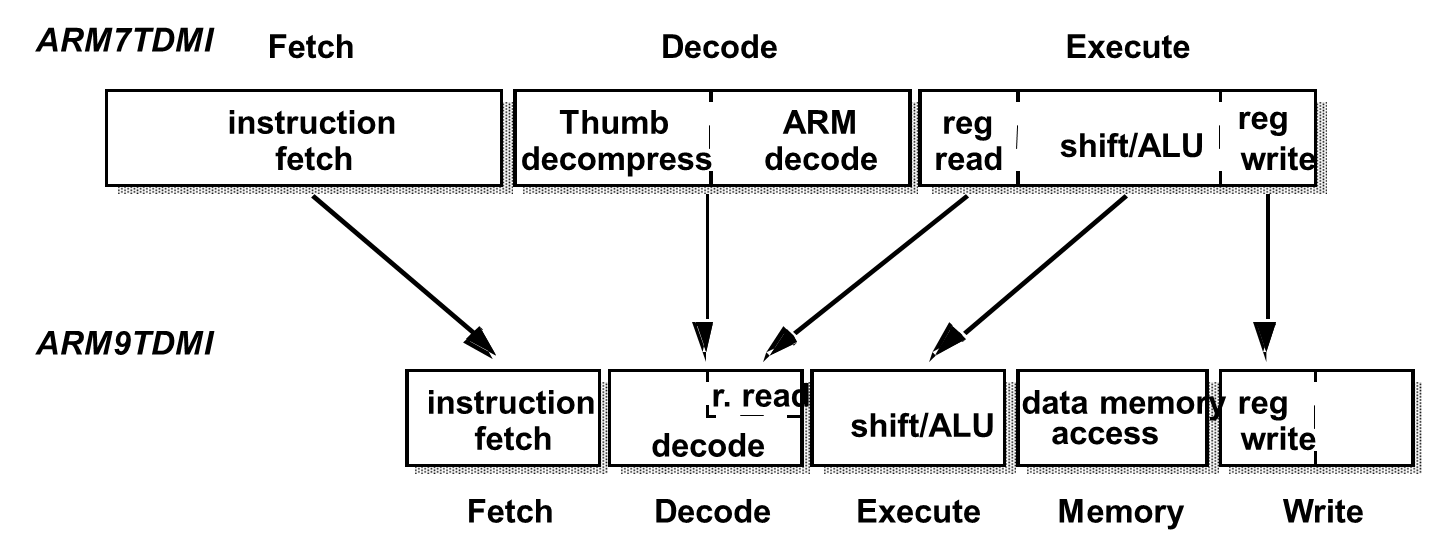
减小CPI

根本问题——**存储器瓶颈**（指令和数据在同一存储器，性能受到存储器带宽限制）

改进方法：一个时钟周期给出多于32位数据、哈佛结构

增加级数后🡪可能产生后一条指令需要前一条指令的结果，然而前一条指令结果还未回写至寄存器，因此需要**前推通路**，从而可以使得结果能在**级间**传送。

**三级流水线和五级流水线对比**



## Chapter3、ARM数据通路组织结构分析

需要的时钟周期主要取决于要访问寄存器的次数

* 数据处理指令

需要两个操作数、一个时钟完成所有**数据通路**操作

* 数据传送指令

STR需要两个时钟周期（计算地址，存储数据及自动变址）

**LDR**需要**三个**时钟周期

**地址寄存器**是处理器数据通路与外界存储器之间的流水线寄存器

* 转移指令

需要三个时钟周期（计算地址、保存返回地址、填充流水线，修正r14链接地址）

# 第七章、ARM指令集

## Chapter1、ARM处理器寻址方式

有**9种**基本寻址方式

(1).寄存器寻址；

操作数的**值**在寄存器中，指令执行时**直接取出寄存器值**来操作。

(2).立即寻址；

指令中的操作码字段后面即是**操作数本身**（立即数）

(3).寄存器偏移寻址；

将第二个寄存器中的操作数移位后再进行操作

(4).寄存器间接寻址；

寄存器中保存的是数据的地址 LDR R0,**[R2]** 用中括号括起来表示

(5).基址偏移寻址；

将基址寄存器的内容与指令中给出的偏移量相加，形成操作数的有效地址 LDR R2,**[R3,#0x0C]**

(6).多寄存器寻址；

一次传送多个寄存器值（**连续的寄存器用-连接，否则用逗号**，如{r2-r7,r10}）

(7).堆栈寻址； (8).块拷贝寻址；

(9).相对寻址；

由程序计数器PC提供基准地址，指令中的地址码字段作为偏移量，两者相加后得到的地址即为操作数的有效地址。（如调用子程序、跳转等）

## Chapter2、ARM指令集介绍

# 第十章、ARM存储器借口及AMBA总线

## Chapter1、ARM存储器接口

时钟与时钟控制

* MCLK – 输入
  + 处理器工作的时钟
  + 静态设计的ARM通过延长时钟周期来**访问慢速的设备**
* nWAIT –输入（等待时钟，用于照顾系统中最慢的路径，如rom）
  + 在ARM内部与MCLK相与
  + 必须**在MCLK为低的相位阶段改变**
  + 容许该信号从一个周期扩展到另一个周期，**延长总线访问周期**
* ECLK – 输出
  + 核心逻辑的时钟的输出。
  + 在正常和调试状态下反映内部时钟
* ph1 & ph2 – 内部信号
  + 双相位非覆盖的内部时钟
  + 处理器内部工作周期

数据总线

* + 32位双向/单向（由BUSEN0/1决定）；
  + 字节、半字、字访问；
  + 读数据必须有效且稳定到相位2结束，写数据在相位1改变，保持稳定贯穿相位2

地址总线

* + 32位（4G字节）寻址能力
  + 默认时序
    - 在**前一周期的相位2阶段**变为有效，保持稳定贯穿当前周期的相位1阶段
    - 流水线地址
  + 地址时序可以通过APE（或ALE）移位。

**周期类型**（由nMREQ和SEQ决定，提前一个周期建立）

* 非连续 (N)：在接下来的周期中的地址与**前一个地址无关**

如：对于基于DRAM的系统的**初始**的行访问

* 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，即：**处理器将在下一个周期执行内部操作**。不驱动数据总线，驱动地址总线

如：乘操作、寄存器特定的移位操作、从存储器加载数据之后的回写操作

* 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

可以**在内部周期期间启动对地址位置**的访问，在连续(S)周期期间，访问可以结束

**可以驱动**数据总线，从而中间不需要设置等待

## Chapter2、AMBA总线

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，支持突发数据数据传送模式，**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**。



AHB总线和ASB总线有下列不同的特点：

* AHB总线支持**分时处理**
* 使用**单一时钟沿**控制所有操作，有利于综合和设计验证
* 使用**中心多路器**总线方案，而不是三态驱动的双向总线
* 支持更宽的**64位或128位**数据总线配置

AHB总线包括

主单元（可以任何时刻使用总线，一或多个），从单元（响应并反馈），仲裁器，译码器

AHB工作过程：

**地址传送**阶段：一个时钟周期，所有从单元在HCLK采地址

**数据传送**阶段：一到多个时钟周期，HREADY低传输，高的时候结束

APB组成

**APB桥**：**唯一**的主单元，是**AHB/ASB的从单元**，连接APB和ASB/AHB

APB从单元

# 第十三章、ARM JTAG测试与调试结构

## Chapter1、JTAG边界扫描测试结构

JTAG：

**联合测试行动组**，开发了用于**印刷版测试标准**，IEEE1149，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

基本思想：

在靠近芯片的输入输出管脚上增加一个移位寄存器单元（**边界扫描寄存器）**，当芯片处于测试状态的时候，这些边界扫描寄存器可以将芯片和外围的输入输出隔离开来，实现对芯片输入输出信号的观察和控制；在正常的运行状态下，这些边界扫描寄存器对芯片来说是透明的，边界扫描（移位）寄存器单元可以相互连接起来，在芯片的周围形成一个**边界扫描链**（是一个数据寄存器）。

TAP控制器（**测试访问端口**）

控制边界扫描链，是个由TMS控制状态转换的状态机，是一个通用端口，可以访问芯片提供的所有数据寄存器（DR）和指令寄存器（IR）。

**数据寄存器（DR）**

* + 器件ID寄存器：读出固定在**芯片内部的标识码；**
  + 旁路（Bypass）寄存器：将TDI经过1个时钟周期的延迟连接到TDO，**可以快速访问同一电路板上测试环路的另一个器件**（即跳过本器件）；
  + 边界扫描寄存器：截取核逻辑与引脚之间的**所有信号**
  + 芯片上还可以有些其他寄存器用于**其他功能**的测试

**指令**

* JTAG测试系统的过程是**向指令寄存器送入指令**，**使用数据寄存器进行测试**
* 指令分为**公开**和**私用**
* 公开指令已经在IEEE1149.1定义且用于**通用测试**
* 私用指令用于**片上专用测试**（兼容器件必须支持的公开指令）

**最小集的公开指令**（兼容器件必须支持的公开指令）

* **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**

五个信号接口（四输入一输出）

TCK（**独立于系统时钟**的测试时钟） TMS（测试模式选择信号）

TDI（测试数据输入） TRST（测试复位输入）

TDO（测试数据**输出**）

通过TAP接口，对数据寄存器（DR）进行访问的一般过程是：

**通过指令寄存器**（IR），**选定**一个需要访问的数据寄存器，把选定的数据寄存器连接到TDI和TDO之间，由TCK驱动，通过TDI把需要的数据输入到选定的数据寄存器当中去；同时把选定的数据寄存器中的数据通过TDO读出来

PCB测试

* JTAG测试的主要目的是**测试印制板上走线与焊盘之间的连接**
* 支持JTAG测试接口的器件可以通过**EXTEST指令**
* 不支持JTAG测试接口的器件还需要使用**探针技术**（封装减小导致探针技术不适用）
* 探针与JTAG接口相结合可以降低制造产品测试仪的成本及难度

VLSI测试（JTAG能用，但并不是通用解决方案）

* **高复杂度IC芯片**通过非常昂贵的测试仪进行测试；
* JTAG测试电路以串行方式工作，**不能高速**的将测试向量加到核逻辑上，**不能以器件正常速度测试性能**；
* JTAG在VLSI产品测试中作用：
* IC**内部电路功能测试**（INTEST指令）
* 较好的控制IC引脚用于**参数测试**（如引脚的物理参数等）（EXTEST指令）
* 用于**内部扫描路径**，提高从引脚难以访问的内部节点的可控性核可观察性
* 用于**访问片上调试功能（EmbeddedICE）**，不需要额外引脚，也不会干扰系统的功能
* 提供了基于宏单元设计的功能测试方法

宏单元测试

* 系统芯片使用**大量的复杂、已设计好的宏单元**（如ARM处理器核本身）
* 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（同样有速度过低的缺点）

## Chapter2、ARM调试结构

**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，**通常为一个相同的芯片或更多引脚的变型芯片**）**替代目标系统中的处理器，并包含缓冲器以实现硬件跟踪。

几种常见的调试方法：

* 指令集模拟器：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

JTAG与Angel

* JTAG调试：协议转换器**解释**上位机传送过来的命令，通过JTAG**控制ARM执行**。
* Angel调试：协议转换器可以**直接做为目标板的Firmware的一部分**。**直接执行**从**宿主机**传送过来的调试命令；并回送相应的数据。（宿主机调试器需要Angel协议）
* Angel可以节省专门的JTAG仿真器，但是**需要软件**，或者是嵌入式**操作系统的支持**，**做不到完全的实时仿真**。
* JTAG仿真是通过**硬件**和**控制ARM的EmbeddedICE实现**的，可以**做到实时仿真**

ARM调试硬件

ICE方法不能适用于复杂的包含多个宏单元的系统芯片

软件模型仿真具有局限性，不能仿真整个软件系统运行情况与精确描述所有实时约束

ARM调试硬件采取了**EmbeddedICE模块**提供了类似ICE的调试工具

**EmbeddedICE模块：**

能够用较低的成本使处理器停止实现断点、观察点的功能。

* 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）

# 第十四章、ARM处理器核系列介绍

## Chapter1、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

T—Thumb16位指令；

D－支持片上调试（debug）；

M－增强型乘法器（Multiplier），较高的性能和64位结果；

I－EmbeddedICE硬件以支持**片上断点和观察点**

三级流水线，15个协处理器，每个有16个专用寄存器，大小不限于32位，采用L/S体系

典型应用：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）

进一步提高：

增加cache，使用更复杂的更高性能水平的ARM核

## Chapter2、ARM8

相对于ARM7改进：

* **增加时钟速率**：简化每级流水线的逻辑，增加级数，**五级流水线**
* **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

其余特点：

* 保留统一存储器，但可获得双倍带宽
* 使用32位总线，节省芯片面积

应用：**通用**处理器核，没有特定优化

## Chapter3、ARM9TDMI

相对于ARM8

使用分开的指令与数据存储器端口改善CPI（**哈佛结构**）

**硬件直接译码**ARM指令核Thumb指令进行，流水线内没有时间对其进行软译码

可以在异常时设置断点

其它特点：

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提供了系统级的调试特性，支持硬件单步调试。

## Chapter4、ARM10TDMI

相对于ARM9：

两倍性能

增加时钟速率：采用特别方式优化每一级，在译码前**增加发射一级**

降低CPI：主要改进来源：存储器带宽🡪采用64位存储器

# 第十六章、存储器管理单元及保护单元

## Chapter1、存储器管理的一般原理

单核下的多任务是由于程序间的快速切换，而程序切换是通过存储器管理单元支持的。

两种基本方法：段式管理、页式管理

段式管理：允许应用程序将其存储器视为**一系列的段**，每段包含一些特定类别的信息。

每次存储器访问都向MMU提供一个段选择和一个逻辑地址（相对段基地址的偏移）；

各段之间是独立的，因此可能产生**碎片化**的问题，通过系统拼接成大片虽然可以解决，但**效率低；**

ARM体系结构**不支持**段式管理；



**段选择**决定段**基址地址**，通过限制判断**逻辑地址**是否超出限制，然后得出**物理地址**或**访问失败**

页式管理：逻辑和物理地址分为固定大小的页，通常几KB，逻辑和物理页之间的关系保存在页表中。

由于单页表将耗费很大的空间，通常采取**多级页表**（至少两级）；

如：

地址的高10位用于确定**一级页表目录中相应的二级页表**，次高10位确定**页表项，即物理页数**

ARM**支持**页式管理；

## Chapter2、ARM存储器管理单元（MMU）

MMU通过系统控制**协处理器CP15**实现，控制片上Cache、存储器管理或保护单元、写缓冲器、预取缓冲器等操作

**MMU结构**

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

**存储器粒度**：段（1MB）、大页（64KB/16KB子页）、小页（4KB/1KB子页）、微页（1KB）

页域

ARM MMU特有，一组具有特定访问权限的段/页，允许多个不同过程用同一个转换表执行，同时不同程序之间具有保护功能。提供了一种**更为轻便的过程转换机制**，不必使每个过程都有自己的转换表。

**转换过程**

转换从第一级开始，转换**基址寄存器[31:14]**结合**虚拟地址的位[31:20]**形成一个存储器地址，访问**第一级描述符**



第一级描述符可能是**一个段的描述符**，也可以是**第二级页表的指针**，取决于最低两位：

00——产生转换故障的描述符、01——第二级粗页表指针

10——段描述符、11——第二级精细页表指针

若转换为段，访问权限检查通过后，则存储器地址由**段描述符的位[31:20]**结合**虚拟地址的位[19:0]**级联构成



段转换

小页转换同理可得



访问权限

* MMU可能产生对齐、转换、页域以及权限故障
* 外部存储器系统可能产生Cache行取、不可Cache或不可缓冲访问以及转换表访问故障
* **所有故障都称为中止**，由处理器作为预取或数据**中止异常**来处理
* 数据访问故障使故障状态寄存器（CP15寄存器5）和故障地址寄存器（CP15寄存器6）更新，以提供故障产生原因和位置的信息
* 指令访问故障只会在指令执行时产生异常，故障地址可由返回的在R14链接寄存器中的地址推断出来



## Chapter3、ARM保护单元

定义了不同存储器区域的各种保护及Cache功能。通过修改保护单元寄存器CP15实现

保护单元的结构

* 保护单元允许将ARM的**4GB寻址空间映射为8个区域**
* 每个区域都有可编程的起始地址及大小、可编程的保护及Cache性质
* 这些区域可以重叠，对**重叠区域的寻址有固定的优先级**

区域定义

* 通过写**CP寄存器6**可以定义8个区域中每个起始地址及大小（大小为倍数关系，即可不需要加减法，提高速度）
* 区域**最小为4KB，最大为4GB**