	保存PC的方式	服务程序入口（用于更新PC实现跳转）	如何返回?	保存PSR和更新PSR条件码以外的部分?
JSR（调用用户编写的子程序）	R7=PC（指向JSR指令的下一条指令）	指令提供，有两种寻址模式（JSR为当前PC+立即数Offset得到入口地址，JSRR为从指定寄存器中得到入口地址）	RET或JMP R7	否
TRAP（系统调用）	R7=PC（指向TRAP指令的下一条指令）	零扩充TRAP指令中提供的8位TRAP矢量到16位得到矢量表地址，查位于0x00xx的矢量表得到服务程序入口地址	RET或JMP R7	否
硬件I/O中断（键盘和显示器）	PC-1（指向被打断而未能执行的指令）压入特权栈（PSR同样被压入）	经过中断优先级选择电路，把优先级最高的中断的8位中断矢量（I/O设备提供）的高位补上0x01，查位于0x01xx的矢量表得到服务程序入口地址	RTI	是（中断时用户态->特权态，特权态中发生嵌套中断则不改变特权级（保持特权态）；返回时通过从特权栈弹出PSR恢复最新一次中断前的特权级）
处理器异常（非法指令和权限模式冲突）	PC-1（指向被打断而未能执行的指令）压入特权栈（PSR同样被压入）	和硬件I/O中断相仿，只不过“中断矢量”（这里不如称异常矢量）是由处理器硬件生成的常量（权限模式冲突：x00；非法指令：x01）	RTI	是（同硬件I/O中断）

Note：

- JSR和TRAP为一类方式（调用类，意料之中、有计划的打断），硬件I/O中断和处理器异常为另一类方式（中断类，实时的不可预料的打断）
- TRAP的作用类似x86 ISA中的软中断（e.g.int 21h请求DOS系统服务），但实现方式却类似子程序调用（不过依然沿用了通过查矢量表获得服务程序入口的方式，一个8位的矢量可以看成一个系统调用的服务号）
- 中断类方式都具有优先级的特征，高优先级能中断掉低优先级（将请求中断的设备的优先级相互比较，再与PSR中的优先级字段比较），反之不行；中断类方式被中断后会更新PSR中的优先级为成功中断者的优先级
- 外部I/O中断的优先级和中断矢量都由外部I/O设备提供，或存在中断控制器对这些中断信息进行二次管理
- PSR在物理上被分割成特权位单bit寄存器、优先级寄存器和条件码寄存器三个部分，其数据在总线上直接合成为16位以方便入栈和出栈；没有任何直接读取和修改PSR的办法
- 目前已知唯一的从用户态进入特权态的办法就是触发中断或异常

- 用户态下禁止RTI，否则触发权限模式冲突（目前已知的唯一一种触发该异常的方式）；特权态（一定是被中断了才有可能进入特权态）下RTI可能返回用户态，也可能返回特权态（嵌套中断的最后一次返回会回到用户态，否则回到特权态）
- 用户态栈指针和特权态栈指针公用一个寄存器R6，特权级切换时，处理器自动完成“热替换”的操作，Saved.USP和Saved.SSP这两个物理寄存器用于存放不活动（未装载进R6）的副本；使用R6指向的栈空间需要自己完成，LC-3没有设计现成的PUSH和POP指令
- 疑惑：系统刚上电，操作系统还没有开始Booting的时候机器是处于用户态还是特权态？特权态的特权指令只有RTI一种且没有硬件寄存器操作和操作系统内存区域读写保护机制为什么还要设计特权态呢？