●外
扩的程序存储器和外扩 I/O 端口地址重叠，80C51 如何区分这些重叠地址？
80C51 单片机对片外数据存储器、片内数据存储器及程序存储器采用不同的指令，会产生不同的控制信号。片外数据存储器有读 RD 和写 WR 控制信号，程序存储器有读 PSEN 控制信号，因此，扩展时虽然数据线和地址线重复，但由不同的控制信号加以区别。片内数据存储器地址采用 MOVC 指令，不会产生 RD 和写 WR 控制信号。

●INT0 的中断响应过程？ 中断采样：中断采样是针对外部中断请求信号进行的，可以直接置位 TCON 或 SCON 中的中断请求标志；2、中断查询：若查询到某中断标志为 1，则按优先级的高低进行处理，即响应中断；3、中断响应：首先将当前程序计数器 PC 的内容压入堆栈进行保护，再将对应中断源的中断矢量地址装入 PC，执行中断服务程序。运行直到遇到 RETI 指令为止，最后恢复原程序的断点地址执行，且恢复中断触发器原先状态。

1、**RISC 的特点？** 指令集：减少了指令种类，每条指令的长度是固定的，指令格式和寻址模式相当少，一个周期就可以执行一条指令；2、流水线：指令的处理过程被拆分成几个更小的。能够被流水线并行执行的单元，允许流水线在当前指令译码器阶段去取其下一条指令；3、寄存器：RISC 拥有更多的通用寄存器，每个寄存器都可以存放数据或地址，寄存器可为所有的数据操作提供快速的局部储存访问；4、Load-Store 结构：处理器只处理寄存器中的数据。独立的 load 和 store 指令用来完成数据在寄存器和外部存储器之间的传送（5、充分利用 VLSI 芯片的面积；6、提高计算机运行速度；7、便于设计，可靠性高；8、有效支持高级语言程序）

●CPSR 与 SPSR 的关系？ 在处理器所有的运行模式下均可以访问当前的程序状态寄存器（CPSR），每一种异常模式下又都有一个专用的程序状态保存寄存器（SPSR），当异常发生时，SPSR 用于保存当前程序状态寄存器 CPSR 的状态，以便从异常退出时，由 SPSR 来恢复 CPSR，从而进行异常处理。

●大端、小端模式？ ARM 体系结构可以用两种方法存储字数据，称之为大端格式和小端格式。大端格式：字数据的高字节存储在低地址中，而字数据的低字节则存放在高地址中。小端格式：与大端储存格式相反，在小端储存格式中，低地址中存放的是字数据的低字节，高地址存放的是字数据的高字节。

●程序存储器和数据存储器地址冲突，如何区分？
不发生数据冲突的原因是：MCS-51 中访问程序存储器和数据存储器指令不一样；程序存储器访问指令为 MOVC；数据存储器访问指令为 MOVX；选通信号不同，前者为/PSEN,后者为/WR 与/RD。

●中断响应的现场保护？ 所谓现场是指中断发生时单片微机中存储单元、寄存器、特殊功能寄存器中的数据或标志位等。因此，在编写中断服务程序时必须考虑保护现场的功能。在 80C51 单片微机中，现场一般包括累加器 A\工作寄存器 R0~R7,以及程序状态字 PSW 等。保护的方法与子程序相同，可以有以下几种：1.通过堆栈操作指令 PUSH direct。2.通过工作寄存器区的切换。3.通过单片微机内部存储器单元暂存。

●ARM 处理器异常中断响应过程？ 当异常中断发生时，处理器挂起正常模式的执行，首先自动保存当前状态，即返回地址存入链接寄存器 R14，当前程序状态寄存器 CPSR 存入 SPSR 中，然后进入相应的工作模式，把程序寄存器 PC 设置为一个特定的存储器地址，这一地址放在一个被称为中断向量表的特定的地址范围内，中断向量表的入口是一些跳转指令，跳转到专门处理某个异常或中断的子程序。

●ARM 的通用寄存器（存储器），R12、R13、R14？ 在 ARM 状态下，通用寄存器包括 R0~R15.这些寄存器又可以分为以下三类：1.未分组寄存器 R0~R7。2.分组寄存器 R8~R14。R12 为写入寄存器，用作子程序间的中间结果寄存器，记录着 IP。R13 通常在 ARM 指令中用作堆栈指针，简称 SP。R14 用作子程序返回指针寄存器，称为链接寄存器（LR）：可以保存每一种运行模式下子程序的返回地址 3.程序寄存器 R15。

●ARM 的运行模式及特点？ 用户（usr）:正常程序执行模式。系统（sys）:运行操作系统的特权任务。3.快中断（FIq）:支持高速数据传输及通道处理。中断（irq）:用于通用的中断处理。管理（svc）:操作系统保护模式。中止（abt）:用于支持虚拟内存和/或存储器保护。未定义（und）:支持硬件协处理器的软件仿真。

●中断的执行与子程序相似点？
1.中断当前正在执行的程序。2.硬件把断点压栈，软件现场保护。3.通过软件恢复现场，重新返回到断点处，继续执行主程序。4.二者可嵌套，如中断嵌套和子程序嵌套。

●中断执行与子程序调用的差别？
1.中断请求由外设发出，是随机的；子程序调用是编排好的。2.中断响应后由固定的矢量地址转入中断服务程序，而子程序地址由软件设定。3.中断响应是受控的，其响应时间会受一些因素影响；子程序响应时间是固定的。

●程序存储器的特点？
1、存放经调试正确的应用程序和表格之类的固定常数；
2、采用 16 位的程序计数器 PC 和 16 位的地址总线，可扩展的地址空间为 64 KB；
3、64KB 地址是空间连续且统一的。

●●●●●●●单片微型计算机原理与接口技术●●●●●●●

●2.3.1 中央控制器是识别指令，并根据指令性质控制计算机各组成部件进行工作的部件，与运算器一起构成中央处理器
功能：控制指令的读出、译码和执行，对指令的执行过程进行定时控制，并根据执行结果决定是否分支转移。
组成：程序计数器 PC、数据指针 DPTR、指令寄存器 IR、指令译码器、条件转移逻辑电路及定时控制逻辑电路。

●程序计数器 PC：一个独立的计数器，是中央控制器中最基本的寄存器。
内容：指令地址。工作过程：PC 变化的轨迹决定程序的流程。PC 的宽度决定了程序存储器可直接寻址的范围。
顺序指令
条件转移指令或无条件转移指令
调用指令或响应中断
数据指针 DPTR
16 位特殊功能寄存器.
主要功能:用作片外数据存储器或 I/O 寻址用的地址寄存器
访问片外数据存储器或 I/O 的指令为：

```
MOVX  A，@DPTR     读
MOVX  @DPTR，A     写
```

既可以作为一个 16 位寄存器处理，或者两个 8 位寄存器，其高 8 位用 DPH 表示，低 8 位用 DPL 表示。

●PC 与 DPTR 不同之处：
(1) 与地址有关的 16 位寄存器
PC：程序存储器的地址； DPTR：数据存储器或 I/O 的地址。
PC 的输出与 ALE 及 PSEN 有关；DPTR 输出与 ALE、WR、RD 信号有关。
(2) PC 为 16 位寄存器，不可访问。DPTR 为 16 位寄存器，也可作两个 8 位特殊功能寄存器，可访问。

●3. 指令寄存器 IR、指令译码器及控制逻辑
IR:存放指令操作码的专用寄存器。
指令译码器对指令进行译码，译码结果送定时控制逻辑电路。

●2.3.2 运算器
组成:算术逻辑运算单元 ALU、累加器 A、暂存寄存器、B 寄存器、程序状态标志寄存器 PSW 以及 BCD 码运算修正电路等
1.算术逻辑运算单元 ALU
ALU 有两个输入：
1)暂存器 1 的输入：
2)暂存器 2 或累加器 A 的输入
ALU 有两个输出：

1）通过内部总线送回
累加器 A；
2) 标志位输出至程序状态
字 PSW。

2. 累加器 A
主要功能：存放操作数，暂存运算结果。
单片机中大部分数据操作都要通过累加器 A 进行，容易产生“瓶颈”现象。

3. B 寄存器
乘/除法指令中用作 ALU 的一个输入。
乘法的两个输入为 A、B，运算结果，A 中放积的低 8 位，B 中放积的高 8 位。
除法中，被除数取自 A，除数取自 B，商数存放于 A，余数存放于 B

● 4. 程序状态字 PSW （Program Status Word）
8 位寄存器，内容是算术逻辑运算单元（ALU）的输出
（1）P—奇偶标志位
表示累加器 A 中值为 1 的个数的奇偶性：若累加器值为 1 的位数是奇数,P 置位(奇校验);否则 P 清除(偶校验) 。
如(A)=00001010，则 P=0。
在串行通信中，常以传送奇偶校验位来检验传输数据的可靠性。
（2）OV —溢出标志位
指示运算结果是否溢出。
OV=1：运算结果超出了寄存器 A 所能表示的带符号数的范围（一 128～＋127）。
（3）RS1、RS0 —工作寄存器组选择位
（4）AC —辅助进位标志位
加减法运算时，低 4 位向高 4 位数进位或借位时，AC 将被硬件置位；否则，被清除。
在十进制调整指令 DA 中要用到 AC 标志位状态。
（5） CY —进位标志位。
在进行算术运算时，表示运算结果中高位是否有进位或借位，可以被硬件置位或清除。
（6）F0 —用户标志位。
开机时该位为“0”。
用户可根据需要，通过位操作指令置“1”或者清“0”。

●时序定时单位：节拍、状态、机器周期和指令周期。
(1)节拍 P： 振荡脉冲的周期称为节拍。
(2) 状态 S ：一个状态 S 包含两个节拍。
(3)机器周期：宽度为 6 个状态，并依次表示为 S1～S6。一个机器周期有 12 个振荡脉冲周期。是单片机的时间单位。
(4)指令周期： 执行一条指令所需要的时间。是最大的时序定时单位。80C51 的指令周期有 1、2、4 个机器周期。

●单片机存储器的两种基本结构：
1. 普林斯顿(Princeton)结构：程序和数据合用一个存储器空间
2. 哈佛（Harvard）结构：程序存储器和数据存储器截然分开，分别寻址的结构。
4 个物理存储器空间
•程序存储器：①片内程序存储器； ②片外程序存储器。
•数据存储器：③片内数据存储器； ④片外数据存储器。
3 个逻辑存储器地址空间
①片内、片外统一的 64 KB 程序存储器地址空间；
②片内 256B 数据存储器地址空间；
③片外 64 KB 的数据存储器地址空间。
三种基本寻址空间：
• 64 KB 的片内、外程序存储器寻址空间；
• 64 KB 的片外数据存储器寻址空间；
• 256B 的片内数据存储器寻址空间，包括 SFR 寻址空间。

●2.4.1 程序存储器
功能：存放程序和固定常数。
PC 和地址总线为 16 位 ，可扩展的地址空间为 64 KB。
1.片内和片外程序存储器
EA 引脚高电平，从片内程序存储器 0000H 开始执行；
当 PC 值超出 4K，自动转向片外程序存储器空间执行。
EA 引脚低电平，从片外程序存储器 0000H 开始执行。
2. 程序入口地址（中断）
系统复位后 PC 为 0000H，系统从 0000H 单元开始取指，执行程序。
0003H～002DH 用于 5 个中断源的中断服务程序入口地址。
P0 口：多功能 8 位口，字节访问地址：80H，位访问地址：80H～87H。
2. P0 口的功能
(1) I／O 口 ：输出锁存、输入缓冲，输入时需先将口置 1；每根口线可以独立定义为输入或输出。
(2) 地址／数据复用总线：作数据总线用时，输入／输出 8 位数据 D0～D7;作地址总线用时,输出低 8 位地址 A0～A7。
P2 口的工作状态是输出高 8 位地址。
P2 口的功能：
(1) I／O 口
(2) 程序存储器或片外数据存储器的高 8 位地址

●80C51 单片机的工作方式
复位：单片机的初始化操作，把 PC 初始化为 0000H，单片机从 0000H 单元开始执行。
(1) 复位信号
RST 引脚： 高电平有效，有效时间应持续 2 个机器周期以上
(2)复位方式： 上电自动复位、按键电平复位和外部脉冲复位

程序执行
单片机的基本工作方式。
复位后 PC＝0000H，程序执行从 0000H 开始。
一般在 0000H 开始的单元中存放一条无条件转移指令，跳转到实际主程序入口去执行。
低功耗
两种低功耗方式：待机方式和掉电保护方式。
1. 待机方式
(1) IDL=1，80C51 进入待机方式
振荡器仍运行，并向中断逻辑、串行口和定时器/计数器电路提供时钟，中断功能继续存在 。
向 CPU 提供时钟的电路被阻断，CPU 不工作，SP、PC、PSW、ACC 及通用寄存器冻结在原状态。
(2) 采用中断方式或硬件复位来退出待机方式。
若产生一个外部中断请求信号，单片机响应中断，PCON.0 位（IDL 位）被硬件自动清“0”， 单片机退出待机方式进入正常工作方式。
2. 掉电保护方式
(1) PD 位控制单片机进入掉电保护方式
当 80C51 检测到电源故障时,进行信息保护,把 PCON.1 位置“1”，进入掉电保护方式。
单片机一切工作停止，只有内部 RAM 单元内容被保护。
(2) 依靠复位退出掉电保护方式
当 Vcc 恢复正常后，只要硬件复位信号维持 10ms，就能使单片机退出掉电保护方式，CPU 则从进入待机方式的下一条指令开始重新执行程序。
编程和校验。
3.1.1 寻址方式
寻址方式：指令中给出的寻找操作数或操作数地址的方法。
1. 立即寻址
在指令中直接给出操作数，出现在指令中的操作数称为立即数。
立即数前面必需加上前缀“＃”。
如：指令 MOV DPTR，＃1234H
2. 直接寻址
在指令中直接给出操作数的地址
例：指令 MOV A，3AH
直接寻址是访问特殊功能寄存器的唯一方法
3.寄存器内容是操作数
例如：指令 INC R0
4. 寄存器间接寻址
寄存器内容是操作数地址。示形式：@寄存器符号
例如：指令 ANL A，@R1
5. 相对寻址
指令给出的操作数为程序转移的偏移量。
在相对转移指令中，给出地址偏移量。
目的地址＝(转移指令所在地址＋转移指令字节数)＋rel
如：指令 JC 80H
6. 变址寻址
以 DPTR 或 PC 为基址寄存器，累加器 A 为变址寄存器，以两者内容相加后形成的 16 位程序存储器地址作为操作数地址。
称基址寄存器＋变址寄存器间接寻址。
例如： MOVC A，@A＋DPTR
7. 位寻址
寻址范围：
1）片内 RAM 位寻址区：
如 MOV C，2BH
●数据传送类指令
指令助记符：MOV、MOVX、MOVC、XCH、XCHD、SWAP、PUSH、POP
1）A 内容送外部数据存储器或 I/O
MOVX @Ri，A
MOVX @DPTR，A
2）存储器数据传送指令(或查表指令)
MOVC A，@A＋PC
MOVC A，@A+DPTR
3）交换指令 XCH 组
XCH A，Rn ；(A)(Rn) 、direct、@Ri
将累加器 A 与源操作数的字节内容互换。
例：设（R0）＝30H，(A)＝3FH，片内（30H）＝BBH。
执行指令 XCH A，@R0
4）节交换指令组
(1) XCHD A，@Ri
Ri 间接寻址单元的低 4 位与累加器 A 的低 4 位互换，而高 4 位不变。
例：设(R0)＝20H， (A)＝36H(00110110B)， (20H)＝75H(01110101B)。
执行指令：XCHD A，@R0
(2) SWAP A
将累加器 A 的高、低半字节交换
例：设（A）＝36H（0011 0110B）
执行指令： SWAP A
XCHD 和 SWAP 主要用于十六进制数或 BCD 码的数位交换。
5）操作指令组
PUSH direct
POP direct
入栈指令：(SP) (SP)+1,(SP) (direct)
出栈指令： (direct) (SP)， (SP) (SP-1)
例：中断响应时(SP)＝30H，DPTR 的内容为 0123H，执行入栈指令：

```
PUSH    DPL          ；DPL 内容入栈
PUSH    DPH          ；DPH 内容入栈。
```

●算术运算类指令
助记符 8 种：ADD、ADDC、INC、DA、SUBB、DEC、MUL、DIV
●逻辑运算类指令
助记符 9 种：ANL、ORL、XRL、RL、RLC、RR、RRC、CPL、CLR。
逻辑“与”助记符为 **ANL**，用符号“∧”表示；
逻辑“或”助记符为 **ORL**，用符号“∨”表示；
逻辑“异或”助记符为 **XRL**，用符号“⊕”表示；
循环右移指令：**RR** A（累加器内容逐位循环右移一位，a0 移到 a7，不影响标志位）
带进位循环右移指令：**RRC** A（将累加器内容和进位位一起循环右移，a0 移入 CY，CY 移到 a7）
循环左移指令：**RL** A（累加器的内容逐位循环左移一位，a7 移到 a0。不影响标志位）
带进位循环左移指令：**RLC** A（累加器的内容和进位位一起循环左移一位，a7 移入进位位 **CY**，CY 的内容移到 a0）
累加器按位取反指令：**CPL** A（累加器的内容逐位取反，结果仍存在 A 中。不影响标志位）
累加器清 0 指令：**CLR** A（对累加器进行清 0，不影响标志位）
●控制程序转移类指令
助记符 12 种：AJMP、LJMP、SJMP、JZ、JNZ、CJNE、DJNZ、ACALL、LCALL、RET、RETI、NOP
短转移指令：**SJMP** rel （目标地址是由当前 PC 值和（8 位带符号）相对地址 rel 相加）
绝对转移指令：**AJMP** addr11（目标地址由指令第 1 字节的高 3 位 a10～a8 和指令第 2 字节的 a7～a0 所组成。以 11 位地址取代当前 PC 低 11 位，形成新的 PC 值。）
长转移指令：**LJMP** addr16（目标地址由指令第 2 字节和第 3 字节组成。目标地址为 64KB 空间）
间接转移指令：**JMP** @A+DPTR（目标地址是累加器 A 中的 8 位无符号数与数据指针 DPTR 的内容相加）
累加器判零转移指令：**JZ** rel ； （ 若 (A)=0 ， 则(PC)=(PC+2)+rel；若(A)≠0，则(PC)=(PC)+2）
JNZ rel;(若(A)≠0,则(PC)=(PC+2)+rel
若(A)=0，则(PC)=(PC)+2）
数值比较转移指令：**CJNE** A，direct，rel（指令格式为： CJNE（操作数 1），（操作数 2），rel
数值比较指令的第 1 字节为操作码(或操作码+操作数 1），第 2 字节为操作数 2，第 3 字节为偏移量 rel。具有比较转移和数值大小比较的功能。）
循环转移指令：**DJNZ** Rn，rel
DJNZ direct，rel
每执行一次本指令，先将操作数减 1，判别是否为 0。
◇不为 0，转向目标地址；
◇为 0，则结束循环程序，程序往下执行。
绝对调用指令：**ACALL** addr11（无条件地调用首址为 addr11 处的子程序。操作不影响标志位）
长调用指令：**LCALL** addr16（无条件地调用首址为 addr16 处的子程序。操作不影响标志位）
子程序返回指令：**RET**（表示结束子程序，返回 ACALL 或 LCALL 的下一条指令(即断点地址)，继续往下执行）
中断返回指令：**RETI**（中断服务程序返回，从断点处继续执行，清除内部相应的中断状态寄存器。中断服务程序必须以 RETI 为结束指令）
中断源：能产生中断的外部和内部事件。
80C51 有 5 个中断源：
◆ 两个外部中断源 INT0 和 INT1
◆ 三个内部中断源
1. 外部中断
两种信号触发方式：
◆电平有效方式：若引脚上采样到有效的低电平，则向 CPU 提出中断请求；
◆跳变有效方式：若引脚上采样到有效负跳变，则向 CPU 提出中断请求。
(1)INT0：外部中断 0。
当 IT0＝0 时，低电平有效；
当 IT0=1 时，下降沿有效。
(2)INT1：外部中断 1。
当 IT1＝0 时，低电平有效；
当 IT1=1 时，下降沿有效。
2. 定时中断
当计数器发生计数溢出时，表明设定的定时时间到或计数值已满，这时可以向 CPU 申请中断。
80C51 有两个源，即：
(1) TF0：T0 溢出中断。
(2) TF1：T1 溢出中断。
3. 串行中断
每当串行口发送或接收一帧串行数据时，就产生一个中断请求。
RXD，TXD：串行中断。
●中断矢量：当 CPU 响应中断时，由硬件产生一个固定的地址，即矢量地址，由矢量地址指出每个中断源的中断服务程序的入口。
●中断响应时间：在单级中断系统中，中断响应时间：3～8 个机器周期。
中断请求标志位查询占 1 个机器周期，而且是指令的最后一个机器周期，在这个机器周期结束后，CPU 即响应中断，产生硬件长调用 LCALL 指令，执行这条长调用指令需要 2 个机器周期，中断响应时间为 3 个机器周期。
●外部中断源的扩展：80C51 只有两个外部中断请求输入端 INT0 和 INT1，可以通过两种方式进行扩展：
1) 可通过增加“OC 门”＋软件来扩展；
2) 定时器/计数器
●●●●●●●●●●●●嵌入式设计及应用●●●●●●●●●●
●RISC 的设计准则：
1.指令集：选用使用频率较高的一些 简单指令 复杂指令的功能由简单指令来组合一个时钟周期 内完成一条指令
2.流水线：指令的处理过程被拆分成更小的、能够被流水线执行的单元。
3.寄存器：拥有多个通用寄存器
4.LOAD/STORE 结构:实现寄存器与外存之间的数据传送。
●RISC 架构的优点：
1.充分利用 VLSI 芯片的面积
2.提高计算机运行速度：指令：通用寄存器：流水线
3.便于设计、可靠性高
4.有效支持高级语言程序
●ARM 内核 RISC 架构的特别技术：
1.所有的指令可根据前面的执行结果决定是否被执行
2.可用加载/存储指令批量传输数据
3.可在一条数据处理指令中同时完成逻辑处理和移位处理
4.在循环处理中使用地址的自动增减来提高运行效率
●ARM 的流水线技术：
指令的二级流水线结构：若取指和执行阶段时间上完全重叠，指令周期减半，速度提高 1 倍
ARM7:三级流水（预取，译码，执行）
ARM9:五级流水（预取，译码，执行，访存，写回）
ARM10:六级流水（预取，发射，译码，执行，访存，写回）
ARM11:八级流水（预取，预取，发射，译码，转换，执行，访存，写回）
●ARM 处理器支持两种指令集
32 位的 ARM 指令集：处理 32 位数据时性能较高
16 位的 Thumb 指令集：ARM 指令集的功能子集。
特点：密度高，所占存储空间较小；可以使用 32 位的寻址空间、32 位寄存器、32 位位移与逻辑运算、32 位的内存存取访问功能。
●从编程角度看，ARM 微处理器的工作状态有两种：
ARM 状态，此时处理器执行 32 位的字对齐的 ARM 指令；
Thumb 状态，此时处理器执行 16 位的、半字对齐的 Thumb 指令。
可以相互调用，只要遵循一定的调用规则

●Thumb 指令与 ARM 指令的时间效率和空间效率关系为:
存储空间约为 ARM 代码的 60%～70%
存储器为 32 位时 ARM 代码比 Thumb 代码快约 40%
存储器为 16 位时 Thumb 比 ARM 代码快约 40～50%
使用 Thumb 代码，存储器的功耗会降低约 30%
●状态切换方法
ARM 指令集和 Thumb 指令集均有切换处理器状态的指令，并可在两种工作状态之间切换，切换不影响处理器的运行模式和寄存器内容
在开始执行代码时，应该处于 ARM 状态。
●状态寄存器:
当前程序状态寄存器 CPSR (Current Program Status Register)，可以在任何工作模式下被访问；程序状态备份寄存器 SPSR (Saved Program Status Register)，只有在异常模式下，才能被访问；(各种异常模式专有的)当异常发生时，SPSR 用于保存当前程序状态寄存器 CPSR 的状态，从异常退出时，用于恢复 CPSR。
功能包括：
保存 ALU 中的当前操作信息
控制允许和禁止中断
设置处理器的运行模式。
●ARM 状态下寄存器的子集。Thumb 状态下寄存器和 ARM 状态下的寄存器组的关系：
R0~R7 是相同的
CPSR 和所有的 SPSP 是相同的
Thumb 状态下的 SP 对应于 ARM 状态下的 R13
Thumb 状态下的 LP 对应于 ARM 状态下的 R14
Thumb 状态下的程序计数器对应于 ARM 状态下的 R15
●S3C2410A 的 DMA 控制器
采用 DMA 方式进行数据传输的具体过程如下：
1. 外设向 DMA 控制器发出 DMA 请求。
2. DMA 控制器向 CPU 发出总线请求信号
3. CPU 执行完总线周期，向 DMA 控制器发出相应的回答信号；
4. CPU 将控制总线、地址总线及数据总线交给 DMA 控制器控制；
5. DMA 控制器向外部设备发出 DMA 请求回答信号；
6. 在 DMA 控制器控制下进行 DMA 传送；
7. 数据传送完，DMA 控制器通过中断请求线发出中断信号。CPU 接收到中断信号后，转入中断处理程序进行处理；
8. 中断处理结束后，CPU 返回断点继续执行，并获得总线控制权
●ARM 指令系统及编程技术
ARM 指令的一般格式：
<opcode>{<cond>}{S}　<Rd>,<Rn>{,<operand2 >}
　其中< >内的项是必须的，｛ ｝内的项是可选的。
opcode：指令助记符
cond：执行条件
S：是否影响 CPSR 寄存器的值
Rd：目标寄存器
Rn：第 1 个操作数的寄存器
operand2：第 2 个操作数
●ARM 指令的条件域<cond>

cond	CPSR 中标志位	含义
EQ	Z 置位	相等
NE	Z 清零	不想等
CS	C 置位	无符号数大于或等于
CC	C 清零	无符号数等于
MI	N 置位	负数
PL	N 清零	正数或零
VS	V 置位	溢出
VC	V 清零	未溢出
HI	C 置位 Z 清零	无符号数大于
LS	C 清零 Z 置位	无符号数小于或等于
GE	N 等于 V	带符号数大于或等于
LT	N 不等于 V	带符号数小于
GT	Z 清零且 N 等于 V	带符号数大于
LE	Z 置位或 N 不等于 V	带符号数小于或等于
AL	忽略	无条件执行

●寻址方式:
1. 寄存器寻址
2. 立即数寻址
3. 寄存器移位寻址
　5 种移位操作：
（1）LSL 逻辑左移；　（2）LSR 逻辑右移
（3）ASR 算术右移；　（4）ROR 循环右移；
（5）RRX 带扩展的循环右移。
例：　MOV R0, R1, LSL #5;　　R0=R1 逻辑左移 5 位
4. 寄存器间接寻址
5. 基址寻址
6. 多寄存器寻址
一条指令实现一组寄存器值的传送，连续的寄存器用“-”连接，否则用“,”分隔
例：LDMIA　R0, {R1-R5};
功能：R1=[R0]，R2=[R0+4]，R3=[R0+8]，R4=[R0+12]，R5=[R0+16]
从以 R0 为起始地址的存储单元中取出 5 个字的数据送到 R1 到 R5 寄存器中。
7. 堆栈寻址
8. 块拷贝寻址
9. 相对寻址
●ARM 指令分类说明
1. 数据处理指令
2. 跳转指令：
**　B　转移指令**
功能：跳转到目的地址。
跳转范围：当前指令的±32M 字节地址内（ARM 指令为字对齐，最低 2 位地址固定为 0）。
**　BL　带链接的转移指令**
功能：PC 拷贝到链接寄存器，然后跳转到指定地址。
跳转范围：当前指令的±32M 字节地址内。
**　BX　带状态切换的转移指令**
功能：跳转到目标地址；处理器工作状态切换。
目标地址：寄存器 Rn 和 0xFFFFFFFE 相与的结果。Rn 的第 0 位拷贝到 CPSR 中 T 位，位[31：1]移入 PC。
跳转范围：当前指令的±32M 字节地址内。
**　BLX　带链接和状态切换的转移指令**
功能：跳转到目标地址；PC 值保存到 LR 寄存器；处理器工作状态切换。
3. Load/Store 指令
**　A. 单寄存器的存取指令（LDR，STR）**
LDR：从内存中读取单个字或字节数据存于寄存器；
LDR　Rd，[Rn，Rm]　；将内存中的地址为 Rn+Rm 的字数据装入寄存器 Rd 中
LDR　Rd，[Rn,Rm]！；将内存中的地址为 Rn+Rm 的字数据装入寄存器 Rd 中，并将新地址 Rn+Rm 写入 Rn
LDR{cond}T <Rd>,<addr>;
T：若指令有 T，即使处理器是在特权模式下，也将存储系统访问看成是在用户模式的。
STR：将寄存器中的单个字或字节数据保存到内存。
STRH:半字数据存储指令
STR{cond}H <Rd>,<addr>;
STRT:用户模式的字数据存储指令
STR{cond}T <Rd>,<addr>;
**　B. 多寄存器存取指令（LDM，STM）**
LDM：批量数据加载指令:
LDM {cond}{<type>}　Rn{!}, reglist {^}
指令中，type 字段有以下几种：
IA 每次传送后地址加一
IB 每次传送前地址加一
DA 每次传送过后地址减一
DB 每次传送前地址减一
FD 满递减堆栈
ED 空递减堆栈
FA 满递增堆栈
EA 空递增堆栈
**　C. 单寄存器交换指令（SWP）**
**　D. SWP 字数据交换指令**
**　SWP{cond} Rd,Rm,[Rn];**
功能：从 Rn 所表示的内存装载一个字并把这个字放置到目的寄存器 Rd 中，然后把寄存器 Rm 的内容存储到同一内存地址中，即 Rd=[Rn],[Rn]=Rm,其中 Rm,Rn 均为寄存器。
例：SWP R0,R1,[R2];将 R2 所表示的内存单元中的字数据装载到 R0，然后将 R1 中的字数据保存到 R2 所表示的内存单元中。
4. 程序状态寄存器指令
MRS 指令：对状态寄存器 CPSR 和 SPSR 进行读操作。CPSR：了解当前处理器的工作状态。SPSR：了解进入异常前的处理器状态。
MRS 指令格式
MRS{cond}　Rd,CPSR/SPSR
5. 协处理器指令
6. 软件中断指令
●定时器 中断程序
这时 T0 采用方式 3 工作，其中，TL0 产生 100μs 定时，由 P1．0 输出方波 1；TH0 产生 200μs 定时，由 P1．1 输出方波 2；T1 设置为方式 2，作波特率发生器用。fOCC=9．216 MHz。

```
ORG 0000H
AJMP    MAIN
ORG     000BH ; TL0 的中断入口
AJMP  ITL0
ORG     001BH; TH0 的中断入口
AJMP ITH0
ORG 0100H
MAIN:   MOV     SP, # 60H; 设栈指针
MOV     TMOD, # 23H 设 T0 为方式 3,T1 为 2
MOV     TL0, #0B3H 设 TL0 初值(100μs 定时
MOV     TH0, #66H 设 TH0 初值(200μs 定时)
MOV     TL1, #0F6H 设 TL1 初值(波特率为 2400)
MOV     TH1, # 0F6H 设 TH1 初值
SETB    TR0 ; 启动 TL0
SETB    TR1 ; 启动 TH0
SETB    ET0 ; 允许 TL0 中断
SETB    ET1 ; 允许 TH0 中断
SETB    EA  ; CPU 中断开放
AJMP    $
ORG     0200H
ITL0: MOV   TL0, # 0B3H; 重装定时常数
CPL     P1. 0; 输出方波 1(200μs)
RETI
ITH0: MOV TH0, #66H; 重装定时常数
 CPL    P1. 1; 输出方波 2(400μs)
       RETI
例. 飞读
RDTIME:         MOV A, TH0; 读 TH0
MOV     R0, TL0 ; 读 TL0 并存入 R0
CJNE    A, TH0, RDTIME ; 再读 TH0
MOV     R1, A  ; 存 TH0 在 R1 中
RET
```

●并行输出 流水灯程序

```
MOV     SCON, #00H      ; 设串行口为方式 0
        CLR     ES                      ; 禁止串行口中断
        MOV     A, #80H         ; 先显示最左边发光二极管
LED:    MOV     SBUF, A         ; 串行输出
        JNB     TI, $                   ; 输出等待
        CLR     TI                      ;软件清中断标志
        ACALL   DELAY   ; 轮显间隔
        RR      A                       ; 发光右移
        AJMP    LED                     ; 循环
DELAY: MOV R6,#200H
        NOP
        DJNZ R6,&                       ; 延时子程序
        RET
```

将 2000H 单元开始的一批数据传送到从 3000H 开始的单元中，数据长度在内部 RAM 的 30H 中。

```
MOV  DPTR, #2000H; 源数据区首址
PUSH  DPL           ; 源数据区首址压栈保护
PUSH  DPH
MOV  DPTR, #3000H; 目的数据区首址
MOV  R6, DPL    ; 目的数据区首址存入寄存器
MOV  R7, DPH
LP:    POP  DPH; 取数据区地址指针
POP  DPL
MOVX  A, @DPTR  ; 取源数
INC  DPTR
PUSH  DPL
PUSH  DPH
MOV  DPL, R6        ; 取目的数据区地址指针
MOV  DPH, R7
MOVX  @DPTR, A     ; 存入目的数据区
INC  DPTR
MOV  R6, DPL
MOV  R7, DPH
DJNZ  30H, LP    ; 若数据块未移完，则继续
POP  DPH
POP  DPL
SJMP  $
```

●单片机与 8255A 的接口电路，PA 口作输出口，接 8 个 LED 发光二极管，PB 作输出口，接 8 个按键开关，PC 口不用，都工作在方式 0，要实现“按下任意键，对应的 LED 发光”，相应的程序如下：
读 PB 口开关状态，送 PA 口输出控制 LED，循环

```
MOV DPTR,#0FF7FH;指向 8255A 的控制口
MOV A,#82H;       工作方式控制字
MOVX @DPTR,A;向控制口写控制字，PA 输出，PB 输入
LOOP:MOV DPTR,#0FF7DH;指向 8255A 的 PB 口
MOVX A,@DPTR;读 PB 口按键状态
MOV DPTR,#0FF7CH;指向 8255A 的 PA 口
MOVX @DPTR,A;从 PA 口输出，驱动 LED 发光
SJMP LOOP
```

●试编写串行接口以工作方式 2 发送数据的中断

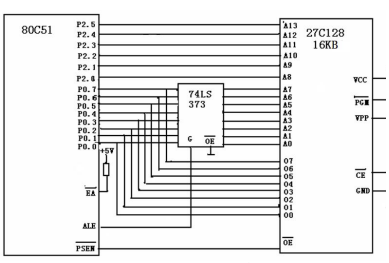
```
    ORG 0023H
        AJMP    SPINT
 SPINT: CLR     EA                      ; 关中断
        PUSH  PSW                       ; 保护现场
        PUSH  ACC
        SETB   EA                       ; 开中断
        SETB   PSW.4            ; 切换寄存器工作组
        CLR    TI                       ;
清除发送中断请求标志
        MOV    A, @R0           ; 取数据，置奇偶标志位
        MOV    C, P       ; 奇偶标志位 P 送 TB8
        MOV    TB8, C
        MOV    SBUF, A  ; 数据写入发送缓冲器，启动发送
        INC     R0              ; 数据地址指针加 1
        CLR     0AFH            ;
恢复现场
        POP     ACC
        POP     PSW
        SETB    0AFH
        CLR     PSW .4  ; 切换寄存器工作组
        RETI
```

●采用延时等待 A/D 转换结束方式，分别对 8 路模拟信号轮流采样一次，并依次把结果存入数据存储器。

```
        ORG     0000H
        SJMP  MAIN
        ORG     0030H
MAIN: MOV  R1, #20H
        MOV       DPTR, #7FF8H ; 指向通道 0 地址
        MOV  R7, #08H  ; 共需转换 8 个通道
LOOP: MOVX  @DPTR, A; 启动 A/D 转换①
        LCALL   D128μs  ; 延时等待 A/D 转换结束②
        MOVX   A, @DPTR  读入 A/D 转换值③
        MOV     @R1, A
        INC     DPTR ; 指向下一通道地址
        INC     R1
        DJNZ    R7, LOOP ; 8 个通道未转换完则继续
D128μs:         …; 延时 128μs 子程序
        RET
```

●中断方式 AD 采样 采集 8 路模拟量，并存入 20H 地址开始的内部 RAM 中

```
ORG     0000H
SJMP    MAIN
ORG   0003H  ; 外部中断 0 入口地址
LJMP    INTDATA
 ORG    0100H ; 数据采集程序
MAIN: MOV R0, #20H ; 数据缓冲区首址
MOV     R2, #8  ; 8 通道计数器
MOV     DPTR, #7FF8H ; 指向 0 通道
START: CLR      F0  ; 清中断发生标志
MOVX    @DPTR, A; 启动 A/D (P2.7=0, /WR=0) ①
SETB    IT0     ; 置外部中断 0 为边沿触发
SETB    EX0  ; 允许外部中断 0
SETB    EA ; 开中断
LOOP: JNB       F0, LOOP        ; 中断发生标志是否为 0 ②
DJNZ    R2, START ; 8 个通道转换是否结束
SJMP    MAIN
INTDATA: MOVX  A, @DPTR ; 读数据 (P2.7=0, RD=0),; 硬件撤销中断③
MOV     @R0, A  ; 存数据
INC     R0
INC      DPTR  ; 指向下一通道
SETB    F0  ; 置中断发生标志
RETI
```



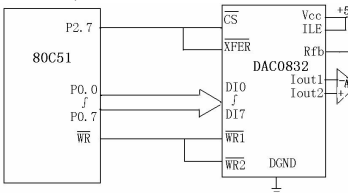
8255A 与 80c51 的接口电路图

例：对 8255A 各口作如下设置：A 口方式 0，B 口方式 0，从 A 口输入，从 B 口、C 口输出。
工作方式控制字为 10010000，即 90H。
初始化程序段：

```
MOV  A, #90H   ; 设 A 口、B 口为方式 0 ; A 口输入，B 口、 C 口输出
MOV     DPTR, #7FFFH
MOVX    @DPTR, A
MOV             DPTR, #7FFCH   ; 从 A 口输入
MOVX    A, @DPTR
MOV     DPTR, #7FFDH   ; 从 B 口输出
MOVX    @DPTR, A
MOV     DPTR, #7FFEH   ; 从 C 口输出
MOVX    @DPTR, A
```

例:把 C 口的第 5 位 pc 端置为 1

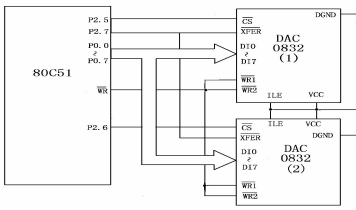
```
MOV    DPTR,#7FFFH
MOV    A,#00001011      ; PC5 置位
MOVX   @DPTR,A
```



例：产生锯齿波，DAC0832 的地址为 7FFFH(P2.7=0) 。

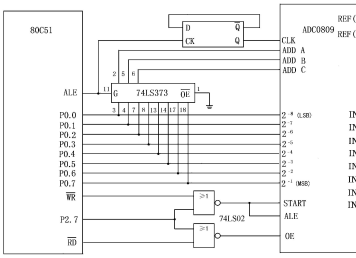
```
ORG          0000H
SJMP    MAIN
ORG     0030H
MAIN: MOV  DPTR, #7FFFH        ; DAC 寄存器地址

MOV     R0, #0          ; 转换初值
LP: MOV A, R0
MOVX    @DPTR, A        ; 送出模拟量
LCALL   DELAY
INC     R0
SJMP    LP
DELAY: ...
RET
```



双缓冲

```
MOV  DPTR, #0DFFFH; 把数据送第一片 DAC0832    输入锁存器
MOV             A, R0
MOVX    @DPTR, A
MOV     DPTR, #0BFFFH; 把数据送第二片 DAC0832 的输入锁存器
MOV     A, R1
MOVX    @DPTR, A
MOV     DPTR, #7FFFH; 两片 0832 同时输出模拟量
MOVX    @DPTR, A
```



ADC0809 与 80c51

例：采用延时等待 A/D 转换结束方式，分别对 8 路模拟信号轮流采样一次，并依次把结果存入数据存储器。

```
ORG          0000H
SJMP  MAIN
ORG          0030H
MAIN: MOV  R1, #20H
MOV          DPTR, #7FF8H ; 指向通道 0 地址
MOV  R7, #08H                   ; 共需转换 8 个通道
LOOP: MOVX  @DPTR, A; 启动 A/D 转换①
LCALL   D128μs          ; 延时等待 A/D 转换结束②
MOVX    A, @DPTR    ; 读入 A/D 转换值③
MOV     @R1, A
INC     DPTR          ; 指向下一通道地址
INC     R1
DJNZ    R7, LOOP ; 8 个通道未转换完，则继续。
D128μs:         …                       ; 延时 128μs 子程序
RET
```

例：采集 8 路模拟量，并存入 20H 地址开始的内部 RAM 中。

```
ORG     0000H
SJMP    MAIN
ORG     0003H                   ; 外部中断 0 入口地址
LJMP    INTDATA
ORG     0100H           ; 数据采集程序
MAIN: MOV       R0, #20H        ; 数据缓冲区首址
MOV     R2, #8          ; 8 通道计数器
MOV     DPTR, #7FF8H ; 指向 0 通道
START: CLR      F0              ; 清中断发生标志
MOVX    @DPTR, A        ; 启动 A/D (P2.7=0, /WR=0) ①
SETB            IT0                     ; 置外部中断 0 为边沿触发
SETB            EX0             ; 允许外部中断 0
 SETB           EA              ; 开中断
LOOP: JNB       F0, LOOP        ; 中断发生标志是否为 0 ②
DJNZ    R2, START       ; 8 个通道转换是否结束
SJMP            MAIN
INTDATA: MOVX A, @DPTR          ; 读数据 (P2.7=0, RD=0)，硬件撤销中断③
 MOV    @R0, A          ; 存数据
 INC    R0
 INC     DPTR                   ; 指向下一通道
SETB    F0              ; 置中断发生标志
 RETI
```