ARM架构

专注于v7A和Cortex-A8

主要内容：

1. ARM处理器概述
2. ARM v7A架构/编程模型
3. Cortex-A8内存管理
4. Cortex-A8管道

# 1 ARM处理器概述

## 1.1 ARM

ARM公司始建于1990年11月，是从破产的英国Acorn计算机公司分拆出来的。Apple，Acorn和VLSI是其早期创始人和投资人。

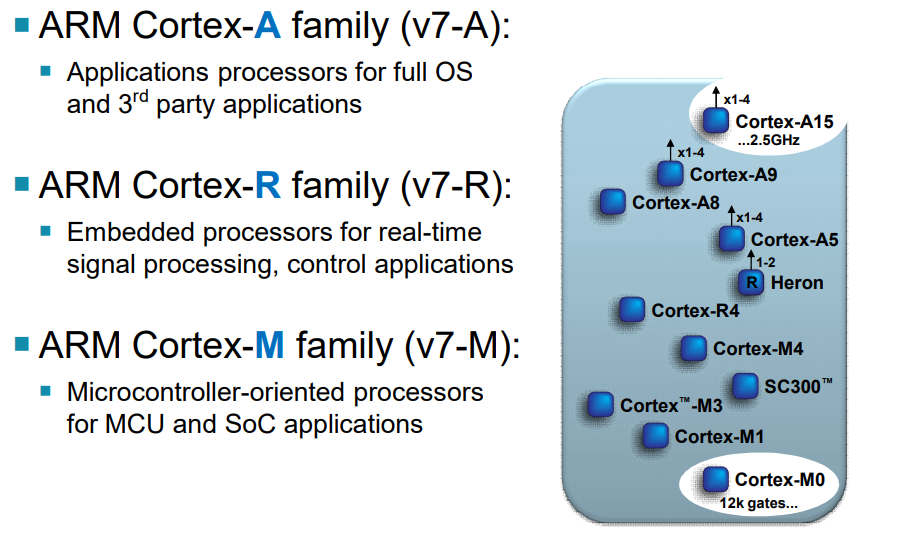
该公司主要设计基于RISC指令集的处理器内核的ARM系列。他们本身并不制造处理器，而仅仅是设计处理器的IP核，然后授权给半导体厂商，由他们进行自己的再设计之后，制造，销售给客户。

另外，还开发辅助基于ARM架构进行设计的技术，比如，

1. 软件工具，开发板，debug硬件
2. 应用软件
3. 总线架构
4. 外设等

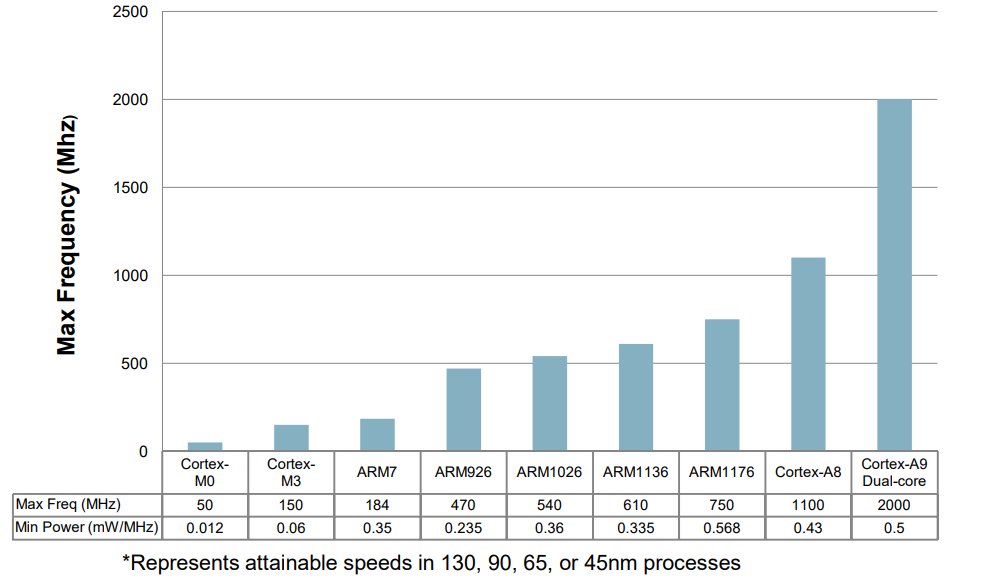
## 1.2 Cortex 系列处理器(v7)

参考下图：



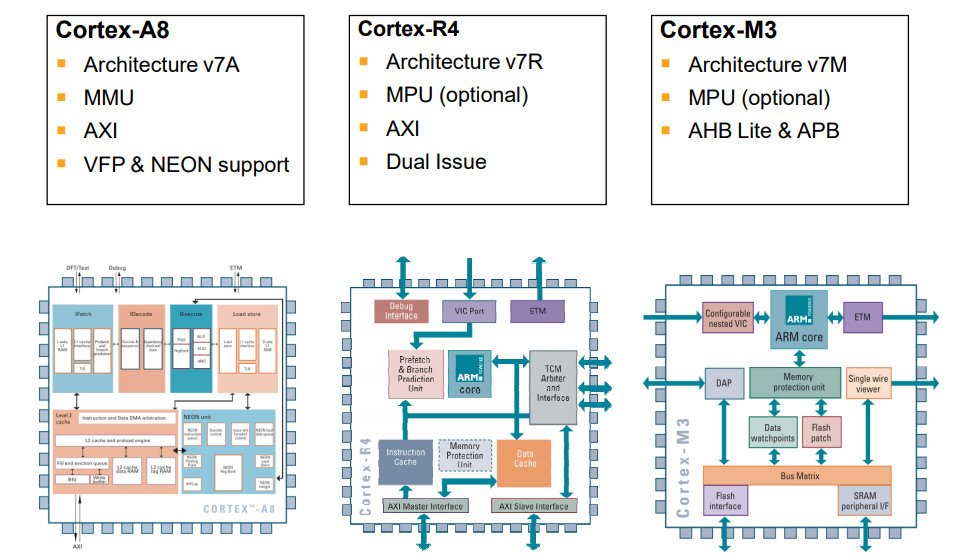
## 1.3 性能比较

相关的性能，可以参考下图：



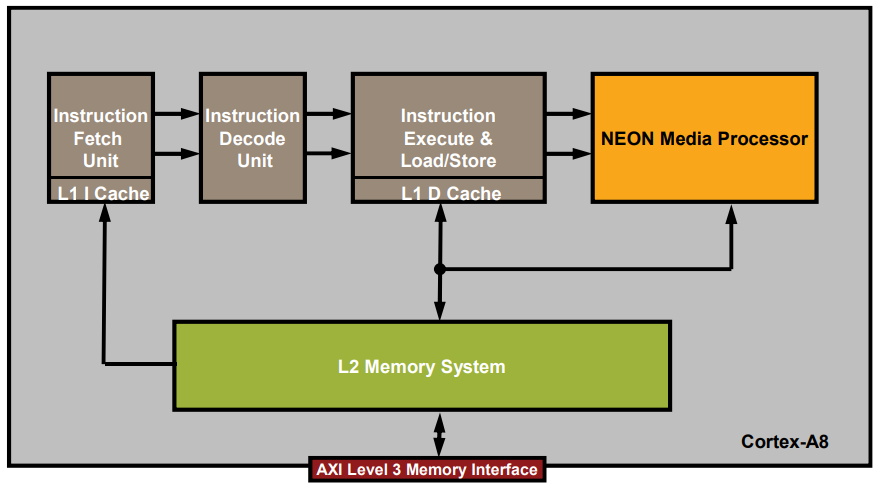
## 1.4 功能比较

主要的功能：

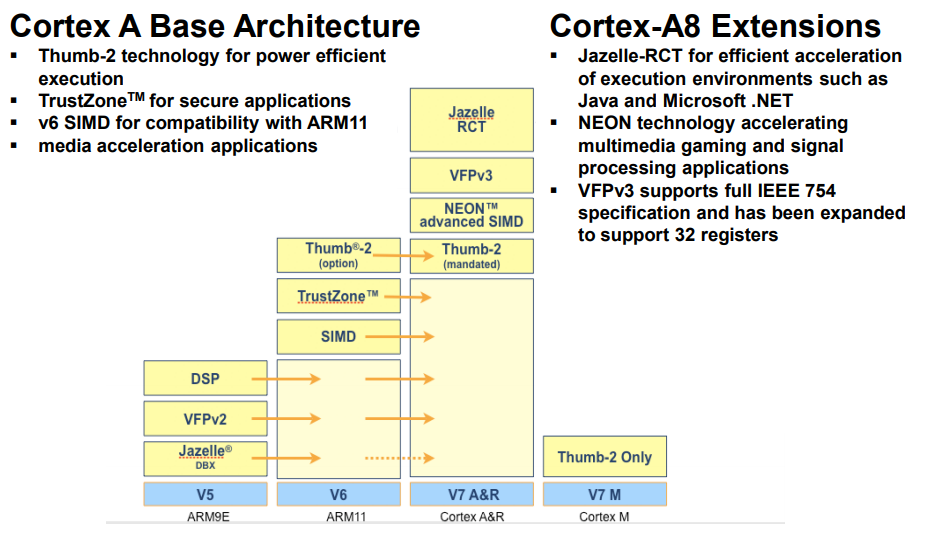


# 2 ARM v7A架构/编程模型

## 2.1 Cortex-A8 框图



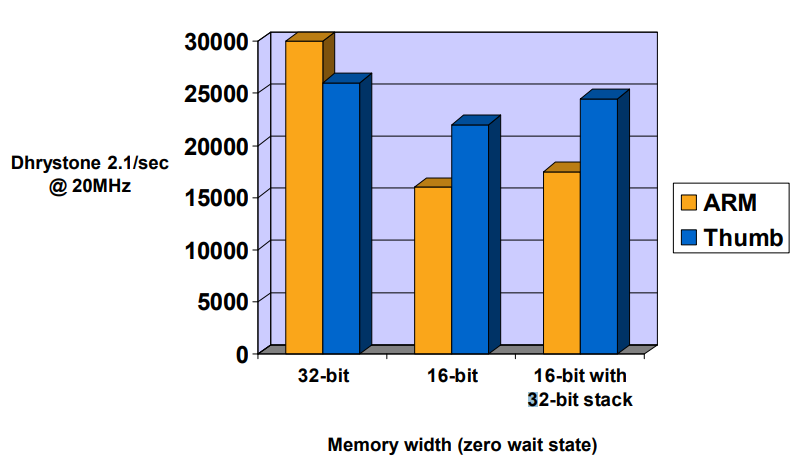
## 2.2 Cortex-A架构



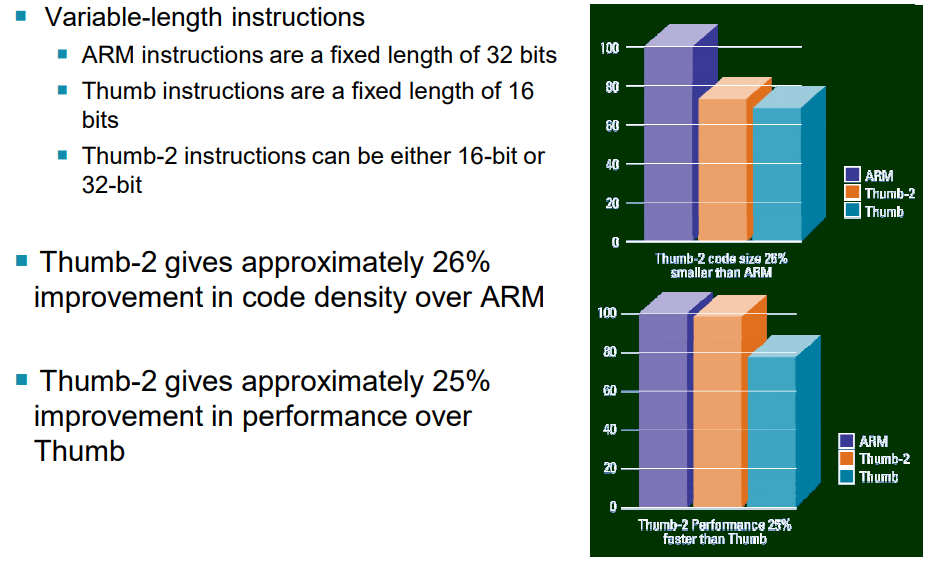
## 2.3 数据大小和指令集

1. ARM是32位架构
2. ARM使用的数据大小：
   1. Byte：8位
   2. HalfWord：16位（2个字节）
   3. Word：32位（4个字节）
3. 大部分大ARM可以执行2套指令集：32位ARM指令集和16位Thumb指令集
4. Jazelle核也可以执行Java字节码

## 2.4 ARM 和Thumb性能



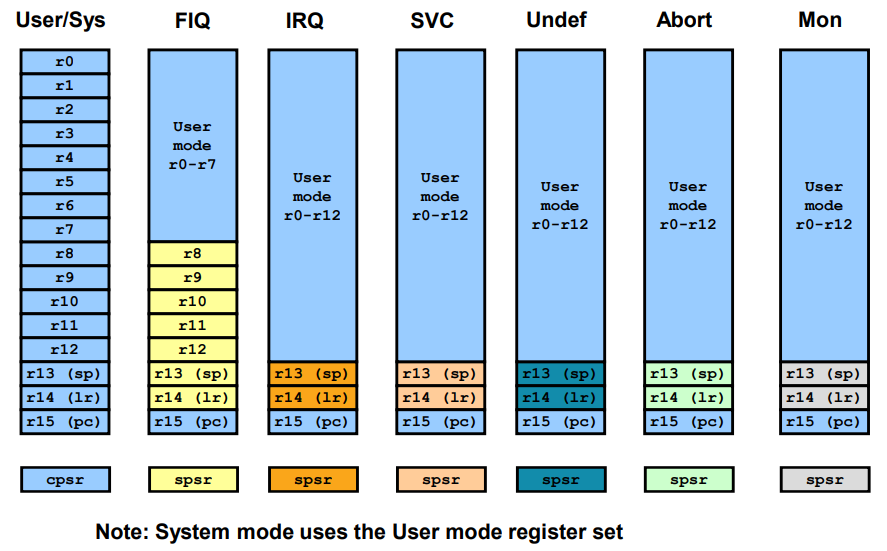
## 2.5 Thumb-2指令集



## Cortex-A8处理器工作模式

1. 用户：用来执行大部分的应用程序
2. FIQ：用来处理快速中断
3. IRQ：用于通用目的的中断处理
4. 超级权限：为操作系统而定的一种保护模式
5. 未定义：未定义的指令异常进入的状态
6. Aort：数据或者预取抛弃的时候进入该模式
7. System：操作系统的特权用户模式
8. Monitor：TrustZone的一种安全模式

## 2.7 Cortex-A8寄存器



1. r13（SP）：堆栈指针。每一种异常模式都有自己独立的r13寄存器，它通常指向异常模式所专用的堆栈，也就是说5种异常模式，以及正常模式（用户模式和系统模式），都有各自独立的堆栈，用不同的堆栈指针来索引。这样当ARM进入异常模式的时候，程序就可以把一般通用寄存器压入堆栈，返回时再出栈，保证了各种模式下程序的状态的完整性。
2. r14（LR）：每种模式下都有自己的LR寄存器，它有两个特殊功能。
   1. 保存子程序的返回地址。比如说，使用BL或BLX时，跳转指令自动把返回地址放入r14中；子程序通过把r14复制到PC中实现返回，通常使用下面的指令：

|  |
| --- |
| MOV PC，LR  BX LR |

通常子程序这样写，保证了子程序中还可以调用子程序。

|  |
| --- |
| stmfd sp!, {lr}  ……  ldmfd sp!, {pc} |

LDM/STM指令主要用于现场保护，数据复制，参数传递。

**STMFD指令**

STMFD Rn{!}, {reglist}{^}

其中，寄存器Rn为基址寄存器，装有传送数据的初始地址，Rn不允许为R15（PC）；后缀“！”表示最后的地址写回到Rn中；寄存器列表reglist可包含多于一个寄存器或寄存器范围，使用“，”分隔开，如{R1，R2，R6-R9}，寄存器排列由小到大排列；“^”后缀不允许在用户模式和系统模式下使用，若在LDM指令用的寄存器列表中包含PC寄存器时使用，那么除了正常的多寄存器传送外，将SPSR拷贝到CPSR中，这可用于异常处理返回；使用“^”后缀进行数据传送且寄存器列表不包含PC寄存器时，加载/存储的是用户模式的寄存器，而不是当前模式的寄存器。

**举例说明**：

|  |
| --- |
| STMFD SP！，{R0-R7，LR}  对于这条指令伪代码的解释，大体是这样的：  SP = SP-9×4; // SP的指针始终指向栈顶的位置  address = SP; // 从栈顶开始存放  for i = 0 to 7  Memory[address] = Ri; // 循环把R0-R7保存在栈中  address = address + 4; // 地址+4  Memory[address] = LR; // 把LR寄存器的内容保存到栈中 |

假设此时的SP寄存器内的地址为：0x40000460，那么前面的伪代码可以解释如下：



**LDMFD指令**

LDMFD Rn{!}, {reglist}{^}

这条指令的意思是以Rn为基址（起始地址），取值写入寄存器列表。

**举例说明：**

|  |
| --- |
| LDMFD SP！，{R0-R7，PC}^  对于这条指令伪代码的解释，大体是这样的：  address = SP;  for i = 0 to 7  Ri = Memory[address ,4]  address = address + 4;  SP = address; |

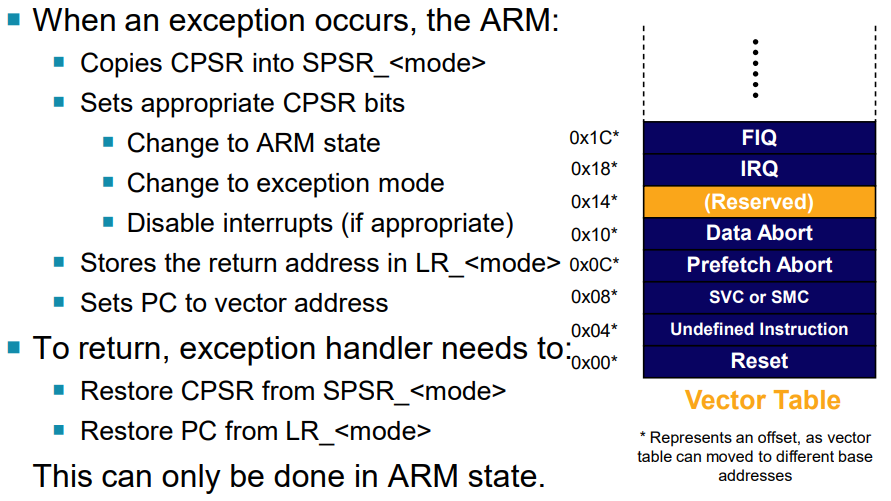
假设此时的SP寄存器内的地址为：0x4000043C，那么前面的伪代码可以解释如下：



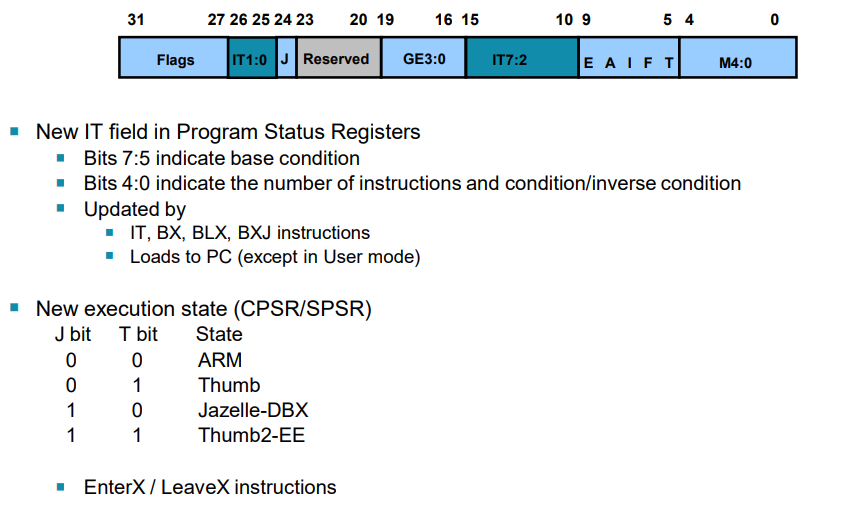
* 1. 当异常发生时，异常模式的r14用来保存异常返回地址，将r14入栈可以处理嵌套中断。

1. r15（PC）：程序计数器。PC是有读写限制的。当没有超过读取限制的时候，读取的值是指令的地址加8后读取的内容，因为ARM指令总是以字对齐的，故bit[1:0]总是00。当用str或stm存储PC的时候，偏移量有可能是8或12等其它值。在V3及以下版本中，写入bit[1:0]的值将被忽略，而在V4及以上版本写入r15的bit[1:0]必须为00，否则后果不可预测。

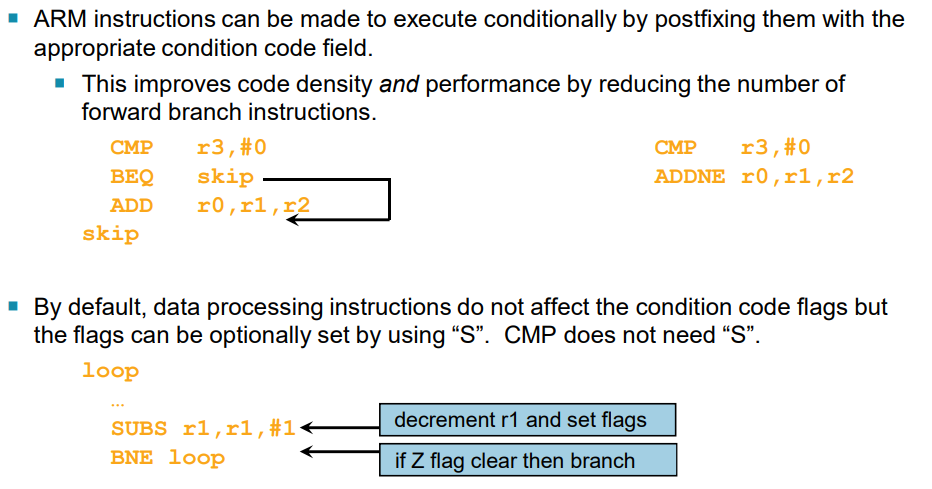
## 2.8 Cortex-A8异常处理



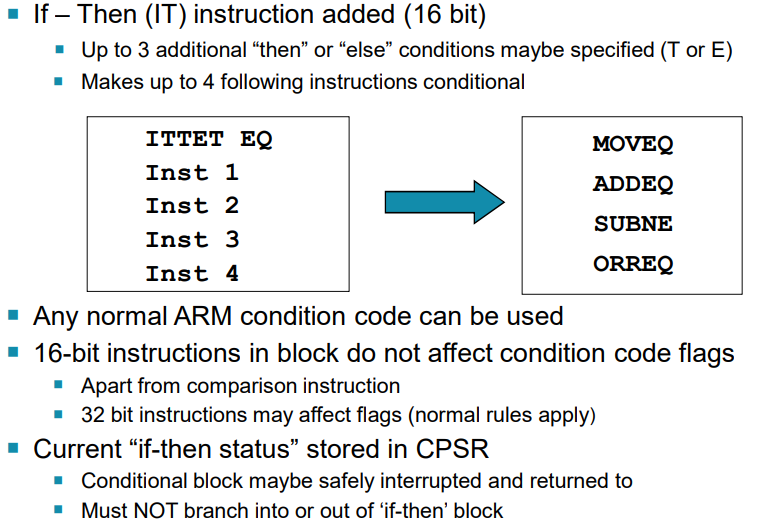
## 2.9 Cortex-A8 程序状态寄存器



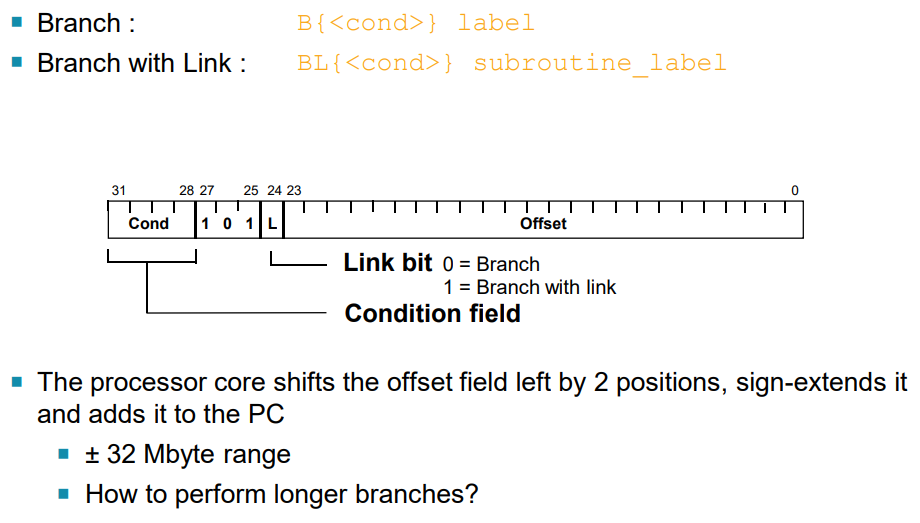
## 2.10 条件执行和标志



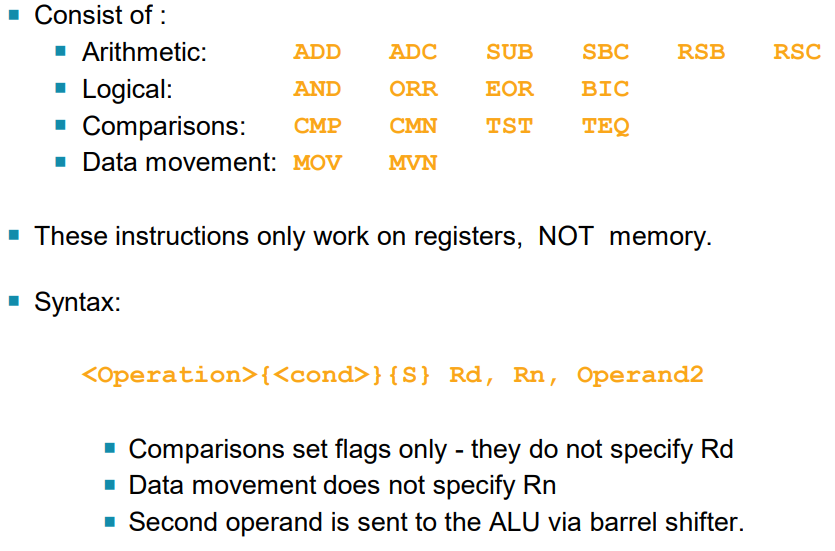
### 2.10.1 16位条件执行



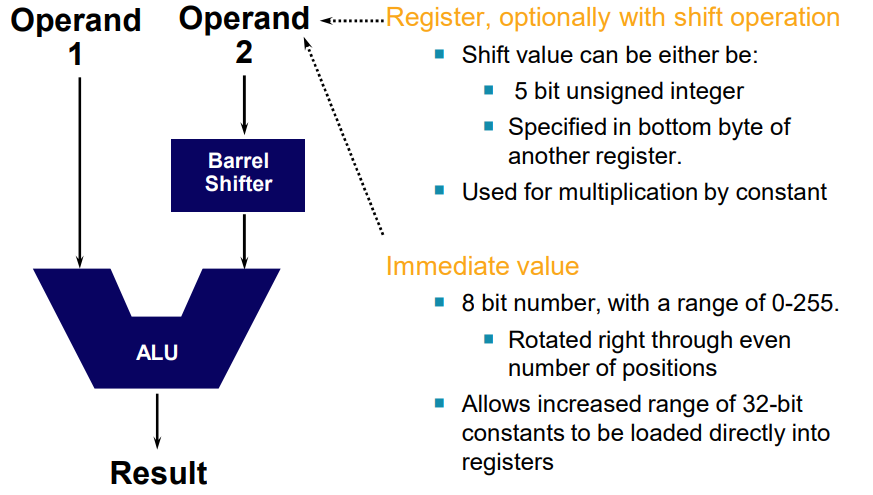
## 2.11 分支指令



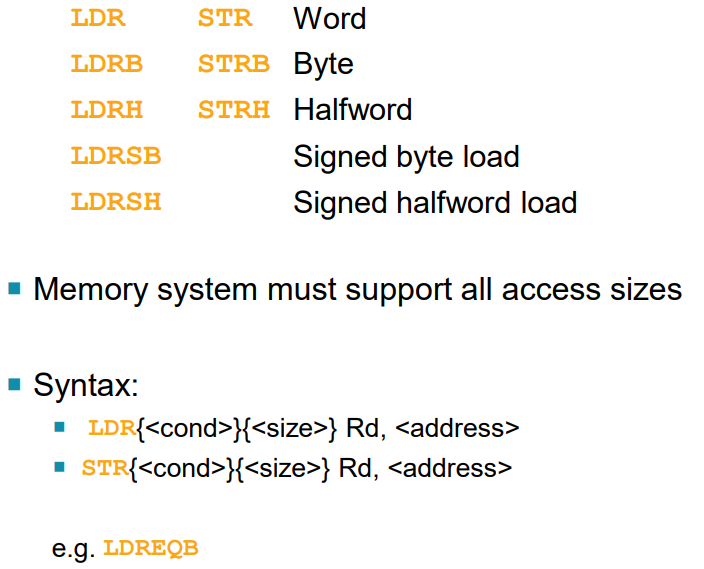
## 2.12 数据处理指令



## 2.13 Using a Barrel Shifter: The 2nd Operand

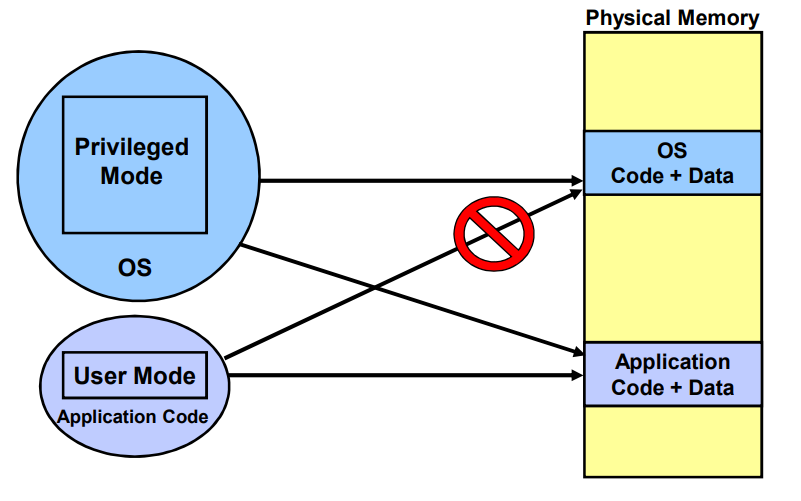


## 2.14 单个寄存器数据传送指令

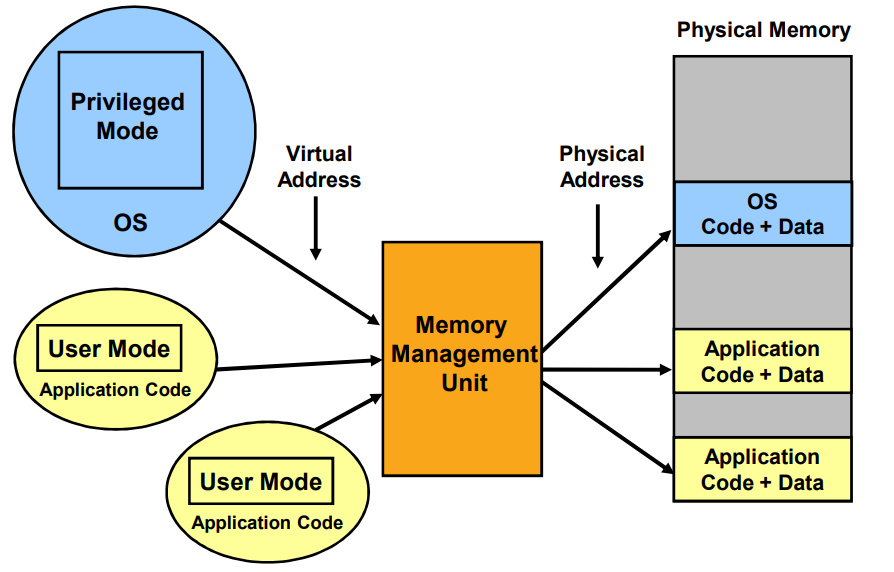


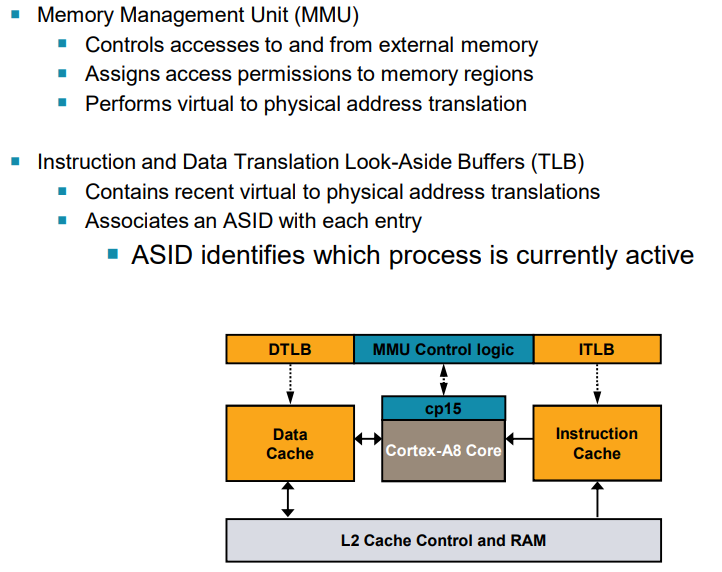
# 3 内存管理

## 3.1内存保护

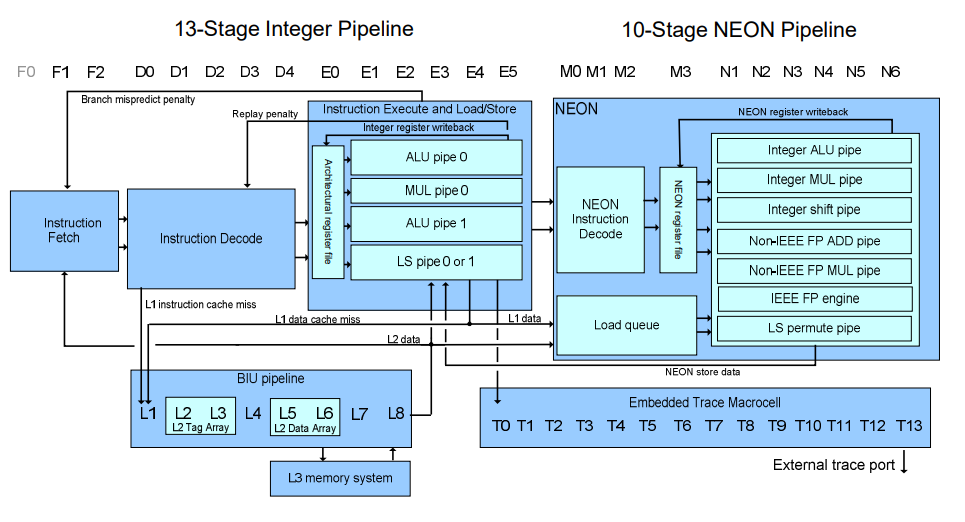


3.2 内存分配

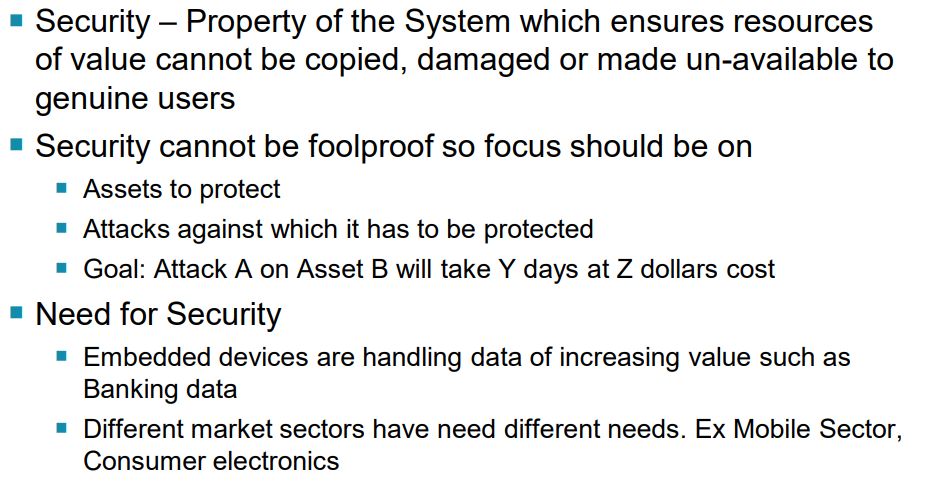




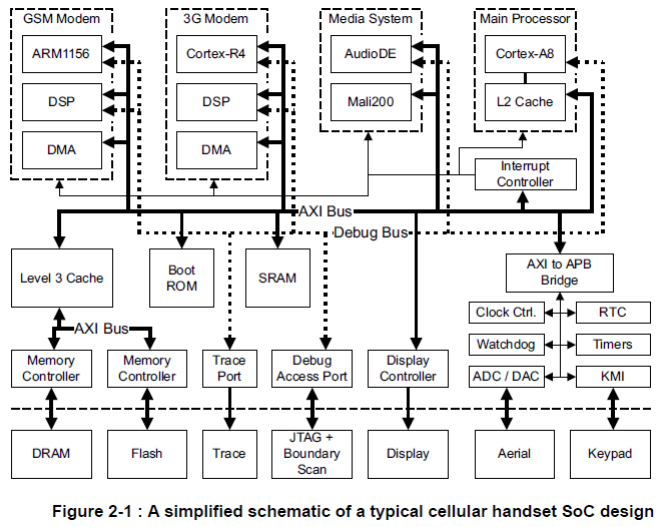
# 4 Cortex-A8流水线



# 5 安全性-TrustZone



## 5.1 Cellular Handset SoC Design



# 参考

1. Cortex-A8 Technical Reference Manual
2. ARM Architecture Reference Manual v7-AR
3. RealView Compilation Tools Compiler Reference Guide
4. RealView Compilation Tools Compiler User Guide
5. <http://infocenter.arm.com>
6. <http://www.arm.com/support/university/>
7. [University@arm.com](mailto:University@arm.com)