《嵌入式系统原理与应用技术》

袁志勇 王景存
章登义 刘树波

北京: 北京航空航天大学出版社, 2009.11

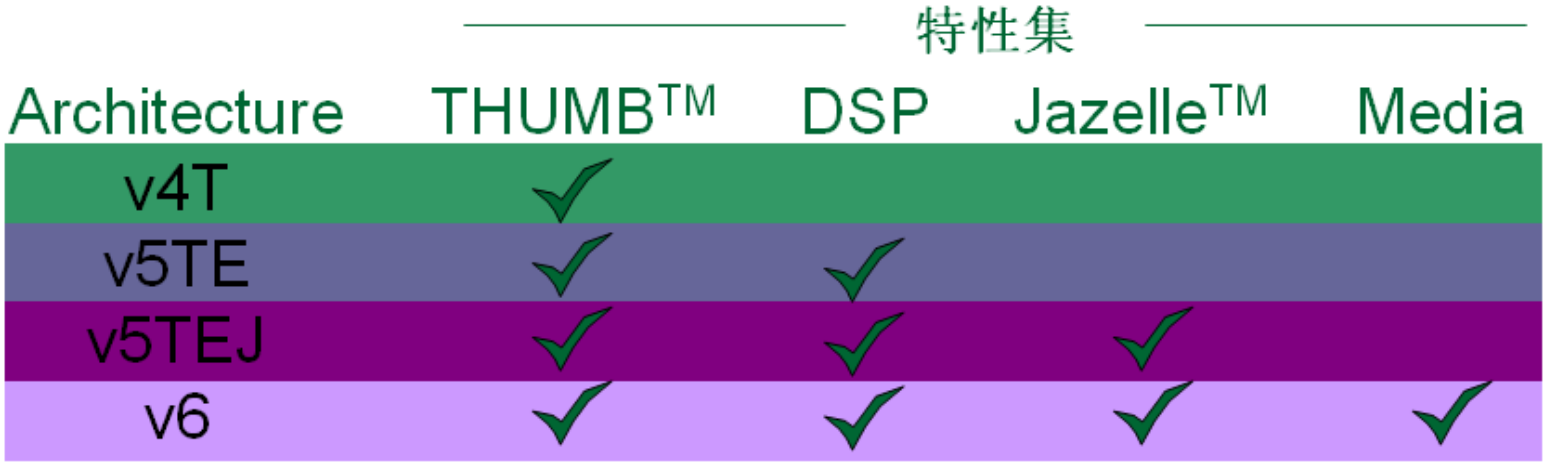
PPT教学课件

第2章 ARM9体系结构

2.1 ARM9嵌入式微处理器

2.1.1 ARM9的结构特点

ARM 架构(Architecture)



- 不断创新以提升性能
  - **THUMB™: 35%** 代码压缩
  - **DSP** 扩充: 定点 **DSP** 的高性能
  - **Jazelle™:** **Java** 性能显著提高, 最高到**8**倍
  - **Media** 扩充: 音频/视频性能显著提高, 最高到**4**倍
- 向下兼容以保护软件投入

2.1.1 ARM9的结构特点

1．ARM9处理器简介

ARM9系列嵌入式微处理器主要有ARM9 TDMI、ARM9 E-S等系列。

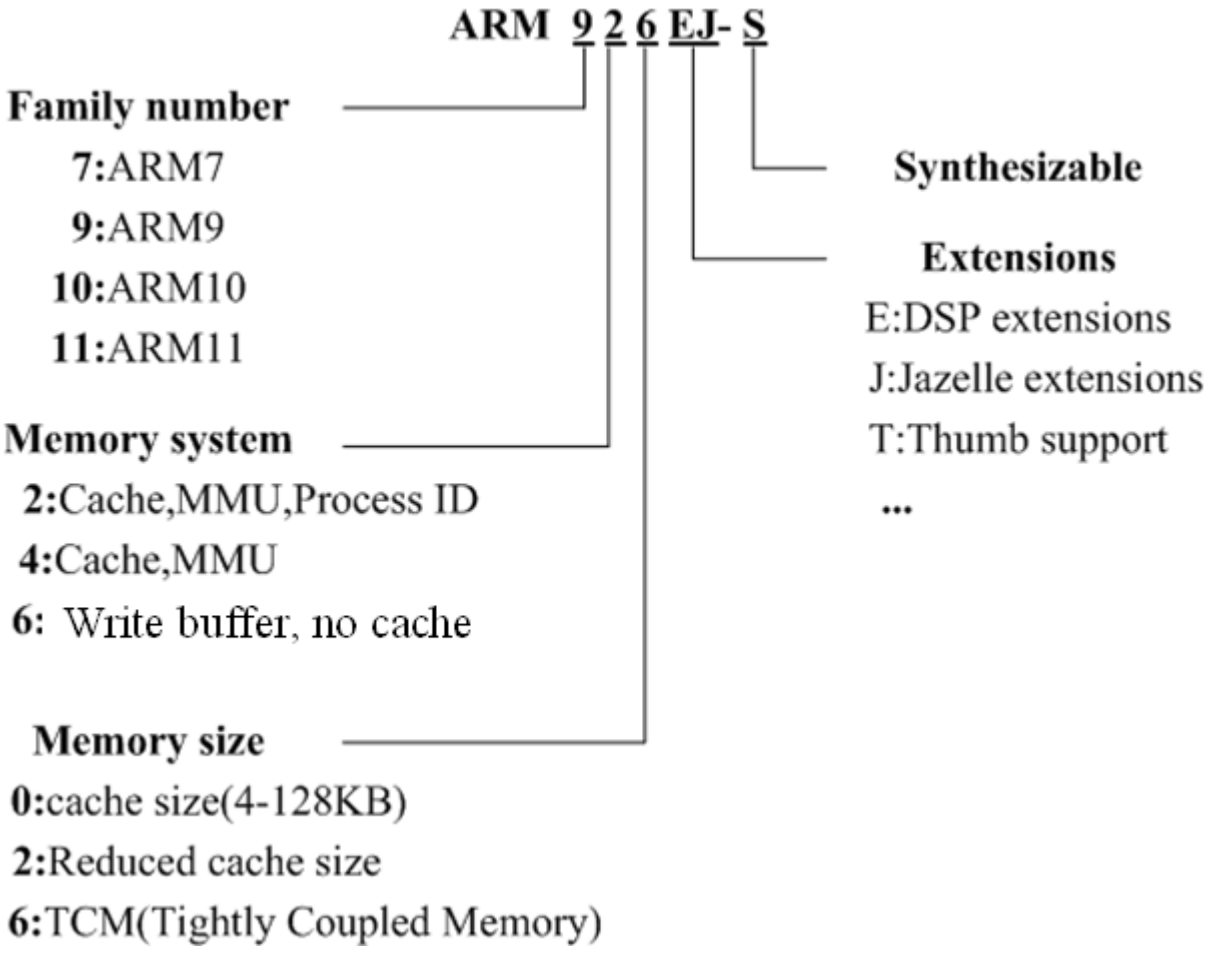


图2.1 ARM嵌入式微处理器命名规则示意图

表2.1 ARM系列微处理器扩展命名符号的含义

标志	含义	补充说明
T	16位Thumb指令集	Thumb指令集版本1：ARMv4T Thumb指令集版本2：ARMv5T Thumb-2：ARMv6T (支持16位Thumb压缩指令集)
D	片上调试	支持片上Debug，允许处理器响应调试请求暂停
M	支持增强型乘法器	32位乘32位得到64位
I	嵌入式ICE部件	提供片上断点和调试点的支持
E	增强型DSP指令	增加了DSP算法处理器指令：16位乘加指令，饱和的带符号数的加减法，双字数据操作，Cache预取指令
J	Java加速器Jazelle	提高Java代码的运行速度；与无加速器相比，最高可达到8倍
S	可综合	提供VHDL或Verilog HDL硬件描述语言设计文件

ARM系列有T变种、M变种，E变种、J变种和SIMD*变种。

A、**T变种**（Thumb指令集）

Thumb指令集是将ARM指令集的一个子集重新编码。ARM指令长度为32位，Thumb指令长度位为16位。

当数据总线宽度小于32位时，使用Thumb指令集。代码尺寸小，只占ARM编译结果的65％左右，可以节省存储器空间。

与ARM指令集相比，Thumb指令集有以下局限。

(1) Thumb指令通常需要更多的指令。因此，ARM指令集更为适合对系统运行时间要求苛刻的应用场合。

(2)Thumb指令集没有包含进行异常处理时需要的一些指令，Thumb指令需要和ARM指令配合使用。

使用ARM指令集还是使用Thumb指令集，需要从存储器开销和性能要求两方面加以权衡考虑。

B、**M变种**（长乘法指令）

增加了两条用于进行长乘法操作的指令。

一条用于32位整数乘以32位整数，生成64位整数的长乘法操作；另一条用于32位整数乘以32位整数，然后再加上32位整数，生成64位整数的长乘加操作。

C、**E变种**（增强型DSP指令）

E变种包含了一些附加的指令，增强处理器对一些典型的DSP算法的处理性能。

(1)几条实现16位数据乘法和乘加操作的指令。

(2)实现饱和的带符号数的加减法操作的指令。所谓饱和的带符号数的加减法操作是在加减法操作溢出时，结果并不进行**返转**(Wrapping around)，而是使用最大的整数或最小的负数来表示。

(3)进行双字数据操作的指令，包括双字读取指令LDRD，双字写入指令STRD和协处理器的寄存器传输指令MCRR／MKRC。

D、**J变种**（Java加速器Jazelle）

ARM Jazelle技术提供了Java加速功能，可以得到比普通 Java虚拟机高8倍的性能，功耗降低了80%，在一个单独的处理器上同时运行Java应用程序、已经建立好的操作系统、中间件以及其他的应用程序。

E、SIMD*变种（ARM媒体功能扩展）

SIMD（Single Instruction Multiple Data：单指令多数据流）：是能够复制多个操作，高效完成多媒体应用等数据密集型运算，如语音和图像编码器。

ARM9采用ARMv4T（Harvard)结构，五级流水处理以及分离的Cache结构，平均功耗为0.7mW/MHz。时钟速度为120MHz-200MHz，每条指令平均执行1.5个时钟周期。其中，ARM920T、ARM940T和ARM9E为含Cache的CPU核。性能为132MIPS（120MHz时钟，3.3V供电）或220MIPS（200MHz时钟）。

ARM11是基于ARMv6架构建成的。基于ARMv6架构的处理器包括ARM1136J(F)-S，ARM1156T2(F)-S，以及 ARM1176JZ(F)-S。ARMv6是 ARM进化史上的一个重要里程碑：存储器系统加入了很多新特性，单指令多数据流（SIMD）指令也是从v6开始首次引入的。经过优化的Thumb-2指令集。

基于从ARMv6开始，扩展出了ARMv7架构。在ARMv7版本中，内核架构首次从单一款式变成3种款式。

① 款式A：设计用于高性能的“开放应用平台”--越来越接近电脑了。

② 款式R：用于高端的嵌入式系统，尤其是那些带有实时要求的嵌入式系统。

③ 款式M：用于深度嵌入的单片机/MCU风格的系统中。

A\R\M 3种款式：

① 款式A（ARMv7-A）：支持大型嵌入式操作系统，比如Symbian、Linux、 Windows CE和智能手机操作系统(微软Windows Mobile、GOOGLE Android等)。这些应用需要高的处理性能，并且需要硬件MMU实现的虚拟内存机制，还基本上会配有Java支持，有时还要求一个安全程序执行环境。如高端手机和手持仪器，电子钱包以及金融事务处理机。Cortex－A8

② 款式R（ARMv7-R）：硬实时且高性能的处理器。目标是高端实时市场。处理器要很强大，还要极其可靠，对事件的反应也要极其敏捷。如高档轿车的组件，大型发电机控制器，机器手臂控制器等。 Cortex－R4

③ 款式M（ARMv7-M）：针对单片机/MCU，实时控制系统，低成本、低功耗、极速中断反应以及高处理效率。 Cortex－M3

ARM9处理能力的提高是通过增加时钟频率和减少指令执行周期实现的。

（1）时钟频率的提高

ARM9采用了五级流水线。在同样的加工工艺下，ARM9 TDMI处理器的时钟频率是ARM7 TDMI的2倍左右。

（2）指令周期的改进

处理器性能提高的幅度依赖于代码执行时指令的重叠。

① load指令和store指令

指令周期数改进最明显的是load指令和store指令。

② 互锁(interlock)技术

当指令需要的数据因为以前的指令没有执行完，将产生管道互锁。管道发生互锁时，硬件将停止该指令的执行，直到数据准备就绪为止。

③ 分支指令

ARM9和ARM7的分支指令周期相同。

2．ARM9体系结构的五级流水线

ARM7的三级流水线：

(1)取指：从程序存储器中取指令，放入指令流水线。(占用存储器访问操作)

(2)译码：指令译码。(占用译码逻辑)

(3)执行：执行指令/读写REG。(占用ALU及数据路径)

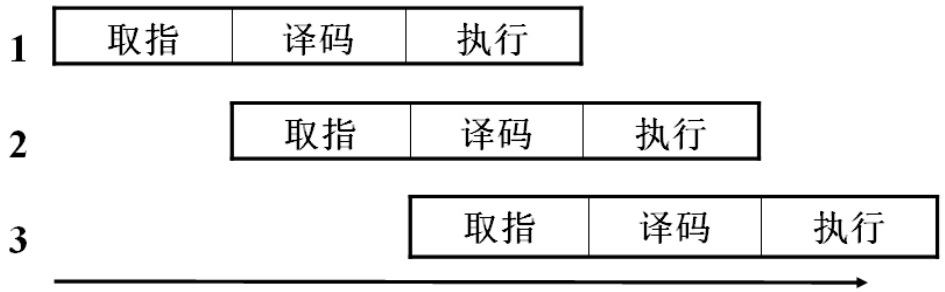
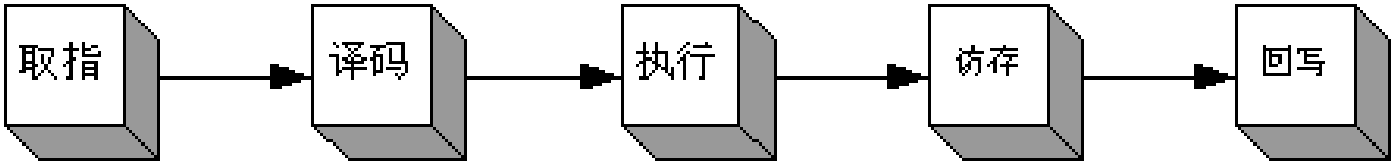


图 ARM单周期指令的3级流水线操作

PC值如何计算？

- **PC 指向处于读取级的指令地址，而不是处于执行级的指令地址**
- **PC=当前执行指令地址+8**

下图是ARM9体系结构的五级流水线：



(1)取指：从存储器中取出指令(fetch)，并将其放入指令流水线。

(2)译码：对指令进行译码(dec)。

(3)执行：执行运算ALU(exe)

(4)访存(缓冲/数据)：如果需要，则访问数据存储器（acc mem）；否则ALU的结果只是简单地缓冲1个时钟周期，以便所有的指令具有同样的流水线流程。

(5)回写：将指令产生的结果回写到寄存器(wtbk res)，包括任何从存储器中读取的数据。

3．AMBA总线接口

ARM嵌入式微处理器使用的是AMBA（Advanced Microcontroller Bus Architecture）总线体系结构。定义了三种总线：

●AHB（Advanced High-performance Bus）：用于连接高性能系统模块。它支持突发数据传输方式及单个数据传输方式；另外，它还支持分离式**总线事务处理**。

● ASB总线（Advanced System Bus）：用于连接高性能系统模块，它支持突发数据传输模式。

● APB总线（Advanced Peripheral Bus）：是一个简单接口，支持低性能的外围接口。

ASB总线是旧版的系统的总线，而新版的AHB总线增强了对性能、综合及时序验证的支持。

4.ARM9的结构特点

ARM9 TDMI处理器内核的符号含义：

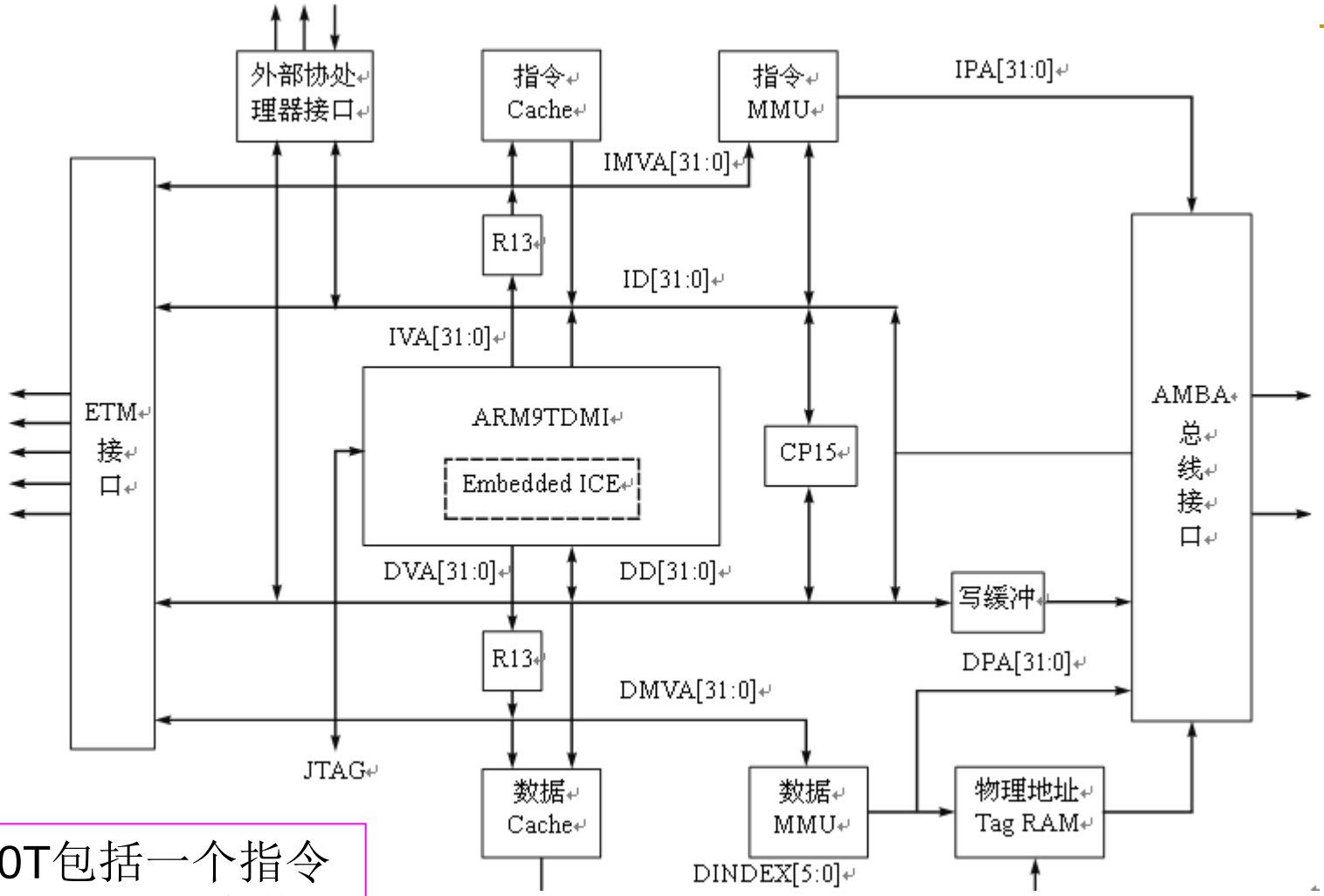
●ARM9：采用哈佛结构，ARMv4T指令集，五级流水线处理以及分离的Cache；

● T：支持16位宽度的Thumb压缩指令集；

● D：支持片上Debug，允许处理器响应调试请求暂停；

● M：支持增强型乘法器，可生成全64位的结果；

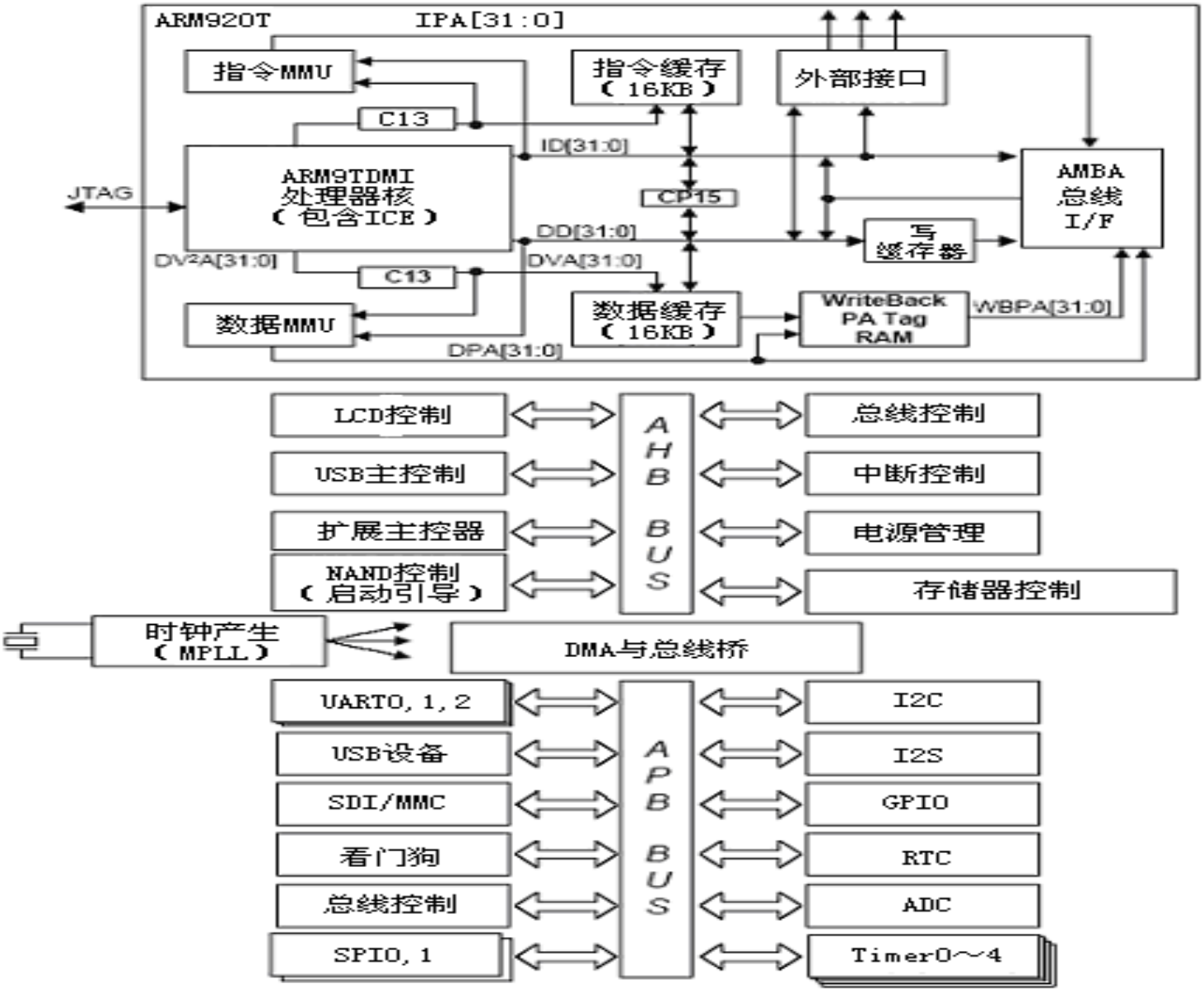
● I：嵌入式ICE部件，提供片上断点和调试点的支持。



ARM920T结构示意图

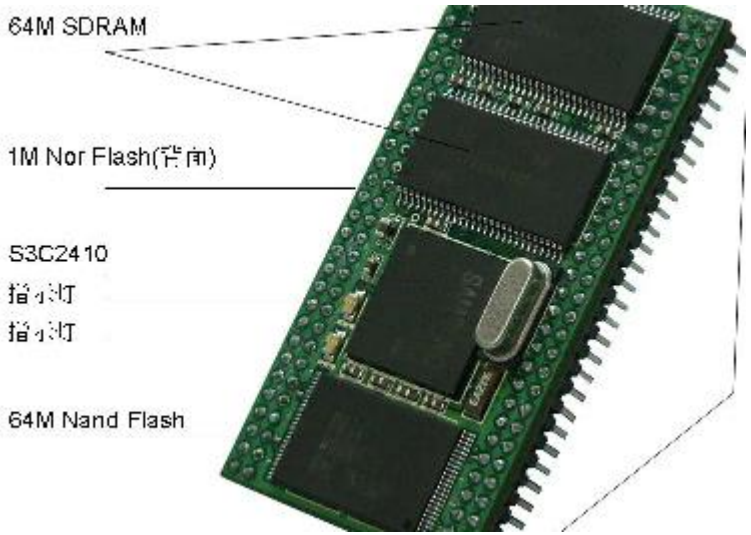
ARM920T包括一个指令缓存(ICache)，一个数据缓存(DCache)，一个**写缓冲**器与一个物理地址(PA) **Tag RAM**来减轻主存储器带宽与等待时间对性能的影响。

CP15协处理器主要是处理MMU和CACHE功能的实现，在ARM中任何涉及到虚拟存储地址到物理地址的转换都是通过CP15协处理器硬件来实现的。



以ARM920为内核**S3C2410芯片系统结构框图(P38)**

- ARM920T核内部具有指令缓存和数据缓存，允许处理器同时进行取指和读写数据操作。数据可以是：8位（字节）、16位（半字）、32位（字）。字必须是4字节边界对准(字对齐)，半字必须是2字节边界对准(半字对齐)。



ARM9 E-S内核：

●32位定点RISC处理器，改进型ARM / Thumb代码，增强型乘法器设计，支持实时调试；

●片内指令和数据SRAM，而且指令和数据的存储器容量可调；

●片内指令和数据高速缓冲器（cache）容量从4K字节到1M字节；

●设置保护单元（protection unit），非常适合嵌入式应用中对存储器进行分段和保护；

●采用AMBA AHB总线接口，为外设提供统一的地址和数据总线；

●支持外部协处理器，指令和数据总线有简单的握手信令支持；

●支持标准基本逻辑单元扫描测试方法，并且支持内建自测试技术（BIST: Built-In-Self-Test）；

●支持嵌入式跟踪宏单元（ETM，Embedded Trace Macrocell），支持实时跟踪指令和数据。

2.1.2 ARM9指令集特点

ARM9的指令集是依据RISC原理而设计的，指令集和相关译码机制较为简单。传统的微处理器体系结构中，指令代码的宽度（位数）和数据的宽度（位数）通常是相同的，而ARM9的指令系统中有一种16位的指令集（Thumb指令集）。通常情况下，16位体系结构与32位体系结构比较而言，在操作32位数据时的性能大约只有32位体系结构的一半，且有效的寻址空间相对较小。而Thumb指令集在32位体系结构中实现了16位指令集，以提供比16位体系结构更高的性能和更高的代码密度。

2.1.2 ARM9指令集特点

1．ARM指令集概况

ARM指令集为32位指令集，可以实现ARM架构下所有功能。

2．Thumb指令集概况

Thumb指令集实现的功能只是32位ARM指令集的子集，它仅仅把常用的ARM指令压缩成16位的指令编码方式，大约是标准ARM指令代码密度的两倍。在指令的执行阶段，16位的指令被重新解码，完成对等的32位指令所实现的功能。

2．ARM指令集与Thumb指令集比较

在Thumb指令集中，指令的操作数和地址仍然是32位的，但Thumb指令集为实现16位的指令长度，舍弃了ARM指令集的一些特性。

使用Thumb指令可以在代码密度方面改善大约30%，但这种改进是以代码的效率为代价的。

尽管每个Thumb指令都有相对应的ARM指令，但是，执行相同的功能，需要更多的Thumb指令才能完成。因此，当指令预取需要的时间没有区别时，ARM指令相对Thumb指令具有更好的性能。

2.1.3 ARM9工作模式

ARM9 TDMI处理器核共支持7种工作模式，它们分别是：

- 用户模式(usr)：ARM处理器正常执行程序时的处理。
- 快速中断模式(fiq)：用于高速数据传输或通道处理。
- 外部中断模式(irq)：用于通用的中断处理。
- 管理模式(svc)：操作系统使用的保护模式。
- 指令/数据访问终止模式(abt)：当数据或指令预取终止时进入该模式，可用于虚拟存储及存储保护。
- 系统模式(sys)：运行具有特权的操作系统任务时的模式。
- 未定义指令中止模式(und)：当未定义的指令执行时进入该模式，可用于支持硬件协处理器的软件仿真。

除用户模式以外，其余的所有6种模式称之为非用户模式，或**特权模式**；其中除去用户模式和系统模式以外的5种又称为**异常模式**，常用于处理中断或异常，以及需要访问受保护的系统资源等情况。

当某种异常发生时，ARM9 TDMI处理器核即进入相应的工作模式。例如，若发生了IRQ中断并响应IRQ中断，则ARM9 TDMI核将进入IRQ模式。每种工作模式下均有其附加的某些寄存器，因此，即使有异常情况发生，异常模式下的处理程序也不至于破坏用户模式的数据及状态。

在程序的执行过程中，ARM9处理器核可以随时在ARM状态和Thumb状态之间切换，并且处理器工作状态的改变不影响处理器的工作模式和相应寄存器中的内容。

ARM状态和Thumb状态之间切换的三种方法：

当操作数寄存器的状态位(位0)为1时，可以采用执行BX指令的方法，使ARM处理器从ARM状态切换到Thumb状态；

当操作数寄存器的状态位(位0)为0时，执行BX指令可以使ARM处理器从Thumb状态切换到ARM状态。

另外，在处理器进行异常处理时，将PC指针放入异常模式链接寄存器中，并从异常向量地址开始执行程序，也可以使处理器切换到ARM状态。

例2.1 用BX实现状态切换(*：第3章学完学后，回过来再看此处代码)。
汇编程序段(教材P24)。

```
                ……
                ADR R0,THUMBCODE+1 ; 将R0的bit[0]置1
                BX R0   ;   跳转,并根据R0的bit[0]实现状态切换
                CODE16                  ;16位Thumb代码
THUMBCODE  MOV R2,#2
                ……
                ADR R0,ARMCODE      ;加载ARMCODE地址到R0中
                BX R0
                CODE32                  ;32位ARM代码
ARMCODE     MOV R4,#4
                ……
```

注释:
BX{<cond>} Rm ； 带状态切换的分支指令
功能：PC=Rm&0xfffffffe, T=Rm[0]&1

2.2 ARM9存储器组织结构

ARM9体系结构采用32位长度地址，字节地址范围是0~2^{32}（16进制地址范围：x00000000~0xffffffff，存储容量：2^{32}=4GB字节）。因此，ARM9体系结构允许使用现有的存储器和I/O器件进行各种存储器系统设计。

2.2.1 大端存储和小端存储

2.2.2 I/O端口的访问方式

2.2.3 内部寄存器

2.2.1 大端存储和小端存储

ARM9体系结构可以有两种格式存储字数据，分别称为大端格式（big-endian）和小端格式（low-endian），见图2.4(a)和(b)所示。

31	...	24	23	...	16	15	...	8	7	...	0
地址X的字节			地址X＋1的字节			地址X＋2的字节			地址X＋3的字节		
地址X的半字						地址X＋2的半字					
地址X的字											

(a) 大端存储格式

31	...	24	23	...	16	15	...	8	7	...	0
地址X＋3的字节			地址X＋2的字节			地址X＋1的字节			地址X的字节		
地址X＋2的半字						地址X的半字					
地址X的字											

(b) 小端存储格式

图2.4 大端和小端存储格式

在大端存储格式中，字的地址对应的是该字中最高有效字节所对应的地址；半字的地址对应的是该半字中最高有效字节所对应的地址。通俗地说，在大端存储格式中，32位字数据的最高字节存储在低字节地址中，而其最低字节则存储在高字节地址中。

在小端存储格式中，字的地址对应的是该字中最低有效字节所对应的地址；半字的地址对应的是该半字中最低有效字节所对应的地址。通俗地说，在小端存储格式中，32位字数据的最高字节存储在高字节地址中，而其最低字节则存储在低字节地址中。

小端存储格式是ARM9默认的格式。ARM9汇编指令集中，没有相应的指令来选择是采用大端存储格式还是小端存储格式，但可以通过硬件输入引脚来配置它。若要求ARM9目标系统支持小端存储格式，则将引脚BIGEND接低电平，否则接高电平。

ARM9体系结构对于存储器单元的访问需要适当地对齐，即访问字存储单元时，字地址应该是字对齐（地址能被4整除）；访问半字存储单元时，半字地址应该半字对齐（地址能被2整除）。如果不按对齐的方式访问存储单元，称作非对齐的存储器访问。非对齐的存储器访问可能会导致不可预知的状态。

2.2.2 I/O端口的访问方式

I/O端口的访问有两种方式，一种是端口地址和存储器统一编址，即存储器映射方式；另一种是I/O端口地址与存储器分开独立编址，即I/O映射方式（独立编址）。

ARM9体系结构使用存储器映射方式实现I/O端口的访问。由于存储器映射方式是为每个I/O端口分配特定的存储器地址，当从这些地址读出或向这些地址写入时，实际上就完成了I/O功能。即从存储器映射的I/O加载即是输入，而向存储器映射的I/O地址存储即是输出。

2.2.3 内部寄存器

ARM9处理器内部共有37个32位寄存器，可分成通用寄存器和状态寄存器两大类。

通用寄存器用于保存数据或地址；

状态寄存器用来标识或设置存储器的工作模式或工作状态等功能。

ARM9处理器的37个寄存器中，31个用作通用寄存器，6个用作状态寄存器，每个状态寄存器只使用了其中的12位。

这37个寄存器根据处理器的工作状态及工作模式的不同而被分成不同的组。程序代码运行时涉及的工作寄存器组是由ARM9处理器的工作模式确定的。

ARM状态下的通用寄存器与程序计数器

System &User	FIQ	Supervisor	Abort	IRQ	Undefind
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_SVC	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_SVC	R14_abt	R14_irq	R14_und
R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)	R15(PC)

ARM状态下的程序状态寄存器

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
= 分组寄存器	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

图2.5 ARM状态下寄存器的组织

1．通用寄存器

通用寄存器包括R0～R15寄存器，可分成未分组寄存器、分组寄存器及程序计数器三种。

（1）未分组寄存器R0～R7

未分组寄存器包括R0～R7，在所有工作模式下，它们在物理上是同一个寄存器。

（2）分组寄存器R8～R14

分组寄存器包括R8～R14。对于分组寄存器，它们每一次所访问的物理寄存器与处理器当前的工作模式有关。

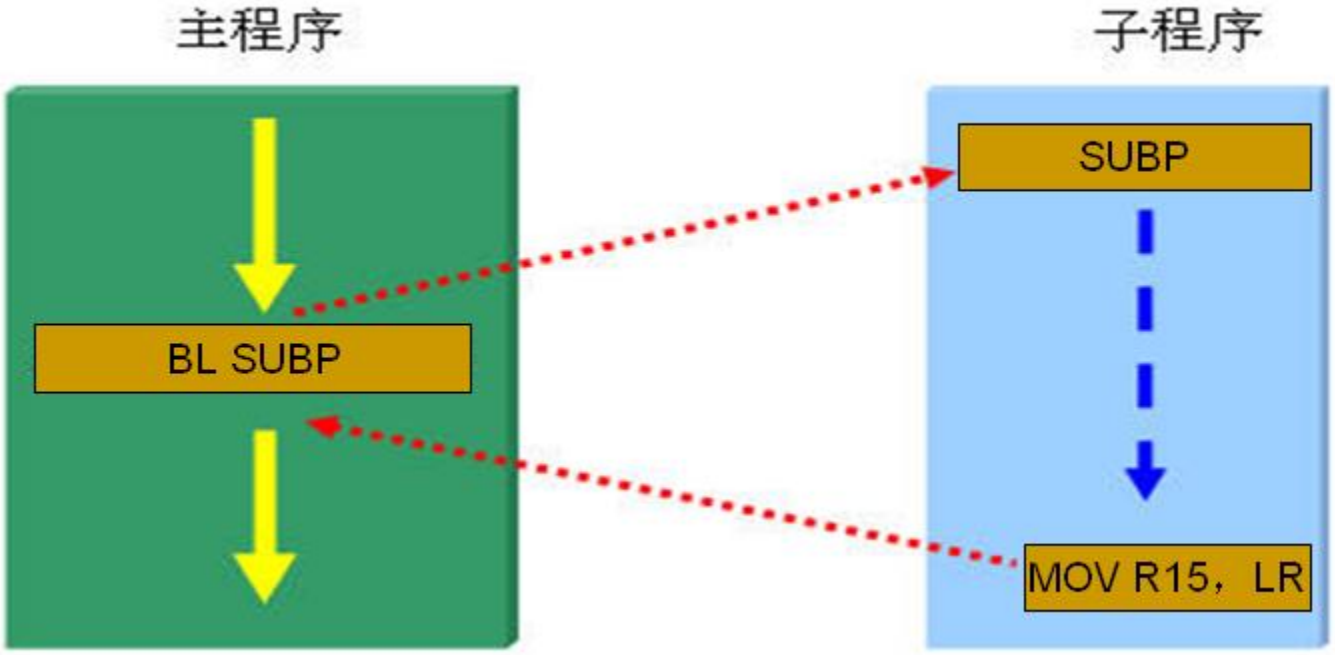
对于R8～R12寄存器，每个寄存器对应两个不同的物理寄存器。当使用fiq模式时，访问寄存器R8_fiq～R12_fiq；当使用fiq模式以外的其他模式时，访问寄存器R8_usr～R12_usr。

对于R13、R14寄存器而言，每个寄存器对应6个不同的物理寄存器，用户模式与系统模式共用1个；另外5种不同的工作模式对应5个，采用R13_<mode>或R14_<mode>记号来区分不同的物理寄存器，其中，<mode>为usr、fiq、irq、svc、abt、und这6种模式之一。

R13寄存器在ARM指令中常用作堆栈指针，又称为SP（Stack Pointer），但这只是一种习惯用法，用户也可使用其他寄存器作为堆栈指针。而在Thumb指令集中，某些指令强制性地要求使用R13作为堆栈指针。

R14寄存器可用作子程序链接寄存器（Subroutine Link Register）或链接寄存器LR（Link Register）。R14有两种特殊功能：一是每种工作模式下所对应的那个R14可用于保存子程序的返回地址；二是当异常发生时，该异常模式下的那个R14被设置成异常返回地址。

LR寄存器作用示意图



（3）程序计数器R15

R15寄存器的用途是程序计数器（PC）。在ARM状态下，R15的位［1:0］是0，位[31:2]保存PC的值；在Thumb状态下，位[0]为0，位[31:1]保存PC的值。**读R15寄存器的结果是读到的值为当前指令地址加8（ARM状态）或加4（Thumb状态）**。一般不用作通用寄存器。

2．程序状态寄存器

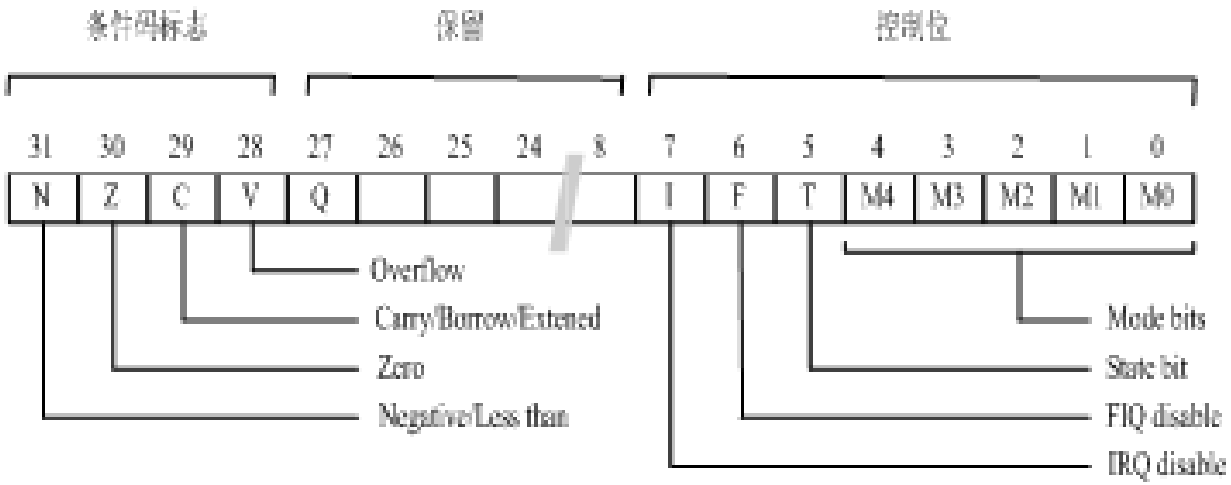


图2.6 程序状态寄存器格式

ARM9体系结构包含1个**当前程序状态寄存器**CPSR(Current Program Status Register)和5个备份的程序状态寄存器SPSR(Saved Program Status Register)，CPSR又称为R16。在任何工作模式下，CPSR都是同一个物理寄存器，它保存了程序运行的当前状态，如条件码标志、控制允许和禁止中断、设置处理器的工作模式以及其他状态和控制信息等。每种异常模式都有一个备份的程序状态寄存器SPSR；当异常发生时，SPSR用于保存CPSR的状态。

（1）条件码标志

CPSR寄存器的高4位是N、Z、C、V（Negative、Zero、Carry、oVerflow），称为条件码标志位。

它们的内容可被算术或逻辑运算的结果所改变，并且可以决定某条指令是否被执行。

CPSR中的条件码标志可由大多数指令检测以决定指令是否执行。在ARM状态下，绝大多数的指令都是有条件执行的。在Thumb状态下，仅有分支指令是有条件执行的。

通常条件码标志可以通过执行比较指令（CMN、CMP、TEQ、TST）、一些算术运算、逻辑运算和传送指令进行修改。

（2）控制位

CPSR寄存器的低8位是I、F、T和M[4:0]，称为控制位。

●中断禁止位：包括I和F，用来禁止或允许IRQ和FIQ两类中断。当I=1时，表示禁止IRQ中断；I=0时，表示允许IRQ中断。当F=1时，表示禁止FIQ中断；F=0时，表示允许FIQ中断。

●T标志位：当T=1时，表示程序运行于Thumb状态；当T=0时，表示程序运行于ARM状态。

●工作模式位：工作模式位（M[4:0]）用于标识或设置处理器的工作模式。M4、M3、M2、M1、M0决定了处理器的工作模式。

表2-3 CPSR寄存器的工作模式

M[4:0]	模式	可访问的寄存器
10000	用户(usr)	PC、R14~R0、CPSR
10001	FIQ(fiq)	PC、R14_fiq~R8_fiq、R7~R0、CPSR、SPSR_fiq
10010	IRQ(irq)	PC、R14_irq、R13_irq、R12~R0、CPSR、SPSR_irq
10011	管理(svc)	PC、R14_svc、R13_svc、R12~R0、CPSR、SPSR_svc
10111	中止(abt)	PC、R14_abt、R13_abt、R12~R0、CPSR、SPSR_abt
11011	未定义(und)	PC、R14_und、R13_und、R12~R0、CPSR、SPSR_und
11111	系统(sys)	PC、R14~R0、CPSR

例2.2：设在程序运行某时刻， CPSR寄存器的值如图2.7所示。试说明处理器的条件标志、中断允许情况、工作状态以及工作模式。

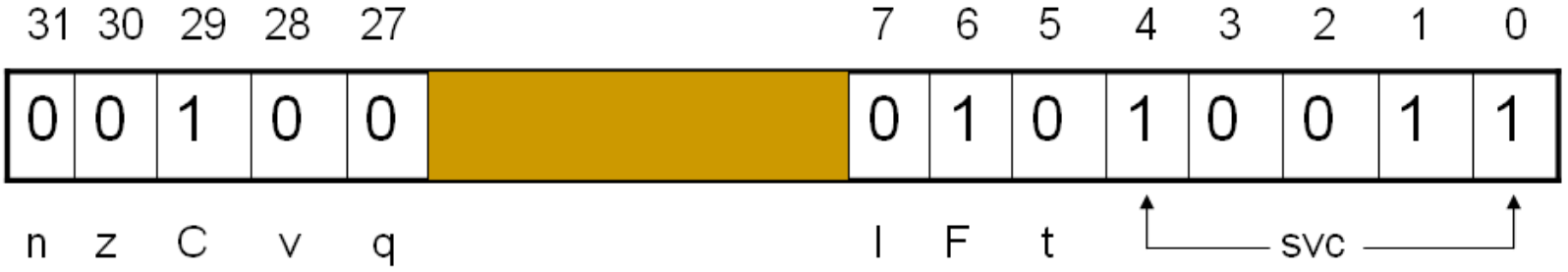


图2.7 CPSR的值

条件标志nzCvq， 即C标志位置1，其他标志位为0。因为bit[7-6]为I和F，所以IRQ中断被使能，即允许CPU响应IRQ中断，FIQ中断被禁止。因为bit[5]为t，所以处理器工作在ARM状态。因为bit[4-0]为10011，系统工作于管理模式(svc)。

（3）保留位

CPSR寄存器中的其余位是保留位，当改变CPSR中的条件码标志位或者控制位时，保留位不需要被改变，在程序中也不要使用保留位来存储数据。保留位主要用于ARM版本的扩展。

（4） SPSR寄存器

除系统模式和用户模式外，其他5种工作模式都有一个对应的专用SPSR寄存器。当异常发生时，SPSR用于保存CPSR的当前值，从异常退出时则可由SPSR来恢复CPSR。由于用户模式和系统模式不属于异常模式，它们没有SPSR；在这两种情况下访问SPSR，结果是未知的。CPSR和SPSR通过特殊指令进行访问。

3．Thumb寄存器

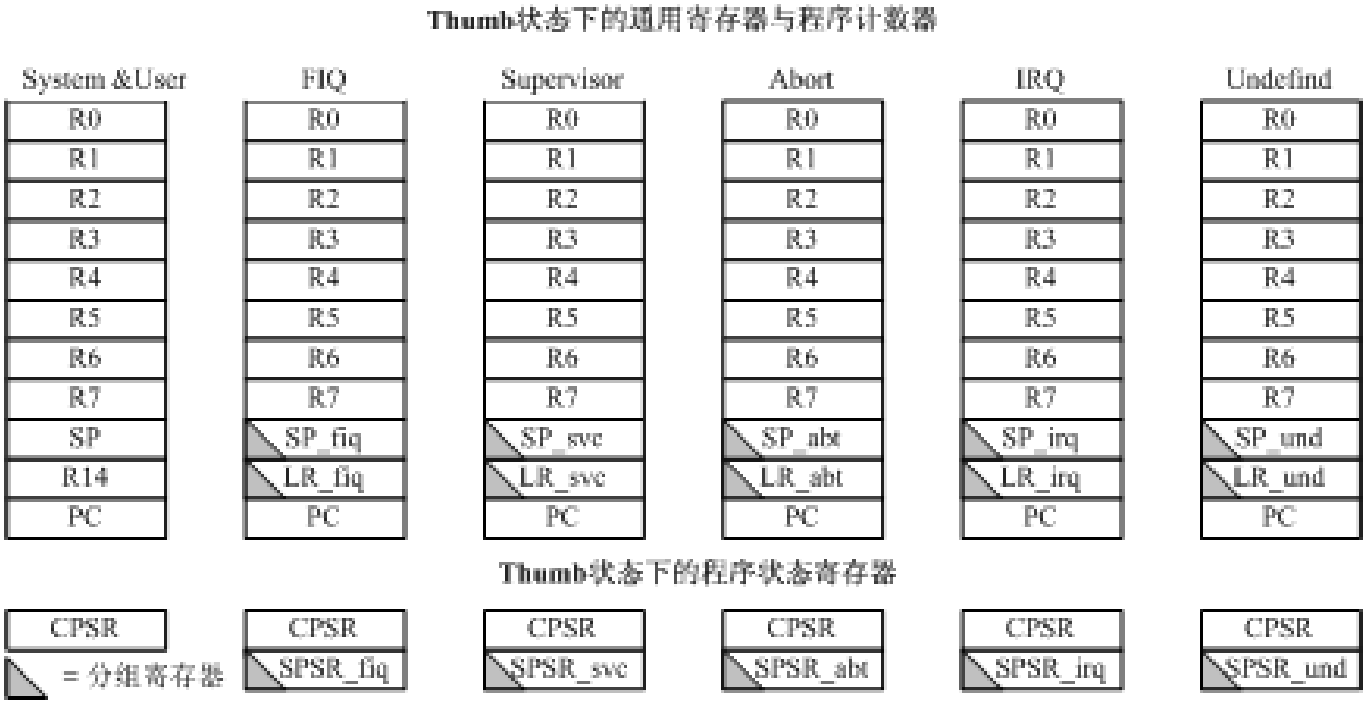


图2.8 Thumb状态下寄存器的组织

Thumb状态下的寄存器集是ARM状态下寄存器集的一个子集，程序可以直接访问通用寄存器R0～R7、程序计数器PC、堆栈指针SP、连接寄存器LR和CPSR。同时，在每种特权模式下都有对应的SP、LR和SPSR。

Thumb状态下寄存器的组织与ARM状态下寄存器的组织之间的关系如下：

- Thumb状态和ARM状态的R0～R7是相同的；
- Thumb状态和ARM状态的CPSR以及所有的SPSR是相同的；
- Thumb状态的SP对应于ARM状态的R13；
- Thumb状态的LR对应于ARM状态的R14；
- Thumb状态的PC对应于ARM状态的R15。

需要说明的是，在Thumb状态下， R8～R15寄存器并不是标准寄存器集的一部分，但可以使用汇编语言受限制地访问这些寄存器，将其用作快速的暂存器。

2.3 ARM9异常

所谓异常（**Exception: 异常中断）是指处理器由于内部或外部的原因，停止执行当前的程序，转而处理特定的事件，处理完毕返回原来的程序继续执行。**只要正常的程序流程被暂时停止，则异常发生。在处理异常之前，处理器状态必须保留，以便在异常处理程序完成后，原来的程序能够重新执行。同一时刻可能会出现多个异常，处理器会按固定的优先级对多个异常进行处理。ARM9体系结构中的异常与8位/16位体系结构的中断有很大的相似之处，但异常与中断的概念并不完全等同。

2.3.1 异常的类型及向量地址

2.3.2 异常的优先级

2.3.3 进入和退出异常

2.3.1 异常的类型及向量地址

表2.4 ARM9的异常处理模式

异常名称	对应模式	正常向量	高地址向量
复位	管理(svc)	0x00000000	0xFFFF0000
未定义指令	未定义(und)	0x00000004	0xFFFF0004
软件中断(SWI)	管理(svc)	0x00000008	0xFFFF0008
指令预取中止（取指令存储器中止）	中止(abt)	0x0000000C	0xFFFF000C
数据中止	中止(abt)	0x00000010	0xFFFF0010
IRQ（中断）	IRQ(irq)	0x00000018	0xFFFF0018
FIQ（快速中断）	FIQ(fiq)	0x0000001C	0xFFFF001C

（1）复位：处理器上一旦有复位信号输入，ARM处理器立刻停止执行当前指令，复位后，ARM处理器在禁止中断的管理模式下，从地址0x00000000或0xFFFF0000开始执行程序。

（2）未定义指令异常：有两种情况:①当ARM处理器执行协处理器指令时，它必须等待任一外部协处理器应答后，才能真正执行这条指令。若协处理器没有响应，会出现未定义指令异常。②试图执行未定义的指令，也会出现未定义指令异常。

（3）软件中断异常：是由软件中断指令SWI引起的。软件中断异常指令SWI进入管理模式，以请求执行特定的管理功能。

（4）指令预取中止(prefetch abort)：**指令预取访问存储器失败时产生的异常**称为指令预取中上异常。此时，存储器系统向ARM处理器发出存储器中止（abort）信号，响应取指激活的中止，预取的指令被标记为无效，若处理器试图执行无效指令，则产生预取中止异常；若指令未执行，则不发生预取中止。

（5）数据中止(data abort)：ARM处理器访问数据存储器失败时产生的异常称为数据中止异常。此时，存储器系统向ARM处理器发出存储器中止(Abort)信号，响应数据访问(加载/存储)激活的中止，数据被标记为无效。

（6）IRQ（中断请求）：通过处理器上的nIRQ引脚输入低电平产生。IRQ异常的优先级比FIQ异常的低。当进入FIQ处理时，会屏蔽掉IRQ异常。

（7）FIQ（快速中断请求）：通过处理器上的nFIQ引脚输入产生。

2.3.2 异常的优先级

- 当某时刻同时出现多个异常时，ARM处理器按优先级的高低顺序处理。异常的优先级如表2.5所示，从表中可知，复位异常的优先级最高，未定义异常和软件中断异常的优先级最低。

表2.5 ARM9异常的优先级排列顺序

优先级	异常	优先级	异常
1(最高)	复位	4	IRQ
2	数据中止	5	预取中止
3	FIQ	6(最低)	未定义指令、SWI

异常向量

- 异常向量是异常服务程序的入口，在某些ARM的应用中，允许异常向量的位置由32位地址空间低端的正常位置(0x00000000~0x0000001C)，移到地址空间高端的另一地址范围(0xFFFF0000~0xFFFF001C)。这些改变后的地址位置称为**高端向量**。
- 由Implementation Defined决定目标系统是否支持高端向量。如果支持，则在输入硬件配置时，选择是使用正常向量(normal vectors)还是高端向量(high vectors)。

- 应用程序中的异常处理：**当系统运行时，异常可能会随时发生**。为保证在ARM处理器发生异常时不至于处于未知状态，在应用程序的设计中，首先要进行异常处理。

1、在异常向量表中的特定位置放置一条跳转指令，跳转到异常处理程序。

2、当ARM处理器发生异常时，程序计数器PC会被强制设置为对应的异常向量，从而跳转到异常处理程序。

3、当异常处理完成以后，返回到主程序继续执行。各异常向量地址如前表2.4所示。

2.3.3 进入和退出异常

1. 进入异常

当处理一个异常时，ARM9完成以下动作(参考英文版S3C2410A.PDF文档, Pages: 2-10)：

（1）将下一条指令的地址保存在相应的LR寄存器中。如果异常是从ARM状态进入，则保存在LR中的是下一条指令的地址(当前PC+4或PC+8，与异常的类型有关)。如果异常是从Thumb状态进入，则保存在LR中的是当前PC的偏移量。这样异常处理程序就不需要确定异常是从何种状态进入的(如：在软件中断异常SWI产生时，指令MOV PC, R14_svc总是返回到下一条指令，不管SWI是在ARM状态下执行还是在Thumb下执行)。

（2）将CPSR复制到相应的SPSR中。

（3）迫使CPSR模式位M[4：0]的值设置成对应的异常模式值。

（4）迫使PC从相关的异常向量取下一条指令。

（5）也可以设置中断禁止位来阻止其他无法处理的异常嵌套。如果异常发生时，处理器处于Thumb状态，那么当用中断向量地址加载PC时，自动切换到ARM状态。

2．退出异常

在完成异常处理后，ARM9完成以下动作：

（1）将LR寄存器的值减去相应的偏移量（偏移量根据异常的不同而不同）后，送到PC中。

（2）将SPSR复制回CPSR中。

（3）清除中断禁止位标志。

3. 异常的返回过程

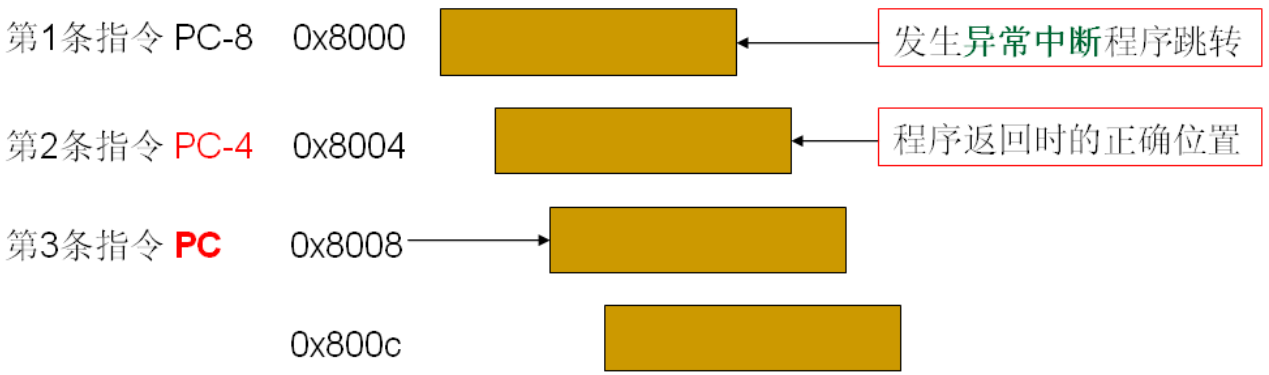


图2.9 异常及返回

表2.6 转移发生时的R14内容及返回操作

发生转移条件	返回操作	R14当前内容	
		ARM状态	Thumb状态
子程序调用	MOVS PC，R14	PC+4	PC+2
软件中断异常	MOVS PC,R14_svc	PC+4	PC+2
未定义异常	MOVS PC,R14_und	PC+4	PC+2
FIQ异常	SUBS PC,R14_fiq,#4	PC+4	PC+4
IRQ异常	SUBS PC,R14_irq,#4	PC+4	PC+4
指令预取中止异常	SUBS PC,R14_abt,#4	PC+4	PC+4
数据中止异常	SUBS PC,R14_abt,#8	PC+8	PC+8
复位	_	_	_

1.SWI和未定义指令异常中断处理程序的返回

当SWI和未定义指令异常中断发生时，ARM920T核将值(PC-4)保存到异常模式下的寄存器LR/R14_<Exception_Mode>中。这时(PC-4)即指向当前指令的下一条指令。

图 SWI的返回地址

返回操作可通过指令“MOVS PC, LR”来实现。同时，SPSR_<Exception_Mode>的内容被复制到当前程序状态寄存器中。

2. 指令预取中止异常中断处理程序的返回

当指令预取中止异常中断发生时，ARM920T核将值(PC-4)保存到异常模式下的寄存器LR/R14_<Exception_Mode>中。这时(PC-4)即指向当前指令的下一条指令。

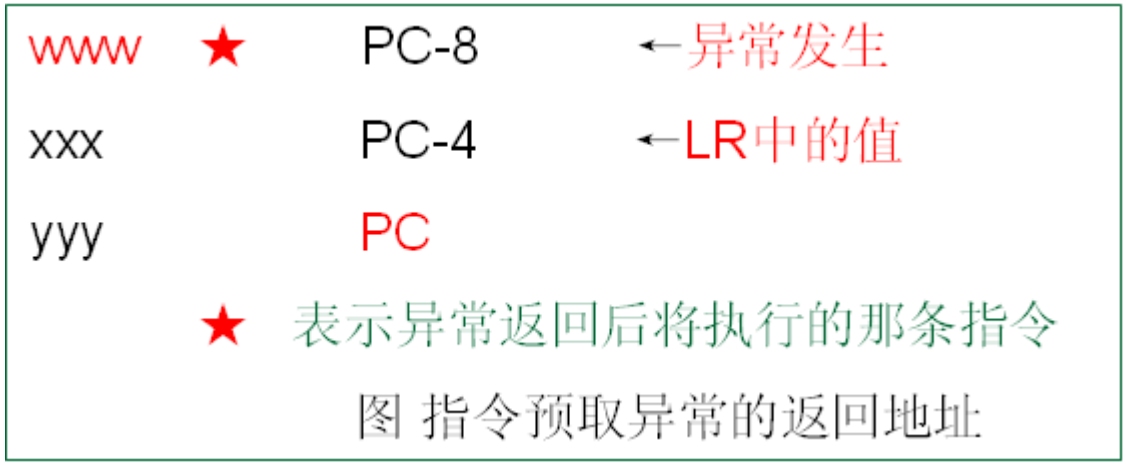


图 指令预取异常的返回地址

返回操作可通过指令“SUBS PC, LR, #4”来实现。同时，SPSR_<Exception_Mode>的内容被复制到当前程序状态寄存器中。

3. 数据访问中止异常中断处理程序的返回

当数据中止异常发生时，ARM920T核将值(PC-4)保存到异常模式下的寄存器LR/R14_<Exception_Mode>中。这时(PC-4)即指向当前指令后的第2条指令。

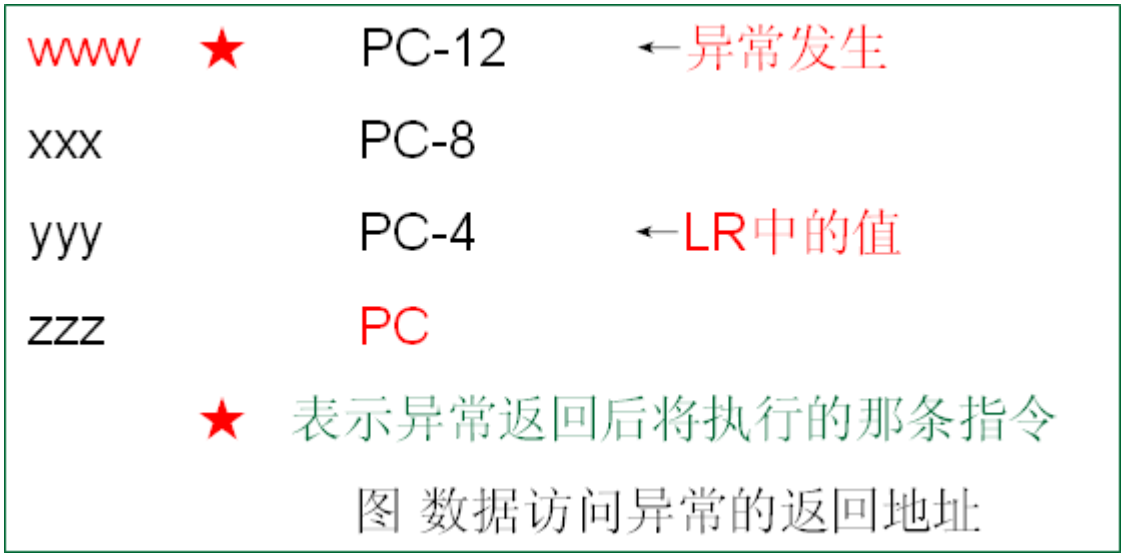


图 数据访问异常的返回地址

返回操作可通过指令“SUBS PC, LR, #8”来实现。同时，SPSR_<Exception_Mode>的内容被复制到当前程序状态寄存器中。

4. IRQ和FIQ异常中断处理程序的返回

当IRQ或FIQ异常中断发生时，ARM920T核将值(PC-4)保存到异常模式下的寄存器LR/R14_<Exception_Mode>中。这时(PC-4)即指向当前指令后的第2条指令。

图 IRQ/FRQ的返回地址

返回操作可通过指令“SUBS PC, LR, #4(R14_irq/R14_fiq)”来实现。同时，SPSR_<Exception_Mode>的内容被复制到当前程序状态寄存器中。

当IRQ/FIQ异常中断时保护和恢复现场。

```
SUBS LR, LR, #4
STMFD SP!, {Reglist, LR}   ;数据入栈，保护现场和断点
……
LDMFD SP!, {Reglist, PC}^   ;数据出栈，恢复现场和断点
```

在上面指令中，Reglist是异常中断处理程序中使用的寄存器列表。标识符^指示将SPSR_<Exception_Mode>寄存器内容复制到当前程序状态寄存器CPSR中。该指令只能在特权模式下使用。

2.4 S3C2410嵌入式微处理器

2.4.1 S3C2410及片内外围简介

2.4.2 S3C2410引脚信号

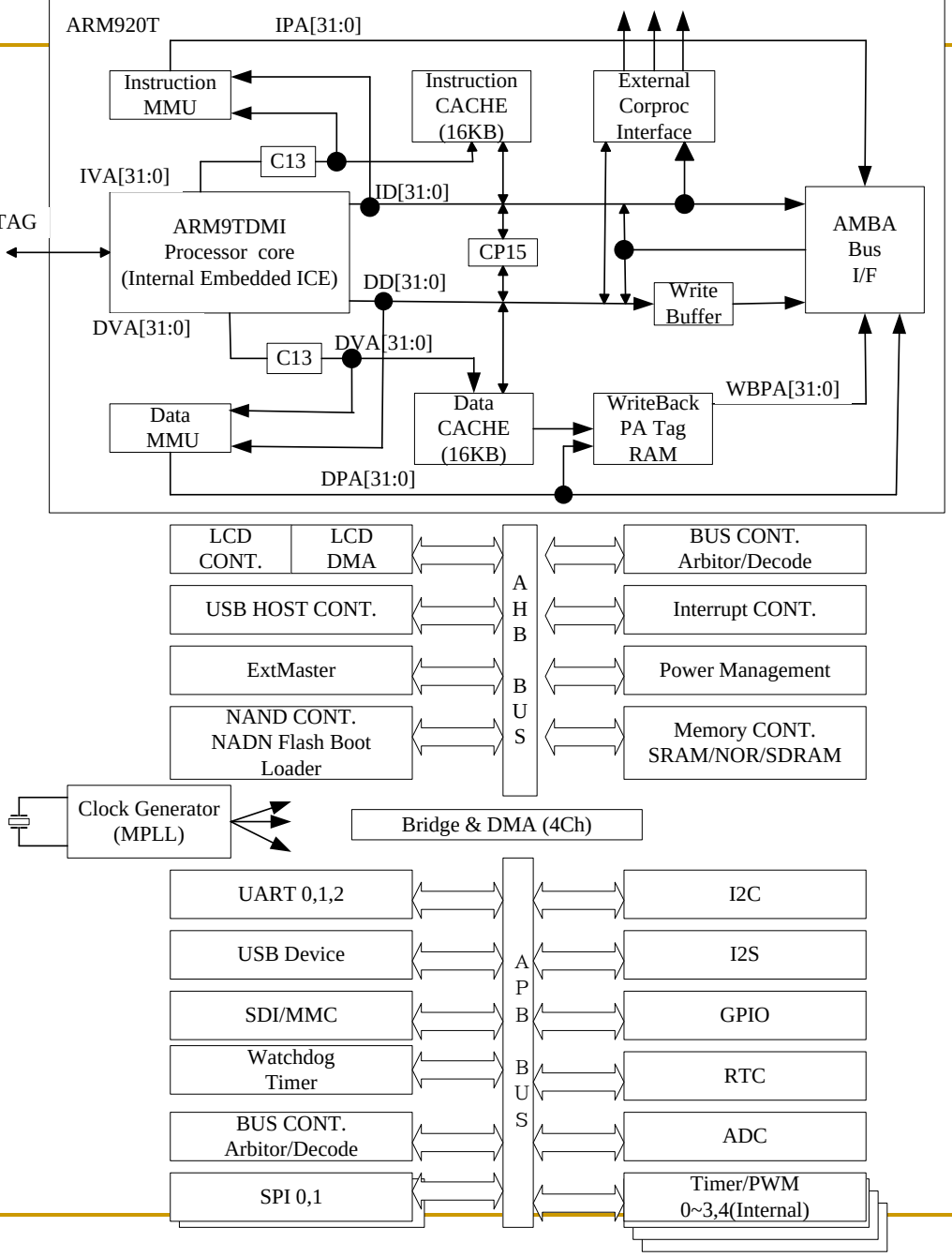
2.4.3 S3C2410专用寄存器

2.4.1 S3C2410及片内外围简介

S3C2410有两个具体的型号，分别是S3C2410X和S3C2410A，A是X的改进型，相对而言具有更优的性能和更低的功耗。

- S3C2410内含一个ARM920T 内核。
- ARM920T内核实现了ARM9 TDMI
- 存储器管理单元MMU
- 哈佛Cache高速缓存体系结构
- AMBA总线

MMU用于管理虚拟内存，Cache包括独立的16KB指令Cache和16KB数据Cache，每个Cache由8字长的行组成。 S3C2410片内外围设备接口可以分成高速外设和低速外设两种，分别使用AHB总线和APB总线。



本图可参见三星公司S3C2410A.PDF英文文档的1-5页。

图2.14　S3C2410结构框图

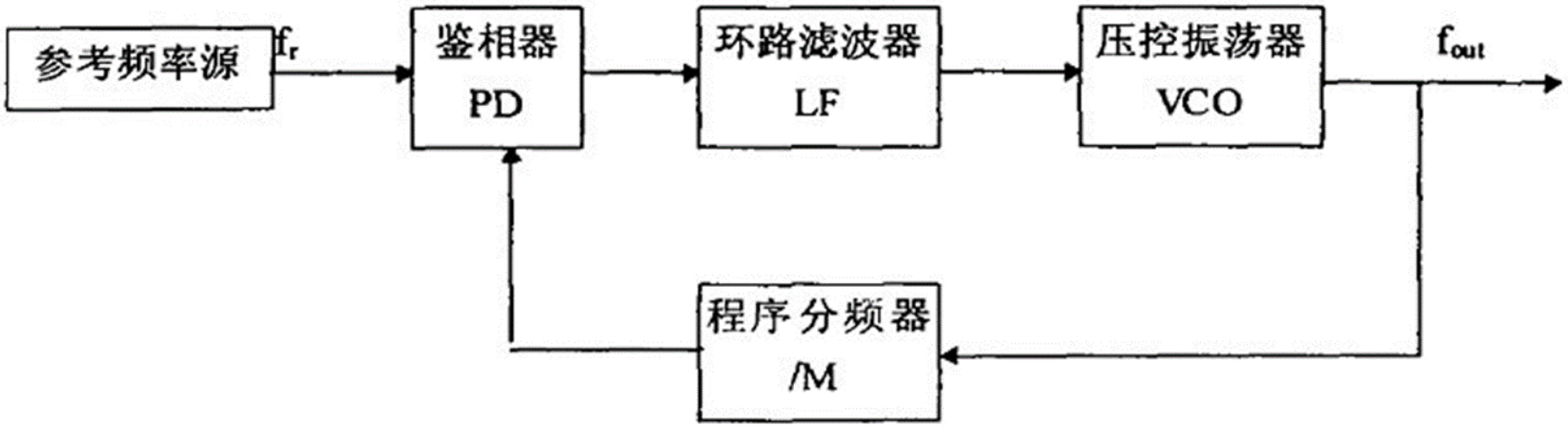
S3C2410片内外围资源主要有：

- 1个LCD控制器（支持4K色STN和256K色TFT带有触摸屏的液晶显示器）；
- SDRAM控制器；
- 3个通道的UART（IrDA1.0，16字节TxFIFO，16字节RxFIFO）；
- 4个通道的DMA；
- 4个具有脉冲带宽调制PWM（Pulse Width Modulation）功能的计时器和1个内部时钟；
- 8个通道的10位ADC；
- 触摸屏接口；
- 集成电路互联IIC（Inter Integrated Circuit）总线接口；
- 1个USB主机接口，1个USB设备接口；
- 2个串行外围设备SPI（Serial Peripheral Interface）接口；
- SD接口和多媒体卡MMC（MultiMedia Card）接口；
- 117位通用I/O口GPIO和24通道外部中断源。

ARM920T的MMU和Cache都集成在CP15协处理器中，MMU和Cache的联系非常密切，MMU、Cache和ARM920T核是协同工作的。使用MCR和MRC两条协处理器指令操作。

寄存器编号	功能
0	ID CODE, Cache Type (RO)
1	Control register
2	Translation table base (TTB) register
3	Domain access control register
4	--Reserved--
5	Fault status register
6	Fault address register
7	Cache operation
8	TLB operation
9	Cache lock down register
10	TLB lock down register
11-12 & 14	--Reserved--
13	ProcID
15	--Reserved for test purpose--

S3C2410集成了一个具有日历功能的RTC (Real Time Clock) 和具有PLL(MPLL和UPLL)的芯片时钟发生器。MPLL产生主时钟，能够使处理器工作频率最高达到203MHz。UPLL产生实现主从USB功能的时钟。



PLL原理图

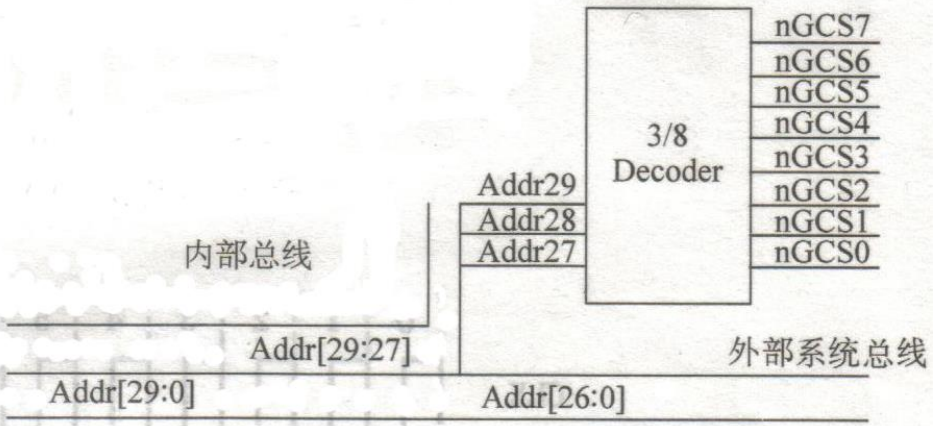
S3C2410将系统的存储空间分成8组/块（Bank）：

- 每组大小是128MB，共1GB。
- Bank0到Bank5的开始地址是固定的，用于ROM和SRAM。
- Bank6和Bank7用于ROM、SRAM或SDRAM，这两个组可编程且大小相同。Bank7的开始地址是Bank6的结束地址。
- 所有内存块的访问周期都可编程。
- 采用**nGCS**[7:0]8个通用片选信号选择这些组。

S3C2410支持从NAND Flash启动，有三种启动方式，可通过OM[1:0]管脚进行选择。

8个通用片选信号nGCS[7:0]的理解：

0x4000 0000		
0x3800 0000	SRAM/SDRAM (nGCS7)	128MB
0x3000 0000	SRAM/SDRAM (nGCS6)	128MB
0x2800 0000	SRAM (nGCS5)	128MB
0x2000 0000	SRAM (nGCS4)	128MB
0x1800 0000	SRAM (nGCS3)	128MB
0x1000 0000	SRAM (nGCS2)	128MB
0x0800 0000	SRAM (nGCS1)	128MB
0x0000 0000	SRAM (nGCS0)	128MB



2410独立地给每个块(bank)一个片选信号(nGCS7~nGCS0)。在理解上，这8个片选信号可看作是2410内部30根地址线的最高3位(A29、A28、A27)所做的地址译码的结果。

2.4.2 S3C2410引脚信号

1．S3C2410芯片封装

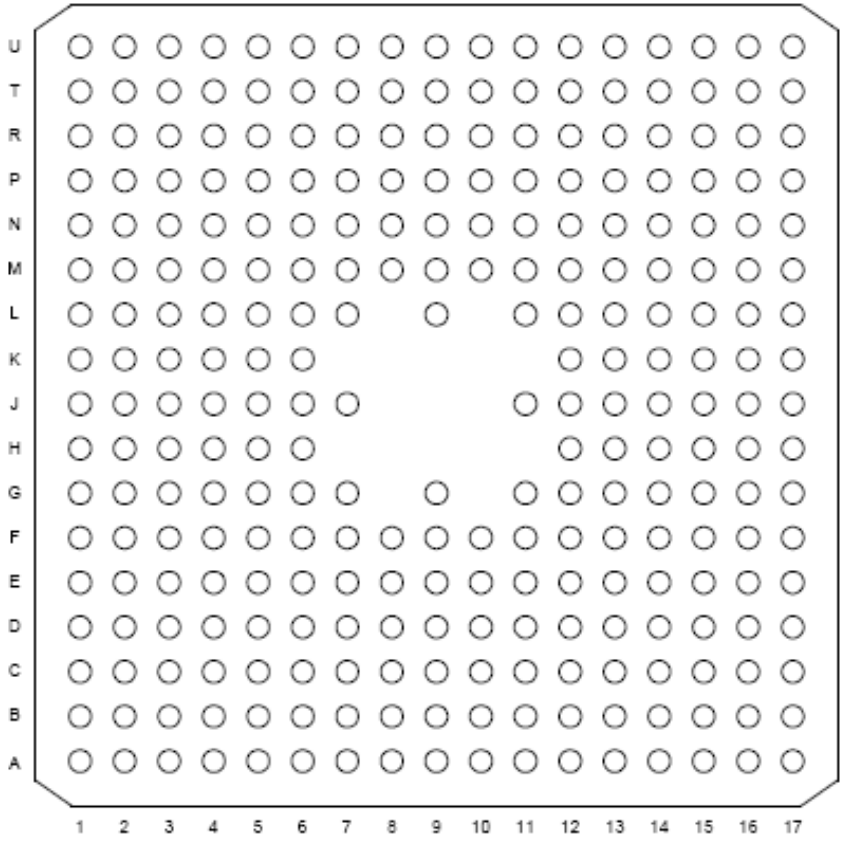


图2.15 S3C2410引脚排列底视图

2．S3C2410引脚信号介绍

S3C2410引脚信号分成地址总线、数据总线、各种接口控制信号和电源等。

（1）总线控制类信号

- OM[1:0] (I)：启动方式(Operating Mode)选择。00-处理器从NAND Flash启动；01-处理器从16位宽度的ROM启动；10-处理器从32位宽度的ROM启动；11-处理器为测试模式。
- ADDR[26:0] (O)：地址总线。输出对应存储器组的地址。
- DATA[31:0] (I/O)：数据总线。可编程为8位、16位或32位。
- nGCS[7:0] (O)：通用片选信号。控制存储器各组/块的访问。
- nWE (O)：写使能。当前总线周期处于写周期时有效。
- nOE (O)：输出使能。当前总线周期处于读周期时有效。
- nXBREQ (I)：总线保持请求(Bus Hold Request)。允许其他总线主控端请求本地总线的控制。
- nXBACK (O)：总线保持应答(Bus Hold Acknowledge)。有效时表示S3C2410已允许其他总线主控端控制本地总线。
- nWAIT (I)：等待信号。用于请求延长当前总线周期，低电平表示当前总线周期没有结束。

（2）SDRAM/SRAM接口信号

- nSRAS (O)：SDRAM行地址选通信号(SDRAM Row Address Strobe)。
- nSCAS (O)：SDRRAM列地址选通信号(SDRAM Column Address Strobe)。
- nSCS [1:0] (O)：SDRAM片选信号。
- DQM [3:0] (O)：SDRAM数据屏蔽/掩码信号(SDRAM Data Mask)。
- SCLK [1:0] (O)：SDRAM时钟信号。
- SCKE(O)：SDRRAM时钟使能(SDRAM Clock Enable)。
- nBE [3:0] (O)：高字节/低字节使能(Upper Byte/Lower Byte Enable，在16位SRAM中使用)。
- nWBE [3:0] (O)：写字节使能。

（3）NAND Flash接口信号

- CLE (O)：命令锁存使能。
- ALE (O)：地址锁存使能。
- nFCE (O)：NAND Flash芯片使能。
- nFRE (O)：NAND Flash读使能。
- nFWE (O)：NAND Flash写使能。
- NCON (I)：NAND Flash配置(NAND Flash Configuration)。若不用NAND Flash控制器，则该引脚接上拉电阻。
- R/nB (I)：NAND Flash准备好/忙信号。若不用NAND Flash控制器，则该引脚接上拉电阻。

（4）通用I/O端口信号

- GPn [116:0] (I/O)：通用输入/输出端口（注：某些端口只能用于输出）

（5）中断接口信号

- EINT [23:0] (I)：外部中断请求。

（6）DMA接口信号

- nXDREQ [1:0] (I)：外部DMA请求(External DMA request)。
- nXDACK [1:0] (O)：外部DMA应答(External DMA acknowledge)。

（7）定时器接口信号

- TOUT [3:0] (O)：定时器输出。
- TCLK [1:0] (I)：外部定时器时钟输入。

（8）UART接口信号

- RxD [2:0] (I)：UART接收数据。
- TxD [2:0] (O)：UART发送数据。
- nCTS [1:0] (I)：UART清除发送(UART clear to send input signal)。
- nRTS [1:0] (O)：UART请求发送。
- UEXTCLK(I)：UART时钟信号(UART clock signal)。

（9）IIC接口信号

- IICSDA (I/O)：IIC总线数据。
- IICSCL (I/O)：IIC总线时钟

（10）IIS接口信号

IIS： Inter-IC Sound Bus， 集成电路互连音频总线。

- IISLRCK (I/O)：IIS总线通道选择时钟 (IIS-bus channel select clock)。
- IISSDO (O)：IIS总线串行数据输出 (IIS-bus serial data output)。
- IISSDI (I)：IIS总线串行数据输入 (IIS-bus serial data input)。
- IISSCLK (I/O)：IIS总线串行时钟 (IIS-bus serial clock)。
- CDCLK (O)：CODEC编解码系统时钟 (CODEC system clock)。

（11）USB接口信号

- DN [1:0] (I/O)：USB主机的DATA (-) (需接15K下拉电阻)。
- DP [1:0] (I/O)：USB主机的DATA (+) (需接15K下拉电阻)。
- PDN0 (I/O)：USB设备的DATA (-) (需接470K下拉电阻)。
- PDP0 (I/O)：USB设备的DATA (+) (需接15K上拉电阻)。

（12）SPI接口信号

- SPIMISO [1:0] (I/O)：当SPI配置为SPI的主设备时，SPIMISO是主设备的数据输入线；当SPI配置为SPI的从设备时，SPIMISO是从设备的数据输出线。
- SPIMOSI [1:0] (I/O)：当SPI配置为SPI的主设备时，SPIMISO是主设备的数据输出线；当SPI配置为SPI的从设备时，SPIMISO是从设备的数据输入线。
- SPICLK [1:0] (I/O)：SPI时钟信号。
- nSS [1:0] (I)：SPI片选信号（仅用于从设备方式）。

（13）SD存储卡接口信号

- SDDAT [3:0] (I/O)：SD接收/发送数据。
- SDCMD (I/O)：SD接收响应/发送命令。
- SDCLK (O)：SD时钟信号。

（14）ADC接口信号

- AIN [7:0] (AI)：A/D转换输入，若不使用，引脚需接地。
- Vref (AI)：参考电压。

（15）LCD数据及控制信号

- VD [23:0] (O)：LCD数据总线（STN/TFT/SEC TFT）。
- LCD_PWREN (O)：LCD屏电源使能控制信号（STN/TFT/SEC TFT）。
- VCLK (O)：LCD时钟信号（STN/TFT）。
- VFRAME (O)：LCD帧信号（STN专用）。
- VLINE (O)：LCD行信号（STN专用）。
- VM (O)：改变行和列的电压极性（STN专用）。
- VSYNC (O)：垂直同步信号（TFT专用）。
- HSYNC (O)：水平同步信号（TFT专用）。
- VDEN (O)：数据使能信号（TFT专用）。
- LEND (O)：行结束信号（TFT专用）。
- STV (O)：LCD屏控制信号，垂直启动脉冲（三星SEC TFT专用）。
- CPV (O)：LCD屏控制信号，垂直移位脉冲（三星SEC TFT专用）。
- LCD_HCLK (O)：LCD屏控制信号，水平采样时钟（三星SEC TFT专用）。
- TP (O)：LCD屏控制信号，源驱动器数据加载脉冲（三星SEC TFT专用）。
- STH (O)：LCD屏控制信号，水平启动脉冲（三星SEC TFT专用）。
- LCDVF [2:0] (O)：LCD时序控制信号（三星SEC TFT或特殊的TFT专用）。

（16）触摸屏接口信号

- nXPON (O)：加X轴开关控制信号。
- XMON (O)：减X轴开关控制信号。
- nYPON (O)：加Y轴开关控制信号。
- YMON (O)：减Y轴开关控制信号。

（17）JTAG接口信号

- nTRST (I)：TAP控制器复位信号。若使用调试器，需接10K上拉电阻；若不使用调试器，通常将nTRST接至nRESET引脚。
- TMS (I)：TAP控制器模式选择信号。用于控制TAP控制器状态序列，需接10K上拉电阻。
- TCK (I)：TAP控制器时钟信号。需接10K上拉电阻。
- TDI (I)：TAP控制器数据输入。用于测试指令和数据的串行输入线，需接10K上拉电阻。
- TDO (O)：TAP控制器数据输出。用于测试指令和数据的串行输出线。

在嵌入式软件调试中，常使用JTAG接口。

（18）复位、时钟和电源控制信号

- nRESET (ST)：复位信号。在处理器电源稳定后，该信号需保持低电平至少4个FCLK周期(FCLK: CPU时钟，由PLL产生)。
- nRSTOUT (O)：复位输出。用于外部设备的复位控制(nRSTOUT = nRESET & nWDTRST & SW_RESET)。
- PWREN (O)：2.0V内核电源开关控制信号。
- nBATT_FLT (I)：电池状态检测。低电压状态时不唤醒掉电模式。
- OM [3:2] (I)：时钟信号模式。00b-晶振用于MPLL CLK和UPLL CLK时钟源；01b-晶振用于MPLL CLK时钟源，EXTCLK用于UPLL CLK时钟源；10b-EXTCLK用于MPLL CLK时钟源，晶振用于UPLL CLK时钟源；11b-EXTCLK用于MPLL CLK和UPLL CLK时钟源。
- EXTCLK (I)：外部时钟源。
- XTIpll (AI)：内部振荡电路的晶振输入。若不使用，需将其接至3.3V高电平。
- XTOpll (AO)：内部振荡电路的晶振输出。若不使用，需将其悬空。
- MPLLCAP (AI)：主时钟回路滤波电容。
- UPLLCAP (AI)：USB时钟回路滤波电容。
- XTIrtc (AI)：用于RTC的32.768 kHz晶振输入。若不使用，需将其接至高电平（RTCVDD = 1.8V）。
- XTOrtc (AO)：用于RTC的32.768 kHz晶振输出。若不使用，需将其悬空。
- CLKOUT [1:0]：时钟输出信号。由特殊功能寄存器MISCCR的CLKSEL[1:0]控制。

（19）电源

- VDDalive (P)：s3c2410复位电路和端口状态寄存器（1.8V/2.0V）。无论是正常模式还是掉电模式都应供电。
- VDDi/VDDiarm (P)：CPU内核逻辑电源（1.8V/2.0V）。
- VSSi/VSSiarm (P)：CPU内核逻辑地。
- VDDi_MPLL (P)：MPLL模拟和数字电源（1.8V/2.0V）。
- VSSi_MPLL (P)：MPLL模拟和数字地。
- VDDOP (P)：I/O端口电源（3.3V）。
- VDDMOP (P)：存储器接口电源VDD（3.3V: SCLK达133MHz）。
- VSSMOP (P)：I/O端口地。
- VSSOP (P)：存储器接口地。
- RTCVDD (P)：RTC电源（1.8V，不支持2.0V和3.3V）。
- VDDi_UPLL (P)：UPLL模拟和数字电源（1.8V/2.0V）。
- VSSi_UPLL (P)：UPLL模拟和数字地。
- VDDA_ADC (P)：ADC电源（3.3V）。
- VSSA_ADC (P)：A/DC地。

2.4.3 S3C2410专用寄存器/特殊功能寄存器

S3C2410的地址空间0x48000000至0x60000000之间的单元区供专用寄存器使用，用于存放硬件各功能部件的控制命令、状态或数据等。因这些寄存器的功能已作专门规定，故而称为专用寄存器或特殊功能寄存器SFR（Special Function Register）。

如，S3C2410的工作频率可达203Mhz，但决不是只工作于该频率。可以通过修改与处理器时钟相关的专用寄存器的值，使处理器工作在不同的频率，我们通常所说的超频就是通过此方法实现的。

S3C2410专用寄存器按硬件功能部件划分如下：

- 存储器控制器专用寄存器组；
- USB主设备专用寄存器组；
- 中断控制器专用寄存器组；
- DMA控制器专用寄存器组；
- 时钟和电源管理专用寄存器组；
- LCD控制器专用寄存器组；
- NAND Flash专用寄存器组；
- UART专用寄存器组；
- PWM定时器专用寄存器组；
- USB从设备专用寄存器组；
- 看门狗定时器专用寄存器组；
- IIC专用寄存器组；
- IIS专用寄存器组；
- 通用I/O口专用寄存器组；
- RTC专用寄存器组；
- A/D转换器专用寄存器组；
- SPI专用寄存器组；
- SD接口专用寄存器组。

表2.7 存储器控制寄存器

寄存器	地　址	功　能	操作	复位值
BWSCON	0x48000000	总线宽度和等待控制	读/写	0x0
BANKCON0	0x48000004	BANK0控制	读/写	0x0700
BANKCON1	0x48000008	BANK1控制	读/写	0x0700
BANKCON2	0x4800000C	BANK2控制	读/写	0x0700
BANKCON3	0x48000010	BANK3控制	读/写	0x0700
BANKCON4	0x48000014	BANK4控制	读/写	0x0700
BANKCON5	0x48000018	BANK5控制	读/写	0x0700
BANKCON6	0x4800001C	BANK6控制	读/写	0x18008
BANKCON7	0x48000020	BANK7控制	读/写	0x18008
REFRESH	0x48000024	SDRAM刷新控制	读/写	0xAC0000
BANKSIZE	0x48000028	可变的组大小设置	读/写	0x0
MRSRB6	0x4800002C	BANK6模式设置	读/写	xxx
MRSRB7	0x48000030	BANK7模式设置	读/写	xxx

以存储器控制器(Memory Controller)专用寄存器组为例，该控制器专用寄存器组共有13个专用寄存器。

(1) 总线宽度和WAIT控制寄存器(BWSCON)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ST7	WS7	DW7		ST6	WS6	DW6		ST5	WS5	DW5		ST4	WS4	DW4	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ST3	WS3	DW3		ST2	WS2	DW2		ST1	WS1	DW1		X	DW0		X

STn：控制存储块n的UB/LB引脚输出信号。

1：使UB/LB与nBE[3:0]相连(用UB/LB)

0：使UB/LB与nWBE[3:0]相连(不用UB/LB)

WSn：使用/禁用存储块n的WAIT状态

1：使能WAIT；0：禁止WAIT

DWn：控制存储块n的数据线宽

00：8位；01：16位；10：32位；11：保留

(2) BANKn--存储块控制寄存器（n=0~5）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Tacs		Tcos		Tacc			Tcoh		Tcah		Tacp		PMC	

Tacs：设置nGCSn有效前地址的建立时间

00：0个；01：1个；10：2个；11：4个时钟周期

Tcos：设置nOE有效前片选信号的建立时间

00：0个；01：1个；10：2个；11：4个时钟周期

Tacc：访问周期

000：1个；001：2个；010：3个；011：4个时钟

100：6个；101：8个；110：10个；111：14个

Tcoh：nOE无效后片选信号的保持时间

00：0个；01：1个；10：2个；11：4个时钟

Tcah： nGCSn无效后地址信号的保持时间

00：0个；01：1个；10：2个；11：4个时钟

Tacp：页模式的访问周期

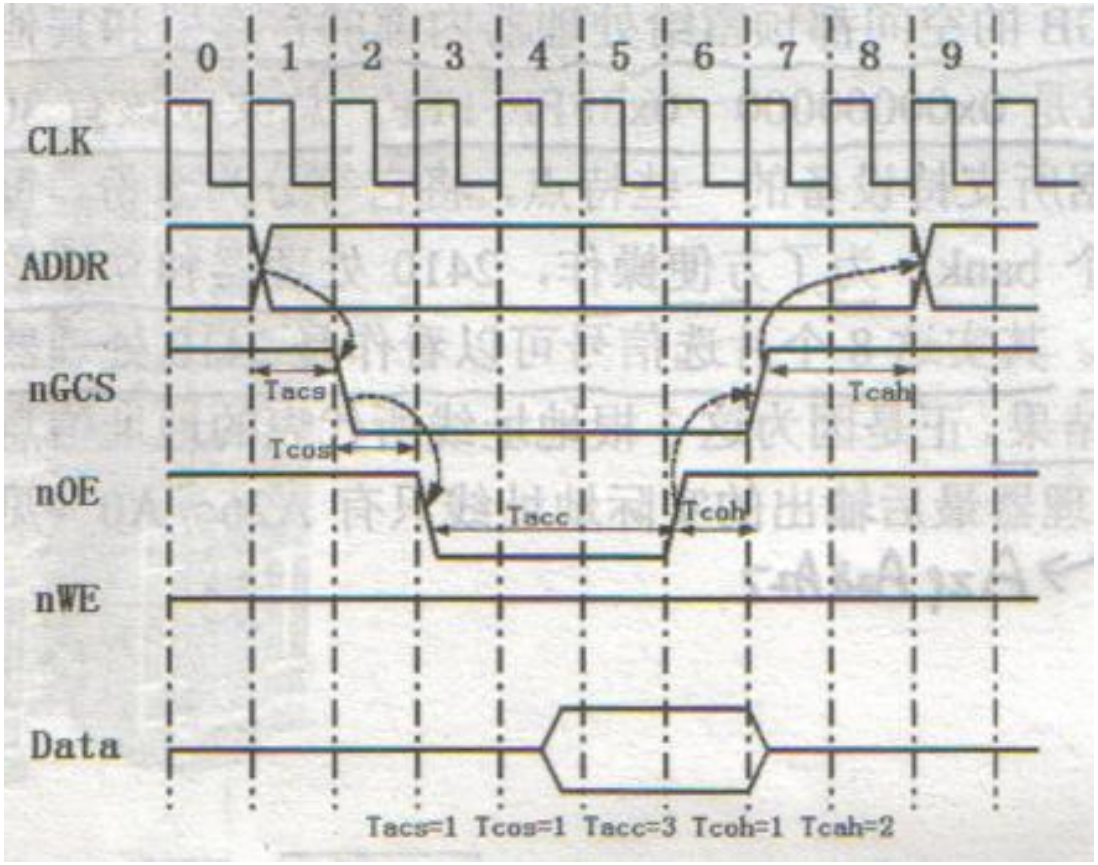
00：2个；01：3个；10：4个；11：6个时钟

PMC：页模式的配置(Page Mode Configuration)， 每次读写的数据数

00：1个；01：4个；10：8个；11：16个

(00为常规/通用模式)

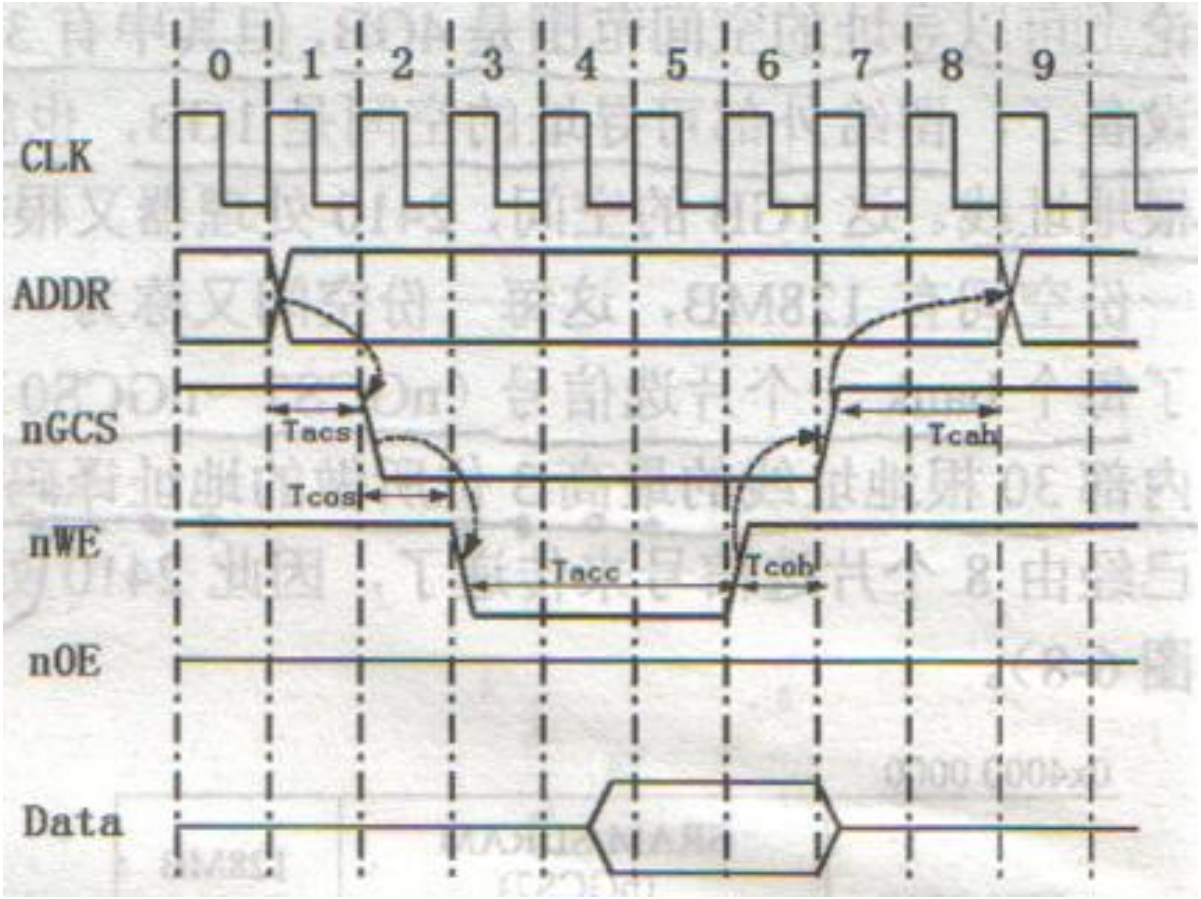
2410总线操作时序



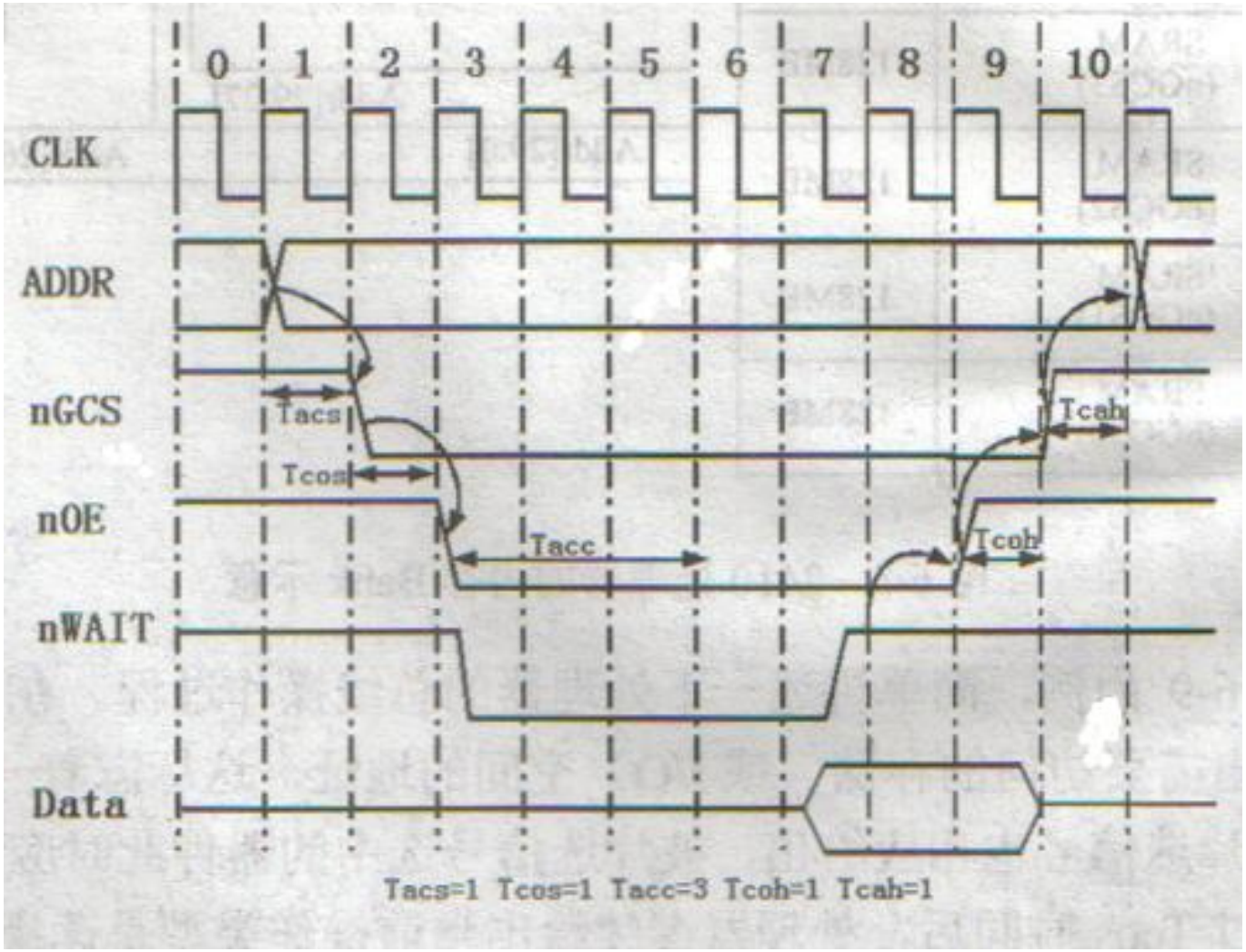
Notes:
Tacs: address set-up time before nGCSn
Tcos: chip selection set-up time before nOE
Tacc: access time
Tcoh: chip selection hold time after nOE
Tcah: Address hold time after nGCSn

(a) 总线读操作基本时序

写操作过程与读操作过程类似(见图(b))，此处不再详述。



(b) 总线写操作基本时序



(c) 带nWAIT信号的总线操作读操作时序

(3) BANK6/7--存储器块6/7控制寄存器

31													17	16	15
														MT	
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Tacs		Tcos		Tacc			Tcoh		Tcah		Tacp/ Trcd		PMC/ SCAN	

MT：设置存储器类型

00：ROM或者SRAM，[3：0]为Tacp和PMC；

11：SDRAM， [3：0]为Trcd和SCAN；　　01、10：保留

Trcd：由行地址信号切换到列地址信号的延时时钟数

00：2个时钟；01：3个时钟；10：4个时钟

SCAN：列地址位数(Column Adderss Number)

00：8位；　　01：9位；　　10：10位

(4) REFRESH--刷新控制寄存器

31					24	23		22		21	20	19	18	17	16
						REFEN		TREFMD		Trp		Tsrc		保留	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
保 留					Refresh_count										

REFEN：刷新控制。　　1：使能刷新；0：禁止刷新
TREFMD：刷新方式。　　1：自刷新　　0：自动刷新

Trp(RAS Precharge Time)：设置SDRAM行刷新时间（时钟数）
00：2个时钟；01：3个；10：3个；11：4个时钟

Tsrc(Semi Row Cycle Time)：设置SDRAM行操作时间（时钟数）
00：4个时钟；01：5个；10：6个；11：7个时钟
注： SDRAM的行周期= Trp + Tsrc。

Refresh_count：刷新计数器值
刷新周期计算公式如下：
刷新周期=(211-Refresh_count+1)/HCLK
例如，设刷新周期=15.6μs，HCLK=60MHz，则刷新计数器值=211+1-60×15.6=1113=0x459=0b10001011001。

(5) BANKSIZE--存储块6/7大小控制寄存器

7	6	5	4	3	2	1	0
BURST_EN	X	SCKE_EN	SCLK_EN	X	BK76MAP		

存储块6/7大小控制寄存器高24位未用，低8位说明如下。

BURST_EN：ARM突发操作控制

0：禁止突发操作；1：可突发操作

SCKE_EN：SCKE使能控制SDRAM省电模式

0：关闭省电模式；1：使能省电模式

SCLK_EN：SCLK省电控制，使其只在SDRAM访问周期内使能SCLK

0：SCLK一直有效；1：SCLK只在访问期间有效

BK76MAP：控制BANK6/7的大小及映射

100：2MB；101：4MB；　　110：8MB

111：16MB；　　000：32MB；　　001：64MB　　010：128MB

(6) MRSRB6/7—存储块6/7模式设置寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						WBL	TM		CL			BT	BL		

WBL(Write Burst Length)：写突发的长度。

0：固定长度；1：保留

TM：测试模式。 00：模式寄存器集；其它保留

CL(CAS Latency)：列地址反应时间

000：1个时钟；010：2个时钟；

011：3个时钟；其它保留

BT(Burst Type)：突发类型

0：连续； 1：保留

BL(Burst Length)：突发时间 000：1个时钟；其它:保留

(7)存储器控制寄存器的编程

存储器控制寄存器共包含13个特殊功能寄存器,在空间分布上是连续的,可以将配置数据连续存放,13个SFR寄存器配置代码如下:

```
LTORG
SMRDATA    DATA  ;定义13个字的数据区(52字节)
DCD  0x22111124  ;GCS0= GCS1=GCS6= GCS7=32bit, others=16bit
DCD   0x0700     ; GCS0 Tacs=Tcos=Tcoh=0clk,Tacp=2clk,Tacc=14clk,PMC=0(1data)
DCD   0x0700     ; GCS1 Tacs=Tcos=Tcoh=0clk,Tacp=2clk,Tacc=14clk,PMC=0(1data)
DCD   0x0700     ; GCS2 Tacs=Tcos=Tcoh=0clk,Tacp=2clk,Tacc=14clk,PMC=0(1data)
DCD   0x0700     ; GCS3 Tacs=Tcos=Tcoh=0clk,Tacp=2clk,Tacc=14clk,PMC=0(1data)
DCD   0x0700     ; GCS4 Tacs=Tcos=Tcoh=0clk,Tacp=2clk,Tacc=14clk,PMC=0(1data)
DCD   0x0700     ; GCS5 Tacs=Tcos=Tcoh=0clk,Tacp=2clk,Tacc=14clk,PMC=0(1data)
DCD   0x18005    ; GCS6 SDRAM, Tacs=1clk, Tcos=2clk, Trcd=3clk, SCAN=9bit
DCD   0x18005    ; GCS7 SDRAM, Tacs=1clk, Tcos=2clk, Trcd=3clk, SCAN=9bit
DCD   0x8c0459 ; REFEN=1,TREFMD=0,Trp=0,Tsrc=3,REFCNT=1113
DCD   0x32       ; SCKE_EN=SCLK_EN=1 SCLK power saving mode, BANKSIZE 128M
DCD   0x30       ; MRSR6,CL=3clk
DCD   0x30       ; MRSR7,CL=3clk
```

(7)存储器控制寄存器的编程（Cont.）

GCS0的数据宽度由外部引脚OM[1:0]设定，不需要程序进行设定。但在访问其他的外部存储器之前，需要通过程序来配置存储器控制寄存器，设定外部存储器的工作状态，程序如下：

```
        ldr     r0,=SMRDATA
        ldr     r1,=0x48000000      ;BWSCON地址
        add     r2,r0,#52           ;存储器控制寄存器组的长度
0       ldr     r3,[r0],#4
        str     r3,[r1],#4
        cmp     r2,r0
        bne     %B0                 ;此处的%B0表示转至向后的0标号处
```

课外作业：

1. P48 第2题
2. P48 第3题
3. P48 第8题

End of Chapter 2

附：JTAG接口简介(教材P78)：

JTAG是Joint Test Action Group(联合测试行动组)的缩写，JTAG最初用来对芯片进行测试，它的基本原理是在器件内部定义一个TAP（Test Access Port，测试访问口）通过专用的JTAG测试工具对内部节点进行测试。JTAG测试允许多个器件通过JTAG接口串联在一起，形成一个JTAG链，能实现对各个器件进行分别测试。目前，JTAG接口还常用于实现ISP（In-System Programmable：在线编程）、对Flash等器件进行编程等。

JTAG标准接口主要有4根线：分别是(1)TMS(Test Mode Selection Input，模式选择)-用来控制TAP状态机的转换。通过TMS信号，可以控制TAP在不同的状态间相互转换。TMS信号在TCK的上升沿有效。TMS在IEEE 1149.1标准中是强制要求的。 (2)TCK(Test Clock Input，测试时钟)-为TAP的操作提供了一个独立的、基本的时钟信号，TAP的所有操作都是通过这个时钟信号来驱动的。TCK在IEEE 1149.1标准中是强制要求的。 (3)TDI(Test Data Input，测试数据输入)-所有要输入到特定寄存器的数据都是通过TDI接口一位一位串行输入的。TDI在IEEE 1149.1标准中是强制要求的。 (4)TDO(Test Data Output，测试数据输出线)-所有要从特定的寄存器中输出的数据都是通过TDO接口一位一位串行输出的。TDO在IEEE 1149.1标准中是强制要求的。另外，还有一根可选的TRST (Test Reset Input， JTAG复位信号），可以用来对TAP Controller进行复位（初始化）。因为通过TMS也可以对TAP Controller进行复位（初始化），所以该信号接口在IEEE 1149.1标准中是可选的，并不是强制要求的。

附： ARM920T总线接口单元简介

AMBA 2.0规范定义了先进的微控制器总线体系结构AMBA(Advanced Micro-controller Bus Architecture)，它包括两种高性能的系统总线：

- 先进高性能总线AHB(Advanced High-performance Bus)；
- 先进的系统总线ASB(Advanced System Bus)。

ARM 920T核设计成带有一个单向ASB接口，外加必要的额外控制信号以确保AHB和ASB接口的高效实现。在单主控系统中，无需增加任何逻辑电路即可使用单向的ASB接口；此时，ARM 920T为主控器。通过增加三态驱动器，ARM 920T便实现了一个完整的ASB接口；此时，ARM 920T既能作为ASB总线的主控器，也能作为产品测试的从设备。通过增加一个可综合的封装器，ARM 920T便实现了一个完整的AHB接口；此时，ARM 920T既能作为AHB总线主控器，也能作为产品测试的从设备。封装器的引入，使读操作无速度代价、无性能代价，使可缓冲的写操作无性能代价，使不可缓冲的写操作只有极小的性能代价。