ARM地址空间

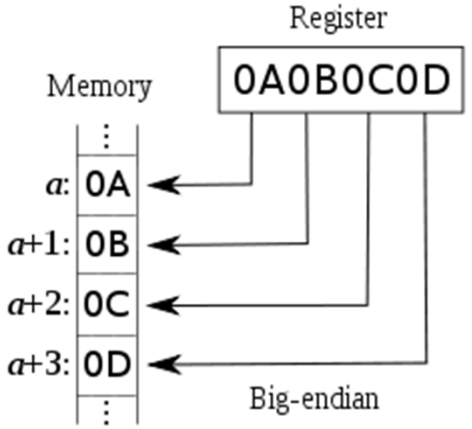
- 存储器按字节编址，寻址范围是4G
  - 单一平面的2^{32}字节的线性地址空间
  - 地址空间大小：
    - 2^{32}个字节
    - 2^{30}个字（32位）
    - 2^{31}个半字（16位）
- I/O地址空间
  - 存储器映射方式，通过访存指令（load/store）访问
  - 通常存储器映射的I/O空间没有Cache和Write buffer
- 端模式（字节顺序）
  - 可以用 little endian/big endian模式存取数据

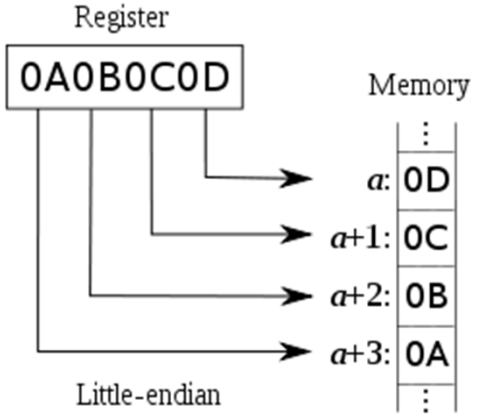
Big Endian and Little Endian

- 定义
  - 大端（Big-Endian）：高字节在低地址，低字节在高地址
  - 小端（Little-Endian）:低字节在低地址，高字节在高地址【ARM9默认模式】
- 例：数字0x0A0B0C0D在内存中
  - 大端：
    - 低地址 ----------> 高地址

      0x0A | 0x0B | 0x0C | 0x0D
  - 小端：低地址 ----------> 高地址

    0x0D | 0x0C | 0x0B | 0x0A

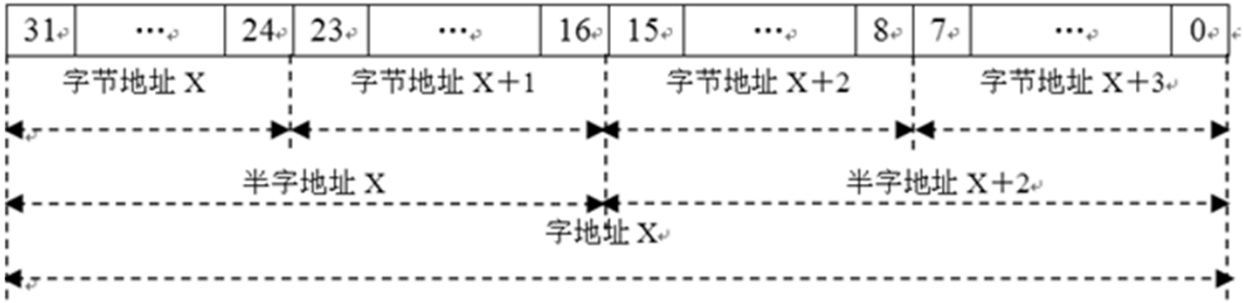




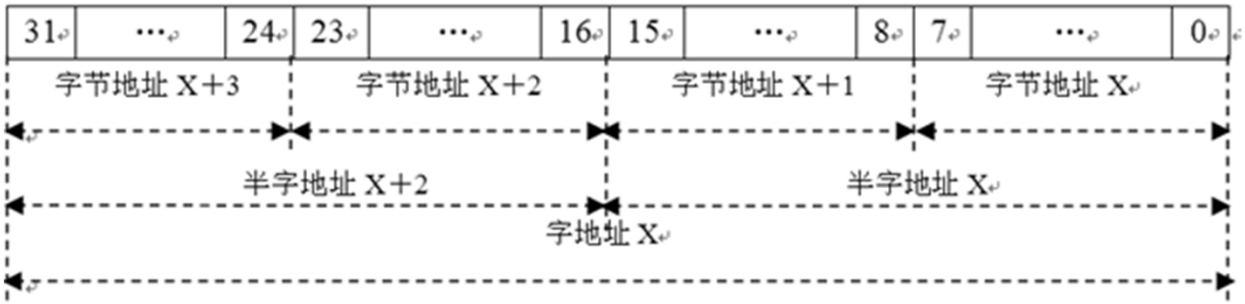
Mapping registers to memory locations

大端存储和小端存储

- 大端模式：存储器字地址是该字中最高有效字节所对应的字节地址



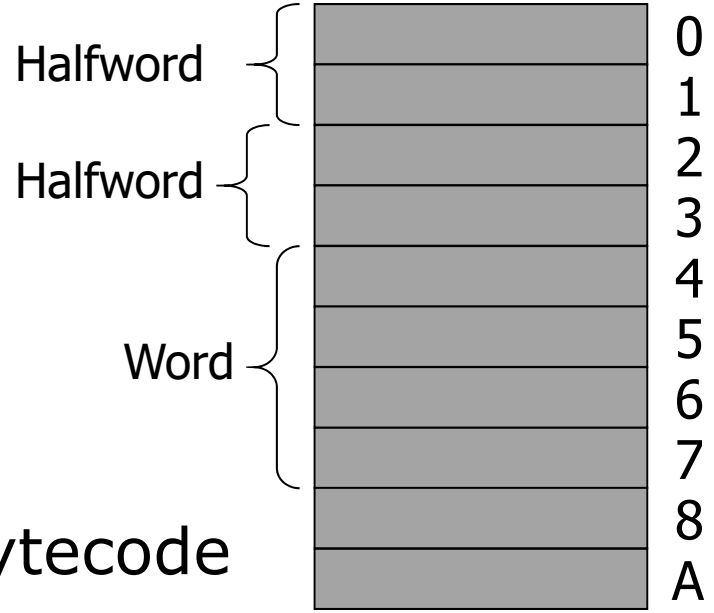
- 小端模式：存储器字地址是该字中最低有效字节所对应的字节地址



数据和指令类型

- **数据类型**
  - Byte：　8**位**
  - Halfword：16**位（2字节边界对齐）**
  - Word：　32**位（4字节边界对齐）**

- **指令**
  - ARM 32**位指令**
    - 4字节边界对齐
  - Thumb 16**位指令**
    - 2字节边界对齐
  - Jazelle**变种内核支持** Java bytecode



处理器工作模式

- **7个基本工作模式**
  - **用户模式**User（usr）：**非特权模式**
    - 大部分任务正常执行在这种模式
  - **系统模式**System（sys）：**操作系统的特权用户模式**
    - 使用和User模式相同寄存器集的特权模式
  - **（普通）中断模式**IRQ（irq）：**响应普通中断时的处理模式**
  - **快速中断模式**FIQ（fiq）：**响应快速中断时的处理模式**
  - **管理员模式**Supervisor（svc）：**操作系统保护模式**
    - 复位或执行软中断指令时进入的模式
  - **中止模式**Abort（abt）：**虚拟存储或存储器保护**
    - 访存异常时进入的模式
  - **未定义指令模式**Undefined（und）:**软件仿真硬件协处理器**
    - 执行未定义指令时进入的模式

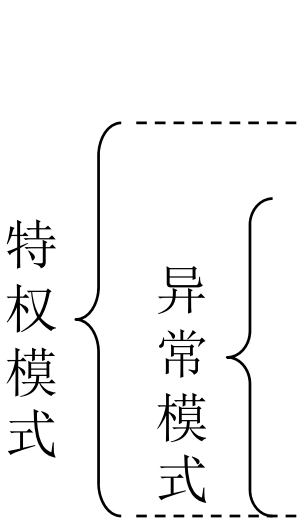
处理器工作模式

特权模式

- User (usr): ARM**正常程序执行模式**
- System (sys): **操作系统的特权用户模式**
- FIQ (fiq): **响应快速中断时的处理模式**
- IRQ (irq): **响应普通中断时的处理模式**
- Supervisor (svc): **操作系统保护模式**
- Abort mode (abt): **虚拟存储或存储器保护**
- Undefined (und): **硬件协处理器的软件仿真**

- 特权模式（Privileged Mode）：程序可以访问所有的系统资源，也可以通过软件控制进行处理器模式的切换
- 用户模式下不能通过软件改变模式，也不能访问被保护的系统资源

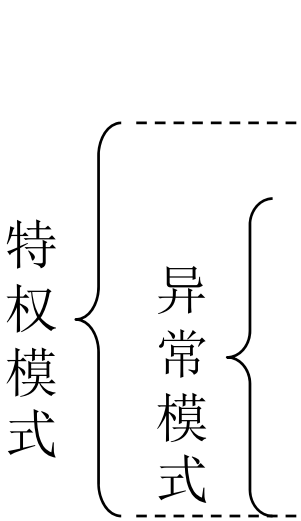
处理器工作模式



- User (usr): ARM**正常程序执行模式**
- System (sys): **操作系统的特权用户模式**
- FIQ (fiq): **响应快速中断时的处理模式**
- IRQ (irq): **响应普通中断时的处理模式**
- Supervisor (svc): **操作系统保护模式**
- Abort mode (abt): **虚拟存储或存储器保护**
- Undefined (und): **硬件协处理器的软件仿真**

- 异常模式：
  - 可以通过程序切换进入，也可以在外部中断或异常发生时进入
  - 每种异常模式都有一些独立的寄存器，以避免异常退出时用户模式的状态不确定

处理器工作模式



- User (usr): ARM**正常程序执行模式**
- System (sys): **操作系统的特权用户模式**
- FIQ (fiq): **响应快速中断时的处理模式**
- IRQ (irq): **响应普通中断时的处理模式**
- Supervisor (svc): **操作系统保护模式**
- Abort mode (abt): **虚拟存储或存储器保护**
- Undefined (und): **硬件协处理器的软件仿真**

- 用户模式与系统模式：
  - 使用完全相同的寄存器组
  - 都不能由异常进入
  - 系统模式：
    - 特权模式，不受用户模式的限制
      - 操作系统的特权任务可以访问受控的资源
    - 操作系统在该模式下可访问用户模式的寄存器

处理器的工作状态

- ARM微处理器的两种工作状态
  - ARM状态：处理器执行32位的字对齐的ARM指令
  - Thumb状态：处理器执行16位、半字对齐的Thumb指令
- 可在两种状态之间切换

ARM 寄存器组织

- ARM 有37个32位寄存器
  - 1个用作PC（program counter）
  - 1个用作CPSR（current program status register）
  - 5个用作SPSR（saved program status registers）
  - 30个通用寄存器
- 根据处理器的状态及工作模式被安排成不同的组
  - “逻辑”寄存器的名称：R0~R15、CPSR、SPSR
  - R0~R7：不分组寄存器
  - R8~R14：根据工作模式分组的寄存器
  - R15：程序计数器，不分组
  - CPSR：当前程序状态寄存器
  - SPSR：各工作模式下保留CPSR值的寄存器

ARM 寄存器组织

- 当前处理器的工作模式决定哪组寄存器可操作
- 任何模式都可以访问的寄存器：
  - 相应模式的R0-R12子集
  - 相应模式的 R13（stack pointer, sp）和R14（link register, lr）
  - 相应模式的 R15（program counter, pc）
  - 相应模式的CPSR（current program status register, cpsr）
- 异常模式（除system模式之外的特权模式）还可以访问：
  - 相应模式的 SPSR（saved program status register）

ARM 寄存器组织

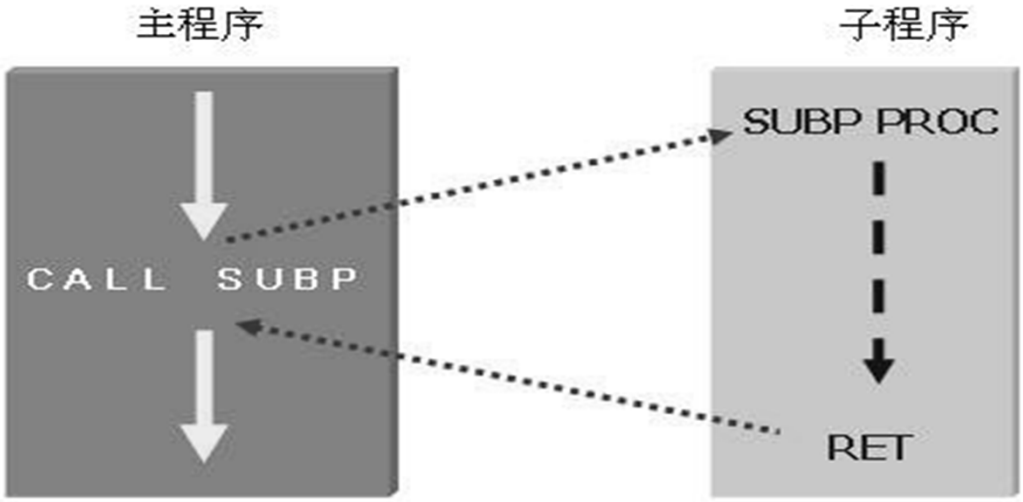
- ARM处理器工作在不同模式时使用的寄存器不同：37
  - 无论何种模式，R15均作为PC使用 1
  - CPSR为当前程序状态寄存器 1
  - R7~R0为公用的通用寄存器 8
  - R8~R12共2组 5*2=10
    - 标有“_fiq”的寄存器为快速中断模式专用，与其他模式的R8~R12使用不同的物理寄存器（“影子寄存器”）
  - R13、R14：
    - 用户模式和系统模式下分别为堆栈指针SP和链接寄存器LR 1+1
    - 其他模式下有专门的物理寄存器 5*2=10
  - 五个异常模式下可以看到各自的SPSR 5

R13：堆栈指针(sp)

- 在ARM指令中，寄存器R13习惯被用作堆栈指针SP
  - 可使用其他的寄存器作为堆栈指针
  - 没有真正的push和pop指令，用批量Load/Store指令实现入栈、出栈操作
  - Thumb指令集中，某些指令强制要求使用R13作为堆栈指针
- 每种处理器模式都有单独的堆栈
  - 用户应用程序初始化时需初始化每种模式下的R13，使其指向该模式的栈空间
  - 当程序进入异常模式时，可以将需要保护的寄存器压入R13所指向的堆栈；当程序从异常模式返回时，则从对应的堆栈中恢复

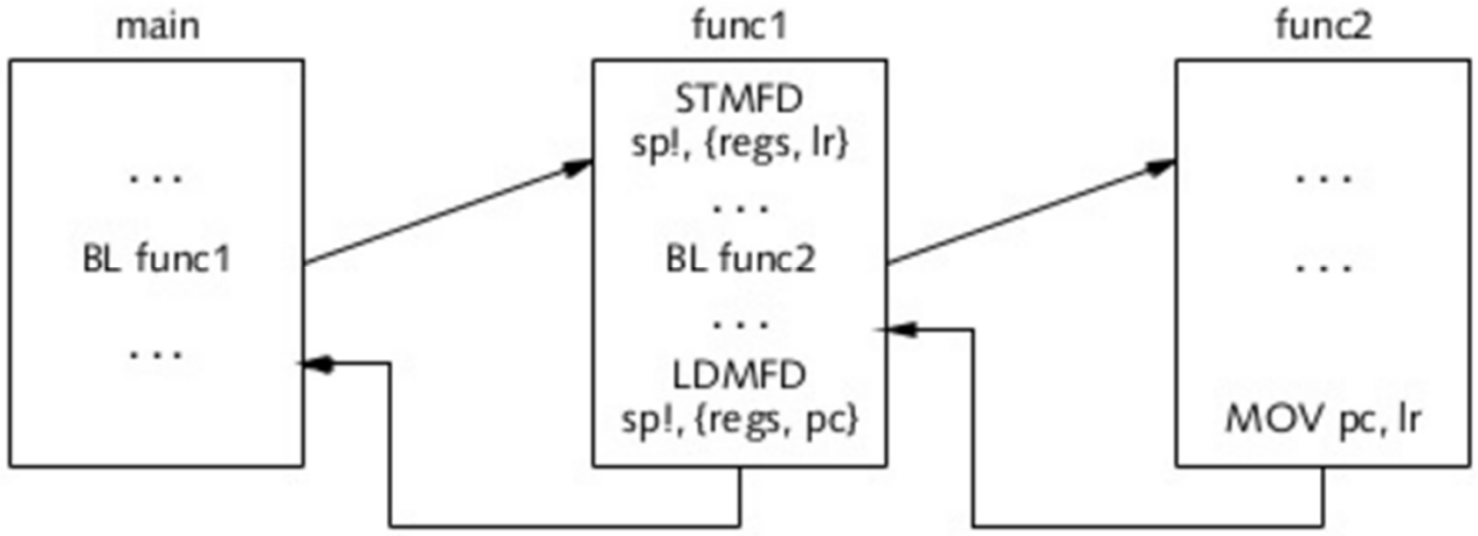
R14：子程序链接寄存器（LR）

- 保存子程序调用或异常引起程序分支时的返回地址
  - 执行子程序调用指令时，R14←R15（程序计数器PC）
  - 当发生中断或异常时，对应的分组寄存器R14_svc、R14_irq、R14_fiq、R14_abt和R14_und用来保存R15的返回值
  - 其他情况下，R14用作通用寄存器



R14：子程序链接寄存器（LR）

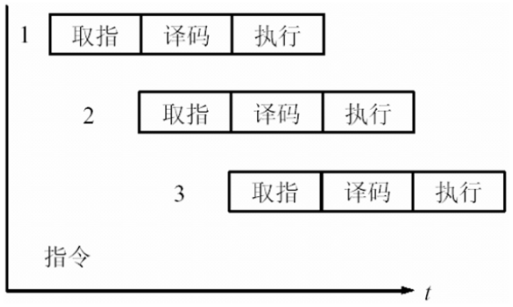
- R14寄存器的嵌套使用
  - 非尾函数需要保存LR到堆栈中
    - 否则当发生嵌套调用时，会发生 R14寄存器冲突：被调用函数会覆盖调用函数的LR
  - 只有在尾函数（不调用任何函数的函数）中，才不需要保存LR

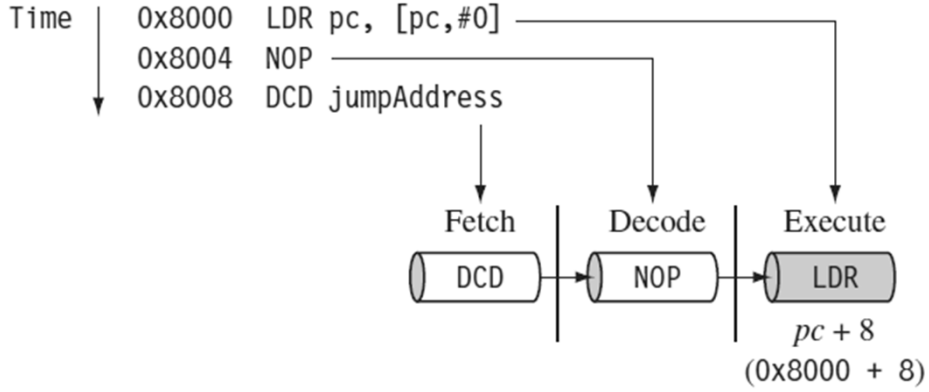


R15：程序计数器(PC)

- **在ARM状态，所有指令均为32位等长指令**
- **所有指令必须字对齐（32位）**
  - R15寄存器的最低两位恒为0
- **PC总是指向“正在取指”的指令**
  - 流水线技术：不是指向“正在执行”的指令或正在“译码”的指令
  - 以“正在执行”的指令作为参考点，PC总是指向第三条指令
    - 兼容ARM7三段流水线架构
  - ARM指令集：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

ARM7 TDMI的3段流水线



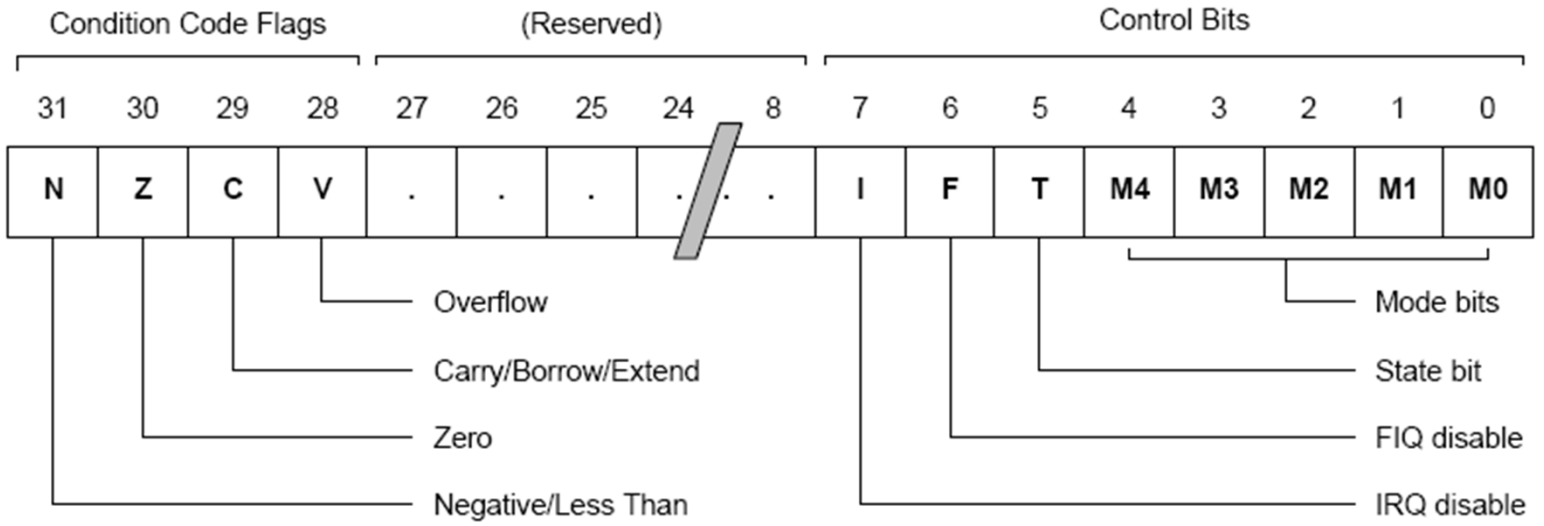


R15：程序计数器 (PC)

- 当处理器处于ARM状态:
  - 所有指令 32 bits 宽，所有指令必须 word 对齐
  - pc值由R15的bits [31:2]决定, bits [1:0]未定义
- 当处理器处于Thumb状态:
  - 所有指令 16 bits 宽，所有指令必须 halfword 对齐
  - pc值由R15的bits [31:1]决定, bits [0] 未定义
- 当处理器处于Jazelle状态:
  - 所有指令 8 bits 宽
  - 处理器执行 word 存取，一次取4条指令

程序状态寄存器（CPSR/SPSR）

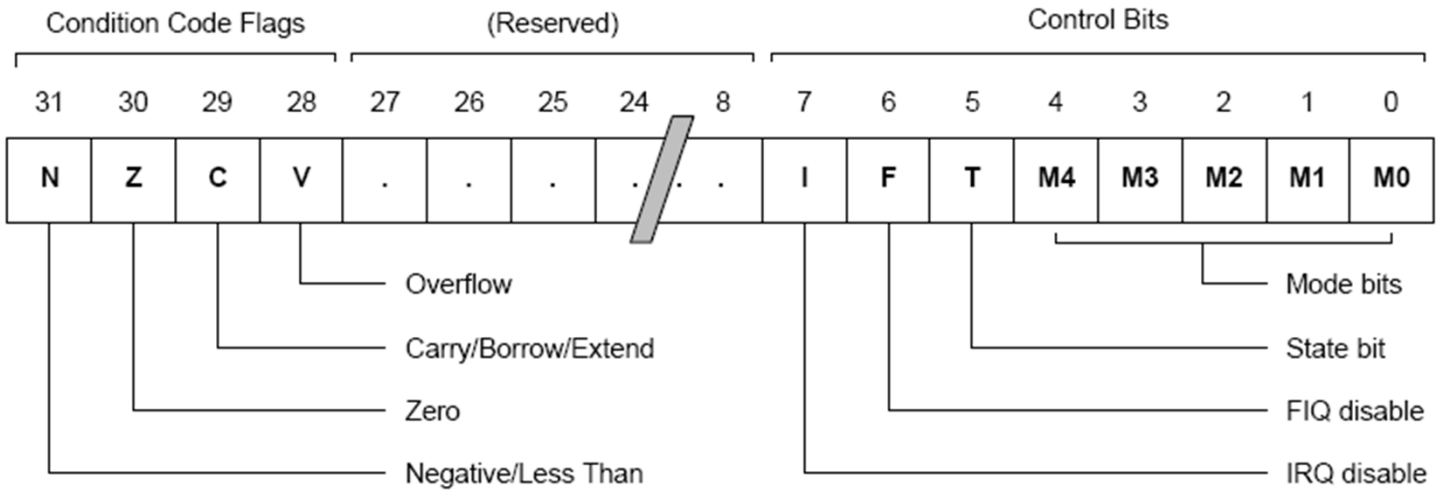
- CPSR（Current Program Status Register）
  - 保存程序运行的当前状态
  - 在所有处理器模式下都是同一个物理寄存器



- 模式位M[4:0]
  - ✓ 10000:用户模式
  - ✓ 10001:FIQ模式
  - ✓ 10010:IRQ模式
  - ✓ 10011:管理员模式
  - ✓ 10111:中止模式
  - ✓ 11011:未定义
  - ✓ 11111:系统模式

程序状态寄存器（CPSR/SPSR）

- CPSR（Current Program Status Register）
- SPSR（saved program status register）
  - **分组的备份程序状态寄存器**
  - **每种异常都有自己的SPSR，共5个物理寄存器**
    - 进入异常时保存异常发生之前的CPSR当前值
    - 异常退出时通过其恢复CPSR



- 模式位M[4:0]
  - ✓ 10000:用户模式
  - ✓ 10001:FIQ模式
  - ✓ 10010:IRQ模式
  - ✓ 10011:管理员模式
  - ✓ 10111:中止模式
  - ✓ 11011:未定义
  - ✓ 11111:系统模式

程序状态寄存器（CPSR/SPSR）

- **条件码标志：指令执行结果的特征**
  - **N（负或小于）标志：带符号数（补码）**
    - 负数：N=1
    - 正数或0：N=0
  - **Z（零）标志：**
    - 0：Z=1
    - 非0：Z=0
  - **V（溢出）标志**
    - 加法或减法指令（带符号数）
      - **有溢出：V=1**
      - **无溢出：V=0**
    - 非加法/减法指令：
      - **V标志不变**

程序状态寄存器（CPSR/SPSR）

- **条件码标志：指令执行结果的特征**
  - **C（进位、借位、扩展）标志：**
    - 加法指令以及比较指令CMN
      - **有进位：C=1**
      - **无进位：C=0**
    - 减法指令以及比较指令CMP
      - **有借位：C=0**
      - **无借位：C=1**
    - 非加法/减法的移位操作指令
      - **C标志值为移出的最后一位的值**
    - 其他非加法/减法指令
      - **C标志不变**

程序状态寄存器（CPSR/SPSR）

- **控制位：**
  - I：IRQ**异常禁止位（关中断位）**
    - I=1时，禁止IRQ异常；I=0时，允许IRQ异常
  - F：FIQ**异常禁止位**
    - F=1时，禁止FIQ异常；F=0时，允许FIQ异常
  - T：**（微处理器）状态位，仅**ARMxT**架构支持**
    - T=0时，为ARM状态；T=1时，为Thumb状态
  - M4、M3、M2、M1、M0：**处理器工作模式**

10000	10001	10010	10011	10111	11011	11111
User	FIQ	IRQ	Super	Abort	Undef	Sys

程序状态寄存器（CPSR/SPSR）

- 控制位：
  - I：IRQ异常禁止位（关中断位）
  - F：FIQ异常禁止位
  - T：（微处理器）状态位，仅ARMxT架构支持
  - M4、M3、M2、M1、M0：处理器工作模式
- 发生异常时，控制位被硬件改变
- 处理器处于特权模式时，可用软件修改控制位

工作模式切换

- 用户模式
  - 通过指令修改CPSR寄存器mode位进入用户模式
  - 除非发生异常，否则在用户模式不能改变工作模式
- 异常模式
  - 发生中断或异常时，处理器自动改变mode位进入相应的异常模式
  - 每一种异常模式下都有一组备份寄存器，可以保证在进入异常模式时，用户模式下的寄存器不被破坏

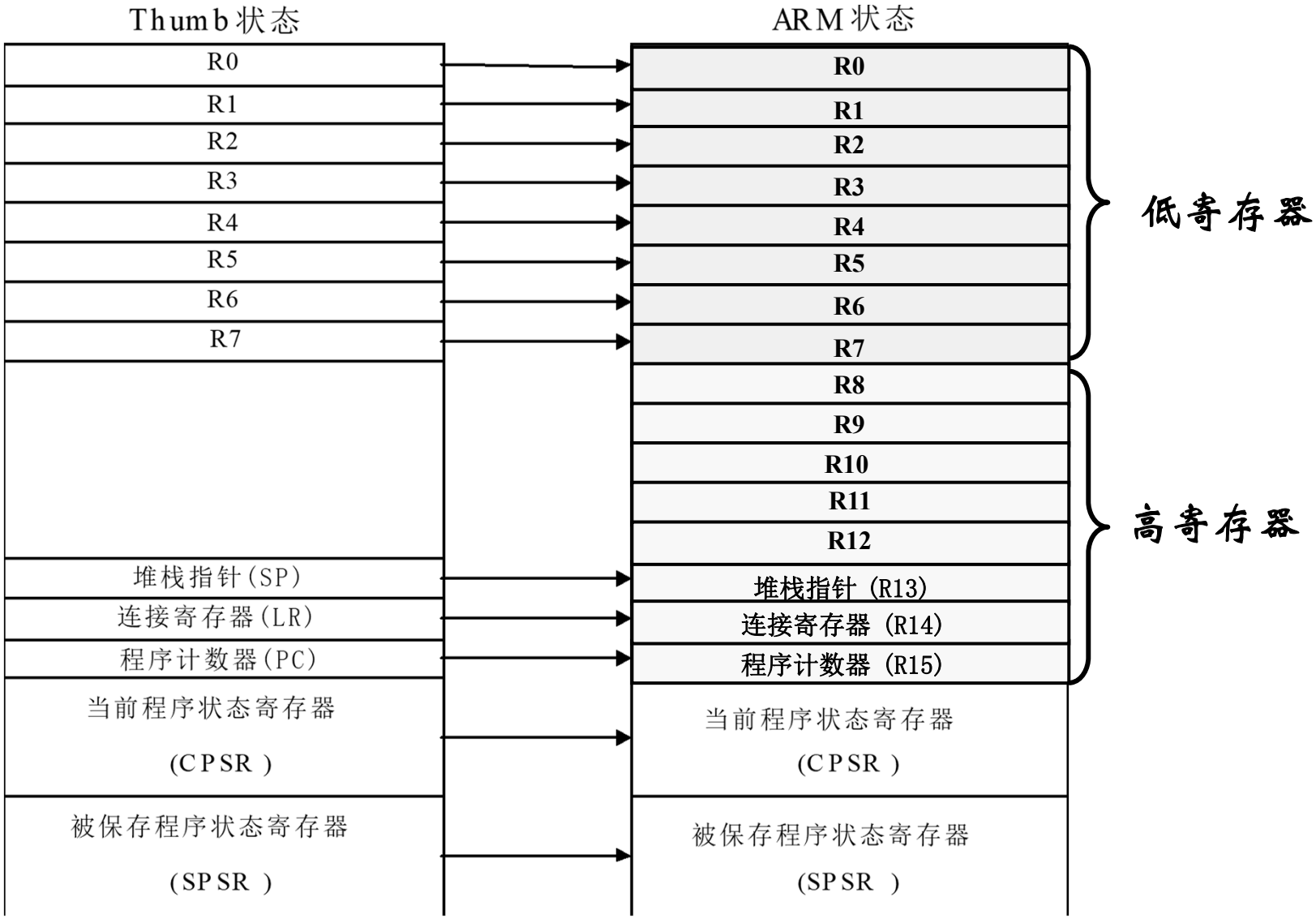
工作模式切换

- **管理员模式**
  - **复位或执行软中断指令时进入**
- **系统模式**
  - **主要供需要特权的操作系统任务使用**
    - 通常操作系统的任务需要访问所有的系统资源
    - 系统模式下的操作系统任务仍使用用户模式的寄存器组，而不是使用异常模式下的寄存器组，保证当异常发生时任务状态不被破坏
  - **通过修改CPSR进入**

ARM状态和Thumb状态之间寄存器的关系

- Thumb状态R0～R7与ARM状态R0～R7相同
- Thumb状态CPSR和SPSR与ARM状态CPSR和SPSR相同
- Thumb状态SP映射到ARM状态R13
- Thumb状态LR映射到ARM状态R14
- Thumb状态PC映射到ARM状态PC（R15）

ARM状态和Thumb状态之间寄存器的关系



ARM的异常

- 异常：内部或外部产生的需要微处理器处理的事件
  - 正常的程序执行流程被暂时中断而引发的过程
- 7种异常
  - 复位：进入管理员模式
  - 未定义指令：进入未定义模式
  - 软件中断（SWI）：进入管理员模式
  - 预取中止：进入中止模式
  - 数据中止：进入中止模式
  - IRQ：进入IRQ模式
  - FIQ：进入FIQ模式
- 不同异常导致处理器进入不同的工作模式
  - 并执行不同的特定地址的指令

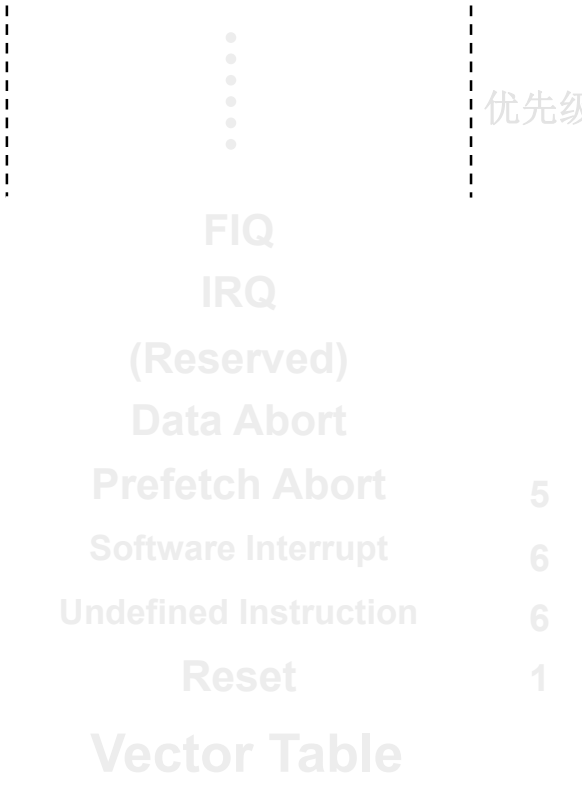
异常向量表和异常优先级

- 中断向量表的位置
  - 低端:0x00000000
  - 高端:0xFFFF0000
  - 由实现决定

Vector table can be at
0xFFFF0000 on ARM720T
and on ARM9/10 family devices

ARM的异常

- 复位
  - 立即停止当前程序，进入禁止中断的管理员模式
  - 从地址0x00000000或者0xFFFF0000处开始执行
- 未定义指令
  - 在ARM处理器或协处理器认为当前指令未定义时发生
- 软件中断
  - 用户模式下的程序执行指令SWI时，处理器产生软件中断
    - 进入管理员模式，调用特权操作



Vector table can be at **0xFFFF0000** on ARM720T and on ARM9/10 family devices

ARM的异常

- 指令预取中止
  - 处理器预取指令的地址不存在，或该地址不允许当前指令访问时，存储器向处理器发出中止信号
  - 只有当预取的指令被执行时，才会产生指令预取中止异常
- 数据访问中止
  - 处理器数据访问指令的地址不存在，或该地址不允许当前指令访问时，产生数据中止异常

Vector table can be at
0xFFFF0000 on ARM720T
and on ARM9/10 family devices

ARM的异常

- 外部中断请求
  - 处理器的外部中断请求引脚irq有效，且CPSR中I位为0时，发生IRQ异常
  - 系统外设可通过该异常请求中断服务
- 快速中断请求
  - 处理器的快速中断请求引脚fiq有效，且CPSR中的F位为0时，发生FIQ异常

异常处理

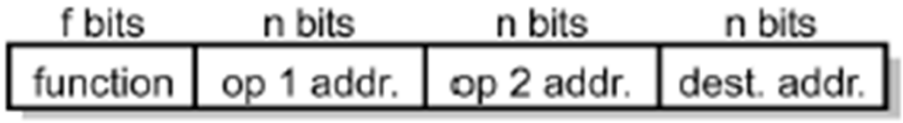
- **当异常产生时**, ARM core:
  - **拷贝** CPSR **到** SPSR_<mode>
  - **设置适当的** CPSR **位:**
    - 改变处理器状态进入ARM态
    - 改变处理器模式进入相应的异常模式
    - 设置中断禁止位禁止相应中断 (如果需要)
  - **保存返回地址到** LR_<mode>
  - **设置PC为相应的异常向量**

异常返回

- **异常结束时，异常处理程序必须：**
  - **从** SPSR_<mode>**恢复**CPSR
    - 恢复CPSR的动作会将T、F和I位自动恢复为异常发生前的值
  - **从**LR_<mode>**恢复**PC
- **复位系统后，开始应用程序的执行，故复位异常处理程序不需要返回**

ARM指令系统特点

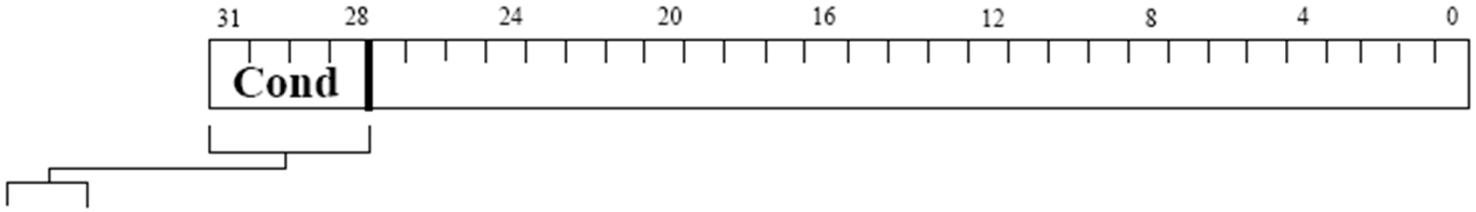
- 32位等长指令字指令格式
- 所有ARM指令都使用4位条件编码决定指令是否执行
- Load/Store体系结构
  - 数据操作只会用到寄存器，不会直接访问内存
- 通过协处理器可以扩充指令
- 3操作数格式



指令语法	目标寄存器（Rd)	源寄存器1(Rn)	源寄存器2(Rm)
ADD r3,r1,r2	**r3**	**r1**	**r2**

ARM指令的条件字段

- 当处理器工作在ARM状态时，几乎所有指令均为条件执行
  - 执行条件满足：指令被执行
  - 执行条件不满足：指令被忽略
  - 条件：指令中的条件字段
  - 状态：CPSR中的条件码

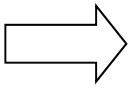


- 每一条ARM指令包含4位条件码，位于指令的最高4位[31:28]

ARM指令中的条件字段

- EQ/NE: **等于/不等于**(equal / not equal)
- HS/LO: **无符号数高于或等于/无符号数小于**(higher or same/lower)
- HI/LS: **无符号数高于/无符号数低于或等于**(higher/lower or same)
- GE/LT: **有符号数大于或等于/有符号数小于**(greater or equal / less than)
- GT/LE: **有符号数大于/有符号数小于或等于**(greater than / less or equal)
- MI/PL: **负/非负**
- VS/VC: **溢出/不溢出**(overflow set / overflow clear)
- CS/CC: **进位/无进位**(carry set / carry clear)

```
if ( (a == b) && (c == d) )
{
        e++;
}
```



```
// r0, r1, r2, r3, r4 : a, b, c, d, e
cmp    r0, r1
cmpeq r2, r3
addeq  r4, r4, #1
```

ARM汇编指令中的操作数符号

- #：立即数前缀
  - 二进制数、十进制数或十六进制数
  - 以十六进制表示的立即数：#0x
  - 以二进制表示的立即数：#0b
  - 以十进制表示的立即数：#0d，或#（缺省）
- ！：更新基址寄存器后缀
  - 指令执行后将最后的地址写入基址寄存器
- -：寄存器列表范围符号
  - 表示多个连续寄存器组成的寄存器列表
  - 例：R0-R7表示R0、R1、R2、R3、R4、R5、R6和R7

ARM指令的寻址方式

- 7种常见的寻址方式
  - 立即寻址
  - 寄存器寻址
  - 寄存器间接寻址
  - 基址加变址寻址（变址寻址）
  - 相对寻址
  - 堆栈寻址
  - 块拷贝（多寄存器）寻址

ARM Addressing Modes

- 立即寻址
  - ADD R0，R0，#1 ；R0←R0＋1
- 寄存器寻址
  - ADD R0，R1，R2 ；R0←R1＋R2
- 寄存器间接寻址
  - LDR R0，[R1] ；R0←[R1]
  - STR R0，[R1] ；[R1]←R0

ARM Addressing Modes

- **基址加变址寻址（变址寻址）**
  - **立即数前变址：** Immediate pre-indexed
    - 基址寄存器存放的地址先变化，然后执行指令的操作
    - LDR R0，[R1，#4]　　；R0←[R1＋4]
    - 基址寄存器R1的内容加上位移量4，所指向的存储单元的值送至R0
    - 可使用一个基址寄存器访问位于同一区域的多个存储单元
  - **带有自动变址的立即数前变址寻址**
    - LDR R0，[R1，#4]!　　；R0←[R1＋4]，R1←R1＋4
    - 实现基址寄存器自动修改，让程序追踪一个数据表
    - 等效于"寄存器间接取数指令+数据处理指令"，但不消耗额外的时间

ARM Addressing Modes

- **基址加变址寻址（变址寻址）**
  - **立即数前变址：** Immediate pre-indexed
    - LDR R0，[R1，#4]　　；R0←[R1＋4]
  - **带有自动变址的立即数前变址寻址**
    - LDR R0，[R1，#4]!　　；R0←[R1＋4]，R1←R1＋4
  - **立即数后变址：** Immediate post-indexed
    - 基址寄存器不加偏移作为传送地址使用，完成操作后再加上偏移量送入基址寄存器
    - LDR R0，[R1]，#4　　；R0←[R1]，R1←R1＋4
    - 等效于"寄存器间接寻址取数指令+改变地址的数据处理指令"
  - **寄存器前变址（偏移变址）：** Register pre-indexed
    - 地址偏移存放在另一个寄存器中
    - LDR R0，[R1，R2]　　；R0←[R1＋R2]
  - ……

ARM Addressing Modes

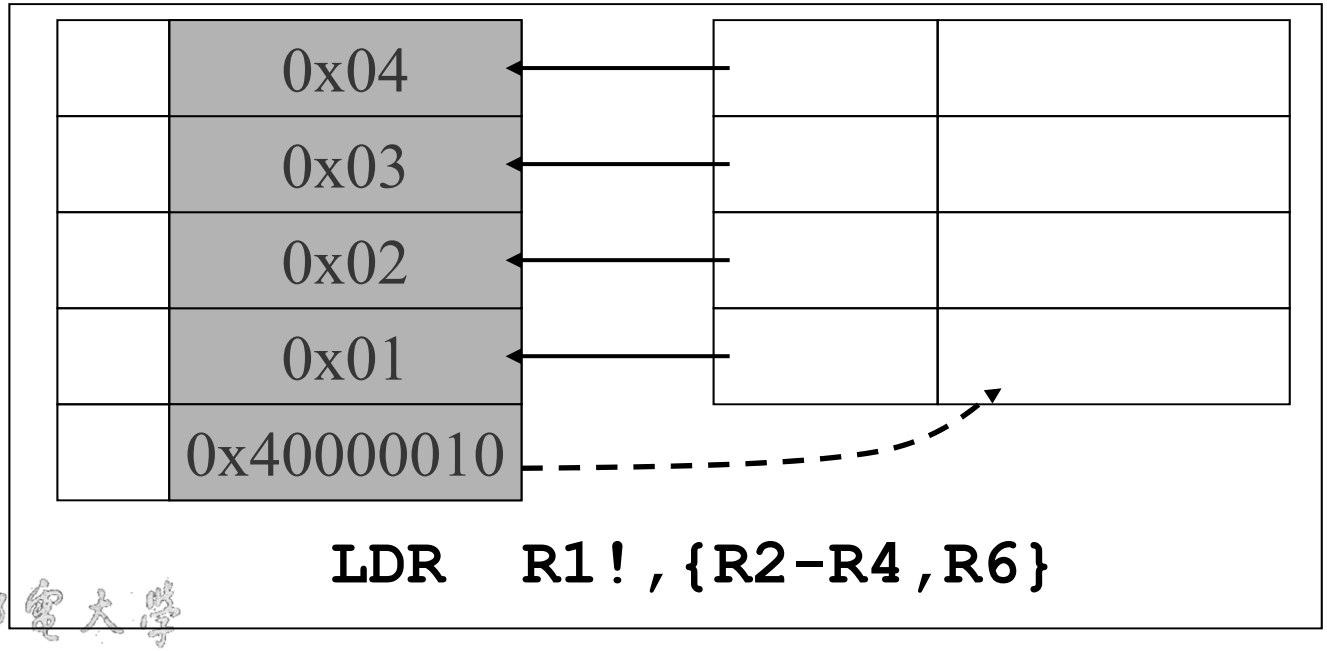
- **相对寻址：**
  - **相对于程序计数器PC**
  - **例：子程序调用**

```
BL    sub_a                ;跳转到子程序suba处执行
.....
sub_a
.....
MOV PC，LR                 ;从子程序返回
```

  - **偏移量由汇编器自动形成**

ARM Addressing Modes

- **块拷贝（多寄存器）寻址**
  - **一条指令可以传送多个寄存器（最多16个）的值**
  - LDMIA R1, {R0, R2, R5} ; R0←[R1], R2←[R1+4]
    ; R5←[R1+8]
  - **将R1所指向的连续3个存储单元中的内容分别送到寄存器R0、R2、R5中**
  - **传送的数据项总是32位的字，基址R1需字对齐**
  - **例：**



ARM Addressing Modes

- **堆栈寻址**
  - **先进后出**
  - **隐含使用堆栈指针寄存器指向存储器区域**
  - **两种类型：**
    - 向上生长（递增，Ascending）堆栈：地址向高地址方向生长
    - 向下生长（递减，Decending）堆栈：地址向低地址方向生长
  - **两种方式**
    - 满堆栈（Full Stack）：SP指向最后压入堆栈的有效数据单元
    - 空堆栈（Empty stack）：SP指向下一个入栈位置（空单元）

ARM Addressing Modes

➢ 堆栈寻址

- ❑ ARM处理器支持上面四种类型的堆栈工作方式：
  - ✉ 满递增堆栈FA（Full Ascending）：堆栈指针指向最后压入的数据单元，且由低地址向高地址生长
  - ✉ 满递减堆栈FD（Full Descending）：堆栈指针指向最后压入的数据单元，且由高地址向低地址生长
  - ✉ 空递增堆栈EA（Empty Ascending）：堆栈指针指向下一个将要放入数据的空单元，且由低地址向高地址生长
  - ✉ 空递减堆栈ED（Empty Descending）：堆栈指针指向下一个将要放入数据的空单元，且由高地址向低地址生长

ARM指令分类

- 数据处理指令：使用和改变寄存器的值
  - 数据传送指令（ MOV等）
  - 算术逻辑运算指令（ADD、SUB、MUL、AND、ORR等）
  - 比较指令（TST、CMP等）
- 数据传送指令（Load/Store）：
  - 存储器单元与寄存器交换数据
    - 把存储器的值拷贝到寄存器中 (load，LDR)
    - 把寄存器中的值拷贝到存储器中(store，STR)
  - 单数据传送（LDR、STR等）
  - 块数据传送（LDM、STM等）

ARM instruction set

- 控制流指令（分支指令）
  - 分支跳转指令（B）
  - 分支和链接，将PC寄存器的值保存到LR寄存器中（BL）
  - 带状态切换的跳转指令（BX）
  - 带返回和状态切换的跳转指令（BLX）
- 程序状态寄存器（PSR）处理指令
  - MRS（Move to Register from State register）
  - MSR（Move to State register from Register）
  - 修改状态寄存器通常通过“读-修改-写回”操作序列实现
- 异常中断指令
  - SWI软中断指令
  - BKPT断点中断指令：停止执行正常指令而进入相应的调试程序
- 协处理器指令
  - CDP、LDC、STC、MCR、MRC