微机原理

本文档采用标准markdown语法编写，推荐使用Marktext、Typedown等软件进行阅读，效果更佳✨✨

使用阅读器打开本文档时，务必将设备连接互联网，否则无法看到文档中的图片

部分考题涉及少的部分，本文少涉及

微处理器

第一颗微处理器：1971：Intel 4004

集成度高，能处理数据，执行指令

运算器，寄存器组，控制器，内部总线

按照位数分类：4位，8位，32位，……

按照应用领域分类：通用高性能微处理器，嵌入式微处理器，数字信号处理器和微控制器

工作原理

取指令，指令译码，取操作数，执行运算，回送结果

周期

时钟周期：晶振或外部CLK的周期

状态周期：工作基本节拍，一般2个时钟周期

机器周期：完成一个基本动作，12个时钟周期

指令周期：完成一条指令所需周期，一般1-4个机器周期

（IO极限翻转速度=机器周期/12 ?）

1T，6T和12T的区别（拓展）：1T为一个机器周期为一个时钟周期，6T和12T同理

单片机

重要指标

位数：4，8，32……

存储器（程序+数据，不同类型（如EEPROM），编程方式（IAP，ISP））

IO口（用户可自定义）

速度（CPU的运行速度，以每秒执行多少条指令来衡量）

工作电压（5v，3.3v，1.5v……）

功耗（采用等待，关断，睡眠等模式降低功耗）

工作温度：分为民用/商业（0-70℃），工业（-40-85℃），军用（-55-150℃）

附加功能（内置ADC，DAC，LCD驱动，串口……用于减少外围器件，提高可靠性）

分类

制造工艺（HMOS，CHMOS）

字长：4，8，32……

片内存储器类型（无ROM，（不）可擦除ROM型，可擦除EPROM，EEPROM，Flash型）

系统结构：**冯·诺依曼**：程序、数据公用一个存储器，MCS-96，**哈佛**：程序、数据分开存储，大部分单片机采用

应用场合：通用性，专用性

MCS-51：书P7-8表格，51-52的区别✨52的内部RAM为51的2倍（256B），比51多一个T2和一个中断源

Atmel的51：都是Flash型，烧录次数1000+，带s表示支持ISP

STC-51：增强型51，可选6T和12T，时钟频率0-35MHz，实际工频可达48MHz，片内2k-12k字节用户应用程序空间，Flash，擦写10w次+，集成512字节RAM，通用IO口（驱动20mA max），支持ISP/IAP，直接串口下载（P3^0和P3^1），内置EEPROM，内置看门狗，0-75℃/-40-85℃，提供多种封装（PDIP，SOP，PLCC）

产品信息：书P9

两种封装：建议记忆对应的VCC、GND、RST、RX/TX、XTAL引脚（小题），PDIP：VCC-40，GND-20，RST-9，RX-10，TX-11，XTAL1、2-19、18；PLCC：VCC-44，GND-22，RST-10，RX-11，TX-13，XTAL1、2-21、20，另外12、23、34、1为空脚（凑数用的/汪汪）

电平：只有两种电平（书上此处未标明P0可以高阻态），为TTL电平

开发流程

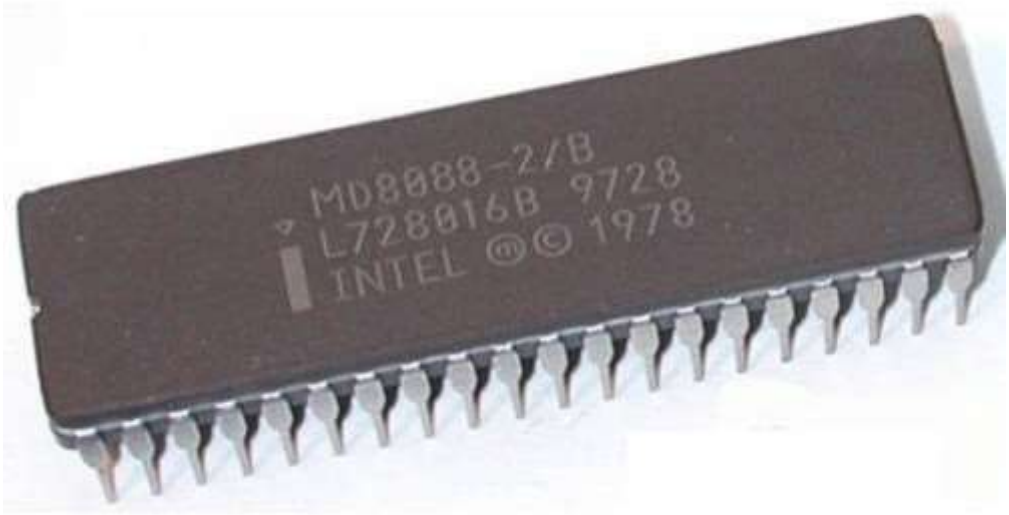
书P11，1-7图

程序开发流程：C→obj→hex→软件仿真/在线仿真/烧录后插入电路板，完成系统设计

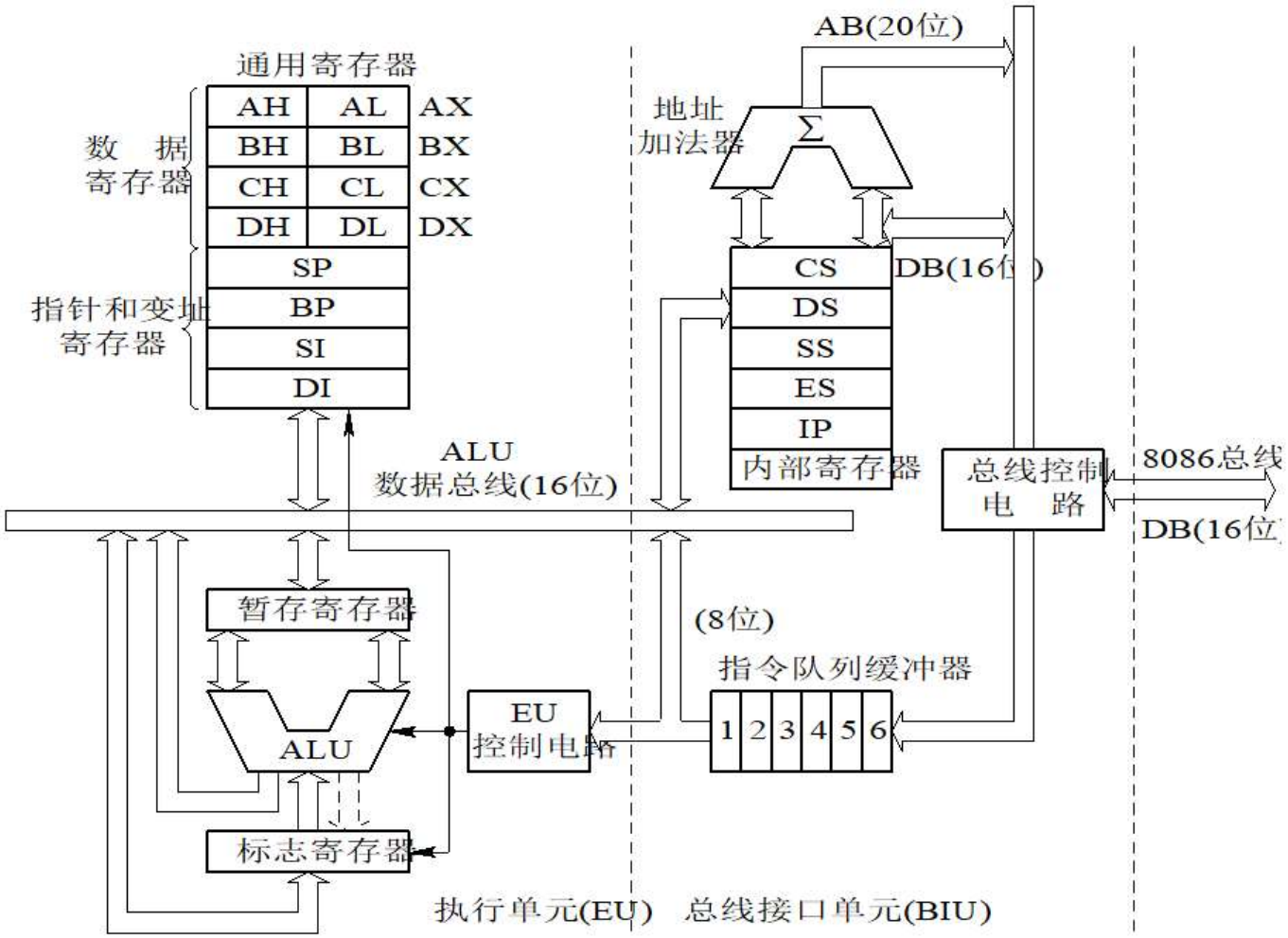
常用工具：Keil C51，Proteus（基于VSM，可以同时仿真外围电路），单片机仿真器（模仿单片机的样子）/仿真程序√（下载进单片机，模拟运行环境）

编程器：专用型，全功能通用型

8086



8086内部结构



8086CPU内部结构框图

总线接口部件BIU

负责信息传送，总线时序控制等

组成：4个16位段寄存器（代码CS、数据段DS（课本此处错？）、堆栈段SS、附加ES），1个16位指令指针（IP，下一条指令的偏移地址），1个20位地址加法器（转换逻辑地址到物理地址，段寄存器左移4位+来自IP/EU的16位偏移地址=20位），6字节指令队列（存放预取的指令，**先进先出**，提高效率），总线控制电路（产生并发出总线控制信号）

执行部件EU

负责指令的执行，从BIU指令队列头部读入指令，译码后向BIU请求操作数，EU执行后返回结果给BIU，同时更新FLAGS中的状态标志位

工作模式

最小工作模式：MN/$\overline{MX}$接VCC，只有一个微处理器，适用小规模系统

最大工作模式：MN/$\overline{MX}$接GND，包含≥2个微处理器，适用中、大规模系统

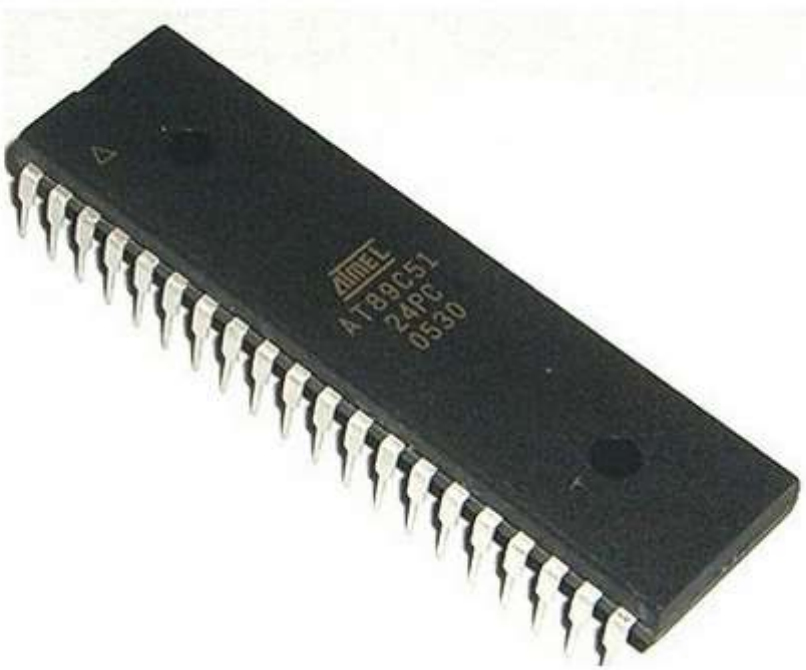
其他参数

引脚功能：书P17-20

存储器：书P20-22，**段地址：偏移地址**

总线时序、寻址方式、内部寄存器、书P22-26

MCS-51



重要引脚

电源引脚：VCC-40（电压根据用户手册定），GND-20（接地）

时钟引脚：XTAL1-19（片内振荡电路输入），XTAL2-18（片内振荡电路输出），内部时钟：晶振（石英晶体，0.5-24MHz）+两个接地电容（10-30pF），外部时钟：CMOS输入至XTAL1，XTAL2悬空

复位引脚：9，高电平>两个机器周期（24个振荡周期），从PC=0000H开始执行，复位后，IO均为高电平，累加器清零，PSW=00H，SP=07H，各个中断源均被关闭并处于低优先级，内部ram

以及SBUF的值不确定（野指针）

$\overline{PSEN}$：外部程序存储器选通，0有效，连接外部ROM的$\overline{OE}$

ALE/$\overline{PROG}$：双功能，控制片外锁存低8位地址/作为编程负脉冲输入，此外，自动输出fosc/6的脉冲信号

$\overline{EA}$/VPP：接1时，访问片内，PC超过0FFFH（4KB）时自动切换外部；接0时，只访问外部，对于内部无ROM的单片机，必须接0

IO口（高低位均为0低7高）

P0（32-39）：8位双向，80H-87H，**三态输出，漏极开路**，可以作为低8位地址/数据总线（无需上拉），作为通用IO（需要上拉），负载能力为8个TTL负载（≥800μA）

P1（1-8）：8位准双向，90H-97H，内置上拉，可作为普通IO，负载能力为每个引脚驱动4个TTL负载（≥400μA）

P2（21-28）：8位准双向，A0H-A7H，内置上拉，作数据高8位（大于256B用，字寻址），也可作为普通通用IO，负载能力为每个引脚驱动4个TTL负载

P3✨（10-17）：8位准双向，B0H-B7H，负载能力为每个引脚驱动4个TTL负载，**每个引脚都具有复用功能**：30-RXD串口receive，31-TXD串口transport，32-$\overline{INT0}$外0输入，33-$\overline{INT1}$外1输入，34-T0定时/计数0输入，35-T1定时/计数1输入，36-$\overline{WR}$写选通，37-$\overline{RD}$读选通

P4：部分型号独有，不做要求

C51需要引用reg51/52.h才能直接使用预定义的变量

硬件结构

中央处理器

8位，负责算术运算，逻辑操作，位运算，位操作，由运算器、控制器和1个布尔（位）处处理器组成

运算器

8位ALU为核心（进行基本四则运算+逻辑运算+循环移位+位操作），包括1个8位ACC累加器（亦称A寄存器，CPU使用最频繁，作为数据的中转站），1个8位B寄存器（ALU乘除法专用，其余时刻可做普通寄存器，乘法时存放被乘数/乘积高8位（A存放乘数/乘积低8位），除法时存放除数/余数（A存放被除数/商）），1个8位程序状态字存储器PSW，和两个8位暂存器TMP1、2（暂存操作数，用户不可见）等

PSW✨：D7H-D0H为CY、AC、F0、RS1、RS0、OV、PSW.1、P(PSW.0)，CY=1表明有进/借位，或作为位累加器；AC为辅助进位标志，用于BCD码计算，与DA A指令配合使用，产生低向高进/借位时硬件置1，其他为0；F0为用户自定义的标志位；RS1与RS0组合选择寄存器组：00-11分别对应第0-3组工作寄存器，分别为00H-07H，07H-0FH，10H-17H，18H-1FH；OV在出现运算溢出的时候会自动硬件置1，否则置0；PSW.1保留位，P为奇偶标志位，1时表示A中1的个数为奇数，奇偶校验时用

控制器

逐条译码，发出控制信号，协调各部分工作

PC：程序计数器，专用寄存器，16位，读取完当前后自动+1，始终指向下一条指令，执行中断时硬件自动暂时压栈，返回后自动恢复，继续执行源程序，**用户无法修改PC内容**，寻址空间最大为2^16=64K

SP：堆栈指针，8位，指示栈顶的位置，复位后07H，**但是从08H开始存放数据**，先入后出，压栈：SP+1，数据进入；出栈：取出栈顶数据，SP-1

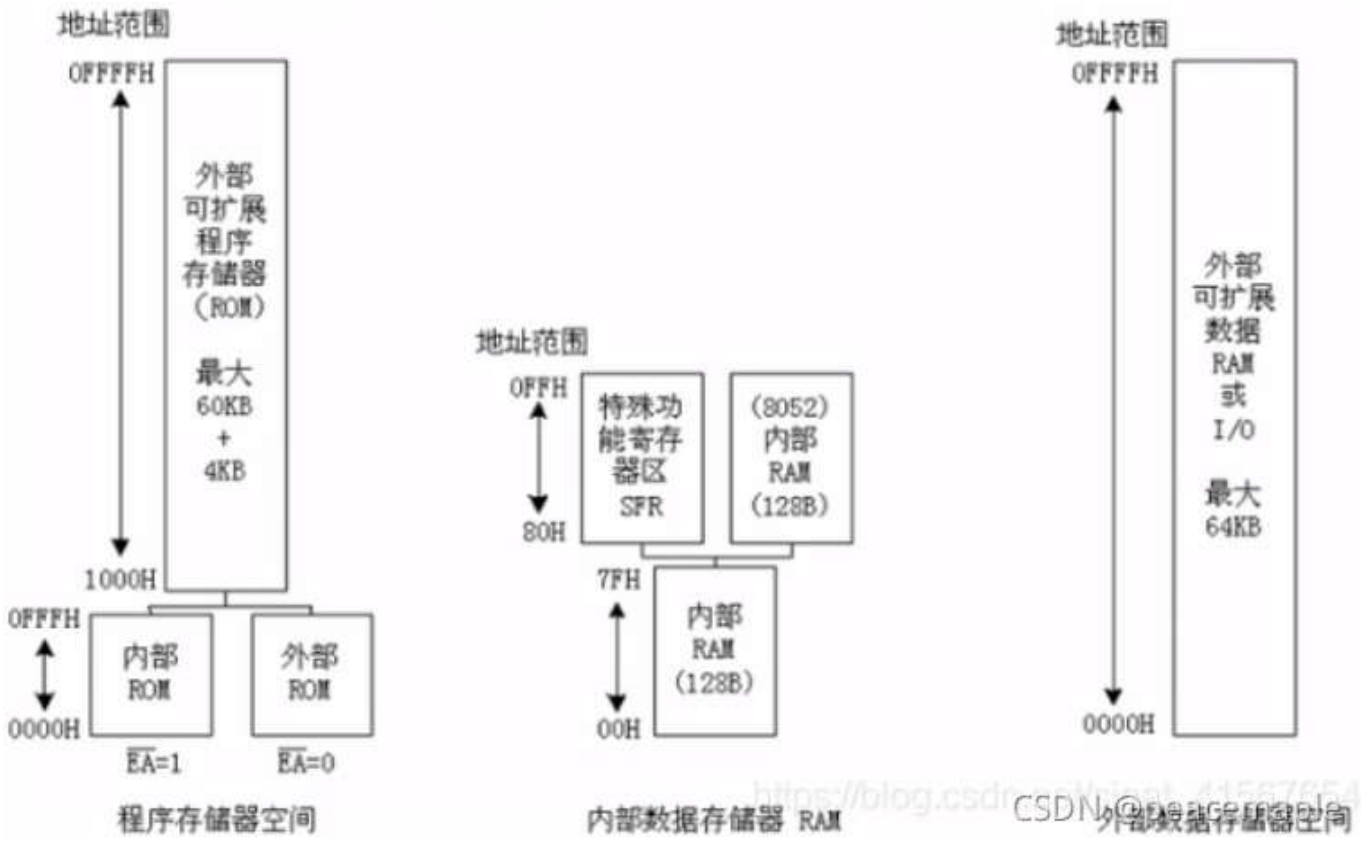
IR:指令寄存器，将指令译码成电平，送往CPU

DPTR：数据指针寄存器，16位，由DPH和DPL组成，作为间接寻址寄存器，也可作为两个独立的8位寄存器使用

布尔（位）处理器

允许51对位变量进行处理，可执行17条位处理指令（位置位，位清零，位取反……），CY为布尔的累加器，操作数均为可直接寻址的位，包括内RAM的位寻址区和11个特殊功能寄存器中可位寻址的位，可以直接对位变量进行处理

存储器结构



图更正：外部地址范围为1000H-FFFFH，以课本为准

51单片机采用哈佛结构进行存储，数据存储器可以直接位寻址访问，更加高效并更容易被8位CPU控制

程序存储器

只读，最大外扩64KB，存放二进制的应用程序和常量数据，0003H-0033H存放中断服务程序（入口地址：外0-0003H，定0-000BH，外1-0013H，定1-001BH，串口-0023H，定2-002BH（52系列独有））；$\overline{EA}$接1时，访问片内，PC超过0FFFH（4KB）时自动切换外部；接0时，只访问外部，无内ROM的如8031，必须接地

数据存储器

存放运算中间结果、数据暂存、缓冲、标志位等，低128B分为3个区

工作寄存器区：00H-1FH，4组×8个工作寄存器，用RS1、0选择，默认第0组

位寻址区：20H-2FH，共128个可按位寻址的位单元，00-7F，支持位寻址+字节寻址

用户RAM区：30H-7FH，共80个单元，只能按字节寻址，堆栈一般设置在30H-7FH的范围内

剩余高128B，为特殊功能寄存器区，分别存放ACC、B、PSW、PC、SP、DPTR、TMOD、TCON、SCON、PCON、IE、IP

下图给出详细位地址的为可按位寻址的SFR，共11个：

表 3　特殊功能寄存器

寄存器名称	字节地址	位地址（16 进制） MSB							LSB	功　能
B	F0H	F7	F6	F5	F4	F3	F2	F1	F0	通用寄存器
ACC	E0H	E7	E6	E5	E4	E3	E2	E1	E0	累加器
PSW	D0H	D7	D6	D5	D4	D3	D2	D1	D0	程序状态字
IP	B8H	BF	BE	BD	BC	BB	BA	B9	B8	中断优先级寄存器
P3	B0H	B7	B6	B5	B4	B3	B2	B1	B0	8 位并行接口
IE	A8H	AF	AE	AD	AC	AB	AA	A9	A8	中断允许寄存器
P2	A0H	A7	A6	A5	A4	A3	A2	A1	A0	8 位并行接口
SBUF	99H									串行口数据寄存器
SCON	98H	9F	9E	9D	9C	9B	9A	99	98	串行口控制寄存器
P1	90H	97	96	95	94	93	92	91	90	8 位并行接口
TH1	8DH									定时器 1 高 8 位数据
TH0	8CH									定时器 0 高 8 位数据
TL1	8BH									定时器 1 低 8 位数据
TL0	8AH									定时器 0 低 8 位数据
TMOD	89H									定时器方式寄存器
TCON	88H	8F	8E	8D	8C	8B	8A	89	88	定时器控制寄存器
PCON	87H									电源控制寄存器
DPH	83H									数据指针高 8 位
DPL	82H									数据指针低 8 位
SP	81H									堆栈指针
P0	80H	87	86	85	84	83	82	81	80	8 位并行接口

补充：52独有的与T2相关的寄存器：T2CON（C8H），T2MOD（C9H），RCAP2L（CAH），RCAP2H（CBH），TL2（CCH），TH2（CDH）

存储器访问

MOV、MOVC、MOVX

C51中存储器管理：

code（程序存储器，0x0000-0xFFFF，共64KB），data（直接寻址的内部数据寄存器，0x00-0x7F，共128B），idata（52独有，间接寻址的内部数据存储器，0x80-0xFF，共128B），bdata（位寻址的内部数据存储器，0x20-0x2F，共16B），xdata（以DPTR寻址的外部数据存储器，0x0000-0xFFFF，共64KB），pdata（以R1、0寻址的外部数据存储器，寻址范围在256B之内）

时钟电路和CPU时序

时钟：外部或者内部提供，频率需要按照要求设置

CPU时序：四种周期，书P44-45

复位电路

上电复位电路：RST经过电阻（一般10kΩ）接地，同时经过电容（一般10μF）接VCC

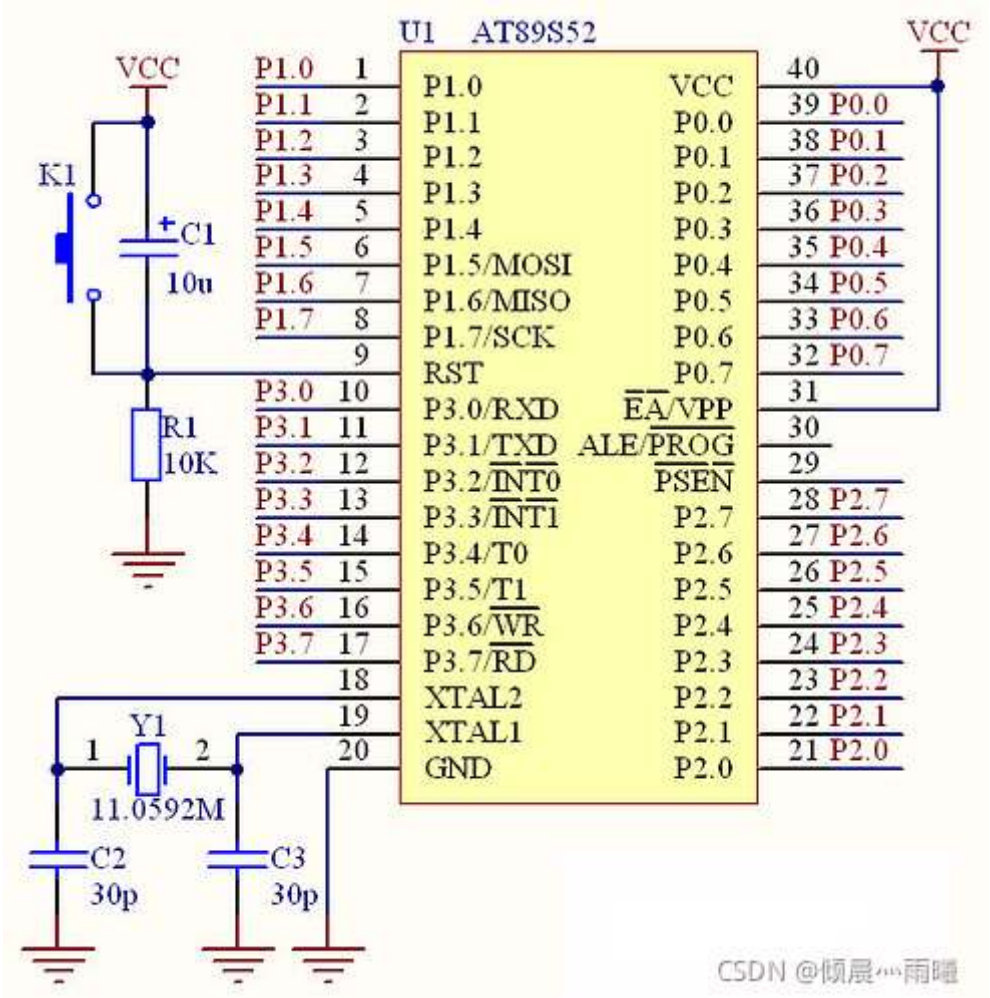
手动复位电路：在上面的基础上，电容两端并联按钮+电阻（可以省略）

看门狗复位电路：部分单片机已经集成，使单片机受到干扰导致程序跑飞后自动复位单片机，正常情况下会有定期的信号到喂狗端，超过时间不喂狗就会产生复位信号。例如，main函数无while循环的话，依然会出现类似于循环的现象，其实是看门狗电路的作用

最小系统✨

包括单片机、晶振电路、复位电路、电源等，为能够使单片机正常工作的最小硬件单元电路

要求会画，记得住



汇编指令系统

```
C_S   SEGMENT
      ASSUME   CS: C_S, DS: C_S
S_T:
      MOV   AX, C_S
      MOV   DS, AX
      LEA   DX, P_S
      MOV   AH, 9
      INT   21H
      MOV   AH, 4CH
      INT   21H
P_S   DB    'Hello World!', 36
C_S   ENDS
      END    S_T
```

指令的格式：指令助记符 操作数1,操作数2,操作数3 ;注释

指令编码最短1字节，最长6字节

寻址方式

8086

立即寻址：

```
MOV AL, 0FFH ;0FFH移动至AL
```

寄存器寻址：

```
MOV CX, AX ;AX的值移动到CX
```

存储器寻址：

1、直接寻址

```
MOV AL, TABLE;
MOV AL, [TABLE];
MOV AL, [1000H] ;如果TABLE的偏移地址是1000H的话，3条指令完全等效
```

2、寄存器间接寻址

```
MOV AX, [BX] ;相当于MOV AX, DS: [BX]
MOV [DI], BX ;相当于MOV DS: [DI], BX
```

3、寄存器相对寻址

```
MOV AL, [BP+Table];
MOV AL, [BP]+[Table];
MOV AL, TABLE [BP];
```

4、基址变址寻址

利用相加得到的操作数作为寻址

```
MOV AX, [BX][SI];
```

5、相对基址变址寻址

相加之后再加一个位移量

```
MOV AX, MASK[BX+DI];
```

51✨

1、立即寻址

```
MOV A, #30H ;将30H送入A中
MOV DPTR, #4567H ;也可送16位
```

2、直接寻址

```
MOV A, 30H ;将30H中的**内容**送到A中
```

3、寄存器寻址

```
MOV A, R0 ;将R0的内容送入A中，同样支持这样操作的有R0-R7，A，B，DPTR等
```

4、寄存器间接寻址

```
MOV A, @R0 ;把R0中的数据作为地址，把地址对应的操作数送入A中，可能有点绕，务必弄清楚了
```

5、变址寻址

```
MOV A, @A+DPTR ;此处的地址为A的地址+偏移量寄存器的地址
```

6、相对寻址

目的地址=转移指令地址+转移指令的字节数+rel，其中rel为带补码的8位二进制数，-128-127

7、位寻址

```
MOV 30H, C ;此处30H是个可以位寻址的单元
```

指令系统

8086

已经确认8086的汇编指令不参与考核，因此此处不再赘述，详见书P60-87

51✨ 建议结合课本P87-100例题一起看，此处不再赘述

1、数据传送

```
;MOV 目的操作数，源操作数
MOV A，Rn;
;MOVX 目的操作数，源操作数
MOVX A，@DPTR;
MOVC A，@A+DPTR ;✨A←((A)+(DPTR))
MOVC A，@A+PC ;✨此处PC+1了，因为要先计算@A+PC，其余同上
```

2、堆栈操作

```
MOV SP，#60H ;设栈底，切记入栈先抬指针再入栈
PUSH ACC ;ACC入栈，进入61H
PUSH B ;B入栈，进入62H
POP DPH ;B的取值出栈，进入DPH中
POP DPL;
```

3、数据交换指令

```
;整字节交换
XCH A，Rn ;A跟Rn换
XCH A，@Ri ;A跟Ri指向的内容换
;半字节交换
XCHD A，@Ri ;A的0-3位跟Ri指向的0-3位换
;自身高低4位交换
SWAP A;
```

4、算数运算✨

;记住此处全是对16进制数字的计算，算不来的可以转成十进制计算
ADD A, Rn ;不进位的加法
ADDC A, Rn ;注意此处(A)+(Rn)后还需要加CY的取值✨
SUBB A, @Ri ;同上，(A)-((Ri))还要-(CY)
;CY在D7有进/借位时为1
;AC在低4位有进/借位时为1
;OV在D7、D6有且仅有一位进/借位时为1
;P在A中1有奇数个时为1
;以上其余情况，对应的变量均为0
;无不带借位的减法指令
;减法时先将操作数变为补码后相加
INC A ;A+1
DEC A ;A-1
DA A ;将结果进行修正，变为BCD码，只能在ADD/ADDC后执行，且加数为BCD码，并只能调节累加器A的结果
MUL AB ;BA=(A)×(B)，低8位→A，高8位→B，此处乘积大于256，OV=1，P的判定照旧，CY始终=0
DIV AB ;无符号整除，商→A，余数→B

5、逻辑指令

CPL A ;取反指令
CLR A ;清零
ANL A, Rn ;按位与，常用于屏蔽某些位
ORL A, Rn ;按位或
XRL A, Rn ;按位异或

6、循环移位

RL A ;左移
RLC A ;CY在A7左边跟着一起左移
RR A ;右移
RRC A ;CY在A7左边跟着一起右移

7、控制转移指令

```
;无条件转移
LJMP addr16 ;长跳转
AJMP addr11 ;绝对转移，PC←(PC)+2,PC10-0←addr11
SJMP rel ;短转移，目的地址=(PC)+2+rel，可双向转移（前后均可）
JMP @A+DPTR ;变址寻址，PC→(A)+(DPTR)
;条件转移
JZ rel ;(A)=0时转移
JNZ rel ;(A)≠0时转移
JB bit, rel ;(bit)=1时转移
JNB bit, rel ;(bit)=0时转移
JBC bit, rel ;(bit)=1时转移并清零bit
JC rel ;(C)=1时转移
JNC rel ;(C)=0时转移
;数值比较不相等转移
CJNE A, direct, rel ;两数不相等，转移，且对于无符号数值，第一<第二，CY=1，(PC)=(PC)+指令字节
;循环转移
DJNZ Rn, rel ;用于构成循环，每次执行这组一次，操作数-1，为0时结束循环，PC=(PC)+指令字节数2/3+
;子程序调用以及返回
ACALL addr11 ;绝对调用，PC+2并入栈，先低后高，转移范围通AJMP
LCALL addr16 ;长调用，PC+3并入栈，同上，可以调用64KB内任意子程序
RET: ;返回，PC出栈，指针-2
NOP ;空指令，单纯消耗程序时间，用于等待或延时
```

8、位操作指令

```
MOV C, 06H
MOV P1.0, C ;位操作无法直接传递，需要中间变量
CLR C ;清零
CPL C ;取反
SETB C ;置1
SETB bit ;位取反
CPL bit ;位置数
ANL C, bit ;按位与
ORL C, bit ;按位或
ANL C, /P1.2 ;(C)←(C)&P1.2取反
```

汇编语言编程

定位伪指令ORG：规定后方程序的起始地址

结束END：END后的指令将不被编译，一个源程序只能包含一个

等值命令EQU：相当于赋值，先定义后使用

数据地址赋值DATA：相较于EQU，无需先定义，但是不能给字符名称赋值，可将一个表达式的值赋给一个字符变量，所以常用于定义数据地址

定义字节DB：定义从标号开始的连续单元

定义字DW：从该地址开始定义若干16位数据，每个占用ROM的两个单元

DB、DW只对程序存储器生效

预留空间DS/DEFS：

位地址符号定义

程序设计步骤：分析题意，选择器件；理清思路，确定算法；编制框图，绘出流程；确定数据结构，分配内存单元；编写源程序；上机调试程序；在单片机上运行

流程图：起止框：圆角矩形；处理框：长方形；判断：菱形；指向线：箭头；连接符：圆圈

源程序的汇编：手动（复杂，容易出错），机器（首选）

C51

表 3-1　Keil C51 支持的数据类型

数据类型	位数	字节数	值 域
signed char	8	1	−128～+127，有符号字符变量
unsigned char	8	1	0～255，无符号字符变量
signed int	16	2	−32 768～+32 767，有符号整型数
unsigned int	16	2	0～65 535，无符号整型数
signed long	32	4	−2 147 483 648～+2 147 483 647，有符号长整型数
unsigned long	32	4	0～+4 294 967 295，无符号长整型数
float	32	4	±1.175494E-38~±3.402823E+38
double	32	4	±1.175494E-38~±3.402823E+38
*	8～24	1～3	对象指针
bit	1		0 或 1
sfr	8	1	0～255
sfr16	16	2	0～65 535
sbit	1		可进行位寻址的特殊功能寄存器的某位的绝对地址

C51程序是遵循ANSI C标准的，不能在for循环内定义循环变量

C51具有多种优点：可读性好，可移植性好，支持模块化开发，无需了解指令系统等

C51程序从主程序开始执行

三种基本结构：顺序、分支、循环

与标准C的区别：库函数不同（stdio.h的printf和scanf除外），数据类型不同，变量存储模式不同，数据存储类型不同，多中断函数，输入输出处理不一样，头文件不同，程序结构的差异（比如不支持递归特性）

基础

标识符：不能以数字开头，不能与关键字重名，区分大小写

关键字：

auto break case char const continue default do

double else enum extern float for goto if

int long register return short signed sizeof static

struct switch typedef unsigned union void volatile while

C51扩充：（此处为避免标识符下划线因为markdown语法而无法显示，因此在下划线与字母之前空开了一格，实际编写不需要）

_ at_ alien bdata bit code compat

data idata intertupt large pdata _priority

reentrant sbit sfr sfr16 small _ task_

using xdata

数据类型：见节首图

data，bdata等见前MCS-51>硬件结构中的描述，此处不再赘述

局部变量：只在函数中生效

全局变量：在全局生效

预处理指令：#include <系统目录中的文件>/"工程目录下的文件"

宏定义：#define

条件编译✨：#IFNDEF/IFDEF #ELSE #ENDIF

基本运算：赋值运算，算术运算，关系运算，逻辑运算，位运算，复合运算

注意：用<<或>>进行位移时，多出的位置将舍弃，空出的位置用0填充，intrins.h中的_crol_(变量,移动的位数)和_cror_(变量,移动的位数)可以很好的解决这个缺陷

注意：先赋值再运算与先运算后赋值的区别（++x和x++）

语句

说明语句：定义变量类型

表达式语句：表达式加上;

空语句：单独的;或者{}

复合语句：多个语句用{}框起来

流程控制语句：分支控制，循环控制，转移

if else switch case break continue do while goto

注意：do while语句至少执行一次，因为是最后判断条件的

函数

用户自定义函数

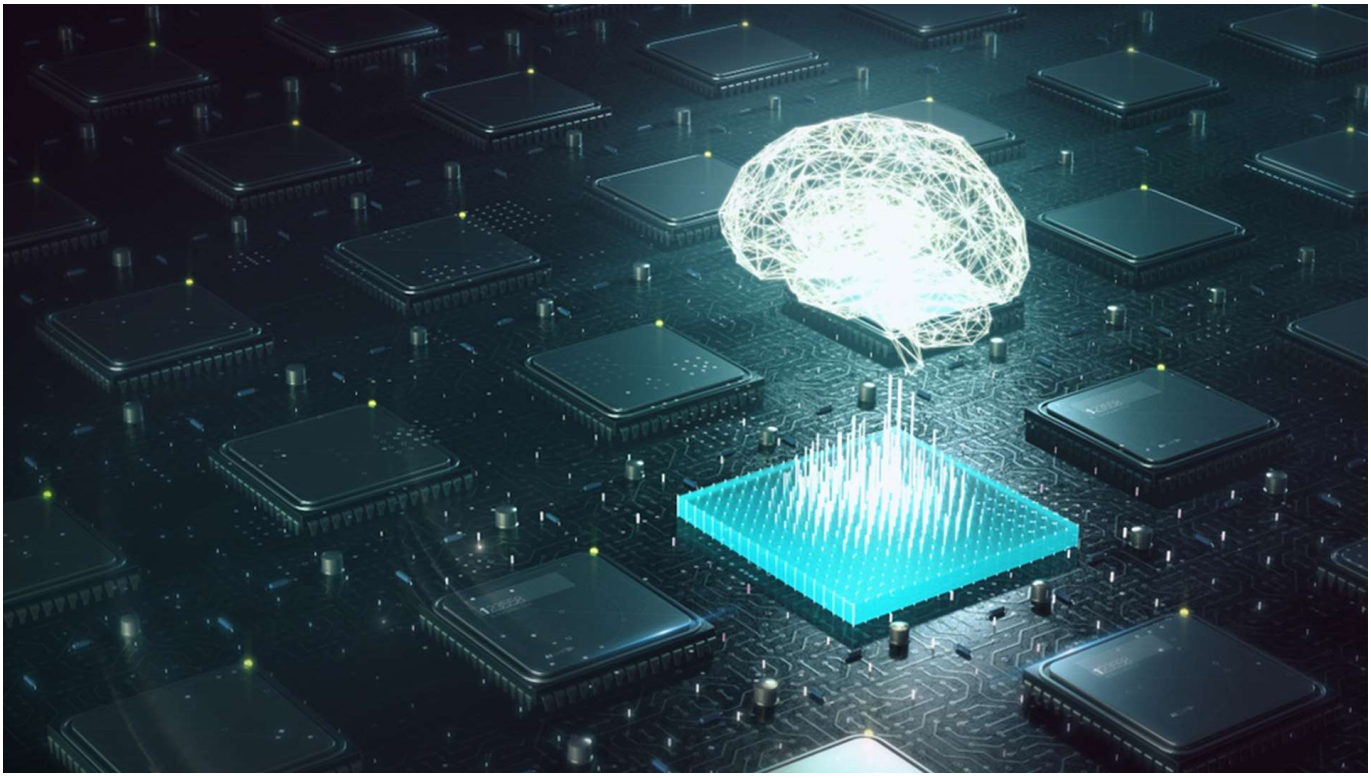
```
类型标识符 函数名(数据类型1， 形参1， ……){
    函数体;
    ……
    return #;//可选
}
```

注意：C51中，函数需要提前声明后再使用

中断服务函数

```
void 函数名字(void) interrupt m using n{
    //对于51单片机，m取值0-4，using用于指明寄存器组，可以省略
}
```

人机接口

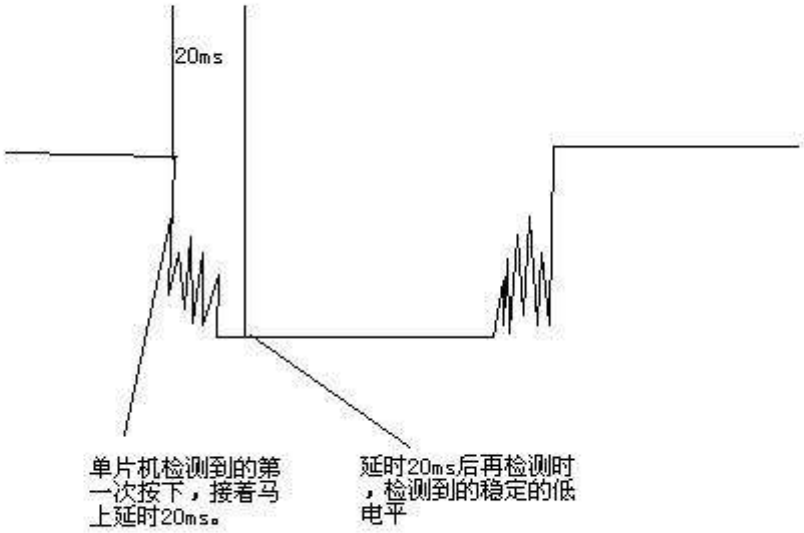


按键

分类：触点式，无触点式，编码式，非编码式……

常用：轻触按键

消抖：



硬件：RS触发器，增加复杂度

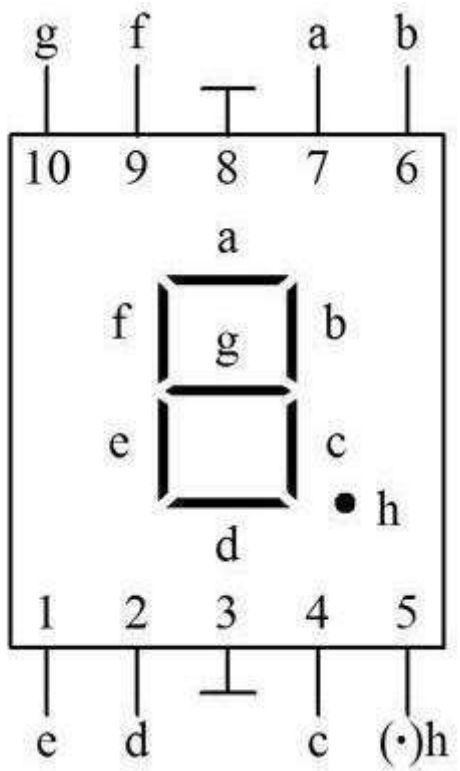
软件：检测到按键后延时再判断，等待松手或者加入变量

独立按键：接上拉/下拉，一般接下拉，方便检测

矩阵按键：逐行扫描（位操作，整组操作），线反转法（扫描两次，需要提前定义键号）

```
{0XEE,0XDE,0XBE,0X7E,
0XED,0XDD,0XBD,0X7D,
0XEB,0XDB,0XBB,0X7B,
0XE7,0XD7,0XB7,0X77};
```

LED数码管



共阴极数码管段码，1点亮

```
{0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};
```

共阳极数码管段码，0点亮

```
{0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};
```

显示小数点：+或者-0x80，根据共阳共阴来判断

静态显示：适用于显示位数少，占用IO

动态显示：适用于显示位数多，编程复杂，一般每位扫描1-2ms，利用视觉暂留产生每位同时点亮的现象，需要注意消隐

液晶显示器

LCD1602（亦称LCM1602）：

编号	符号	引脚说明	标号	符号	引脚说明
1	VSS	电源地	9	D2|	数据
2	VDD	电源正极	10	D3	数据
3	VL	液晶显示偏压	11	D4	数据
4	RS	数据/命令选择	12	D5	数据
5	R/W	读/写选择	13	D6	数据
6	E	使能信号	14	D7	数据
7	D0	数据	15	BLA	背光源正极
8	D1	数据	16	BLK	背光源负极

支持串/并行数据，自带ASCII码以及日文片假名字符（平假名？）字库，也可显示用户自定义字符

显示区域为16×2，每格5×7或10

CGRAM：用于自定义字符，左边需要补3个0，显示的位置填1

DDRAM：数据存储器，对应第一行00-10，第二行10-50共32个显示位置

操作指令：

	指令码		功能	
显示模式	0X38	0011 0111		
显示开/关及光标设置		0000 1DCB	D=1 开显示 C=1 显示光标 B=1 光标闪烁	D=0 关显示 C=0 不显示光标 B=0 光标不闪烁
		0000 01NS	N=1 读或写后地址光标加 1 S=1 整屏移动光标不动	N=0 读或写后地址光标减 1 S=0 整屏不动光标动
数据指针设置		80H+地址码	地址码（0-27H，40H-67H）设置地址指针	
清屏指令	0X01		数据指针清零，显示清零	
	0X02		数据指针清零	

写命令：

```
RS = 0;
RW = 0;
E = 1;
port = com;//port一般为整组端口，com为写入的命令
E = 0;
delayms(1);
```

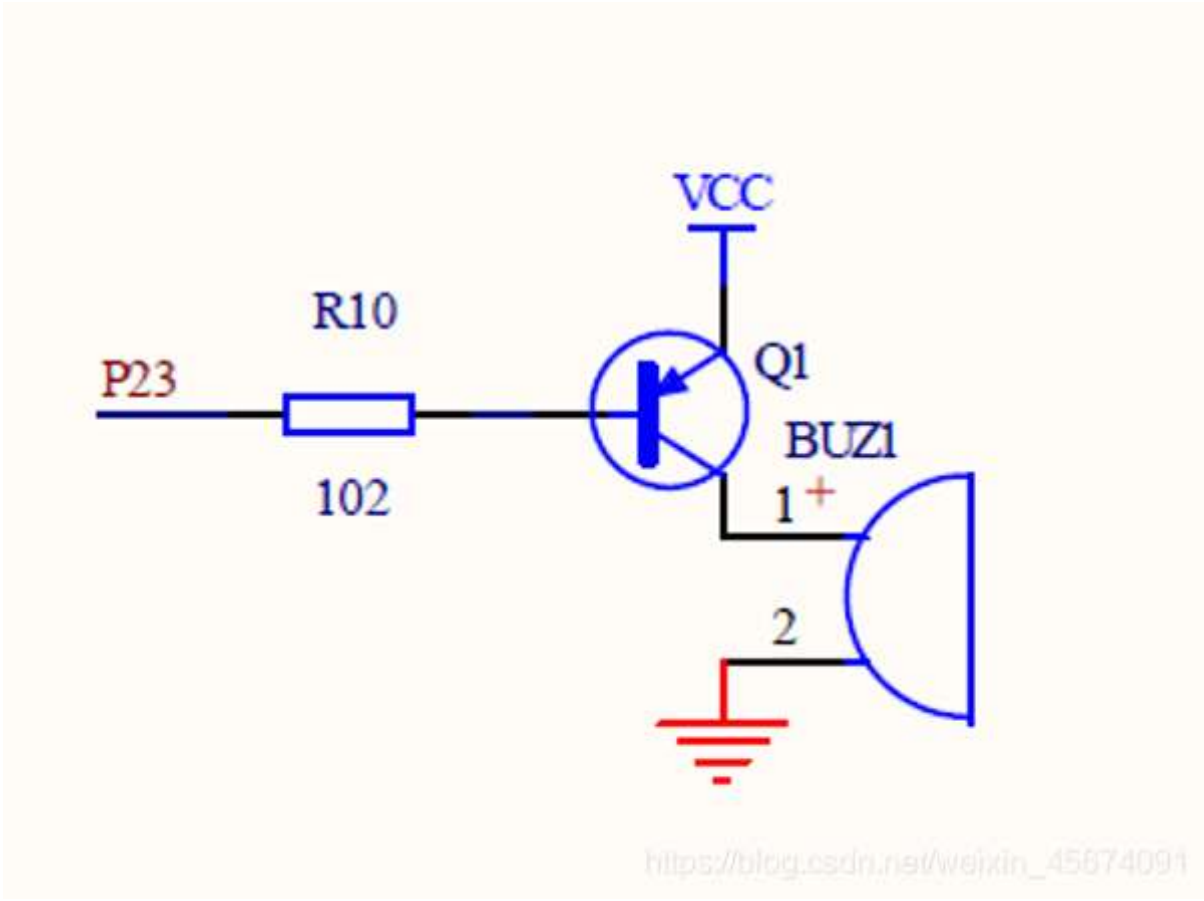
写数据：

```
RS = 1;
RW = 0;
E = 1;
port = data;//data为写入的数据
E = 0;
delayms(1);
```

写字符串（数组）

```
for(i = 0; str[i] != '\0'; i++)
    w_dat(str[i]);
```

蜂鸣器



有源/无源蜂鸣器：有源的通电就发声，无源的需要脉冲信号，有源的不能哆来咪（实测其实可以，但是很难听），无源的可以

注意✨：题目中给出带三极管的电路，**务必小心判断输入什么电平才能让蜂鸣器两端产生电势差**，看清三极管里面的箭头方向，是PNP还是NPN，如图电路在IO输出低电平时蜂鸣器发声，考模电的底子了！

中断、定时/计数器

中断

定义：出现突发情况需要主机干预时，自动暂停当前程序并转入新程序，处理完毕后继续返回被暂停的程序

程序状态字，中断向量

中断系统功能：实现中断响应、返回，优先级排队，中断嵌套

中断源分类：机器故障，程序性，输入/输出设备，外部，调用管理程序

中断源

外部中断0、1：分别从P3^2和P3^3输入，具有电平、脉冲两种触发方式，电平持续时间需要大于一个机器周期

定时/计数器0、1：接收CPU的计数脉冲，或者接收从P3^4和P3^5的计数脉冲，52还有定时器2

串口中断：发送/接收整组数据后，TI/RI置1，需要手动置0

中断寄存器 ✨

定时/计数控制TCON：8F-88分别为TF1、TR1、TF0、TR0、IE1、IT1、IE0、IT0

IT：0为低电平触发，1为下降沿脉冲触发；IE：外部中断申请标志，无需手动操作；TR：定时/计数器的运行控制位，1时启动；TF：定时/计数器溢出标志

串行口控制SCON：9F到98为SM0，SM1，SM2，REN，TB8，RB8，TI，RI

EA为总开关，ES为串口中断开关，ET为允许定时器中断，EX为允许外部中断

中断优先级IP：BF到B8为* 、 * 、 * 、PS、PT1、PX1、PT0、PX0

任何一项为1，则为高优先级

默认优先级从高到低为外0、定0、外1、定1、串口（、定2）

中断入口地址（向量）：外0：0x03、定0：0x0B、外1：0x13、定1：0x1B、串口：0x23（、定2：0x2B？）

中断响应时间：3-8个机器周期

初始化：对应的位置置1，无法按位寻址的整体赋值即可

例：开外0：

```
EX0 = IT0 = EA = 1;
```

定时/计数器0

方式：软件、硬件、可编程定时器……

定时模式：最大频率fosc/12，以此反推定时时长（12MHz，1us定时一次）

计数模式：最大输入频率fosc/24

定/计工作方式控制方式TMOD：**不可位寻址**，位于89H，高低4位分别为T1和T0的控制位，分别为GATE、C/$\overline{T}$、M1、M0

GATE=1时启用INT0、1引脚高电平检测（高电平+TR=1才OK），C/$\overline{T}$=0时为定时，=1时为计数，M1、M0从00-11分别为：13位定时、16位定时、可自动重装8位、定0分为2个8位并关闭定1

定/计控制方式TCON：前一章已经提及，此处不再赘述

方式0和方式1开始下一轮计数之前，就需要**再次将计数初值装入计数寄存器**，再开始计数。重新装入初值的操作会造成定时计数的短暂中断，并产生额外的延迟，影响定时的精度。

方式0：

计数器中设定一个初值X后，那么计数器就从X开始计数，直到计数溢出，一共经历(2^13-X)次计数，定时时间就是(2^13-X)*(12/fclk)。此时，TL=X%32，TH=X/32。（勘误：教材P173页写法2^13-X，X为计数脉冲个数时才可用）

方式1：

同上，只是2^13换成了2^16，32换成了256，教材P174的表达式同样存在着错误

方式2：

当TLx计数溢出时，在溢出标志TFx置1的同时，还自动将THx中的初值送至TLx，使TLx从初值开始重新计数。此工作方式可省去用户软件中 重装初值的指令的执行时间，简化定时初值的计算方法，可以比较精确地控制定时时间。

只有TLx参与计数，THx仅供保存初值，并由TLx自动调用

T=(2^8-X)×Ts，X为初值

方式3（仅用于定时器0）：

方式3是为了使单片机有1个独立的定时器 / 计数器、1个定时器以及 1个串行口波特率发生器的应用场合而特地提供的。这时，可把定时器I用于工作方式2，把定时器0用于工作方式3。

此时，T1只能工作在方式1或2

初值计算同上?

例：12MHz，50ms触发一次，方式1

```
TMOD |= 0x01;//为了防止干扰到其他位所以这样写
TH0 = (65535 - 50000 + 1) / 256;
TL0 = (65535 - 50000 + 1) % 256;
TR0 = 1;
ET0 = 1;
EA = 1;
```

学会编写相应的初始化代码

串口

并行通信：各位同时传输，数据线需要多根，仅适用于近距离，例如LCD1602的并口数据

串行通信：一位一位传输，适用于远距离，例如部分氛围灯的灯控系统

分类：

异步通信：

起始位+5-8位数据+奇偶校验+停止位

同步通信：

同步字符+数据流+CRC

单工：只能单向传输数据

半双工：可以分时发送和接收

全双工：可以同时发送和接收

波特率：单位时间传输二进制代码的位数

常用标准

RS232：采用负逻辑电平，-15--3为1，3-15为0，需要逻辑电平转换，距离一般不超过15m，速率最高为19.2Kbps，速度与距离成反比

RS485：平衡数字多点系统的电气特性标准，最长1219m，最高10Mbps， 速度同样与距离成反比

内部结构

数据缓冲器SBUF（99H）：接收，发送的数据，其实双向的SBUF是分开的

C51可直接SBUF=#或者#=SBUF

串行口控制SCON ✨：可位寻址，9F到98为SM0，SM1，SM2，REN，TB8，RB8，TI，RI

表 7-1　串行口的工作方式

SMO	SM1	工作方式	说　明	波特率
0	0	方式 0	同步移位寄存器	$f_{osc}/12$
0	1	方式 1	10 位移位收发器	由定时器控制
1	0	方式 2	11 位移位收发器	$f_{osc}/32$ 或 $f_{osc}/64$
1	1	方式 3	11 位移位收发器	由定时器控制

SM2：多机控制位，方式0，SM2必须置0，其他方式SM2=1时，只有接收到停止位并且RI=0，或RB8=1时，RI=1；TB8：发送的奇偶校验位；RB8：奇偶校验位（方式2、3）或停止位（方式1）；TI、RI在接收/发送数据后置1，**有且仅有这两个需要手动置0**

中断允许控制IE：AF到A8为EA、*、*、ES、ET1、EX1、ET0、EX0

串口的TI和RI需要手动清零

电源控制寄存器PCON（87H？）：SMOD * * * GF1 GF0 PD IDL

SMOD为串口波特率倍增位，SMOD=1时波特率×2

工作方式

方式0

主要用于串并转换（使用74164等芯片），串口扩展输入

波特率固定fosc/12，RXD输出数据，TXD输出移位脉冲

例：初始化代码

```
SCON = 0x00;
```

方式1

10位为一帧异步通信，全双工，波特率可调，停止位送进RB8

接收的2个条件：RI=0且REN=1

接收生效的2个条件：RI=0且SM2=0/接收到的停止位是1，否则丢弃此帧数据

波特率为(2^SMOD/32)×(fosc/12×(256-X))，初值X=256-fosc×2^SMOD/(384×波特率)

T1作为波特率发生器，工作在方式2

方式2、3

每帧11位异步通信，仅在波特率上区分

接收的2个条件：RI=0且REN=1

接收生效的2个条件：RI=0且SM2=0/接收到的停止位是1，否则丢弃此帧数据

方式2波特率：**2^SMOD/64×fosc**

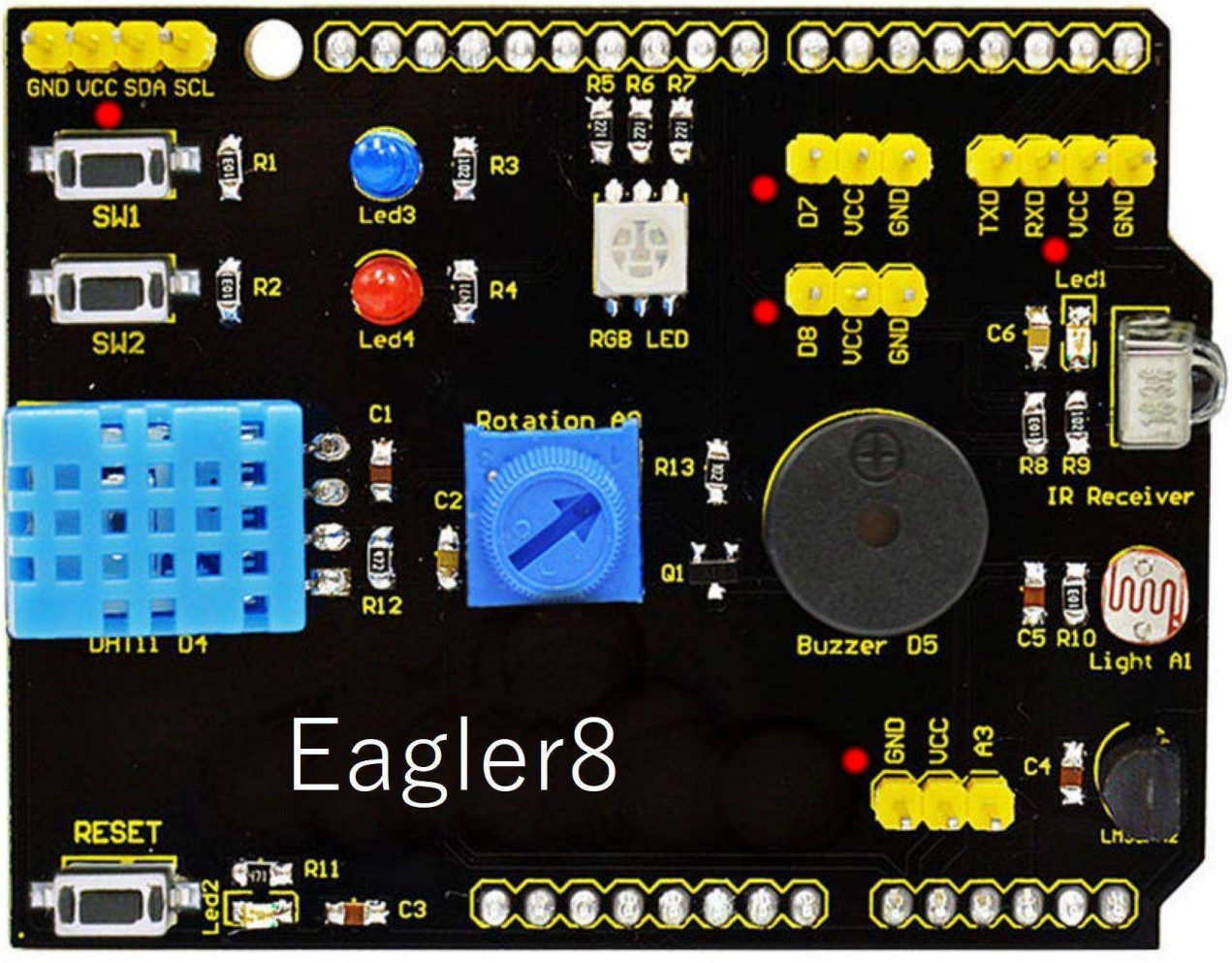
方式3波特率：**波特率为(2^SMOD/32)×(fosc/12×(256-X))，初值X=256-fosc×2^SMOD/(384×波特率)**

T1作为波特率发生器，工作在方式2

例：初始化函数，双机，方式1，11.0592MHz，2400

```
SCON = 0x50;//方式1
PCON = 0;
TMOD = 0x20;//定时器方式2
TH1 = TL1 = 244//计算值
TR1 = 1;//开T1
EA = ES = 1;//开中断
```

接口扩展



并行总线结构

地址总线AB、数据总线DB、控制总线CB

编址技术

线选法：直接IO接在外片的CS上，最大n个

译码法：使用74138等芯片进行译码操作，适用于大容量多芯片存储器的扩展，最大2^n个，根据使用IO的数量还分为部分译码和全译码

AD

部分单片机已经内置

逐次比较型，双积分型，∑-△型（每种的特点，书P205-206）

技术指标：分辨率（VFSR/2^n），量化误差（因为分辨率有限导致真实值与测量值之间的误差），转换精度（实际与理想的差距），转换时间（一次转换所需的时间），温度系数

DA

同样，部分单片机也内置了DAC

分类：电压输出、电流输出（需要加I-V运放）

技术指标：分辨率，建立时间，转换精度

SPI

摩托罗拉推出，同步串行交换信息，主从工作模式，最少4线（单向可3线）

SCK，MOSI，MISO，CS

传输速率高达1.05Mb/s

TLC549

8位AD，内置采样保持器，模数转换器，输出数据寄存器，数据选择器，驱动器等电路

接口函数：

```
u8 TLC549_ADC(){
    u8 i, tmp;
    TLC549_CLK = 0;
    TLC549_CS = 0;
    for(i = 0;i < 8;i ++){
        tmp <<= 1;
        tmp |= TLC549_DO;
        TLC549_CLK = 1;
        TLC549_CLK = 0;
    }
    TLC549_CS = 1;
    delayus(20);
    return tmp;
}
```

注意：本程序仅用作读取芯片的输出，并不意味着单片机自身具备AD转换的功能

TLC5615

10位DA，内置16位移位寄存器，两种工作模式（12位数据序列或16位数据序列）

12位接口函数

```
void TLC5615_DAC(u16 dat){
    u8 i;
    dat <<= 6;
    TLC5615_CLK = 0;
    TLC5615_CS = 0;
    for(i = 0; i < 12; i ++){
        TLC5615_DI = (bit)(dat & 0x8000);
        TLC5615_CLK = 0;
        dat <<= 1;
        TLC5615_CLK = 1;
    }
    TLC5615_CS = 1;
    TLC5615_CLK = 0;
}
```

16位接口函数

```
dat左移2位
拉低CLK与CS
DI = (强制bit)(dat与上0x8000)
拉低CLK
dat左移一位
抬高CLK
以上4条重复16次
抬高CS
拉低CLK
```

同样，此函数无法证明单片机具有DA功能，仅仅作为一个读取芯片的函数

IIC

现在多采用飞利浦的技术规范（另一个为索尼），IIC空闲时SCL和SDA均为高电平，各器件的SCL和SDA都是线与的关系

速率为100Kb/s，高速模式为400Kb/s

数据帧格式

下面加粗的为从到主

主发从：S 从机地址 0 **A** 字节1 **A** …… 字节n-1 **A** 字节n **A/#A** P

从发主：S 从机地址 1 **A 字节1 A …… 字节n-1 A 字节n** #A P

主发从后接主：S 从机地址 0 **A** 数据 **A/#A** Sr 从机地址r 1 **A 数据** #A P

例：IIC接口函数✨

开IIC总线

```
SDA = 1;
SCL = 1;
//延时5μs
SDA = 0;
//延时5μs
SCL = 0;
```

终止信号P

```
SDA = 0;
SCL = 1;
//延时5μs
SDA = 1;
SDA = 0;
```

应答0函数

```
SDA = 0;
SCL = 1;
//延时5μs
//等待SDA被拉高或等待超时
SCL = 0;
//延时5μs
```

非应答1函数

```
SDA = 1;
SCL = 1;
//延时5μs
SCL = 0;
SDA = 0;
```

发送一个字节

```
//dat左移一位
SCL = 0;
//延时5μs
SDA = CY;//左移多的进了CY，所以能这样写
//延时5μs
SCL = 1;
//延时5μs
//以上指令重复8次
SCL = 0;
//延时5μs
SDA = 1;
//延时5μs
```

接收一个字节

```
SCL = 0;
//延时5μs
SDA = 1;
//延时5μs
//以下指令重复8次
SCL = 1;
//延时5μs
temp = (temp << 1) | SDA);
SCL = 0;
//延时5μs
//重复截至此处
//延时5μs
//返回temp
```

AT24C02

EEPROM，256B，具有3个地址引脚，支持字节写入，页写入，指定地址读，指定地址连续读

单字节读写接口函数

```
void write_24c02(unsigned char addr,unsigned char dat)
{
    IIC_Start();
    IIC_SendByte(0XA0);
    IIC_WaitAck();
    IIC_SendByte(addr);
    IIC_WaitAck();
    IIC_SendByte(dat);
    IIC_WaitAck();
    IIC_Stop();
}

unsigned char read_24c02(unsigned char addr)
{
    unsigned char dat=0;
    IIC_Start();
    IIC_SendByte(0XA0);
    IIC_WaitAck();
    IIC_SendByte(addr);
    IIC_WaitAck();
    IIC_Start();
    IIC_SendByte(0XA1);
    IIC_WaitAck();
    dat=IIC_RecByte();
    IIC_Stop();
    return dat;
}
```

PCF8591

AD、DA双功能芯片，支持4路输入AD，1路DA输出，支持AD、DA同时工作（电压跟随），同样有3个地址引脚

AD、DA的接口函数

```
uchar read_adc(uchar add)//Rb2电阻 0x03
{
    uchar temp;
    IIC_Start();
    IIC_SendByte(0x90);
    IIC_WaitAck();
    IIC_SendByte(add);//0x40-0x43
    IIC_WaitAck();
    IIC_Start();
    IIC_SendByte(0x91);
    IIC_WaitAck();
    temp=IIC_RecByte();
    IIC_WaitAck();
    IIC_Stop();
    return temp;
}

void write_dac(uchar add,uchar dat)//必须写0x40，dat为0-255（对应0-5v）
{
    IIC_Start();
    IIC_SendByte(0x90);
    IIC_WaitAck();
    IIC_SendByte(add);
    IIC_WaitAck();
    IIC_SendByte(dat);
    IIC_WaitAck();
    IIC_Stop();
}
```

微处理器开发流程

理解系统功能与技术指标，选择微处理器类型，关键器件的选择，软硬件功能划分，硬件设计，资源分配

完结撒花 cheers~ 🎉