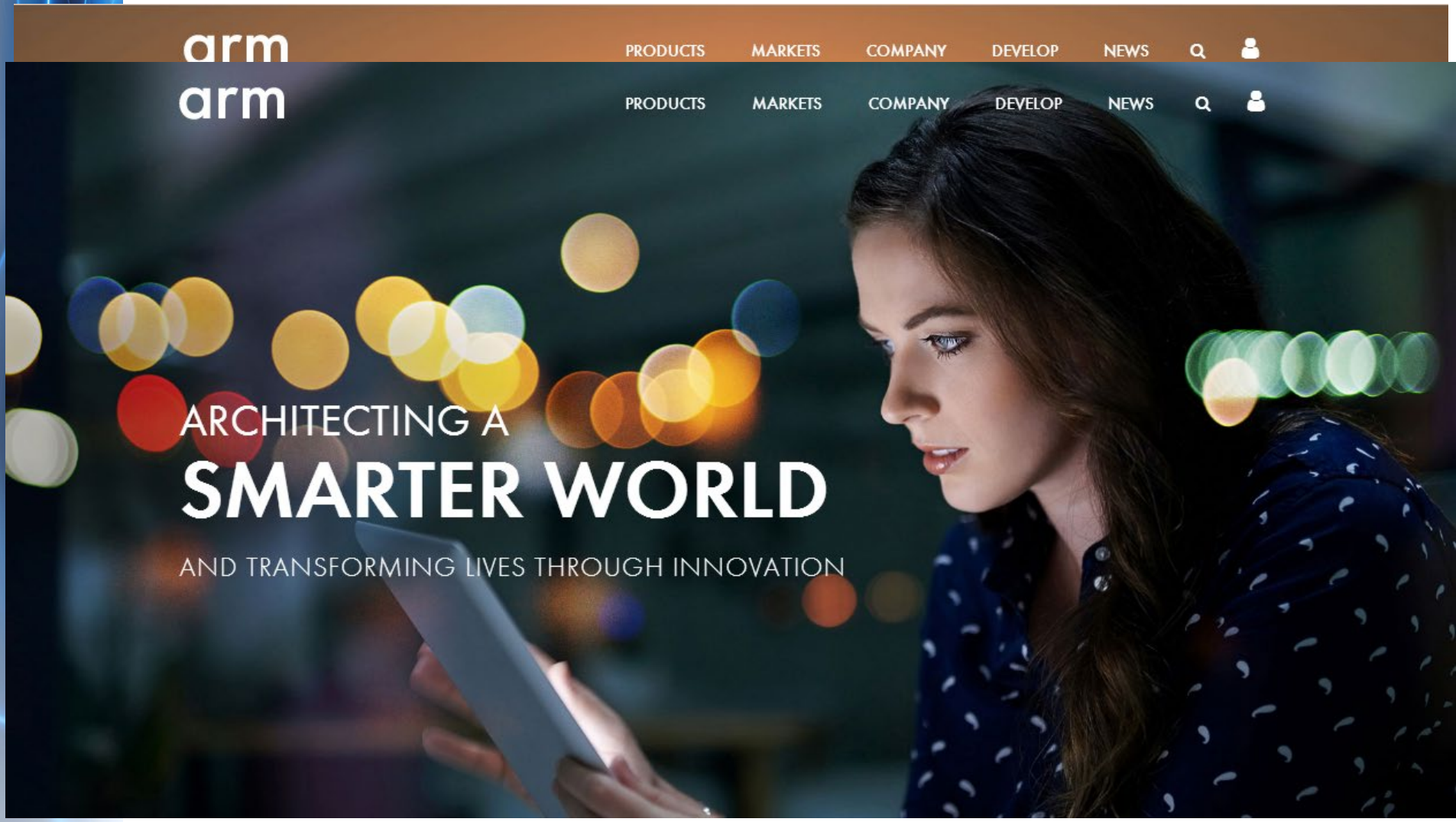
本章的主要内容为：

- 2.1 ARM体系结构和技术特征
- 2.2 ARM体系结构不同版本的发展概述
- 2.3 Thumb技术介绍
- 2.4 ARM处理器工作状态
- 2.5 ARM处理器工作模式
- 2.6 ARM寄存器组成
- 2.7 ARM异常中断

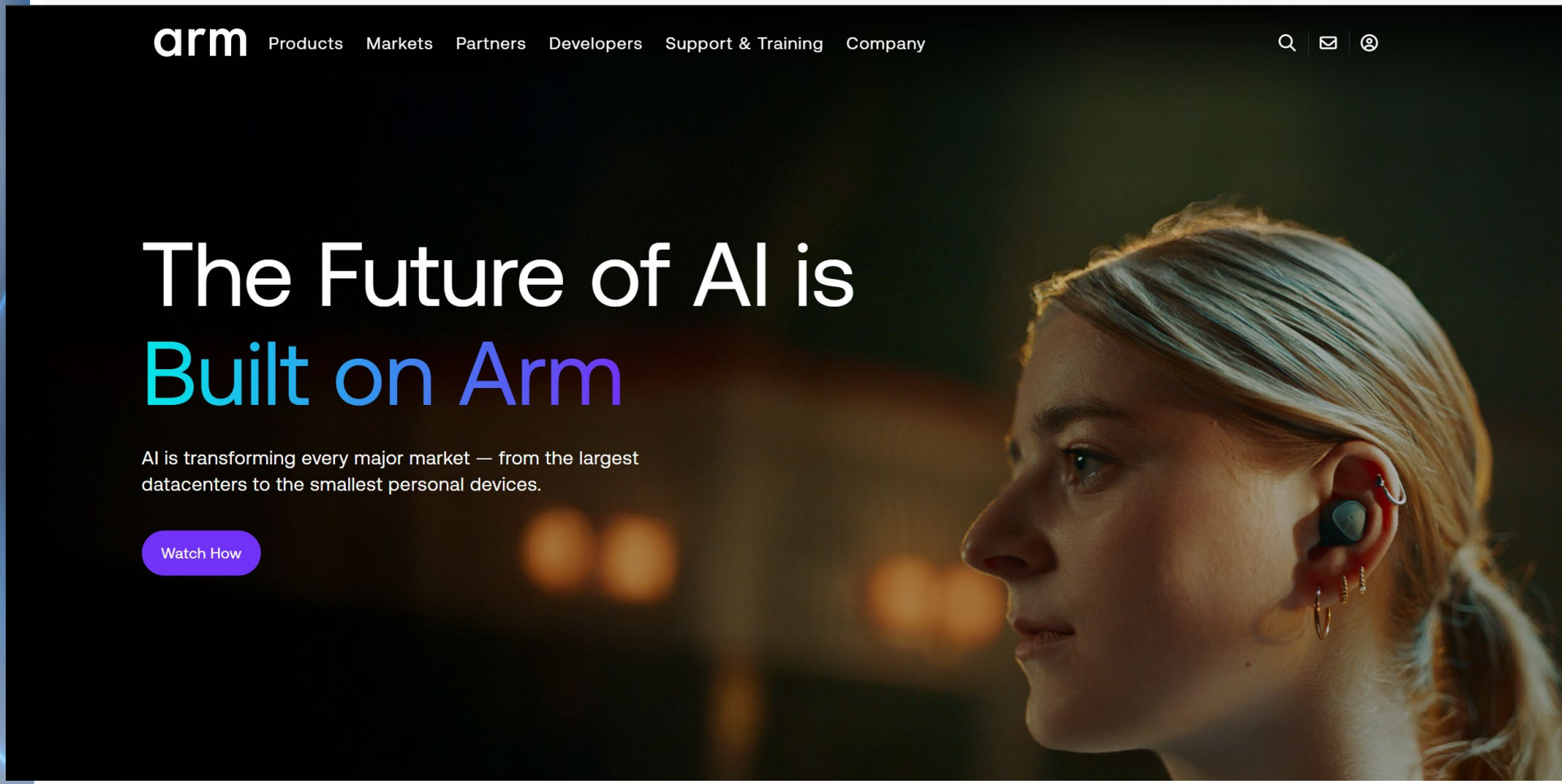
本章的主要内容为：

- 2.8 ARM组织结构简介
- 2.9 ARM存储器接口及存储器层次
- 2.10 ARM协处理器
- 2.11 ARM片上总线AMBA
- 2.12 ARM核综述
- 2.13 基于ARM核的芯片选择

2.1 ARM体系结构和技术特征



2.1.1 ARM发展历程



your chip.

ARM发展历程

- 第一片ARM处理器是1983年10月到1985年4月间由在位于英国剑桥的Acorn Computer公司开发
- 1990年，为广泛推广ARM技术而成立了独立的公司
- 20世纪90年代， ARM快速进入世界市场

ARM发展历程

- 在ARM的发展历程中，从ARM7开始，ARM核被普遍认可和广泛使用
- 1995年 StrongARM 问世
- XScale是下一代StrongARM芯片的发展基础
- ARM10TDMI是ARM处理器核中的高端产品
- ARM11是ARM家族中性能最强的一个系列

2.1.2 ARM体系结构的技术特征

嵌入式微处理器按指令系统分类
- 精简指令集系统RISC
- 复杂指令集系统CISC

RISC结构一般具有如下特点：

（1）单周期的执行。
（2）采用高效的流水线操作。
（3）无微代码的硬连线控制。
（4）指令格式的规格化和简单化。
（5）采用面向寄存器组的指令。
（6）采用Load/Store（装载/存储）指令结构。
（7）注重编译的优化，力求有效地支撑高级语言程序。

2.2 ARM体系结构版本的发展概述

2.2.1 ARM体系结构的基本版本

2.2.2 ARM体系结构的演变

2.2.3 ARM体系结构的命名规则

2.2.1ARM体系结构的基本版本

1. v1版本

v1版本ARM处理器没有商品化，只出现在ARM1原型机上。它的主要特点有：

（1）26位的地址空间，寻址空间64MB；

（2）只有基本的数据处理指令，甚至没有乘法指令；

（3）基于字节、半字和字的Load/Store存储器访问指令；

（4）子程序调用指令（BL）和链接指令；

（5）操作系统调用的软件中断指令（SWI）。

2. v2版本

对v1版本进行了扩展和完善。仍旧采用26位地址空间和64M寻址空间。它的主要特点有：

（1）增加了32位乘法指令和乘加指令；

（2）支持协处理器指令；

（3）对快速中断模式支持；

（4）支持最基本的存储器与寄存器交换指令SWP/SWPB。

3. v3版本

该版本在体系结构上较以前的版本有很大变化。基于该版本的ARM6处理器，做为IP核独立的处理器，具有片上高速缓存、MMU和写缓存的集成CPU。它的主要特点有：

（1）寻址空间增加到32位（4G）；

（2）增加了当前程序状态寄存器（CPSR）保存当前程序运行的状态信息；

（3）增加了备份程序状态寄存器（SPSR），在程序运行被异常中断时保存现场；

（4）增加了MRS/MSR指令，以访问新增的CPSR/SPSR寄存器；

（5）增加了中止和未定义两种异常模式，以方便操作系统使用数据访问中止异常、指令预取中止异常和未定义指令异常；

（6）改进了从异常返回指令。

4. v4版本

该版本在v3版本的基础上做了进一步的扩充，是目前被应用最广的ARM体系结构，ARM7TDMI、ARM9、StrongARM等都采用该结构。它的主要特点有：

（1）增加了对有符号、无符号半字及有符号字节的存/取指令；

（2）增加 T变种，引入Thumb状态，处理器工作在该状态下时，指令集为新增的16位Thumb指令集；

（3）增加了系统模式，该模式下处理器使用用户寄存器；

（4）完善了软件中断（SWI）指令功能；

（5）把一些未使用的指令空间捕获为未定义指令。

5. v5版本

在v4版本的基础上增加了一些新的指令。ARM9E、ARM10和Intel的XScale处理器都采用该版本结构。它的主要特点有：

（1）改进了ARM指令集和Thumb指令集的混合使用效率；

（2）增加了带有链接和交换的转移指令（BLX）、计数前导零指令（CLZ）、软件断点指令（BKPT）；

（3）v5TE版本中增加了DSP（数字信号处理）指令集，包括全部算法和16位指令集；

支持新的Java，提供字节代码执行的硬件和优化软件加速性能。

6. v6版本

该版本2001年发布，并应用在2002年发布的ARM11处理器中。该版本降低耗电量的同时提高了图像处理能力，适合无线和消费类电子产品；高数据吞吐量和高性能的结合。它的主要特点有：

（1）支持多微处理器内核；

（2）Thumb代码压缩技术；

（3）引入Jazelle技术，提高了Java性能，降低了Java应用程序对内存的空间占用；

（4）通过SIMD（单指令多数据流）技术，提高了音/视频处理能力。

7. v7版本

v7版本架构是在v6版本的基础上诞生的，对于早期的ARM处理器软件提供了较好的兼容性。它的主要特点有：

（1）采用了在Thumb代码压缩技术上发展的**Thumb-2**技术，比纯32位代码减少了31%的内存占用，减小了系统开销，能够提供比基于Thumb技术的解决方案高出38%的性能；

（2）首次采用NEON信号处理扩展集，它是一个结合64位和128位的SIMD指令集，对H.264和MP3等媒体解码提供加速，将DSP和媒体处理能力提高了近4倍，并支持改良的浮点运算；

（3）支持改良的运行环境，迎合不断增加的JIT（Just In Time）和DAC（Dynamic Adaptive Compilation）技术的使用。

该架构定义了三大系列：

Cortex-A系列：面向基于虚拟内存的操作系统和用户应用，主要用于运行各种嵌入式操作系统（Linux、WindowsCE、Android、Symbian等）的消费娱乐和无线产品；

Cortex-M系列：主要面向微控制器领域，用于对成本和功耗敏感的终端设备，如智能仪器仪表、汽车和工业控制系统、家用电器、传感器、医疗器械等；

Cortex-R系列：该系列主要用于具有严格的实时响应限制的深层嵌入式实时系统。

8. v8版本

2011年11月，ARM公司发布了新一代处理器架构ARMv8的部分技术细节，这是ARM公司的首款支持64位指令集的处理器架构，将被首先用于对扩展虚拟地址和64位数据处理技术有更高要求的产品领域，如企业应用、高档消费电子产品。目前的ARMv7架构的主要特性都将在ARMv8架构中得以保留或进一步拓展，如TrustZone技术、虚拟化技术及NEON advanced SIMD技术等。ARMv8 架构将64位架构支持引入ARM架构中，其中包括：

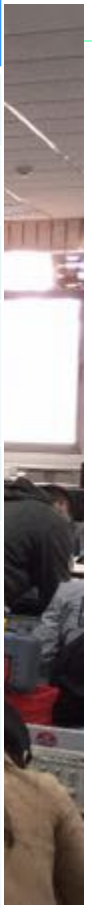
（1）64位通用寄存器、SP（堆栈指针）和PC（程序计数器）；

（2）64位数据处理和扩展的虚拟寻址；

（3）两种主要执行状态：AArch64（64位执行状态）和AArch32（32位执行状态）；

ARM体系结构总结

体系结构	ARM内核版本
v1	ARM1
v2	ARM2
v2a	ARM2aS、ARM3
v3	ARM6、ARM600、ARM610、ARM7、ARM700、ARM710
v4	Strong ARM、ARM8、ARM810
v4T	ARM7TDMI、ARM720T、ARM740T、ARM9TDMI、ARM920T、ARM940T
v5TE	ARM9E-S、ARM10TDMI、ARM1020E
v6	ARM11、ARM1156T2-S、ARM1156T2F-S、ARM1176JZF-S、ARM11JZF-S
v7	ARM Cortex-M、ARM Cortex-R、ARM Cortex-A
v8	Cortex-A53/57、Cortex-A72等



回顾

- 嵌入式软件开发中的两个重要的考虑的方面是提高软件的 [填空1] 和 [填空2] 。
- ARM?
- V?
- ARM9
- ARM11
- CORTEX-?
- CORTEX-A57?

2.2.2ARM体系结构的演变

1）Thumb指令集（T变种）

- Thumb指令集是把32位的ARM指令集的一个子集重新编码后而形成的一个特殊的16位的指令集

2）长乘指令（M变种）

- 长乘指令是一种生成64位相乘结果的乘法指令（此指令为ARM指令），M变种增加了两条长乘指令

ARM体系结构的演变

3）增强型DSP指令（E变种）

- E变种的ARM体系增加了一些增强处理器对典型的DSP算法处理能力的附加指令

4）Java加速器Jazelle（J变种）

- ARM的Jazelle技术是Java语言和先进的32位RISC芯片完美结合的产物

5）ARM媒体功能扩展（SIMD变种）

2.2.3ARM体系结构的命名规则

表示ARM/Thumb体系版本的命名格式的
ARM/Thumb体系版本由下面几部分组成的：

- 基本字符串ARMv。
- 基本字符串后为ARM指令集版本号，目前是1-8的数字字符。
- ARM指令集版本号后为表示所含变种的字符。由于在ARM体系版本4以后，M变种成为系统的标准部件，所以字符M通常也不单独列出来。
- 最后使用的字符x表示排除某种功能。

体系结构版本

名称	ARM指令集版本	Thumb指令集版本	长整型乘法指令	增强型DSP指令
ARMv3	3	无	否	否
ARMv3M	3	无	是	否
ARMv4xM	4	无	否	否
ARMv4	4	无	是	否
ARMv4TxM	4	1	否	否
ARMv4T	4	1	是	否
ARMv5xM	5	无	否	否
ARMv5	5	无	是	否
ARMv5TxM	5	2	否	否
ARMv5T	5	2	是	否
ARMv5TExP	5	2	是	是
ARMv5TE	5	2	是	是

2.3 Thumb技术介绍

ARM的RISC体系结构的发展中已经提供了低功耗、小体积、高性能的方案。而为了解决代码长度的问题，ARM体系结构又增加了T变种，开发了一种新的指令体系，这就是Thumb指令集，它是ARM技术的一大特色。

2.3.1Thumb的技术概述

2.3.2Thumb的技术实现

2.3.3Thumb技术的特点

2.3.1Thumb的技术概述

- Thumb是ARM体系结构的扩展。它有从标准32位ARM指令集抽出来的36条指令格式，可以重新编成16位的操作码。这能带来很高的代码密度
- ARM7TDMI是第一个支持Thumb的核，支持Thumb的核仅仅是ARM体系结构的一种扩展，所以编译器既可以编译Thumb代码，又可以编译ARM代码
- 支持Thumb的ARM体系结构的处理器状态可以方便的切换、运行到Thumb状态，在该状态下指令集是16位的Thumb指令集

2.3.2Thumb技术的特点

在性能和代码大小之间取得平衡，在需要较低的存储代码时采用Thumb指令系统，但有比纯粹的16位系统有较高的实现性能，因为实际执行的是32位指令，用Thumb指令编写最小代码量的程序，却取得以ARM代码执行的最好性能

Thumb技术的特点

与ARM指令集相比．Thumb指令集具有以下局限

- 完成相同的操作，Thumb指令通常需要更多的指令，因此在对系统运行时间要求苛刻的应用场合ARM指令集更为适合；
- Thumb指令集没有包含进行异常处理时需要的一些指令，因此在异常中断时，还是需要使用ARM指令，这种限制决定了Thumb指令需要和ARM指令配合使用。

2.4 ARM处理器工作状态

ARM处理器核可以工作在以下2种状态

- ARM状态

32位，ARM状态下执行字对准的32位ARM指令；

- Thumb状态

16位，Thumb状态下执行半字对准的16位Thumb指令。在Thumb状态下，程序计数器PC使用位1选择另一个半字。

ARM处理器工作状态

在程序执行的过程中，处理器可以在两种状态下切换

- ARM和Thumb之间状态的切换不影响处理器的模式或寄存器的内容。
- ARM指令集和Thumb指令集都有相应的状态切换命令。
- ARM处理器在开始执行代码时，只能处于ARM状态。

ARM处理器工作状态

ARM处理器在两种工作状态之间切换方法

- 进入Thumb状态

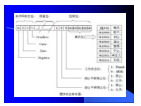
当操作数寄存器Rm的状态位bit［0］为1时，执行BX Rm指令进入Thumb状态。如果处理器在Thumb状态进入异常，则当异常处理（IRQ，FIQ，Undef，Abort和SWI）返回时，自动切换到Thumb状态。

- **进入ARM状态**

当操作数寄存器Rm的状态位bit［0］为0时，执行BX Rm指令进入ARM状态。如果处理器进行异常处理（IRQ，FIQ，Undef，Abort和SWI），在此情况下，把PC放入异常模式链接寄存器LR中，从异常向量地址开始执行也可以进入ARM状态。

2.5 ARM处理器工作模式

CPSR（当前程序状态寄存器）的低5位用于定义当前操作模式， 如图示



CPSR[4:0]	模式	用途	可访问的寄存器
10000	用户	正常用户模式，程序正常执行模式	PC，R14~R0，CPSR
10001	FIQ	处理快速中断，支持高速数据传送或通道处理	PC，R14_fiq~R8_fiq，R7~R0，CPSR，SPSR_fiq
10010	IRQ	处理普通中断	PC，R14_irq~R13_fiq，R12~R0，CPSR，SPSR_irq
10011	SVC	操作系统保护模式，处理软件中断（SWI）	PC，R14_svc~R13_svc，R12~R0，CPSR，SPSR_svc
10111	中止	处理存储器故障、实现虚拟存储器和存储器保护	PC，R14_abt~R13_abt，R12~R0，CPSR，SPSR_abt
11011	未定义	处理未定义的指令陷阱，支持硬件协处理器的软件仿真	PC，R14_und~R13_und，R12~R0，CPSR，SPSR_und
11111	系统	运行特权操作系统任务	PC，R14~R0，CPSR

ARM处理器工作模式

除用户模式外的其他6种模式称为特权模式

特权模式中除系统模式以外的5种模式又称为异常模式，即

- FIQ（Fast Interrupt Request）
- IRQ（Interrupt ReQuest）
- SVC（Supervisor SWI）
- 中止（Abort）
- 未定义（Undefined）

Cortex-A8处理器工作模式

Cortex-A8是基于ARMv7构架的处理器，共有8种工作模式：

处理器模式	模式标识符	备注
用户模式(User)	usr	正常程序执行模式
系统模式(System)	sys	使用和用户模式相同的寄存器组，用于运行特权级操作系统任务
管理模式(Supervisor)	svc	系统复位或软件中断时进入该模式，是供操作系统使用的一种保护模式
外部中断模式(IRQ)	irq	低优先级中断发生时进入该模式，常用于普通的外部中断处理
快速中断模式(FIQ)	fiq	高优先级中断发生时进入该模式，用于高速数据传输和通道处理
数据访问中止模式(Abort)	abt	当存取异常时进入该模式，用于虚拟存储和存储保护
未定义指令中止模式(Undefined)	und	当执行未定义指令时进入该模式，用于支持硬件协处理器的软件仿真
安全监控模式(Monitor)	mon	可在安全模式和非安全模式下转换

用户模式

非用户模式，或特权模式

异常模式

2.6 ARM寄存器组成

- Cortex-A8处理器共有40个32位寄存器，包括33个通用寄存器和7个状态寄存器。其中状态寄存器包括1个CPSR（Current Program Status Register，当前程序状态寄存器）和6个SPSR（Saved Program Status Register，备份程序状态寄存器）。
- 这些寄存器不能同时访问，在不同的处理器工作模式下只能够访问一组相应的寄存器组。

ARM状态下的通用寄存器组

不分组的通用寄存器

系统和用户模式	快速中断模式	管理模式	数据访问中止模式	外部中断模式	未定义指令中止模式	安全监控模式
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)

分组的通用寄存器

ARM状态下的状态寄存器组

= 特权模式

在特权模式下，特定模式下的寄存器阵列才是有效的。

1. 通用寄存器组

R0～R7是不分组的通用寄存器；

R8～R15是分组的通用寄存器；

在ARM状态下，任何时刻，16个数据寄存器R0～R15和1～2个状态寄存器是可访问的。在特权模式下，特定模式下的寄存器阵列才是有效的。

未分组的通用寄存器R0～R7用于保存数据和地址。在处理器的所有工作模式下，它们中的每一个都指向一个物理寄存器，且没有被系统用于特殊用途。在处理器工作模式切换时，由于使用的是相同的物理存储器，可能会破坏寄存器中的数据。

分组的通用寄存器R8～R15则具有不同的处理器工作模式决定访问的物理寄存器不同的特点。每个物理寄存器名字的形式为 Rx_<mode>，<mode>是模式标识符，每个模式标识符指示当前所处的工作模式

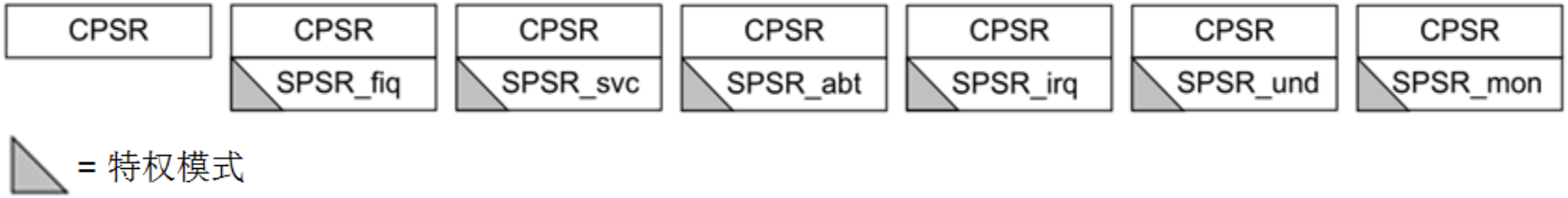
- R8～R12寄存器分别对应两个不同的物理寄存器，分别是快速中断模式下的相应存储器和非快速中断模式下的相应存储器。
- R13、R14寄存器分别对应七个不同的物理存储器，除了用户和系统模式共用一个物理寄存器外，其它六个分别是fiq、svc、abt、irq、und和mon模式下的不同物理寄存器。R13常作堆栈指针(SP: Stack Pointer)；R14子程序链接寄存器(LR: Link Register)，该寄存器由ARM编译器自动使用。在执行BL和BLX指令时，R14保存返回地址。。

- 程序计数器R15（PC），用于记录程序当前的运行地址。ARM处理器每执行一条指令，都会把PC增加4字节(Thumb模式为两个字节)。此外，相应的分支指令(如BL等)也会改变PC的值。在ARM状态下，PC字对齐；在Thumb和ThumbEE状态下，PC半字对齐。
- FIQ模式下有7个分组寄存器映射到R8～R14，即R8_fiq～R14_fiq,所以很多快速中断处理不需要保存任何寄存器。

2. 状态寄存器

ARM处理器有两类程序状态寄存器：1个当前程序状态寄存器CPSR和6个备份程序状态寄存器SPSR。它们的主要功能是：

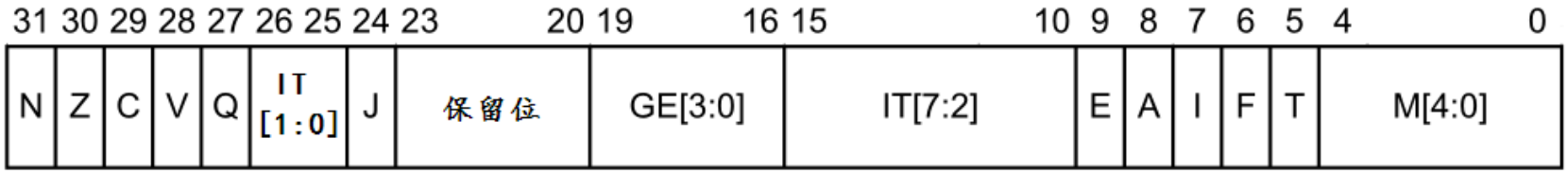
- **保存最近执行的算术或逻辑运算的信息；**
- **控制中断的允许或禁止；**
- **设置处理器工作模式。**



每一种处理器模式下使用专用的备份程序状态寄存器。当特定的中断或异常发生时，处理器切换到对应的工作模式下，该模式下的备份程序状态寄存器保存当前程序状态寄存器的内容。当异常处理程序返回时，再将其内容从备份程序状态寄存器回复到当前程序状态寄存器。

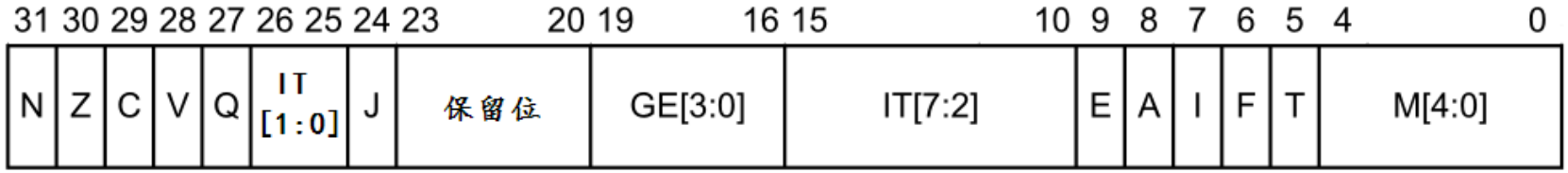
程序状态寄存器的格式如图2-9所示，32位寄存器会被分成四个域：

- 标志位域f (flag field)， PSR[31:24]；
- 状态域s (status field)， PSR[23:16]；
- 扩展域x (extend field)，PSR[15:8]；
- 控制域c (control field)，PSR[7:0] 。



（1）条件标志位：N、Z、C和V统称为条件标志位，这些标志位会根据程序中的算术和逻辑指令的执行结果修改。处理器则通过测试这些标志位来确定一条指令是否执行。

- N（Negative）：N=1表示运算的结果为负数，N=0表示结果为正数或零。
- Z（Zero）：Z=1表示运算的结果为零，Z=0表示运算的结果不为零。
- C （Carry）：在加法指令中，当结果产生了进位，C=1，其它情况C=0；在减法指令中，当运算发生了借位，C=1，其它情况C=0；在移位运算指令中，C被设置成被移位寄存器最后移出去的位。
- V（oVerflow）：对于加/减法运算指令，当操作数和运算结果为二进制补码表示的带符号数时，V=1表示符号位溢出。



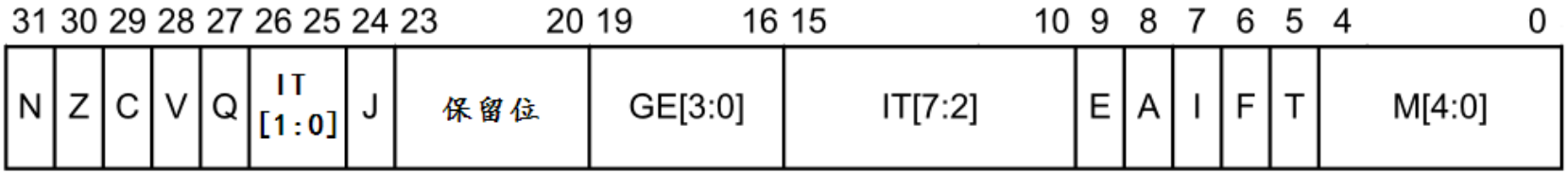
（2）Q标志位：在带有DSP指令扩展的ARMv5及以上版本中，Q标志位用于指示增强的DSP指令是否发生了溢出。Q标志位具有粘性，当因某条指令将其设置为1时，它将一直保持为1直到通过MSR指令写CPSR寄存器明确地将该位清0，不能根据Q标志位的状态来有条件地执行某条指令。

（3）IT块：IT块用于对Thumb指令集中if-then-else这一类语句块的控制。如果有IT块，则IT[7：5]为当前IT块的基本条件码。在没有IT块处于活动状态时，该3位为000。IT[4：0]表示条件执行指令的数量，不论指令的条件是基本条件码或是基本条件的逆条件码。在没有IT块处于活动状态时，该5位为00000。当处理器执行IT指令时，通过指令的条件和指令中Then Else（T和E）参数来设置这些位。

（4）J标志位：用于表示处理器是否处于ThumbEE状态。T=1时，

- J=0，表示处理器处于Thumb状态。
- J=1，表示处理器处于ThumbEE状态。

注意：T=0时，不能够设置J=1；当T=0时，J=0。不能通过MSR指令来改变CPSR的J标志位。



（5）GE[3：0]位：该位用于表示在SIMD指令集中的大于、等于标志。在任何模式下可读可写。

（6）E标志位：该标志位控制存取操作的字节顺序。0表示小端操作，1表示大端操作。ARM和Thumb指令集都提供指令用于设置和清除E标志位。当使用CFGEND0信号复位时，E标志位将被初始化。

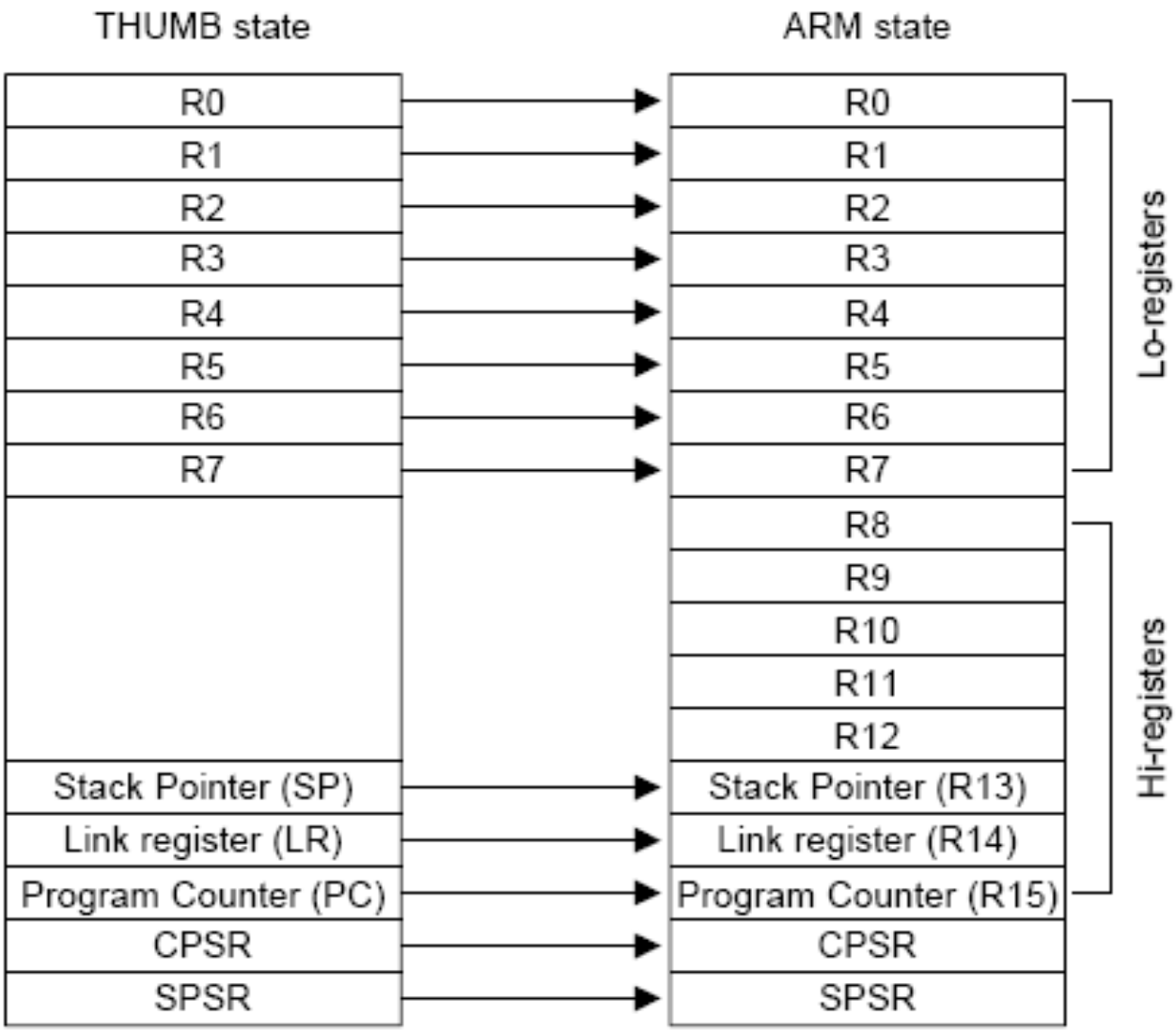
（7）A标志位：表示异步异常禁止。该位自动置为1，用于禁止不精确的数据中止。

（8）控制位：程序状态寄存器的低8位是控制位。当异常发生时，这些位的值将发生改变。在特权模式下，可通过软件编程来修改这些标志位的值。

- 中断屏蔽位：I=1，IRQ中断被屏蔽；F=1，FIQ中断被屏蔽。
- 状态控制位：T=0，处理器处于ARM状态；T=1，处理器处于Thumb状态。
- 模式控制位：M[4：0]为模式控制位，决定处理器的工作模式，如表2-3所示。

M[4：0]	0b10000	0b10001	0b10010	0b10011	0b10111	0b11011	0b11111	0b10110
工作模式	User	FIQ	IRQ	Supervisor	Abort	Undefined	System	Secure Monitor

ARM State 与Thumb State寄存器关系



2.7 ARM的异常中断

在ARM体系结构中，异常中断用来处理软件中断、未定义指令陷阱（它不是真正的“意外”事件）及系统复位功能和外部事件.

这些“不正常”事件都被划归“异常”，因为在处理器的控制机制中，它们都使用同样的流程进行异常处理。

ARM的异常中断

- ARM的异常中断响应过程
- 从异常中断处理程序中返回
- 异常中断向量表
- 异常中断的优先级

ARM的异常中断响应过程

ARM处理器对异常中断的响应过程如下：

- 将CPSR的内容保存到将要执行的异常中断对应的SPSR中
- 设置当前状态寄存器CPSR中的相应位
- 将引起异常指令的下一条指令的地址保存到新的异常工作模式的R14
- 给程序计数器（PC）强制赋值

ARM的异常中断响应过程

- 每个异常模式对应有两个寄存器R13_<mode>、R14_<mode>分别保存相应模式下的堆栈指针、返回地址；堆栈指针可用来定义一个存储区域保存其它用户寄存器，这样异常处理程序就可以使用这些寄存器。
- FIQ模式还有额外的专用寄存器R8_fiq～R12_fiq，使用这些寄存器可以加快快速中断的处理速度。

从异常中断处理程序中返回

从异常中断处理程序中返回时，需要执行以下四个基本操作：

- 所有修改过的用户寄存器必须从处理程序的保护堆栈中恢复（即出栈）
- 将SPSR_mode寄存器内容复制到CPSR中，使得CPSR从相应的SPSR中恢复，即恢复被中断的程序工作状态；
- 根据异常类型将PC变回到用户指令流中相应指令处
- 最后清除CPSR中的中断禁止标志位I/F。

异常中断的优先级

当几个异常中断同时发生时，在ARM中通过给各异常中断赋予一定的优先级来实现处理次序

- 复位（最高优先级）；
- 数据异常中止；
- FIQ；
- IRQ；
- 预取指异常中止；
- SWI、未定义指令（包括缺协处理器）

异常中断向量表

- 中断向量表中指定了各异常中断与其处理程序的对应关系
- 每个异常中断对应的中断向量表的4个字节的空间中存放一个跳转指令或者一个向PC寄存器中赋值的数据访问指令
- 存储器的前8个字中除了地址0x00000014之外，全部被用作异常矢量地址

异常：

1. 由内部或外部源产生以引起处理器处理的一个事件。
2. 异常出现时，异常模式可如下表示：

```
R14_<exception_mode>=return link
SPSR_<exception_mode>=CPSR
CPSR[4:0]=exception mode number
CPSR[5]=0                                   /*在ARM状态运行
*/
If<exception_mode>==Reset or FIQ then
   CPSR[6]=1                                /*禁止快速中断*/
CPSR[7]=1                                   /*禁止正常中断*/
PC=exception vector address
```

复位：

处理器上一旦有复位输入，ARM处理器立即停止执行当前指令。复位完成下列操作：

```
R14_svc=UNPREDICTALE value
SPSR_svc=UNPREDICTALE value
CPSR[4:0]=0b10011          /*进入管理模式*/
CPSR[5]=0                  /*在ARM状态运行*/
CPSR[6]=1                  /*禁止快速中断*/
CPSR[7]=1                  /*禁止正常中断*/
PC=0x00000000
```

中断请求异常（IRQ）

通过处理器上的IRQ输入引脚，由外部产生IRQ异常，IRQ异常的优先级比FRQ异常的低。当进入FRQ处理时会屏蔽掉IRQ异常。

```
R14_irq=address of next instruction to be executed+4
SPSR_irq=CPSR
CPSR[4:0]=0b10010          /*进入IRQ模式*/
CPSR[5]=0                  /*在ARM状态运行*/
/* CPSR[6] 不变 */          /*禁止快速中断*/
CPSR[7]=1                  /*禁止正常中断*/
PC=0x00000018
```

2.8 ARM典型流水线技术简介

2.8.1三级流水线ARM的组织

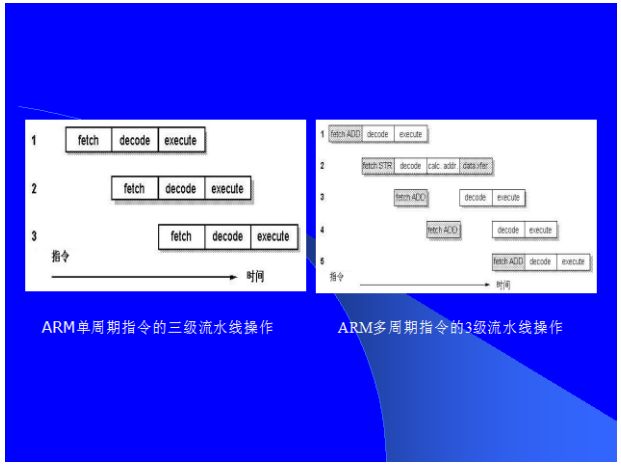
2.8.2五级流水线ARM的组织

2.8.1三级流水线ARM的组织

ARM的3级流水线介绍

到ARM7为止的ARM处理器使用的简单3级流水线分别为

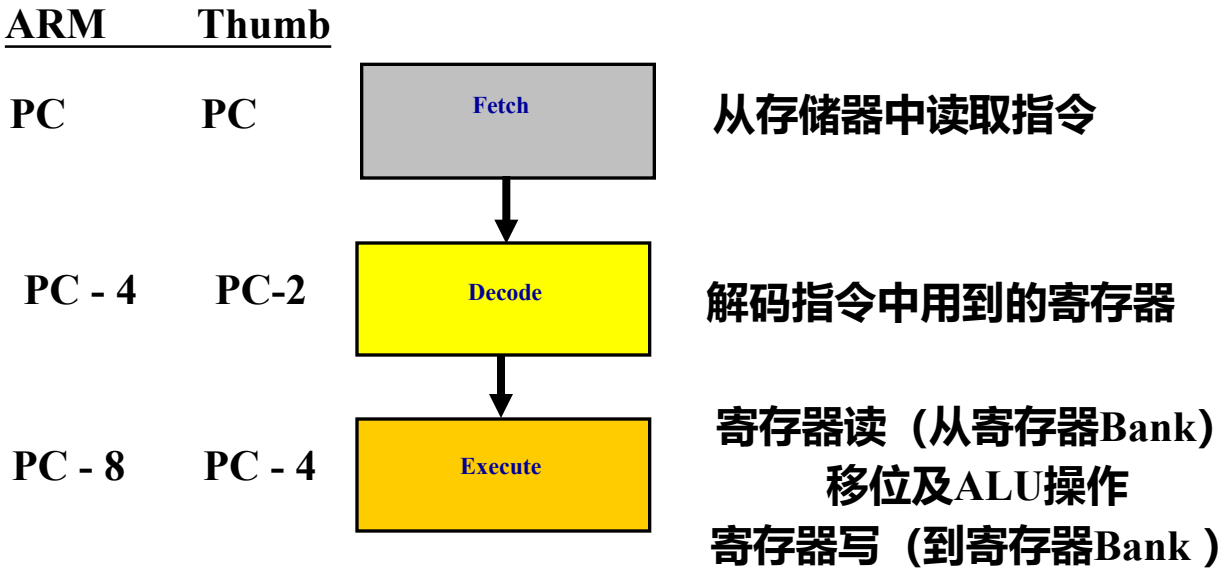
- 取指级
- 译码级
- 执行级



ARM单周期指令的三级流水线操作　　ARM多周期指令的3级流水线操作

ARM7TDMI指令流水线

- 为增加处理器指令流的速度，ARM7 系列使用3级流水线。
  - 允许多个操作同时处理，而非顺序执行。
  - PC指向正被取指的指令，而非正在执行的指令。



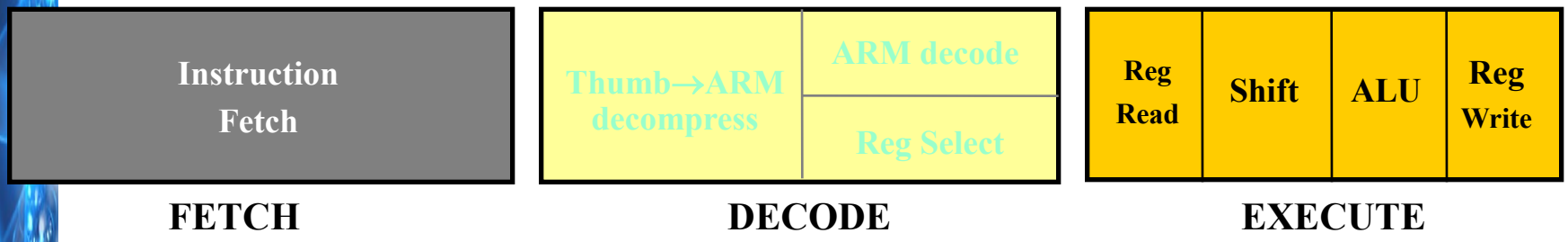
2.8.2五级流水线ARM的组织

使用5级流水线的ARM处理器包含下面5个流水线级

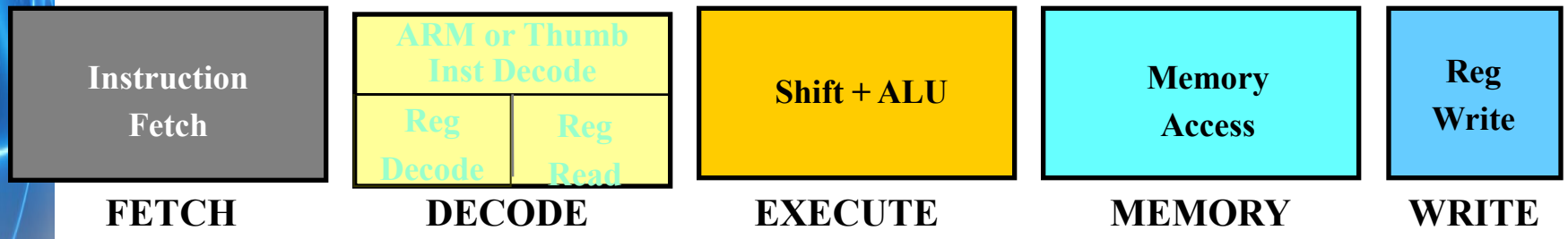
- 取指
- 译码
- 执行
- 缓冲\数据
- 回写

ARM9TDMI指令流水线

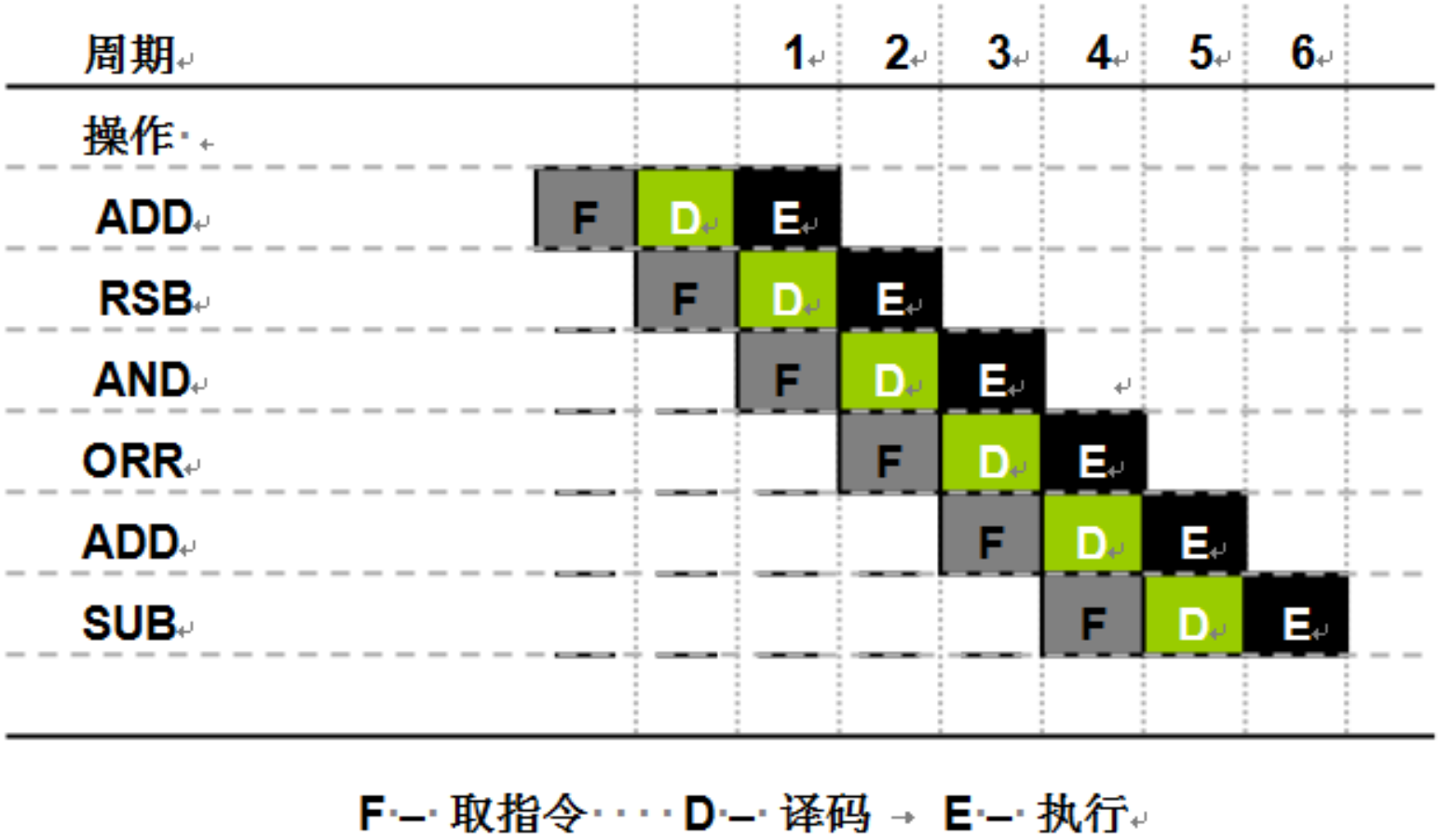
ARM7TDMI



ARM9TDMI



理想的3级流水线（ARM7TDMI:无访存操作）



单选题 10分

设置

设指令由取指、分析、执行3个子部件完成，每个子部件的工作周期均为Dt，采用常规标量单流水线处理机。若连续执行10条指令，则共需时间_？_Dt。

A 8

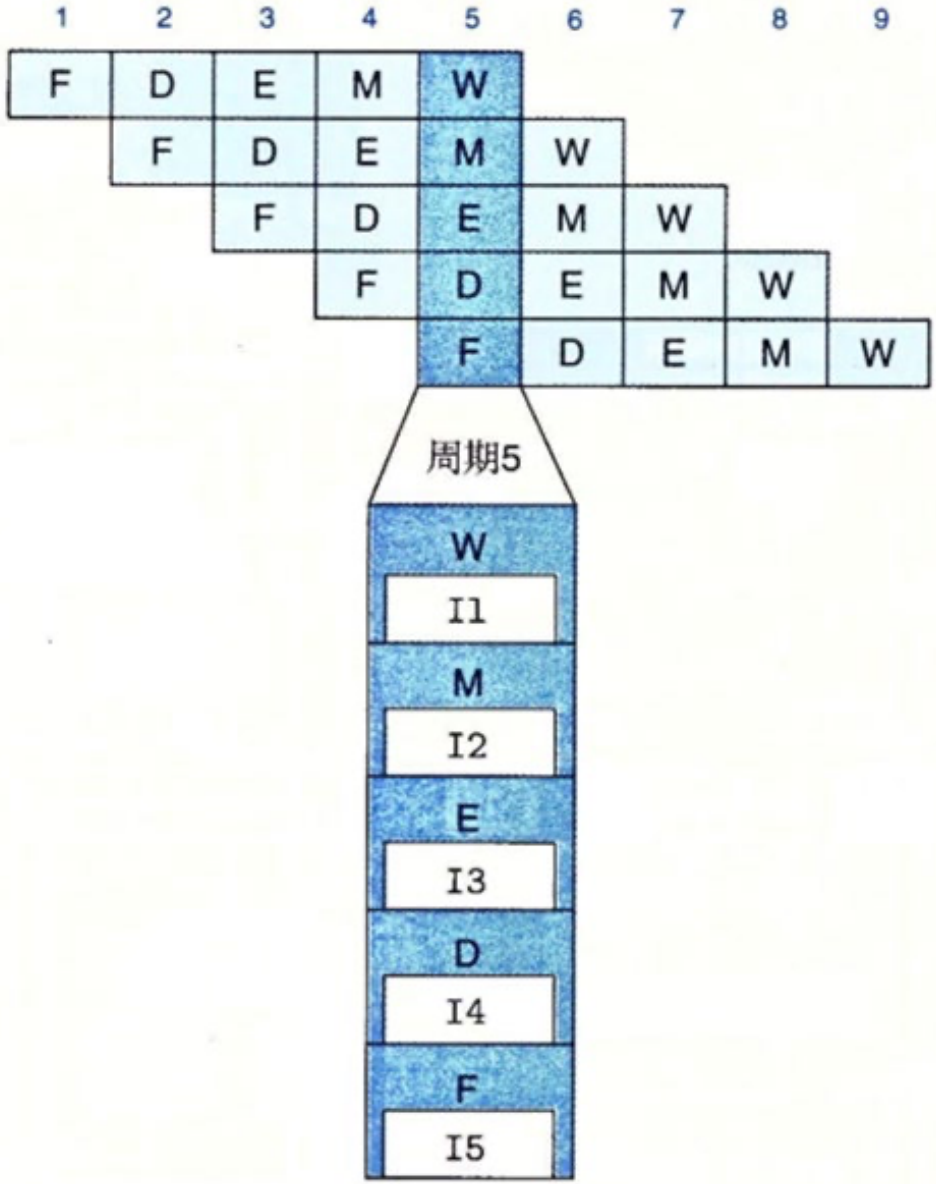
B 10

C 12

D 14

提交

流水线冒险✦



然而，流水线设计也引 计有两种主要类型的流水线冒险。当条件分 可能获取和解码了错误的指令。这可以通过把 实际上表现为无操作。当计算或加载的结果在 会发生数据冒险。对于ALU操作，通过ALU输 载，它不能立即使用，需要一个流水线气泡 可以从两个周期前转发。转发多路复用器使 图顶部所示。

流水线冒险分为三类：

- **数据冒险**：当指令依 会产生数据冒险。数据冒险可以通过数据转 线重新调度（re-scheduling）来解决
- **控制冒险**：当条件分 和译码。控制冒险可以通过分支预测和流水
- **结构冒险**：当多个指 结构冒险可以通过资源分配和流水线暂停来解决。

2.9 ARM存储器接口及存储器层次

- 多级存储器使它包括一个容量小但速度快的从存储器和一个容量大但速度慢的主存储器
- 根据典型程序的实验统计，这个存储器系统的外部行为在绝大部分时间象一个即大又快的存储器。
- 这个容量小但速度快的元件是Cache，它自动地保存处理器经常用到的指令和数据的拷贝。

2.9.1 ARM存储数据类型和存储格式

Cortex-A8(为例)是32位处理器，支持多种数据类型：

- 字节（Byte）：8位；
- 半字（Half word）：16位；
- 字（Word）： 32位；
- 双字（Double word）：64位。

当数据是无符号数时，二进制格式存储，数据范围为：
$0 \sim 2^N-1$，其中N为数据类型长度；
当数据是有符号数时，二进制补码格式存储，数据范围为：
$-2^{N-1} \sim 2^{N-1}-1$，其中N为数据类型长度。

- ARM的体系结构将存储器看成是从0x00000000地址开始的按字节编码的线性存储结构，每个字节都有对应的地址编码。由于数据有不同的字节大小（1字节、2字节、4字节等），导致数据在存储器中存放不是连续的，这样降低了存储系统的效率，甚至引起数据读写错误。因此数据必须按照以下方式对齐：

  - 以字为单位，按4字节对齐，地址最末两位为00。
  - 以半字为单位，按2字节对齐，地址最末一位为0；
  - 以字节为单位，按1字节对齐；

题目：若内存按字节编址，用存储容量为8K*8比特的存储器芯片构成地址编号A0000H～DFFFFH的内存空间，则至少需要多少片？

- 本题考查内存容量的计算。
- 给定起、止地址码的内存容量 = 终止地址 – 起始地址 + 1。
- 将终止地址加1等于E0000H，再减去起始地址，即E0000H – A0000H = 40000H。十六进制的 $(40000)_{16} = 2^{18}$。
- 组成内存储器的芯片数量 = 内存储器的容量/单个芯片的容量。
- $2^{18}/(8*2^{10}) = 2^{18}/2^{13} = 2^5$

大/小端存储模式

Cortex-A8处理器支持大端（Big-endian）和小端（Little-endian）两种存储模式，同时还支持混合大小端模式（既有大端模式也有小端模式）和非对齐数据访问。可以通过硬件的方式设置（没有提供软件的方式）端模式。

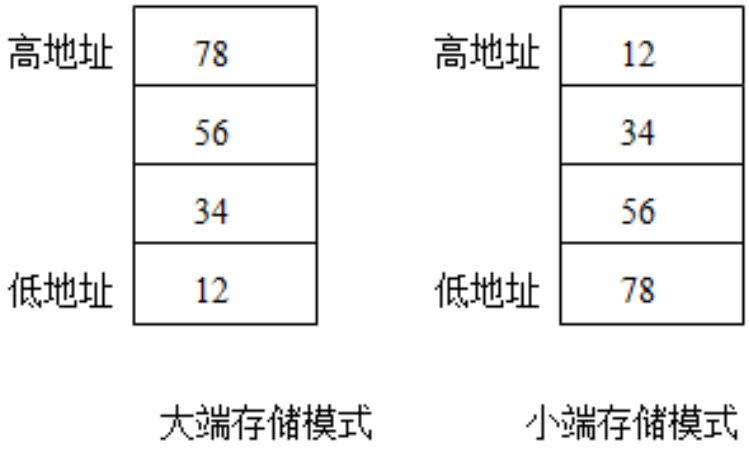
大端模式是被存放字数据的高字节存储在存储系统的低地址中，而被存放的字数据的低字节则存放在存储系统的高地址中。

高对低
低对高

小端模式中，存储系统的低地址中存放的是被放字数据中的低字节内容，存储系统的高地址存放的是被存字数据中的高字节内容。

低对低
高对高

例如，一个32位的字数据
0x12345678



2.9.2 ARM的存储器层次简介

- 寄存器组
- 片上RAM
- 片上Cache
- 主存储器
- 硬盘

2.10 ARM协处理器***

- ARM通过增加硬件协处理器来支持对其指令集的通用扩展，通过未定义指令陷阱支持这些协处理器的软件仿真。简单的ARM核提供板级协处理器接口，因此协处理器可以作为一个独立的元件接入。

- 最常使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ARM720上的高速缓存Cache和存储器管理单元MMU等。ARM也开发了浮点协处理器，也可以支持其它的片上协处理器。ARM体系结构支持通过增加协处理器来扩展指令集的机制。

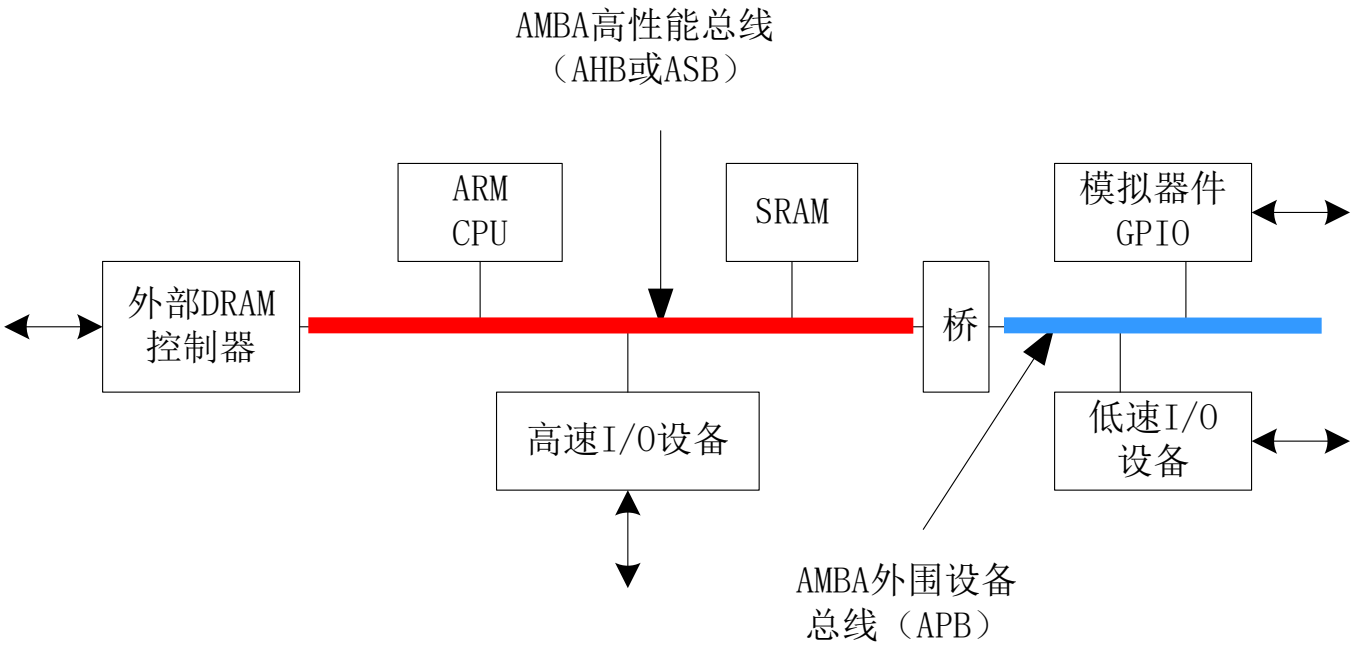
2.11 ARM片上总线AMBA

先进的微控制器总线体系结构

AMBA是ARM公司公布的总线标准

- AHB（Advanced High-performance Bus）：用于连接高性能模块。它支持突发数据传输方式及单个数据传输方式，所有时序参考同一个时钟沿。
- ASB（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（Advance Peripheral Bus）：是一个简单接口支持低性能的外围接口。

AMBA接口


AMBA高性能总线
（AHB或ASB）
ARM
CPU
SRAM
模拟器件
GPIO
外部DRAM
控制器
桥
高速I/O设备
低速I/O
设备
AMBA外围设备
总线（APB）

2.12 ARM核综述

在高性能的32位嵌入式SoC设计中，几乎都是以ARM作为处理器核。ARM核已是现在嵌入式SoC系统芯片的核心，也是现代嵌入式系统发展的方向。

ARM处理器核作为基本处理单元，根据发展需求还集成了与处理器核密切相关的功能模块，如Cache存储器和存储器管理MMU硬件。

ARM核综述

ARM处理器核当前有6 个系列产品：

ARM7　　ARM9

ARM9E　　ARM10E,

SecurCore　　ARM11

Intel公司推出的：

StrongARM

Xscale

高性能的ARM处理器

2.12.1 ARM7系列核介绍

ARM7TDMI是ARM公司最早为业界普遍认可且得到最为广泛应用的处理器核，特别是在手机和了PDA中，随着ARM技术的发展，它已是目前最低端的ARM核。

. 嵌入式ICE-RT逻辑（允许代码的任何部分设置断点，进行调试）

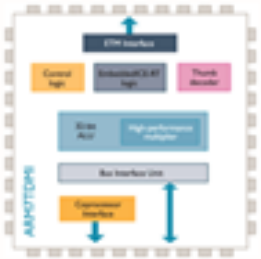
. 非常低的功耗

. 提供0.9MIPS的三级流水线

. 冯.诺依曼结构

ARM7

- AR
- T：
- D：
  止以
- M：
  能且
- I：‘
  点

	 ARM7TDMI
Multicore	No
Architecture	ARMv4T
ARM	√
Thumb	√
Bus Interface	AHB
Process Geometry	TSMC 90G
Process Libraries	ARM SC10T
Performance (Total DMIPS)	189
Performance (DMIPS/MHz)	0.93
Max Frequency	204 MHz
Area (mm^2)	0.34
Power (mW/MHz)	0.07

理器能够停

具有较高的性

断点和观察

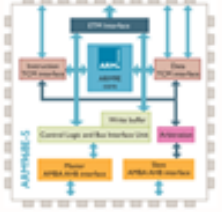
ARM7系列核介绍

1）ARM7TDMI重要的特性

- 实现ARM体系结构版本4T，支持64位结果的乘法，半字、有符号字节存取；
- 支持Thumb指令集，可降低系统开销；
- 32×8 DSP 乘法器；
- 32位寻址空间- 4GB 线性地址空间；
- 它包含了EmbeddedICE模块以支持嵌入式系统调试；
- 调试硬件由JTAG访问端口；
- 与 ARM9 Thumb 系列 ARM10 Thumb 系列和StrongARM处理器相兼容。

2.12.2ARM9系列核介绍

- ARM9TDMI将流水线的级数从ARM7TDMI的3级增加到5级，并使用分开的指令与数据存储器的Harvard体系结构。ARM9TDMI的性能在相同工艺条件下近似达到ARM7TDMI两倍



	ARM946E-S	ARM968E-S
Multicore	No	No
Architecture	ARMv5TE	ARMv5TE
ARM	√	√
DSP	√	√
Floating Point	X	√
Jazelle	√	X
Thumb	√	√
L1 Cache (Max)	1MB	X
TCM (Max)	4KB	4MB
Memory Controller	MPU	X
Bus Interface	AHB	AHB
Process Geometry	TSMC 65LP	TSMC 90G
Process Libraries	ARM SC10T	ARM SC12T
Performance (Total DMIPS)	409	636
Performance (DMIPS/MHz)	1.2	1.2
Max Frequency	341MHz	530MHz
Area With Cache (mm^2)	0.488	X
Area No Cache (mm^2)	0.26	0.42
Power With Cache (mW/MHz)	0.142	X

1）A

- 支持
- 含有 调试；
- 通过 钟速率；
- 分开

2.12.3 ARM10系列核

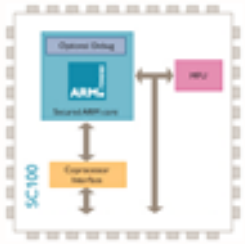
ARM10TDMI属于ARM处理器核中的高端处理器核，ARM10TDMI的性能在相同工艺条件下近似达到也以ARM9TDMI的两倍性能工作。ARM1020E/ARM10200是基于ARM10TDMI核设计的高性能CPU核。

- 增加最高时钟速率。

2.12.4 S

Secur 而设计，其独特的优势 支持。

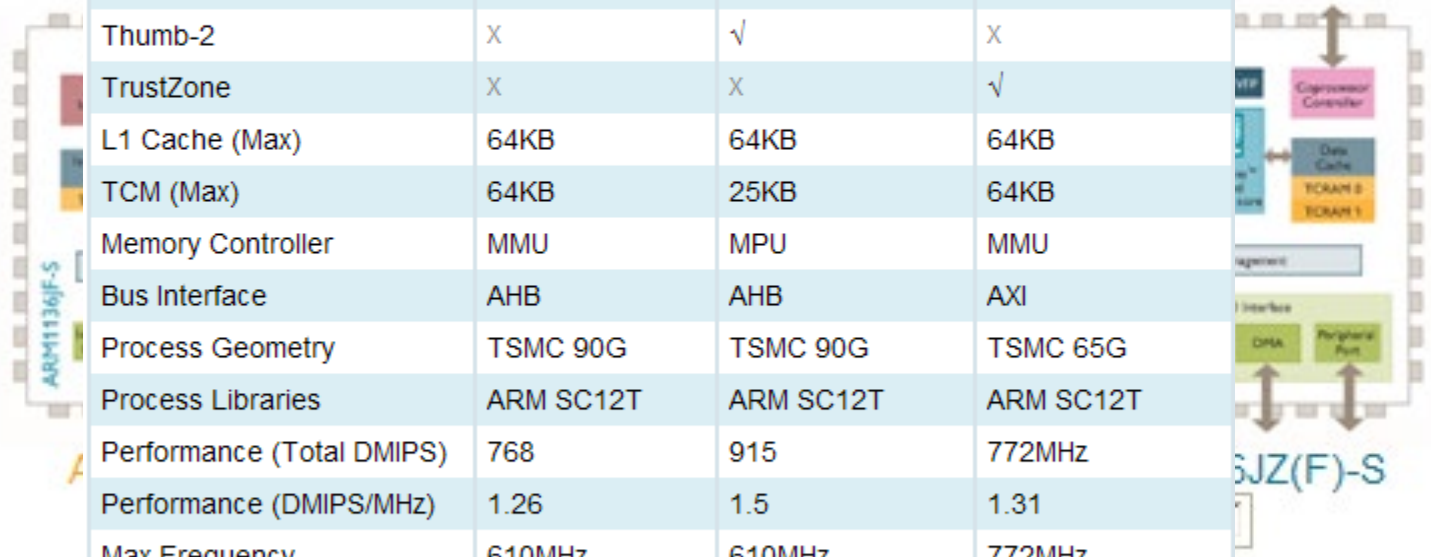
- 带有 统和应用数据
- 采用 扫描探测。
- 可集 处理器。

	 SecurCore SC100
Multicore	No
Architecture	ARMv4T
ARM	√
Thumb	√
Bus Interface	AHB
Process Geometry	TSMC 90G
Process Libraries	ARM SC10T
Performance (Total DMIPS)	186
Performance (DMIPS/MHz)	0.93
Max Frequency	200MHz
Area (mm^2)	0.34
Power (mW/MHz)	0.07

嵌入式安全！

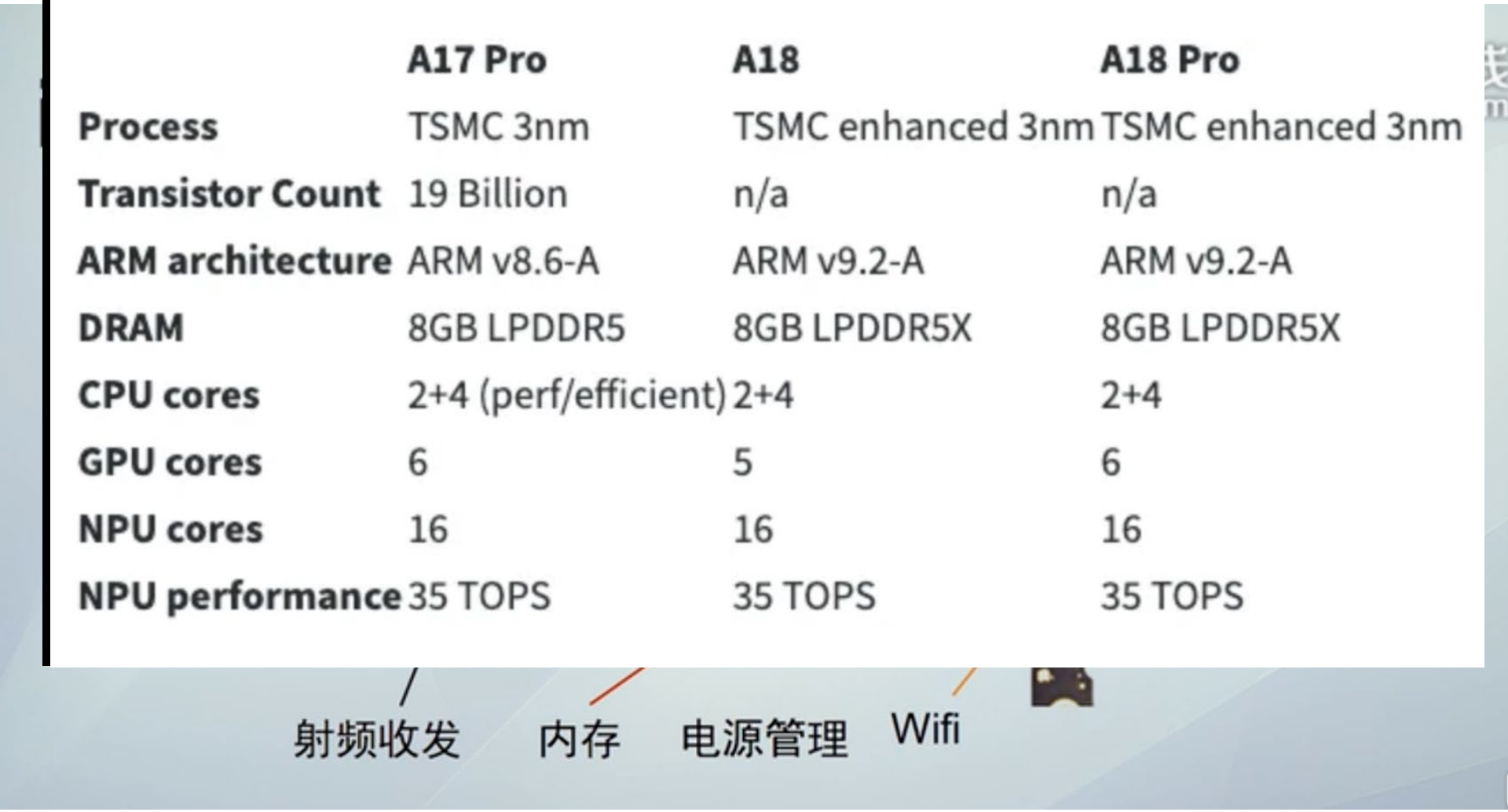
2.12

Multicore	No	No	No
Architecture	ARMv6	ARMv6	ARMv6
ARM	√	√	√
DSP	√	√	√
Floating Point	√	√	√
Jazelle	√	X	√
Thumb	√	√	√
Thumb-2	X	√	X
TrustZone	X	X	√
L1 Cache (Max)	64KB	64KB	64KB
TCM (Max)	64KB	25KB	64KB
Memory Controller	MMU	MPU	MMU
Bus Interface	AHB	AHB	AXI
Process Geometry	TSMC 90G	TSMC 90G	TSMC 65G
Process Libraries	ARM SC12T	ARM SC12T	ARM SC12T
Performance (Total DMIPS)	768	915	772MHz
Performance (DMIPS/MHz)	1.26	1.5	1.31
Max Frequency	610MHz	610MHz	772MHz
Area With Cache (mm^2)	2.5	2.267	1.937
Area No Cache (mm^2)	1.8	1.686	1.31
Power With Cache (mW/MHz)	0.36	0.415	0.208
Power No Cache (mW/MHz)	0.3	0.333	0.153



高性能的ARM嵌入式处理器

	A17 Pro	A18	A18 Pro
Process	TSMC 3nm	TSMC enhanced 3nm	TSMC enhanced 3nm
Transistor Count	19 Billion	n/a	n/a
ARM architecture	ARM v8.6-A	ARM v9.2-A	ARM v9.2-A
DRAM	8GB LPDDR5	8GB LPDDR5X	8GB LPDDR5X
CPU cores	2+4 (perf/efficient)	2+4	2+4
GPU cores	6	5	6
NPU cores	16	16	16
NPU performance	35 TOPS	35 TOPS	35 TOPS



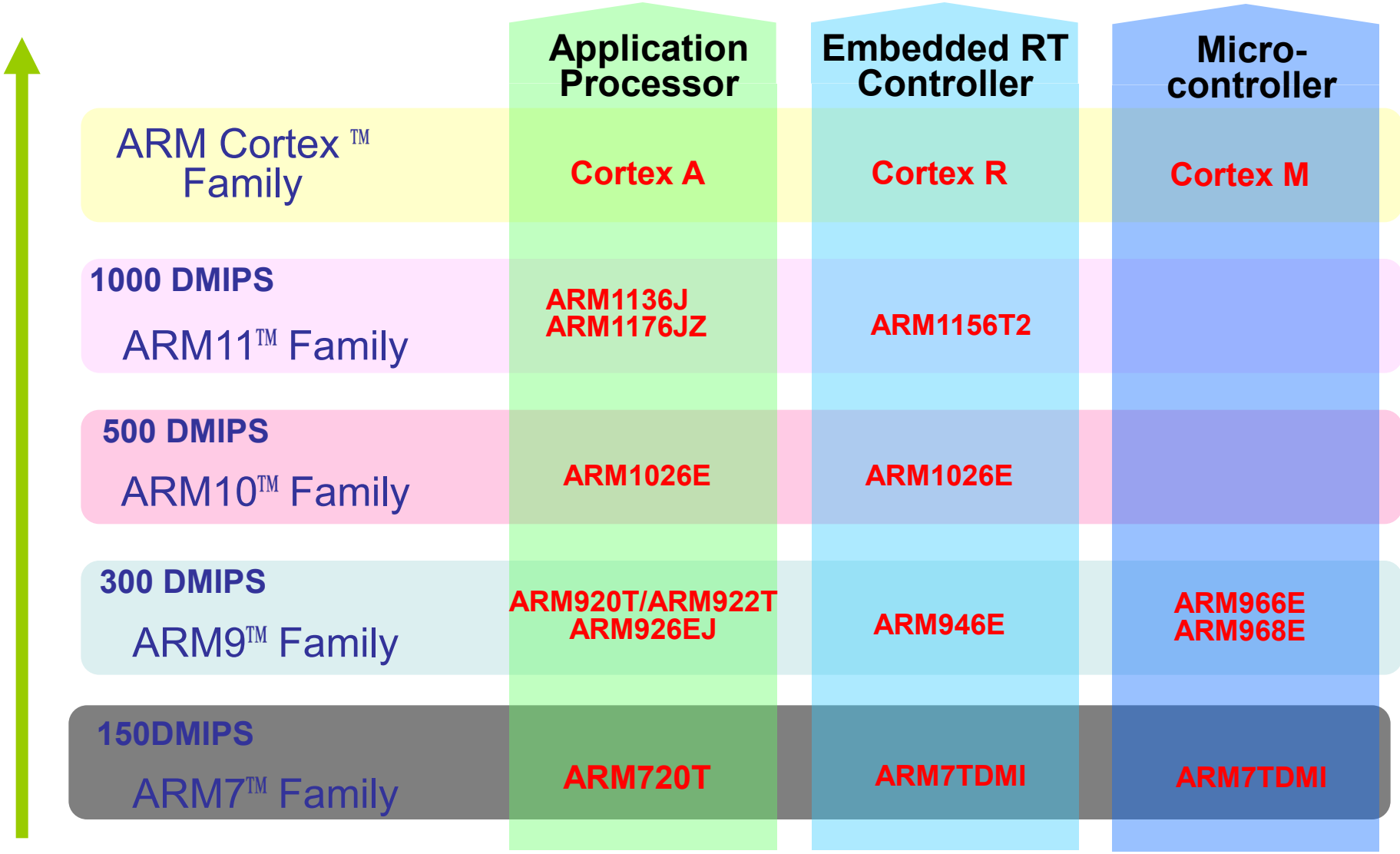
Cortex-M3 实现 $1 ARM芯片

ARM Cortex-M3 微控制器内核，专门针对MCU应用领域而设计，突出低成本、低功耗和高效率。

- ARM Cortex Architecture
- Thumb-2 ISA
- 3 Stage Pipeline
- 1.22 DMIPS/MHz – 30% over ARM7TDMI
- 33K gates – 30% smaller than ARM7TDMI

- **Luminary Micro的Stellaris系列MCU产品售价仅1美元**

ARM Family


Application Processor
Embedded RT Controller
Micro-controller
ARM Cortex ™ Family
Cortex A
Cortex R
Cortex M
1000 DMIPS
ARM11™ Family
ARM1136J
ARM1176JZ
ARM1156T2
500 DMIPS
ARM10™ Family
ARM1026E
ARM1026E
300 DMIPS
ARM9™ Family
ARM920T/ARM922T
ARM926EJ
ARM946E
ARM966E
ARM968E
150DMIPS
ARM7™ Family
ARM720T
ARM7TDMI
ARM7TDMI

2.13 基于ARM核的芯片选择

从应用的角度上ARM芯片选择的一般原则

- MMU
- 处理器速度
- 内置存储器容量
- USB接口
- GPIO数量
- 中断控制器
- IIS（Integrate Interface of Sound）音频接口

处理器选择器

使用处理器选择器可快速选择和比较处理器。

只需从下拉菜单组合中选择所需的系列或功能，选中相应框，然后单击"选择"按钮即可。根据所选的选项，某些选项可能显示为灰色，在所选的情形下无法使用。如果未选择项目，则指南的下一阶段中将显示所有处理器。

列出处理器之后，请单击勾选框以标识要在最终阶段查看的处理器，然后单击"比较"。

可随时单击缩略图以直接转到该处理器的单独页面。

Select

Processor Family :: Select ::

Architecture :: Select ::

Multicore :: Select ::

Instruction Set ☐ ARM ☐ Jazelle ☐ Thumb ☐ Thumb-2

Extra Features ☐ DSP ☐ Floating Point ☐ NEON ☐ TrustZone

Memory System ☐ Cache ☐ TCM

Memory Control ☐ MPU (Memory Protection Unit)

☐ MMU (Memory Management Unit)

59% ↑ 0.01K/S ↓ 0.1K/S

ARM开发工具简介

ARM开发工具就是ARM公司为庞大的各领域工程师和开发人员装备的完整的开发工具链，帮助迅速搭建开发平台，降低开发的成本和难度，缩短开发周期，让工程师们充分针对ARM架构处理器进行开发。

根据开发目标平台的不同，ARM提供不同的工具解决方案。 最常见的是**MDK-ARM、RVDS，ARM DS5**。他们分别针对低端和高端ARM处理器应用。

1. MDK-ARM

RealView Microcontroller Development Kit(MDK) 支持基于包括ARM7，ARM9，Cortex-M3微控制处理器等在内的众多处理器，例如Atmel，Freescale，Luminary，NXP，OKI，Samsung，Sharp，ST，TI等厂家的产品。MDK提供工业标准的编译工具和强大的调试支持。

MDK是专为MCU的用户开发嵌入式软件而设计的一套开发工具。包括根据器件定制的调试仿真支持，丰富的项目模版，固件示例以及为内存优化的RTOS库。MDK上手容易，功能强大，适合微控制器应用程序开发。

MDK主要是为终端客户提供价格低廉，功能强大的开发工具。集成了RealView编译工具，Keil uVision开发环境，支持基于ARM7,ARM9,Cortex-M1,Cortex-M3，Cortex-R4等ARM产品的仿真，提供非常高效的RTOS Kernel，此外，提供的Real-Time库还有TCP/IP网络套件，Flash文件系统，USB器件接口，CAN总线接口等，方便终端用户进行应用开发。因此对于MDK用户来说，他们得到的就是可以对MCU进行仿真和调试，容易使用又没有冗余的功能，关键是价格实惠，而且用户可以先试用再购买。

2.RVDS

RVDS（RealView Development Suite）是ARM公司推出的专为SOC，FPGA 以及ASIC用户开发复杂嵌入式应用程序或者和操作系统平台组件接口而设计的开发工具，被业界称为最好的ARM开发工具。RVDS支持器件设计，支持多核调试，支持基于所有ARM 和Cortex系列CPU的程序开发。RVDS还可以和第三方软件进行很好的连接。

RVDS 是ARM公司继SDT 与ADS1.2之后主推的新一代开发工具，目前最高版本是4.1。RVDS对代码密度的提升、代码执行速度的提高，都可以由ARM开发工具自动实现，而不需要软件开发人员花费过多的时间手动优化高级语言代码。这是RVDS的优势所在。

RVDS包含有四个模块：

（1）IDE：RVDS中集成了Eclipse IDE，用于代码的编辑和管理。支持语句高亮和多颜色显示，以工程的方式管理代码，支持第三方Eclipse功能插件。

（2）RVCT：RVCT是业界最优秀的编译器，支持全系列的ARM和XSCALE架构，支持汇编语言、C语言和C++语言。RVDS的编译器根据最新的ARM架构进行特别的优化，针对每个ARM架构都提供最好的代码执行性能，最优的代码密度。可以根据需要选择调试信息级别，以及不同的代码优化方向和优化级别。

（3）RVD：是RVDS中的调试软件，功能强大，支持Flash烧写和多核调试，支持多种调试手段，快速错误定位。

（4）RVISS：是指令集仿真器，支持外部设备虚拟，可以使软件开发和硬件开发同步进行，同时可以分析代码性能，加快软件开发速度。

3. ARM DS5

ARM DS5，也叫ARM DS-5，是一款支持开发所有ARM内核芯片的集成开发环境，也是一套针对 ARM 支持的linux和android平台的全面的端到端软件开发工具套件。ARM DS5提供具有跟踪、系统范围性能分析器、实时系统模拟器和编译器的应用程序和内核空间调试器。这些功能包含在定制的、功能强大且用户友好的基于Eclipse的IDE中。借助于该工具套件，可以很轻松地为ARM支持的系统开发和优化基于Linux的系统，缩短开发和测试周期，并且可帮助工程师创建资源利用效率高的软件。

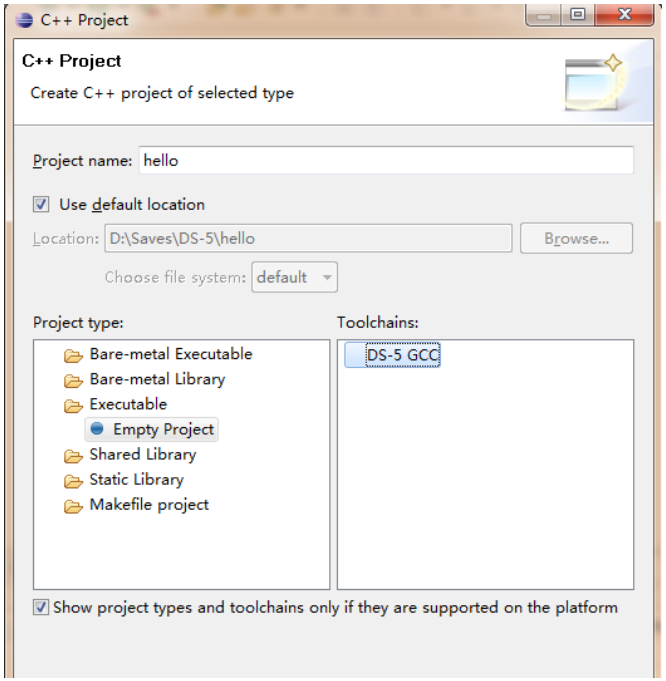


图2-1 ARM DS5的工程配置

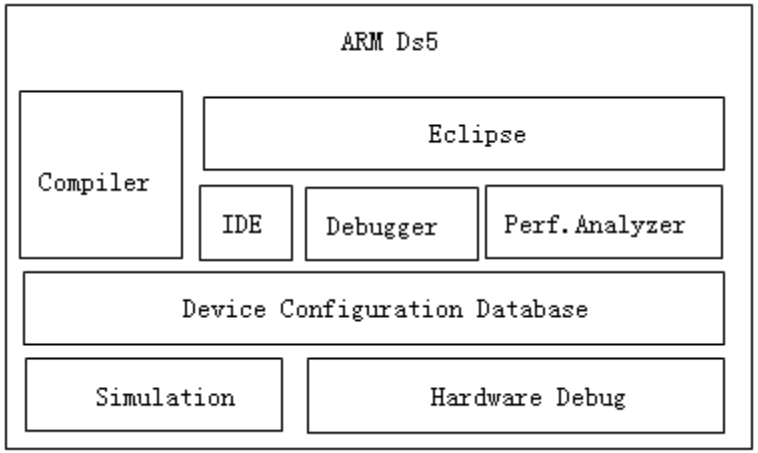


图2-2 ARM DS-5功能框架