ARM微处理器体系结构

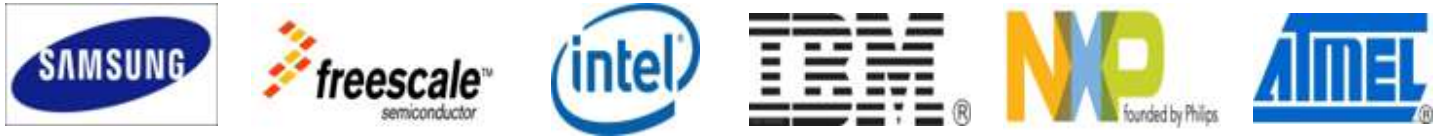
2

- ARM简介
- Cortex-A9内部功能及特点
- Cortex-A9架构的处理器状态
- Cortex-A9内核的工作模式
- Cortex-A9架构的内核寄存器
- ARM的异常中断
- Cortex-A9的存储系统

BACK

2.1 ARM简介

ARM公司是一家知识产权供应商，它与一般的半导体公司最大的不同就是不制造芯片且不向终端用户出售芯片，而是通过转让设计方案，由合作伙伴生产出各具特色的芯片。目前，100多家公司与ARM公司签订了技术使用许可协议，其中包括Intel、IBM、LG、NEC、SONY、NXP、NS、Samsung、Freescale、Atmel等这样的大公司。



2.1.1 RISC结构

RISC即精简指令集计算机，它的指令格式和长度通常是固定的(如ARM是32位的指令)，且指令和寻址方式少而简单，大多数指令在一个周期内就可以执行完毕。RISC的设计重点在于降低处理器中指令执行部件的硬件复杂度，这是因为软件比硬件更容易提供更大的灵活性和更高的智能化。

与RISC架构对应的是CISC（复杂指令集计算机）架构。特点是通过存放在只读存储器中的微码(Microcode)来控制整个处理器的运行。在CISC架构下，一条指令往往可以完成一串运算动作，但却需要多个时钟周期来执行。

RISC和CISC在构架上的不同：

- RISC构架的指令格式和长度通常是固定的；CISC构架下的指令长度通常是可变的；
- RISC在结构设计上是一个载入/存储的构架，只有载入和存储指令可以访问存储器，数据处理指令只对寄存器的内容进行操作。为了加速程序的运算，RISC会设定多组寄存器，并且指定特殊用途的寄存器。CISC架构则允许数据处理指令对存储器进行操作，对寄存器的要求相对不高。

2.1.2 ARM体系架构的发展

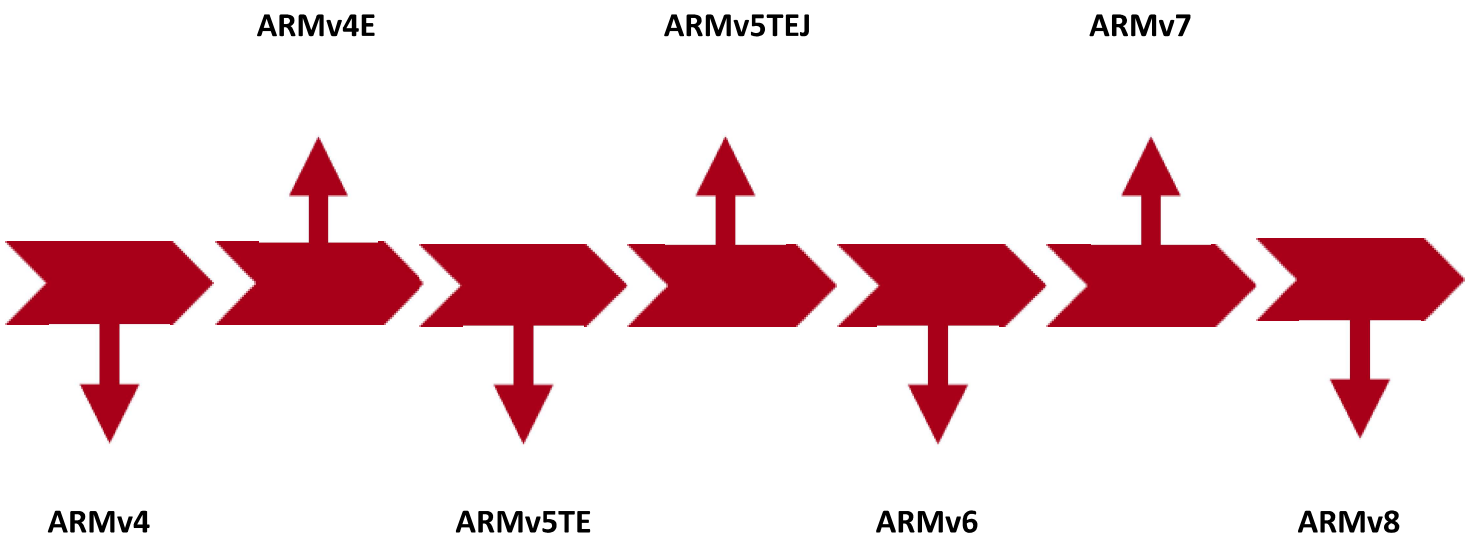


图2.1 ARM体系发展过程

ARMv4:只支持32位的指令集，支持32位的地址空间。

ARMv4E:在ARMv4基础上增加了16位的Thumb指令集，代码更紧凑。

ARMv5TE：改进了Thumb指令集：增加了一些“增强型DSP指令”，用于增强处理器对一些典型的DSP算法的处理性能。

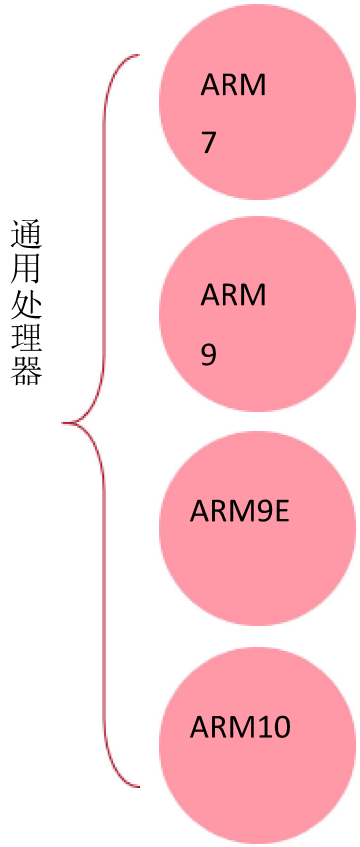
ARMv5TEJ：增加了Jazelle技术，用于提供Java加速功能。

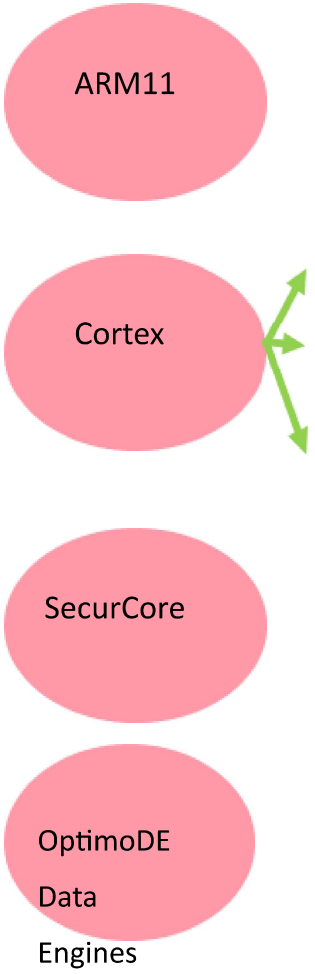
ARMv6：在存储系统、异常处理以及对多媒体功能的支持等多方面都有改进。

ARMv7：使用Thumb-2技术，还使用了NEON技术。

ARMv8：ARM公司的首款支持64位指令集的处理器架构。

2.1.2 ARM处理器系列




ARM11
Cortex
SecurCore
OptimoDE
Data
Engines

2.2 Cortex-A9内部功能及特点

2.2.1 功能特点

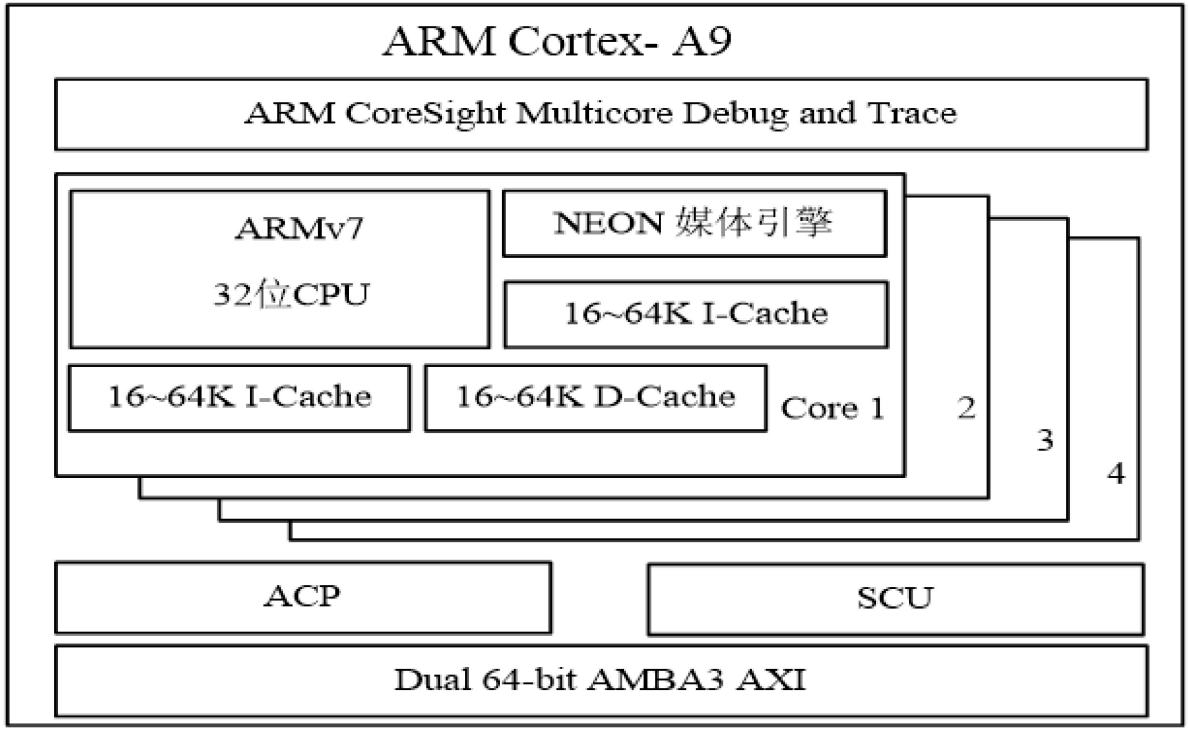
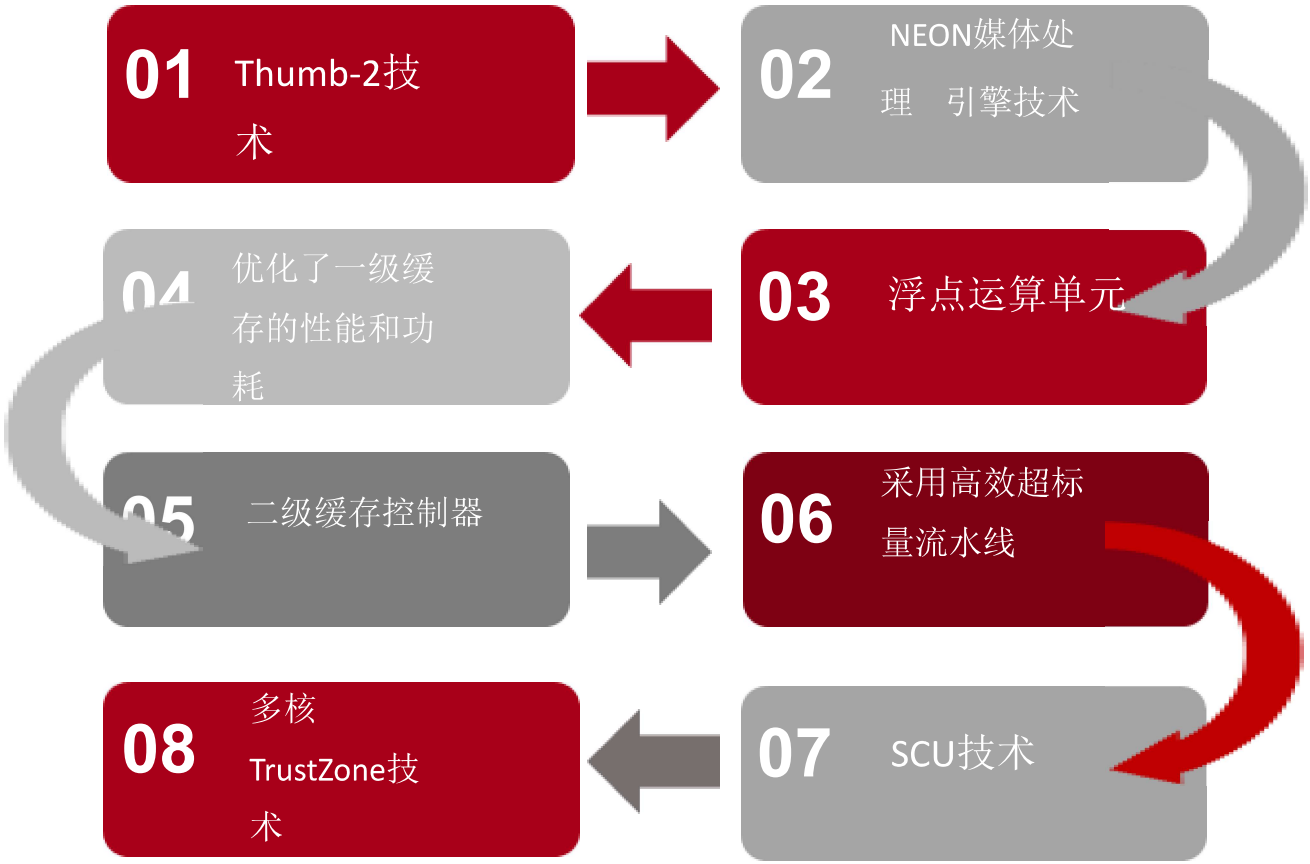


图2.2 Cortex-A9处理器的体系结构


01 Thumb-2技术
02 NEON媒体处理 引擎技术
03 浮点运算单元
04 优化了一级缓存的性能和功耗
05 二级缓存控制器
06 采用高效超标量流水线
07 SCU技术
08 多核 TrustZone技术

2.2.2 Cortex-A9的流水线

传统的单片机(如8051)中，处理器只有完成一条指令的读取和执行后，才会开始下一条指令的处理，所以PC(程序计数器)总是指向正在执行的指令。而在ARM体系架构中则引入了流水线的概念。流水线是ARM体系架构提高执行效率的一种有效策略。

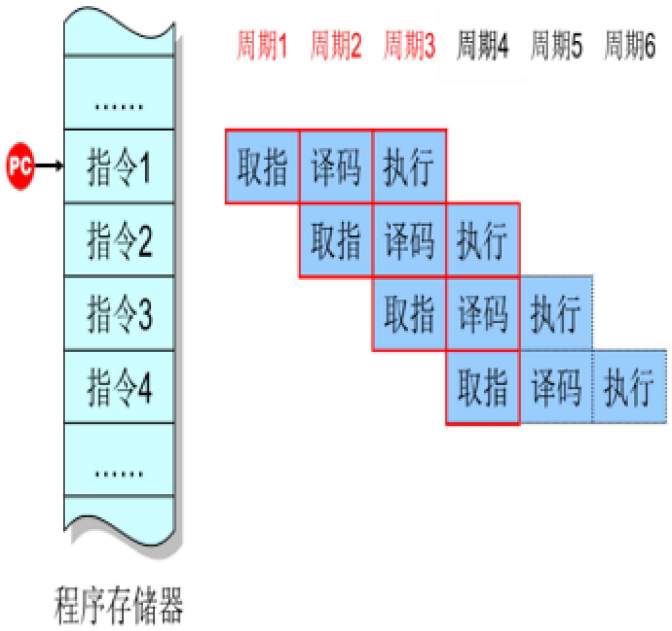


图2.3 三级流水线结构的指令执行顺序

(1) 在第1个周期，PC指向指令1，此时指令1进入三级流水线的取指阶段。

(2) 在第2个周期，PC指向指令2，此时指令1进入三级流水线的译码阶段，同时取出指令2。

(3) 在第3个周期，PC指向指令3，此时指令1进入三级流水线的执行阶段，指令2进入译码阶段，取出指令3。

(4) 在第4个周期，指令1执行完成，指令2和指令3流水线推进一级，同时开始指令4的取指处理

2.3 Cortex-A9架构的处理器状态

四种状态：指令集状态、执行状态、安全状态和调试状态。

指令集状态——提供了4种指令集状态，分别是ARM状态、Thumb状态、Jazelle状态和ThumbEE状态。

执行状态——包括指令集状态和一些控制指令流编码的位。

安全状态——安全状态的数目依赖于是否执行安全扩展。

调试状态——由于调试事件的存在，处理器被配置成暂停调试模式。

2.4 Cortex-A9内核的工作模式

内核工作模式	简称	模式控制位 M[4:0]	优先级	执行状态	执行情况	用　途
用户模式(User)	USR	10000	PL0	皆可	一直	ARM处理器正常程序执行状态
快速中断模式	FIQ	10001	PL1	皆可	一直	进行快速中断请求处理
外部中断模式	IRQ	10010	PL1	皆可	一直	进行通用的中断请求处理
管理模式(Supervisor)	SVC	10011	PL1	皆可	一直	操作系统使用的保护模式
监视模式(Monitor)	MON	10110	PL1	安全状态	安全扩展	进行操作系统的高级处理
预取指令中止异常(Abort)	ABT	10111	PL1	皆可	一直	虚存和存储器保护，包括数据异常和预取指异常
超级管理模式(Hypervisor)	HYP	11010	PL2	非安全状态	虚拟扩展	虚拟内存和存储器保护
未定义指令模式(Undefined)	UND	11011	PL1	皆可	一直	协处理器的软件仿真，未定义指令异常
系统模式(System)	SYS	11111	PL1	皆可	一直	运行具有特权的操作系统任务

表2.1 ARM处理器的模式

2.5 Cortex-A9架构的内核寄存器

从系统角度来看，Cortex-A9架构内部共有42个用户可以访问的32位寄存器，这些寄存器分别为：

(1) 13个通用32位寄存器：R0_usr～R12_usr。特殊的，在FIQ模式下，R8～R12对应为R8_fiq、R9_fiq、R10_fiq、R11_fiq、R12_fiq。

(2) 8个状态寄存器：CPSR、SPSR_hyp、SPSR_svc、SPSR_abt、SPSR_und、SPSR_mon、SPSR_irq、SPSR_fiq。

(3) 针对不同模式的三种寄存器：SP(R13)、LR(R14)、PC(R15)。在不同模式下，它们分别为SP_usr(R13)、SP_hyp、SP_scv、SP_abt、SP_und、SP_mon、SP_irq、SP_fiq、LR_usr(R14)、LR_svc、LR_abt、LR_und、LR_mon、LR_irq、LR_fiq、PC。

2.5.1 ARM状态下的寄存器

寄存器名称	各模式下实际访问的寄存器								
	用户	系统	超级管理	管理	中止	未定义	监测	中断	快中断
R0	R0_usr								
R1	R1_usr								
R2	R2_usr								
R3	R3_usr								
R4	R4_usr								
R5	R5_usr								
R6	R6_usr								
R7	R7_usr								
R8	R8_usr								R8_fiq
R9	R9_usr								R9_fiq
R10	R10_usr								R10_fiq
R11	R11_usr								R11_fiq
R12	R12_usr								R12_fiq
SP(R13)	R13_usr		SP_hyp		SP_scv	SP_abt	SP_und	SP_mon	SP_irq	SP_fiq
LR(R14)	R14_usr				LR_svc	LR_abt	LR_und	LR_mon	LR_irq	LR_fiq
PC(R15)	PC								

表2.2 ARM处理器各种模式下的寄存器

2、一般通用寄存器：

R0～R7为保存数据或地址值的通用寄存器；

R8～R14所对应的物理寄存器取决于当前的物理寄存器模式；

R13、R14在不同模式下分别对应不同的寄存器，除用户模式和系统模式外。

3、堆栈指针寄存器

堆栈指针	用户	系统	超级管理	管理	中止	未定义	监测	中断	快中断
SP(R13)	R13		SP_hyp	SP_scv	SP_abt	SP_und	SP_mon	SP_irq	SP_fiq

4、 链接寄存器LR

堆栈指针	用户	系统	超级管理	管理	中止	未定义	监测	中断	快中断
SP(R14)	R14			LR_scv	LR_abt	LR_und	LR_mon	LR_irq	LR_fiq

5、程序指针寄存器PC

6、程序状态寄存器CPSR和程序状态保存寄存器SPSR

堆栈指针	用户	系统	超级管理	管理	中止	未定义	监测	中断	快中断
CPSR	CPSR								
SPSR	–		SPSR_hyp	SPSR_svc	SPSR_abt	SPSR_und	SPSR_mon	SPSR_irq	SPSR_fiq

CPSR和SPSR的区别

2.6 ARM异常中断

执行低地址	执行高地址	异 常 类 型	进入时的模式	进入时I的状态	进入时F的状态
0x00000000	0xFFFF0000	复位(Reset)	管理	禁止	禁止
0x00000004	0xFFFF0004	未定义指令(Undefined Interrupt)	未定义	I	F
0x00000008	0xFFFF0008	软件中断(SWI)	管理	禁止	F
0x0000000C	0xFFFF000C	预取中止(Prefetch Abort)	中止	I	F
0x00000010	0xFFFF0010	数据中止(Data Abort)	中止	I	F
0x00000014	0xFFFF0014	保留	保留	–	–
0x00000018	0xFFFF0018	外部中断(IRQ)	中断	禁止	F
0x0000001C	0xFFFF001C	快速中断(FIQ)	快速中断	禁止	禁止

表2.3 ARM的异常向量地址一览表

1. 复位异常

当复位异常时，系统主要执行下列伪操作：

```
R14_svc = UNPREDICTABLE value
SPSR_svc = UNPREDICTABLE value
CPSR[4:0] = 0b10011                    ；进入特权模式
CPSR[5] = 0                    ；处理器进入ARM状态
CPSR[6] = 1                    ；禁止快速中断
CPSR[7] = 1                    ；禁止外设中断
If  high vectors configured then
    PC = 0xffff0000
else
 PC = 0x00000000
```

2. 未定义指令异常

当未定义指令异常发生时，系统主要执行下列伪操作：

```
r14_und = address of next instruction after the undefined instruction
SPSR_und = CPSR
CPSR[4:0] = 0b11011          ；进入未定义指令模式
CPSR[5] = 0                  ；处理器进入ARM状态
                    ；CPSR[6]保持不变
CPSR[7] = 1                  ；禁止外设中断
If high vectors configured then
    PC = 0xffff0004
else
    PC = 0x00000004
```

3. 软件中断异常：

软件中断异常发生时，处理器主要执行下列伪操作：

```
r14_svc = address of next instruction after the SWI instruction
SPSR_und = CPSR
CPSR[4:0] = 0b10011；              进入特权模式
CPSR[5] = 0；        处理器进入ARM状态；CPSR[6] 保持不变
CPSR[7] = 1；              禁止外设中断
If high vectors configured then
    PC = 0xffff0008
else
    PC = 0x00000008
```

4. 预取中止异常：

预取中止异常发生时，处理器主要执行下列伪操作：

```
r14_svc = address of the aborted instruction + 4
SPSR_und = CPSR
CPSR[4:0] = 0b10111              ；进入特权模式
CPSR[5] = 0                      ；处理器进入ARM状态；CPSR[6]保持不变
CPSR[7] = 1                      ；禁止外设中断
If high vectors configured then
    PC = 0xffff000C
else
    PC = 0x0000000C
```

5. 数据中止异常：

当数据中止异常发生时，处理器主要执行下列伪操作：

r14_abt = address of the aborted instruction + 8

SPSR_ abt = CPSR

CPSR[4:0] = 0b10111

CPSR[5] = 0　　；处理器进入ARM状态；　　CPSR[6]保　持不变

CPSR[7] = 1　　；禁止外设中断

If high vectors configured then

　PC = 0xffff000C10

else

　PC = 0x00000010

6. 外部中断异常：

当外部中断异常发生时，处理器主要执行下列伪操作：

```
r14_irq = address of next instruction to be executed + 4
SPSR_irq = CPSR
CPSR[4:0] = 0b10010      ；进入特权模式
CPSR[5] = 0              ；处理器进入ARM状态；CPSR[6]保持
不变
CPSR[7] = 1              ；禁止外设中断
If high vectors configured then
    PC = 0xffff0018
else
    PC = 0x00000018
```

7. 快速中断异常：

当快速中断异常发生时，处理器主要执行下列伪操作：

```
r14_fiq = address of next instruction to be executed + 4
SPSR_ fiq = CPSR
CPSR[4:0] = 0b10001          ；进入快速中断模式
CPSR[5] = 0                  ；处理器进入ARM状态
CPSR[6] = 0                  ；允许快速中断
CPSR[7] = 1                  ；禁止外设中断
If high vectors configured
    PC= 0xffff001c
else
    PC = 0x0000001c
```

2.6.3 ARM异常优先级

优先级	异常类型	返回地址
1 (最高)	复位异常	R14
2	数据异常	R14
3	快速中断异常	R14-4
4	外部中断异常	R14-4
5	预取异常	R14-4
6	软件中断异常	R14-8
7 (最低)	未定义指令异常	—

表2.4 ARM异常的优先级

2.7 Cortex-A9的存储系统

2.7.1 ARM存储系统的相关概念

1、存储器的字节、半字、字与双字

2、存储方式：

方式	半字对齐	字对齐
地址	… 0x4002 0x4004 …	… 0x4004 0x4008 …
特征	bit0 = 0，其他位为任意值	bit1 = 0，bit0 = 0，其他位为任意值

表2.5 字对齐和半字对齐地址示例

3、存储格式：

1）大端存储系统

<table>
<tr><td>MSByte</td><td>MSByte-1</td><td>MSByte-2</td><td>MSByte-3</td><td>LSByte+3</td><td>LSByte+2</td><td>LSByte+1</td><td>LSByte</td></tr>
<tr><td colspan="8">在地址 A 处的双字</td></tr>
<tr><td colspan="4">在地址 A 处的双字</td><td colspan="4">在地址 A+4 处的字</td></tr>
<tr><td colspan="2">在地址 A 处的半字</td><td colspan="2">在地址 A+2 处的半字</td><td colspan="2">在地址 A+4 处的半字</td><td colspan="2">在地址 A+6 处的半字</td></tr>
<tr><td>A 处的字节</td><td>A+1 处的字节</td><td>A+2 处的字节</td><td>A+3 处的字节</td><td>A+4 处的字节</td><td>A+5 处的字节</td><td>A+6 处的字节</td><td>A+7 处的字节</td></tr>
</table>

图2.6 大端存储系统示意图

2）小 端存储系统

<table>
<tr><td>MSByte</td><td>MSByte-1</td><td>MSByte-2</td><td>MSByte-3</td><td>LSByte+3</td><td>LSByte+2</td><td>LSByte+1</td><td>LSByte</td></tr>
<tr><td colspan="8">在地址 A 处的双字</td></tr>
<tr><td colspan="4">在地址 A+4 处的双字</td><td colspan="4">在地址 A 处的字</td></tr>
<tr><td colspan="2">在地址 A+6 处的半字</td><td colspan="2">在地址 A+4 处的半字</td><td colspan="2">在地址 A+2 处的半字</td><td colspan="2">在地址 A 处的半字</td></tr>
<tr><td>A+7 处的字节</td><td>A+6 处的字节</td><td>A+5 处的字节</td><td>A+4 处的字节</td><td>A+3 处的字节</td><td>A+2 处的字节</td><td>A+1 处的字节</td><td>A 处的字节</td></tr>
</table>

图2.7 小端存储系统示意图

2.7.1 ARM Cortex-A9存储系统的架构

Cortex-A9采用虚拟存储系统架构（VMSA）的处理器，其存储系统的架构示意图如图2.6所示：

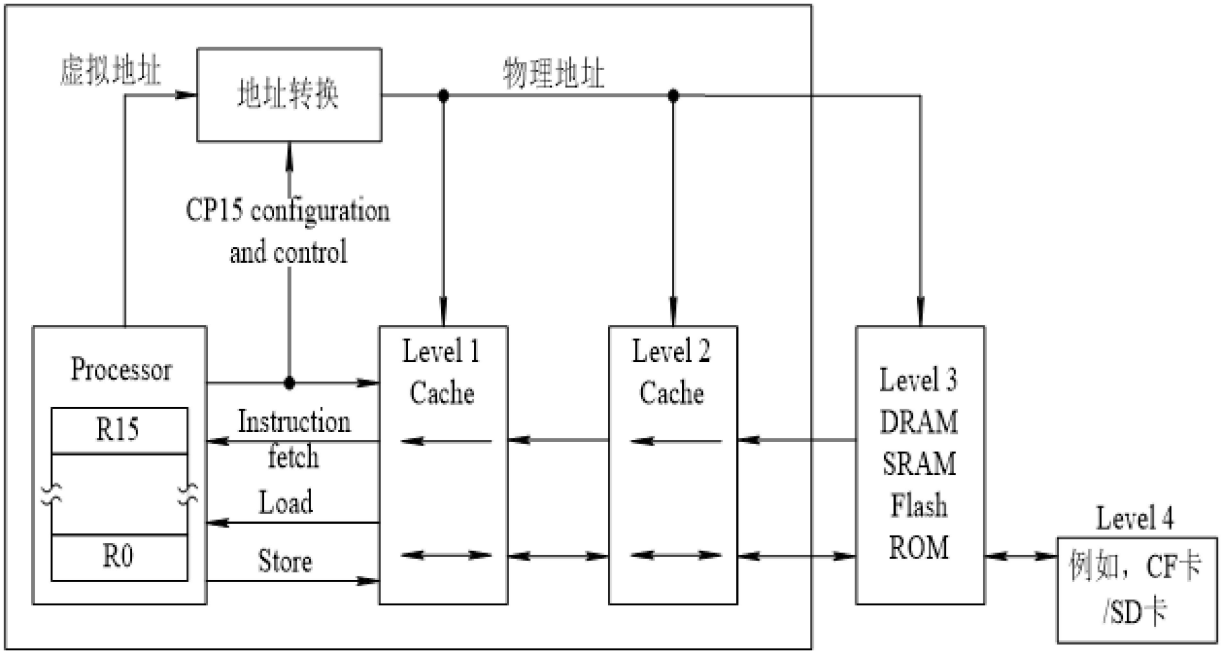


图2.8 多层次高速缓存架构

问题与思考：

1. RISC和CISC架构的区别是什么？

2.简述三级流水线的概念？Cortex-A9面向多核技术的特点有哪些？

3. ARM处理器模式和ARM处理器状态有何区别？列举Cortex-A9处理器的模式和状态。

4. ARMv7的内核工作模式有哪些？哪些属于异常工作模式？5.针对不同的异常模式，SP和LR分别使用哪个寄存器？

6.简述Cortex-A9存储器的体系架构。