ARM学习 - 知识整理 - HQ

[TOC]

注意

-

参考资料《从0学ARM》公众号 [一口Linux]

ARM基础知识节选 - 图片

如何学习ARM?

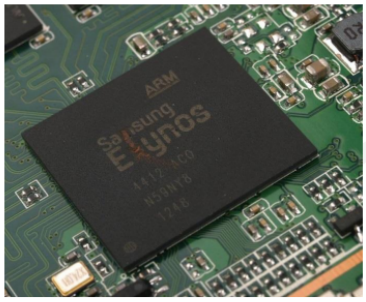
- 1. 了解CPU、SOC、总线、寄存器、异常、内存、中断、并发、调度基础知识
- 2. 掌握基本核心汇编指令
- 3. 熟练掌握C语言
- 4. 能看懂基本的电路图：数据线、信号线，常见的元器件
- 5. 掌握常见的硬件控制器原理，并给对应的外设编写驱动程序

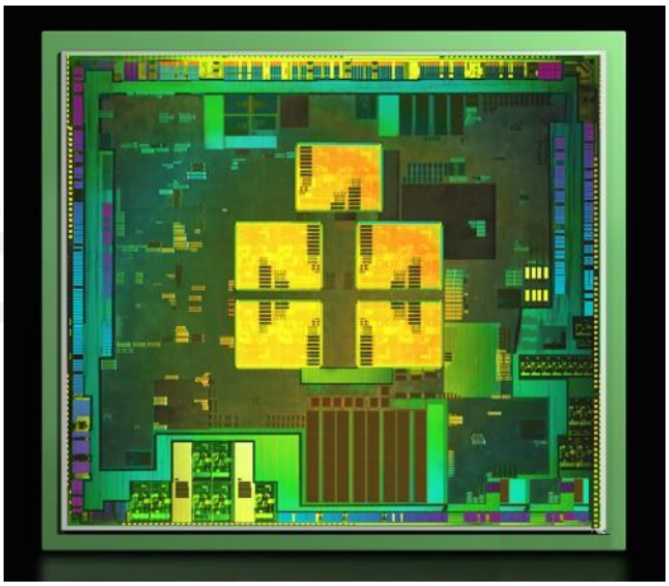
ARM内核

- ARM内核：
- 包括了寄存器组、指令集、总线、存储器映射规则、中断逻辑和调试组件等。
- 内核是由ARM公司设计并以销售方式授权给个芯片厂商使用的（ARM公司本身不做芯片）。
- 比如Cortex A8、A9都是ARMv7a 架构;Cortex M3、M4是ARMv7m架构;
- 前者是处理器（就是内核），后者是指令集的架构（也简称架构）。

SOC

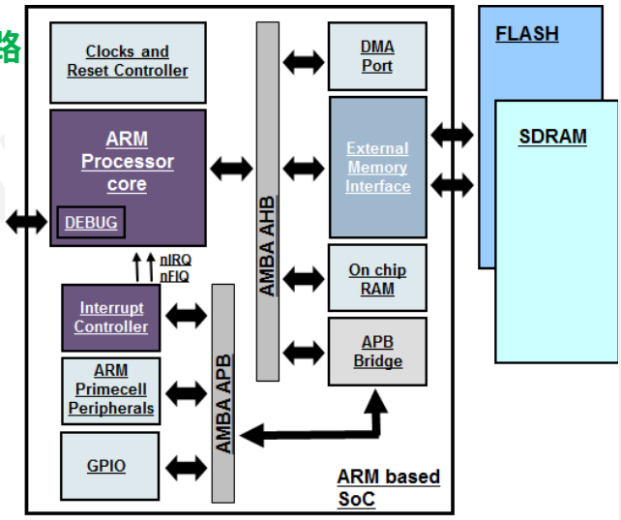
- SoC的全称叫做：
- System-on-a-Chip，
- 把系统都做在一个芯片上





一个典型的基于ARM的Soc架构

ARM Processor core 处理器核
Clocks and Reset Controller 时钟和复位电路
Interrupt Controller 中断控制器
ARM Propherals 外部设备
GPIO
DMA Port
External Memory Interface 外部内存接口
On chip RAM 偏上RAM
AHB、APB总线



存储器

高速缓冲存储器（Cache）：CPU可以直接访问，用来存放当前正在执行的程序中的活跃部分，以便快速地向CPU提供指令和数据。

它分为一级缓存（L1 Cache）、二级缓存（L2 Cache）、三级缓存（L3 Cache）这些数据，它位于内存和 CPU 之间，是一个读写速度比内存更快的存储器。

主存储器：可由CPU直接访问，用来存放当前正在执行的程序和数据。

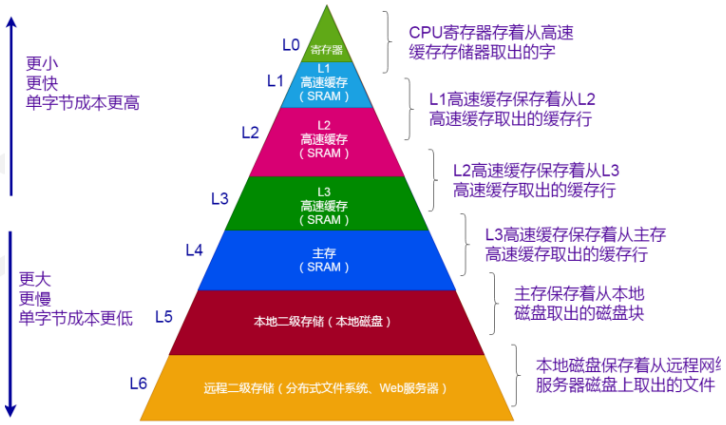
辅助存储器：设置在主机外部，CPU不能直接访问，用来存放暂时不参与运行的程序和数据，需要时再传送到主存。

速度

Cache： 几百到上千GB/s

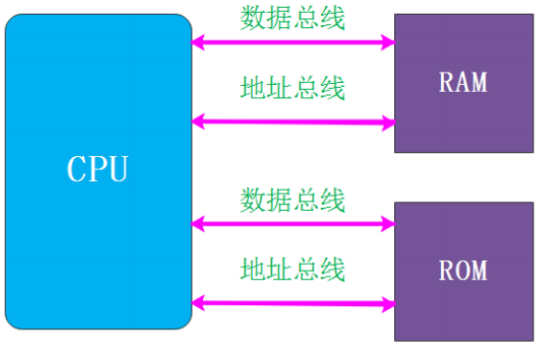
内存： 几十GB/s

Disk: 几百MB/s



哈弗结构

- 冯诺依曼结构是程序存储区和数据存储器都是可以放到内存中，统一编码的，而哈弗结构是分开编址的。



哪些处理器是哈佛架构、哪些处理器是冯诺依曼架构?

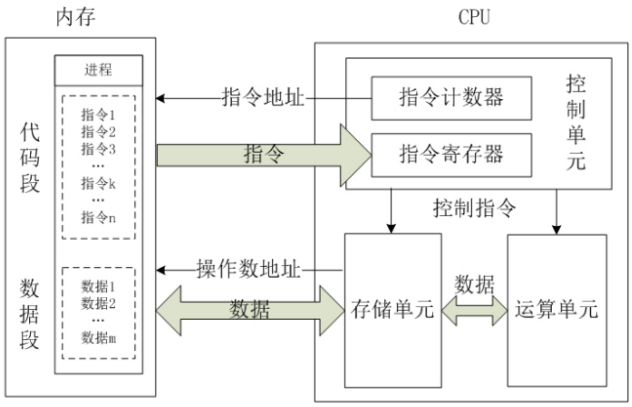
- MCU（单片机）几乎都是用哈佛结构，譬如广泛使用的51单片机、典型的STM32单片机（核心是ARM Cortex-M系列的）都是哈佛结构。
- PC和服务器芯片（譬如Intel AMD），ARM Cortex-A系列嵌入式芯片（譬如三星Exynos 4412，华为的麒麟970等手机芯片）等都是冯诺依曼结构。

混合结构

- 混合结构
- 比如基于Exynos 4412开发板上都配备了1024MB的DDR SDRAM，和8GB的EMMC。
- 正常工作时所有的程序和数据都从EMMC中加载到DDR中，也就是说不管你是指令还是数据，存储都是在EMMC中，运行时都在DDR中，再通过cache和寄存器送给CPU去加工处理。这就是典型的冯诺依曼系统。
- 但是，exynos 4412内部仍然有一定容量的64KB irom和64KB iram，这些irom和iram是用于SoC引导和启动的，芯片上电后首先会执行内部irom中固化的代码，其实执行这些代码时4412就好像一个MCU一样，irom就是他的flash，iram就是他的SRAM，这又是典型的哈佛结构。这其实就是混合式结构设计，而非纯粹设计。

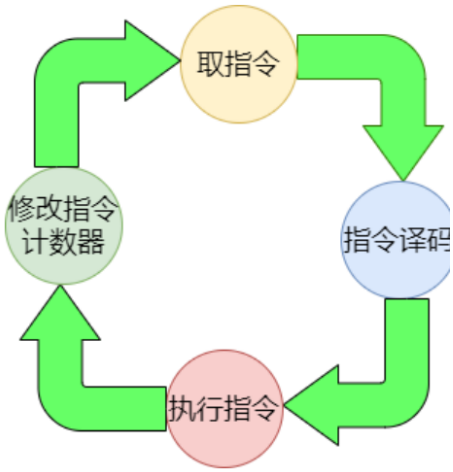
CPU的运行原理

- CPU的控制单元在时序脉冲的作用下，将指令计数器里所指向的指令地址(这个地址是在内存里的)送到地址总线上去，然后CPU将这个地址里的指令读到指令寄存器进行译码。
- 有运算器执行对应的机器指令，并将结果通过地址总线写回数据段



指令执行步骤

- 取指令：
- CPU的控制器从内存读取一条指令并放入指令寄存器。
- 指令译码：
- 指令寄存器中的指令经过译码，决定该指令应进行何种操作(就是指令里的操作码)、操作数在哪里(操作数的地址)。
- 执行指令：
- 分两个阶段“取操作数”和“进行运算”。
- 修改指令计数器：
- 决定下一条指令的地址。



ARM的基本数据类型

- ARM采用的是32位架构，ARM的基本数据类型有以下3种。
- Byte:
  - 字节，8bit。
- Halfword:
  - 半字，16bit(半字必须与2字节边界对齐）。
- Word:
  - 字，32bit(字必须与4字节边界对齐）。
- 存储器可以看做是序号为0~2^32-1的线性字节阵列。每一个字节都有唯一的地址。

ARM处理器工作模式

- Cortex-A系列的ARM处理器工作模式有8种

模式分类	处理器工作模式	异常模式	说明
非特权模式	用户（user）		用户程序运行模式
特权模式 该模式下可以访问系统资源	系统（system）		运行特权级的操作系统任务
	一般中断（IRQ）	异常模式 通常由系统异常状态切换进该组模式	普通中断模式
	快速中断（FIR）		快速中断模式
	管理（supervisor）		提供操作系统使用的一种保护模式，swi命令状态
	中止（abort）		虚拟内存管理和内存数据访问保护
	未定义指令终止（undefined）		支持通过软件仿真硬件的协处理
	monitor		用于执行安全监控代码的模式

1. 用户模式：

- 用户模式是用户程序的工作模式，
- 它运行在操作系统的用户态，
- 它没有权限去操作其它硬件资源，只能执行处理自己的数据，
- 也不能切换到其它模式下，要想访问硬件资源或切换到其它模式只能通过软中断（SWI）或产生异常。

2.系统模式

- 系统模式是特权模式，不受用户模式的限制。
- 用户模式和系统模式共用一套寄存器，
- 操作系统在该模式下可以方便的访问用户模式的寄存器，而且操作系统的一些特权任务可以使用这个模式访问一些受控的资源。

3. 一般中断模式

- 一般中断模式也叫普通中断模式，用于处理一般的中断请求
- 通常在硬件产生中断信号之后自动进入该模式
- 该模式为特权模式，可以自由访问系统硬件资源

4.快速中断模式

- 快速中断模式是相对一般中断模式而言的，它是用来处理对时间要求比较紧急的中断请求
- 主要用于高速数据传输及通道处理中。

5.管理模式(SVC)

- 管理模式是CPU上电后默认模式
- 因此在该模式下主要用来做系统的初始化
- 软中断处理也在该模式下，当用户模式下的用户程序请求使用硬件资源时通过软件中断进入该模式。

6.中止模式

- 中止模式用于支持虚拟内存或存储器保护，
- 当用户程序访问非法地址，没有权限读取的内存地址时，会进入该模式，

- linux下编程时经常出现的segment fault通常都是在该模式下抛出返回的。

7.未定义模式

- 未定义模式用于支持硬件协处理器的软件仿真，
- CPU在指令的译码阶段不能识别该指令操作时，会进入未定义模式。

8. Monitor

- 是为了安全而扩展出的用于执行安全监控代码的模式；也是一种特权模式

模式切换

- ARM微处理器的运行模式可以通过软件改变，也可以通过外部中断或异常处理改变。

- 应用程序运行在用户模式下，当处理器运行在用户模式下时，某些被保护的系统资源是不能被访问的。

异常（Exception）

- 指由处理器执行指令导致原来运行程序的中止，
- 异常与指令运行相关，是CPU执行程序产生的，是同步的，可分为精确异常和非精确异常。
- 异常处理遵守严格的程序顺序，不能嵌套，只有当第一个异常处理完并返回后才能处理后续的异常。

异常源

异常源	描述
Reset	上电时执行
Undef	当流水线中的某个非法指令到达执行状态时执行
SWI	当一个软中断指令被执行完的时候执行
Prefetch	当一个指令被从内存中预取时，由于某种原因而失败，如果它能到达执行状态这个异常才会产生
Data	如果一个预取指令试图存取一个非法的内存单元，这时异常产生
IRQ	通常的中断
FIQ	快速中断

异常源与模式关系

- 快速中断请求异常进入快速中断模式，支持高速数传输及通道处理（FIQ异常响应时进入此模式）；
- 中断请求异常进入中断模式，用于通用中断处理
- 预取指中止，数据中止异常进入中止模式，用于支持虚拟内存和/或存储器保护；
- 未定义指令异常进入未定义模式，
- 支持硬件协处理器的软件仿真软件中断(swi)，复位异常(reset)进入管理模式，操作系统保护代码

异常源与模式关系

- 快速中断请求异常进入快速中断模式，支持高速数传输及通道处理（FIQ异常响应时进入此模式）；
- 中断请求异常进入中断模式，用于通用中断处理
- 预取指中止，数据中止异常进入中止模式，用于支持虚拟内存和/或存储器保护；
- 未定义指令异常进入未定义模式，
- 支持硬件协处理器的软件仿真软件中断(swi)，复位异常(reset)进入管理模式，操作系统保护代码

ARM寄存器

- Cortex A系列ARM处理器共有40个32位寄存器,其中33个为通用寄存器,7个为状态寄存器。
- usr模式和sys模式共用同一组寄存器。

ARM state general registers and program counter

System and User	FIQ	Supervisor	Abort	IRQ	Undefined	Secure monitor
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_fiq	r8	r8	r8	r8	r8
r9	r9_fiq	r9	r9	r9	r9	r9
r10	r10_fiq	r10	r10	r10	r10	r10
r11	r11_fiq	r11	r11	r11	r11	r11
r12	r12_fiq	r12	r12	r12	r12	r12
r13	r13_fiq	r13_svc	r13_abt	r13_irq	r13_und	r13_mon
r14	r14_fiq	r14_svc	r14_abt	r14_irq	r14_und	r14_mon
r15	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)

ARM state program status registers

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

= banked register

关注

3. 寄存器R13（sp）

- 在ARM指令中常用作堆栈指针,用户也可使用其他的寄存器作为堆栈指针,而在Thumb指令集中,某些指令强制性的要求使用R13作为堆栈指针。
- 由于处理器的每种运行模式均有自己独立的物理寄存器R13，在用户应用程序的初始化部分，一般都要初始化每种模式下的R13，使其指向该运行模式的栈空间。
- 这样，当程序的运行进入异常模式时，可以将需要保护的寄存器放入R13所指向的堆栈，而当程序从异常模式返回时，则从对应的堆栈中恢复，采用这种方式可以保证异常发生后程序的正常执行。

4. R14（LR）链接寄存器(Link Register)

- 当执行子程序调用指令(BL)时,R14可得到R15(程序计数器PC)的备份
- 在每一种运行模式下，都可用R14保存子程序的返回地址

- 1.当用BL或BLX指令调用子程序时，将PC的当前值复制给R14，
- 2. 执行完子程序后，又将R14的值复制回PC，即可完成子程序的调用返回。
- 以上的描述可用指令完成。
- MOV PC, LR 或者 BX LR
- STMFD SP！,{,LR}　　LDMFD SP！,{,PC}

5. R15(PC)程序计数器

- 寄存器R15用作程序计数器(PC)

- 在ARM状态下,位[1:0]为0,位[31:2]用于保存PC,在Thumb状态下,位[0]为0,位[31:1]用于保存PC。
- 比如如果pc的值是0x40008001,那么在寻址的时候其实会查找地址0x40008000，低2位会自动忽略掉。

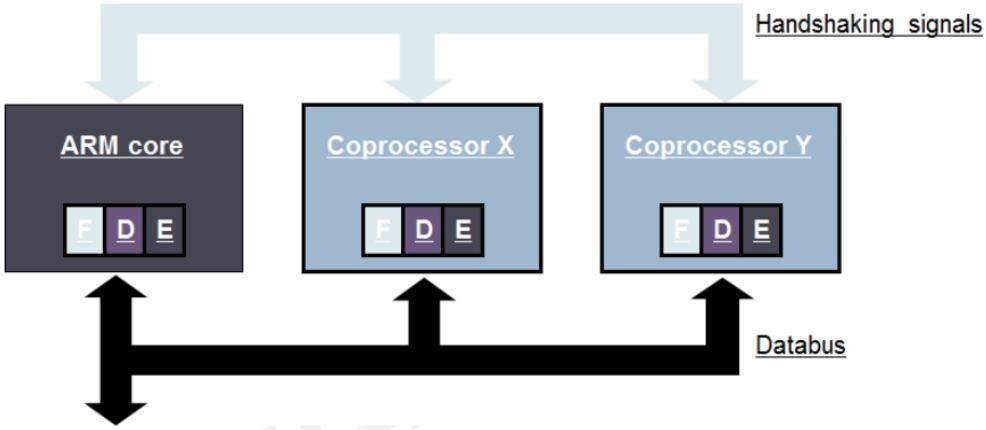
- 由于ARM体系结构采用了多级流水线技术，对于ARM指令集而言，PC总是指向当前指令的下两条指令的地址,即PC的值为当前指令的地址值加8个字节。
- 即：PC值=当前程序执行位置+8

6. CPSR、SPSR

- CPSR(Current Program Status Register，当前程序状态寄存器)，CPSR可在任何运行模式下被访问
- 每一种运行模式下又都有一个专用的物理状态寄存器，称为SPSR(Saved Program Status Register，备份的程序状态寄存器)，
- 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，当在这两种模式下访问SPSR，结果是未知的。

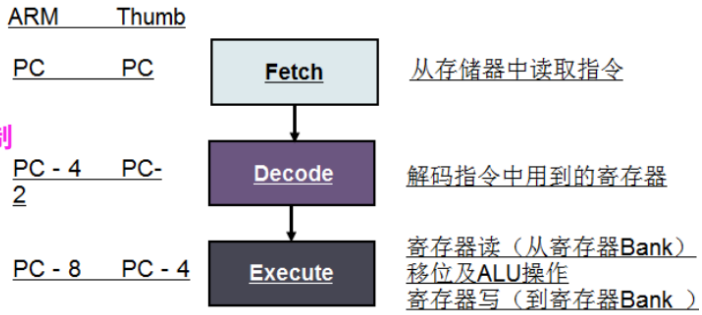
协处理器



ARM体系结构允许通过增加协处理器来扩展指令集。最常用的协处理器是用于控制片上功能的系统协处理器。
例如，控制Cache和存储管理单元MMU的CP15协处理器、设置异常向量表地址的mcr指令。

3级流水线

（1）取指令
从寄存器装载一条指令。
（2）译码（decode）
识别被执行的指令，并为下一个周期准备数据通路的控制信号。在这一级，指令占有译码逻辑，不占用数据通路。
（3）执行
处理指令并将结果写回寄存器。



当处理器执行简单的数据处理指令时，流水线使得平均每个时钟周期能完成1条指令。但一条指令需要3个时钟周期来完成，因此有3个时钟周期的延时，但吞吐率是每个周期一条指令。
对于3级流水线，PC寄存器里的值并不是正在执行的指令的地址，而是预取指令的地址

从经典ARM系列到现在Cortex系列，ARM处理器的结构在向复杂的阶段发展，但没改变的是CPU的取址指令和地址关系，不管是几级流水线，**都可以按照最初的3级流水线的操作特性来判断其当前的PC位置**。

ARM指令助记符

条件码	助记符后缀	标 志	含 义
0000	EQ	Z置位	相等
0001	NE	Z清零	不相等
0010	CS	C置位	无符号数大于或等于
0011	CC	C清零	无符号数小于
0100	MI	N置位	负数
0101	PL	N清零	正数或零
0110	VS	V置位	溢出
0111	VC	V清零	未溢出
1000	HI	C置位Z清零	无符号数大于
1001	LS	C清零Z置位	无符号数小于或等于
1010	GE	N等于V	带符号数大于或等于
1011	LT	N不等于V	带符号数小于
1100	GT	Z清零且（N等于V）	带符号数大于
1101	LE	Z置位或（N不等于V）	带符号数小于或等于
1110	AL	忽略	无条件执行

- {<cond>}：条件码域
- <opcode>：操作码域
- {S}：条件码设置域：这是一个可选项，当在指令中设置{S}域时，指令执行的结果将会影响程序状态寄存器CPSR中相应的状态标志
- <Rd>：目的操作数：总是一个寄存器
- <Rn>：第一操作数：也必须是个寄存器
- <shift_op2>：第二操作数：在第二操作数中可以是寄存器、内存存储单元或者立即数

5.软件中断指令（swi）异常

- 该异常是应用程序自己调用时产生的，用户程序申请访问硬件资源时需要调用该指令
- 例如：printf()打印函数，
  - 用户程序要想实现打印必须申请使用显示器，而用户程序又没有外设硬件的使用权，只能通过使用软件中断指令切换到内核态，
  - 通过操作系统内核代码来访问外设硬件，内核态是工作在特权模式下，操作系统在特权模式下完成将用户数据打印到显示器上。
- 这样做的目的无非是为了保护操作系统的安全和硬件资源的合理使用，该异常在管理模式（SVC）下处理。

6.数据中止访问异常

- 该异常发生在要访问数据地址不存在或者为非法地址时，该异常在中止异常模式下处理。（segment error）

ARM的异常优先级

- 依次递减
  - Reset→
  - Data abort→
  - FIQ→
  - IRQ→
  - Prefetch abort→
  - Undefined instruction/SWI。

异常与模式关系

- reset异常-----》SVC模式
- fiq -----》快中断模式，
  - 支持高速数传输及通道处理（FIQ异常响应时进入此模式）
- irq -----》中断模式
  - 用于通用中断处理
- prefetch预取指中止，数据中止异常-----》中止模式，
  - 用于支持虚拟内存和/或存储器保护
- undef未定义指令异常-----》未定义模式
  - 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）
- swi软件中断，复位异常-----》管理模式
  - 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）

ARM异常处理完整过程

- 一、 硬件阶段：
  - 4大步3小步
- 二、 异常处理
  - 保存现场、异常处理
- 三、 异常返回
  - 回复现场

一、硬件阶段- 4大步3小步

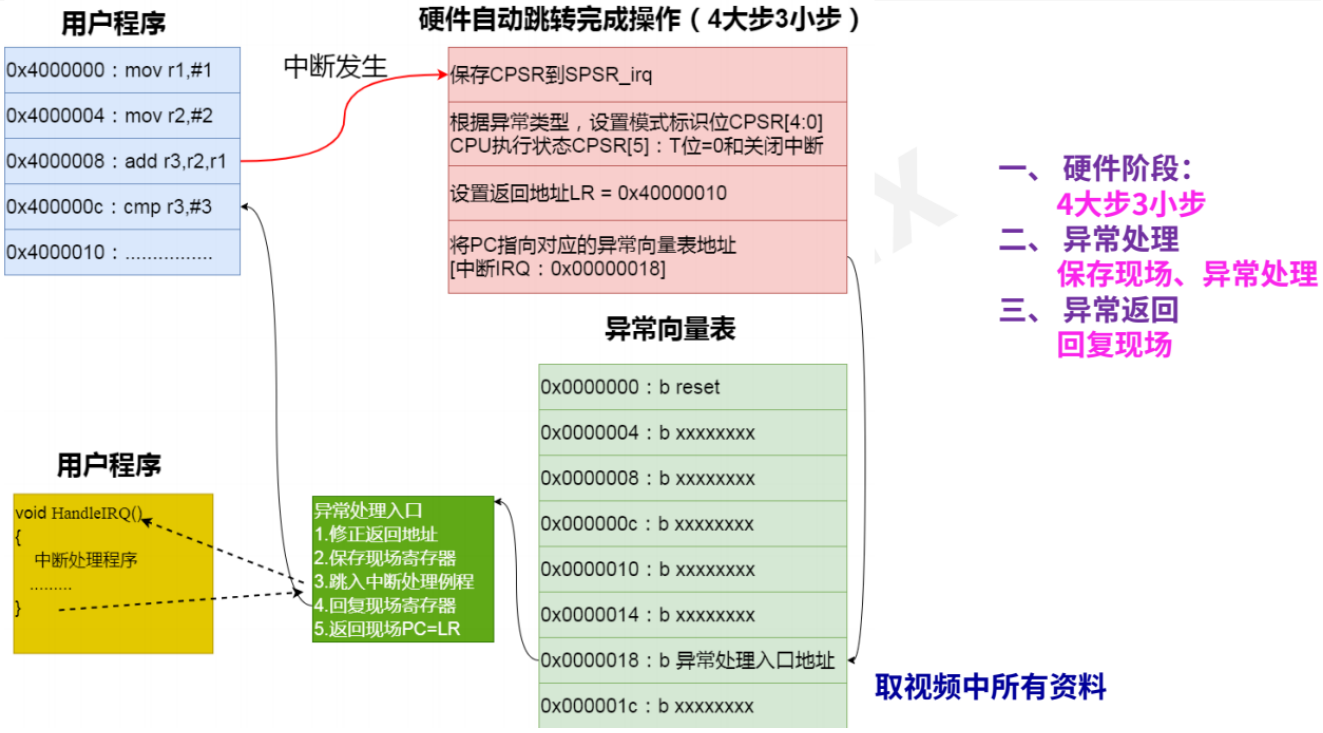
- 1. 保存执行状态：将CPSR复制到发生的异常模式下SPSR中；
- 2. 模式切换：
  - CPSR模式位强制设置为与异常类型相对应的值，
  - 处理器进入到ARM执行模式，
  - 禁止所有IRQ中断，当进入FIQ快速中断模式时禁止FIQ中断；
- 3. 保存返回地址：将下一条指令的地址（被打断程序）保存在LR(异常模式下LR_excep)中。
- 4. 跳入异常向量表：强制设置PC的值为相应异常向量地址，跳转到异常处理程序中。

异常向量表

- 异常向量表是一段特定内存地址空间，每种ARM异常对应一个字长空间（4Bytes），正好是一条32位指令长度，当异常发生时，CPU强制将PC的值设置为当前异常对应的固定内存地址。

地址	异常	进入模式
0x00000000	复位	管理模式
0x00000004	未定义指令	未定义模式
0x00000008	软件中断	管理模式
0x0000000C	中止（预取）	中止模式
0x00000010	中止（数据）	中止模式
0x00000014	保留	保留
0x00000018	中断 IRQ	中断模式
0x0000001C	快中断 FIQ	快中断模式

中断处理流程举例


用户程序
0x4000000：mov r1,#1
0x4000004：mov r2,#2
0x4000008：add r3,r2,r1
0x400000c：cmp r3,#3
0x4000010：.....
中断发生
硬件自动跳转完成操作（4大步3小步）
保存CPSR到SPSR_irq
根据异常类型，设置模式标识位CPSR[4:0]
CPU执行状态CPSR[5]：T位=0和关闭中断
设置返回地址LR = 0x40000010
将PC指向对应的异常向量表地址
[中断IRQ：0x00000018]
异常向量表
0x0000000：b reset
0x0000004：b xxxxxxxx
0x0000008：b xxxxxxxx
0x000000c：b xxxxxxxx
0x0000010：b xxxxxxxx
0x0000014：b xxxxxxxx
0x0000018：b 异常处理入口地址
0x000001c：b xxxxxxxx
用户程序
void HandleIRQ()
{
中断处理程序
.....
}
异常处理入口
1.修正返回地址
2.保存现场寄存器
3.跳入中断处理例程
4.回复现场寄存器
5.返回现场PC=LR
一、 硬件阶段：
4大步3小步
二、 异常处理
保存现场、异常处理
三、 异常返回
回复现场
取视频中所有资料