

微嵌第六章习题参考答案

6.6 一条机器指令应该包含哪些要素？

指令字需要包含操作码、操作数(可能多个)和操作数地址等字段

6.12 指令中的操作数可以存放在哪些地方？

- 操作数包含在指令中（立即数）
- 操作数包含在 CPU 的内部寄存器中
- 操作数在存储器中（分为代码区和数据区）

6.15 对比寄存器偏移寻址、前变址寻址和后变址寻址的异同。

- **相同点：**三个寻址方式均基于寄存器偏移寻址，寻址指令的格式基本相同，均由操作码、目的操作数、基址寄存器和地址偏移量组成
- **不同点：**寻址方式上有差异，寄存器偏移寻址是在执行指令时，操作数地址由基址与偏移量相加而得，指令完成后不改变基址寄存器和地址偏移量寄存器的值，指令格式为：opcode Rd,[< Rn >,< offset >]；前变址寻址是在执行指令时，将基址与偏移量的相加得到的操作数地址写回基址寄存器，再按新的基址寻址，指令格式为：opcode Rd,[< Rn >,< offset >]!；后变址寻址是在执行指令时，操作数地址从基址寄存器获取，指令完成后，再将操作数地址加上偏移量的值写入基址寄存器，指令格式为：opcode Rd,[< Rn >],< offset >。

6.20 解释汇编语法“LDM {addr_mode} <Rn>{!}, <registers>”中各部分要素的含义。

LDM 是多寄存器加载指令；

{addr_mode}是可选后缀，可选择 IA、IB、DA 和 DB 四种方式；

< Rn >是基址寄存器；

{!}是可选项，表示需要将修改后的地址写入基址寄存器< Rn >；

< registers >是载入数据的寄存器集合。

6.24 解释堆栈寻址和 PC 相对寻址两种方式中基地址各存放在哪里？

堆栈寻址基地址存放在堆栈指针寄存器中，PC 相对寻址基地址就是程序计数器寄存器当前值。

6.34 什么是符号扩展？

在进行数据扩展时，扩展位与符号位保持一致以保护有符号数的符号位即为符号扩展。

6.35 MOV 指令是否可以完成从一个存储器单元到一个寄存器的数据传送？为什么？

不能，MOV 指令只能实现处理器内部不同电路单元之间的数据传送，即将数据从一个寄存器送到另外一个寄存器、通用寄存器和特殊寄存器之间传送数据、将立即数送到寄存器。

6.40 为什么“BL”或“BLX”指令适用于函数调用，而“B”指令不适合。

“B”指令是无条件跳转指令，没有对返回跳转前原函数执行位置的保护，执行用于函数调用无法返回主函数；而“BL”和“BLX”指令在执行时将返回地址保存在 LR 中，适用于函数调用。

6.42 “CBZ”指令的作用与“CMP”指令组合“BEQ”指令有什么区别？

CBZ 在执行跳转后会不会影响 APSR，且只能向前跳转，不能向后跳转，跳转范围为当前指令后 4~130 字节，且只能使用 R0 到 R7，常用于小范围的循环控制；“CMP”指令组合“BEQ”指令会更新 APSR 的值，可以用于任意位置的大范围跳转。

6.43 解释饱和和溢出的区别。

- 溢出是在传统运算方式中，数据达到可表示的最大值/最小值时若还需增/减，自动回到计数零点；
- 饱和是当数据达到可表示的最大值/最小值时不再增/减，而是保持在这个最大值/最小值上，只有加减运算才有饱和方式。

6.45 为什么 ARM 的存储器访问指令要提供非特权加载和存储功能？

ARM 处理器区分特权和非特权访问等级。运行于非特权访问等级的程序无法访问特权访问等级程序中的数据。例如在有操作系统的情形下，操作系统的代码运行在特权访问等级，而普通用户的代码则运行在非特权访问等级。用户程序通过操作系统提供的 API 函数访问存储器的时候，可能会因目标存储位置的访问等级问题而无法正常工作。故 ARM 中提供了一组特殊的加载和存储指令，在特权访问等级程序中使用这些指令加载或保存的数据，其权限等同于非特权访问等级的程序。例如，工作于特权等级的 OS 代码在 API 中向非特权等级的普通用户程序传递数据时，可使用这些特殊指令存取目标存储器位置，如果存取过程发生异常，则 OS 可知用户程序是无法访问这些存储器位置的，不能把这些存储器位置上存储的数据传递给普通的用户程序，需要调整。