

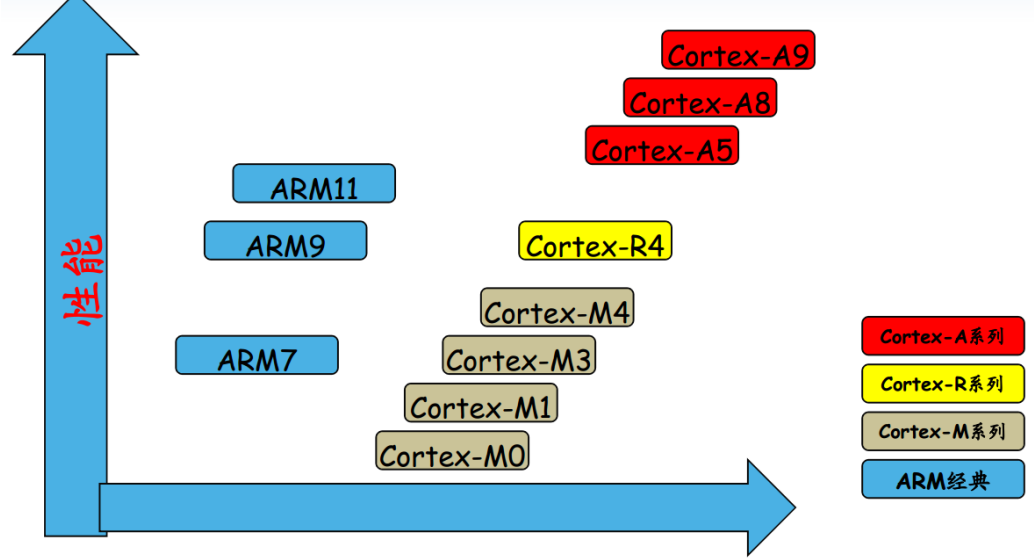
专题2-内部看ARM

一、ARM家族大检阅

1.1、名词

芯片	ARM核	指令架构
S3C2440	ARM920T	ARMv4
S3C6410	ARM1176JZF-S	ARMv6
S5PV210	Cortex-A8	ARMv7

1.2、ARM核演变图



ARM7 <= Cortex-M3
Cortex-R4 < ARM9 < ARM11 < Cortex-A5

1.3、对比说明

芯片	处理数度(MHz)	缓存(KB)	内存接口	支持OS	其他
S3C2440	405~532	16	SDRAM	WinCE/Linux	宣布停产
S3C6410	533~667	16	SDRAM/DDR	WinCE/Linux/Android	宣布停产
S5PV210	800~1000	32	DDR1/DDR2	WinCE/Linux/Android	在产
















二、ARM处理器工作模式






2.1、学会查阅官方文档 (ARM Architecture Reference Manual)


2.2、ARM处理器模式

Processor mode	Abbreviation	Mode number	Description
User	usr	0b10000	Normal program execution mode
FIQ	fiq	0b10001	Supports a high-speed data transfer or channel process
IRQ	irq	0b10010	Used for general-purpose interrupt handling
Supervisor	svc	0b10011	A protected mode for the operating system
Abort	abt	0b10111	Implements virtual memory and/or memory protection
Undefined	und	0b11011	Supports software emulation of hardware coprocessors
System	sys	0b11111	Runs privileged operating system tasks (ARMv4 and above)

三、ARM寄存器详解

Modes						
<div> <div>Privileged modes</div> <div>Exception modes</div> </div>						
User	System	Supervisor	Abort	Undefined	Interrupt	Fast interrupt
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	 R8_fiq
R9	R9	R9	R9	R9	R9	 R9_fiq
R10	R10	R10	R10	R10	R10	 R10_fiq
R11	R11	R11	R11	R11	R11	 R11_fiq
R12	R12	R12	R12	R12	R12	 R12_fiq
R13	R13	 R13_svc	 R13_abt	 R13_und	 R13_irq	 R13_fiq
R14	R14	 R14_svc	 R14_abt	 R14_und	 R14_irq	 R14_fiq
PC	PC	PC	PC	PC	PC	PC

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		 SPSR_svc	 SPSR_abt	 SPSR_und	 SPSR_irq	 SPSR_fiq

 indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode

3.1、ARM寄存器总共有37个寄存器（在不同模式下的同标号寄存器不是同一个寄存器（下三角标注））

3.2、31个通用寄存器

3.2.1、8个不分组寄存器（R0~R7）

3.2.2、22个分组寄存器（R8~R14）

寄存器R8至R12分别具有两个存储体物理寄存器。一个用于除FIQ模式之外的所有处理器模式，另一个用于FIQ模式。在需要具体说明哪个版本被引用时，第一组物理寄存器称为R8_usr到R12_usr，第二组称为R8_fiq到R12_fiq。

寄存器R8到R12在架构中没有任何专门的特殊用途。然而，对于仅使用寄存器R8至R14进行处理的中断，这些寄存器的独立FIQ模式版本的存在允许非常快速的中断处理。

寄存器R13和R14分别有六个存储体物理寄存器。一个用于用户和系统模式，其余五个中的每一个用于五种异常模式之一。如果需要具体说明哪个版本被引用，则使用以下格式的名称：

R13_ <模式>

R14_ <模式>

其中<mode>是usr, svc（用于Supervisor模式），abt, und, irq和fiq中适当的一个。

寄存器R13通常用作堆栈指针，也称为SP。ARMv6中引入的SRS指令是以特殊情况使用R13的唯一ARM指令。在Thumb指令集中还有其他这样的指令，如第A6章Thumb指令集中所述。

每个异常模式都有自己的R13版本。这些R13版本的合适用途取决于架构版本。

寄存器R14（也称为链接寄存器或LR）在架构中具有两个特殊功能：

- 1、在每个模式下，模式自己的R14版本用于保存子程序返回地址。
- 2、发生异常时（如中断），将适当的异常模式的R14版本设置为异常返回地址（由于某些异常的偏移量小于常量）。

3.2.3、1个程序计数器（R15(PC））

3.3、6个程序状态寄存器

3.3.1、1个CPSR程序状态寄存器(Current Program Status Register)

3.3.2、5个SPSR程序状态保存寄存器(Saved Program Status Register)

CPSR(当前程序状态寄存器)在任何处理器模式下被访问。它包含了条件标志位、中断禁止位、当前处理器模式标志以及其他的一些控制和状态位。每一种处理器模式下都有一个专用的物理状态寄存器，称为SPSR（备份程序状态寄存器）。当特定的异常中断发生时，这个寄存器用于存放当前程序状态寄存器的内容。在异常中断退出时，可以用SPSR来恢复CPSR。由于用户模式和系统模式不是异常中断模式，所以他沒有SPSR。当用户在用户模式或系统模式访问SPSR，将产生不可预知的后果。

3.3.3、CPSR和SPSR的格式

The format of the CPSR and the SPSRs is shown below.

31	30	29	28	27	26	25	24	23	20 19		16 15		10 9		8	7	6	5	4	0	
N	Z	C	V	Q	Res	J	RESERVED		GE[3:0]		RESERVED			E	A	I	F	T	M[4:0]		

A2.5.1 Types of PSR bits

PSR bits fall into four categories, depending on the way in which they can be updated:

- Reserved bits

Reserved for future expansion. Implementations must read these bits as 0 and ignore writes to them. For maximum compatibility with future extensions to the architecture, they must be written with values read from the same bits.
- User-writable bits

Can be written from any mode. The N, Z, C, V, Q, GE[3:0], and E bits are user-writable.
- Privileged bits

Can be written from any privileged mode. Writes to privileged bits in User mode are ignored. The A, I, F, and M[4:0] bits are privileged.
- Execution state bits

Can be written from any privileged mode. Writes to execution state bits in User mode are ignored. The J and T bits are execution state bits, and are always zero in ARM state.

Privileged MSR instructions that write to the CPSR execution state bits must write zeros to them, in order to avoid changing them. If ones are written to either or both of them, the resulting behavior is UNPREDICTABLE. This restriction applies only to the CPSR execution state bits, not the SPSR execution state bits.

3.3.3.1、重要位
N (Negative)

Is set to bit 31 of the result of the instruction. If this result is regarded as a two's complement signed integer, then N = 1 if the result is negative and N = 0 if it is positive or zero.

Z (Zero)

Is set to 1 if the result of the instruction is zero (this often indicates an equal result from a comparison), and to 0 otherwise.

I bit

Disables IRQ interrupts when it is set.

F bit

Disables FIQ interrupts when it is set.

M[4:0] (The mode bits)

A2.5.7 The mode bits

M[4:0] are the mode bits. These determine the mode in which the processor operates. Their interpretation is shown in Table A2-2.

Table A2-2 The mode bits

M[4:0]	Mode	Accessible registers
0b10000	User	PC, R14 to R0, CPSR
0b10001	FIQ	PC, R14_fiq to R8_fiq, R7 to R0, CPSR, SPSR_fiq
0b10010	IRQ	PC, R14_irq, R13_irq, R12 to R0, CPSR, SPSR_irq
0b10011	Supervisor	PC, R14_svc, R13_svc, R12 to R0, CPSR, SPSR_svc
0b10111	Abort	PC, R14_abt, R13_abt, R12 to R0, CPSR, SPSR_abt
0b11011	Undefined	PC, R14_und, R13_und, R12 to R0, CPSR, SPSR_und
0b11111	System	PC, R14 to R0, CPSR (ARMv4 and above)

Not all combinations of the mode bits define a valid processor mode. Only those combinations explicitly described can be used. If any other value is programmed into the mode bits M[4:0], the result is UNPREDICTABLE.

四、ARM寻址方式

4.1、概念

寻址方式就是处理器根据指令中给出的信息来找到指令所需操作数的方式。

4.2、寻址方式

4.2.1、立即数寻址

是一种特殊的寻址方式，操作数本身就在指令中给出，只要取出指令也就取到了操作数。这个操作数被称为立即数，对应的寻址方式也就叫做立即数寻址。例如以下指令：

ADD R0, R0, #0x3f ; R0←R0+0x3f

在以上两条指令中，第二个源操作数即为立即数，要求以“#”为前缀。

4.2.2、寄存器寻址

寄存器寻址就是利用寄存器中的数值作为操作数，这种寻址方式是一种执行效率较高的寻址方式。

ADD R0, R1, R2 ; R0←R1+R2

该指令的执行效果是将寄存器R1和R2的内容相加，其结果存放在寄存器R0中。

4.2.3、寄存器间接寻址

寄存器间接寻址就是寄存器中存放的是操作数在内存中的地址。例如以下指令：

LDR R0, [R2] ; R0←[R2]

在第一条指令中，以寄存器R2的值作为操作数的地址，在存储器中取得一个操作数后与R1相加，结果存入寄存器R0中。第二条指令将以R1的值为地址的存储器中的数据传送到R0中。

4.2.4、基址变址寻址

基址变址寻址就是将寄存器里的内容（基地址）与指令中给出的地址偏移量相加，从而得到操作数在内存中的地址：

LDR R0 [R1, #4] ; R0 ←[R1 + 4]

4.2.5、相对寻址

与基址变址寻址方式相类似，相对寻址PC寄存器的当前值为基地址，指令中的地址标号作为偏移量，将两者相加之后得到操作数的有效地址。以下程序段完成子程序的调用和返回，

跳转指令BL采用了相对寻址方式：

BL NEXT ; 跳转到子程序NEXT处执行

.....

NEXT

.....

MOV PC, LR ; 从子程序返回