第一章

1.3

存储器容量对应地址线的根数：8K=2^n,所以 n=13.

1.4

78=01001110B=4EH=01111000BCD

134=10000110B=86H=000100110100BCD

2.1

X=1000100

8 位二进制表示：[x]原=01000100；[x]反=01000100；[x]补=01000100；

16 位二进制表示：[x]原=0000000001000100；[x]反=0000000001000100；

[x]补=0000000001000100；

X=-1000100

8 位二进制表示：[x]原=11000100；[x]反=10111011；[x]补=10111100；

16 位二进制表示：[x]原=1000000001000100；[x]反=1111111110111011；

[x]补=1111111110111100；

X=-0111111

8 位二进制表示：[x]原=10111111；[x]反=11000000；[x]补=11000001；

16 位二进制表示：[x]原=1000000000111111；[x]反=1111111111000000；

[x]补=1111111111000001；

2.2 将下列补码转化成二进制数的真值

1）[X]补=00101100 2）[X]补=11111111 3）[X]补=10000000

(1) X=0101100

(2) X=-0000001

(3) X=-10000000;

2.3 已知下列补码[X]补和[Y]补，分别求[X+Y]补、[X-Y]补。并判断运算结果是否出现溢出。（这里注意周老师书上说的变形补码，高位 1 补 1，高位 0 补 0 ）

(1) [X+Y]补=11000101 110011001

+000101100

111000101 不溢出

[X-Y]补=01101101 110011001

-000101100

101101101 溢出

(2) [X+Y]补=01111111 111111111

+110000000

101111111 溢出

[X-Y]补=01111111 111111111

-110000000

001111111 不溢出 (3) [X+Y]补=00010111 000110111

+111100000

000010111 不溢出

[X-Y]补=01010111 000110111

-111100000

001010111　　不溢出

(4) [X+Y]补=01000111　　110000111

+　111000000

101000111　　溢出

[X-Y]补=11000111　　110000111

-　111000000

111000111　　不溢出

第二章

1.1 89C52 内部资源：

一个 8 位的 CPU，8KB 的 FLASH 程序存储器，256B 的静态 RAM 数据存储器，4 个 8 位的并行 I/O 接口，1 个全双工串行口，中断系统，3 个定时器/计数器，时钟电路

1.2 采用哈佛结构，程序存储器和数据存储器具有独立存储器空间，具有较高的执行效率。

1.3 工作寄存器分布在 00H—1FH 区域。

1.4 工作寄存器分 4 个区，可以改变特殊功能寄存器中的 RS1 和 RS0 的位状态来改变工作寄存器区。

1.5　工作寄存器是暂时存放数据的部件，而特殊功能寄存器是存放一些专用的信息。

1.6　特殊功能寄存器的地址空间：80Ｈ—FFH

1.7　位寻址区域在内部数据寄存器的 20H—2FH

1.8　A0H=P2，B0H=P3，E6H=ACC

1.9　RC 电路中，时间常数 t=RC,如果 C 太小，则时间常数会变小，产生的脉冲信号将变窄，有可能小于单片机的两个机器周期，从而不能复位。

1.10　上电复位时，内部数据存储器的内容是不确定和随机的；人工按钮复位时，内部数据存储器的内容保持不变。

1.12　因为 SP 指示的是当前堆栈存放信息的位置。只有重新设置，才能知道堆栈地址。

2

60K，0000H—EFFFH

第三章

1.1 指令的格式是由哪些部分组成的？每部分的含义是什么？

答：操作码和操作数。

1.2 什么是寻址方式？51 系列指令系统有哪些寻址方式？

答：指令给出操作数的方式即用何种方式找到操作数。

1.3 对于内部数据存储器 00H~1FH 区域的访问有哪些寻址方式？对于外部数据存储器的访问有哪些寻址方式？对于特殊功能寄存器的访问有哪些寻址方式？

答：1）直接寻址和寄存器间接寻址方式

2）寄存器间接寻址方式

3）直接寻址方式

1.4 51 系列单片机有哪些标志位？这些标志位存放在哪里

Cy AC OV P RS0/RS1 F0/F1，PSW

2.1

MOV　　R4，38H　　寄存器寻址、直接寻址

ADD　　A，　@R1　　寄存器寻址、寄存器间接寻址

MOVC A, @A+DPTR 寄存器寻址、变址寻址
MOVX A,@DPTR 寄存器寻址、寄存器间接寻址
DEC B 寄存器寻址
SETB 24H 位寻址
CJNE A,#100, NEXT 立即寻址、相对寻址
ANL 30H, #00H 直接寻址、立即寻址
PUSH P1 直接寻址

2.2
答: 合法：1，5，10
不合法：2，3，4，6，7，8，9

2.3
(1)
```
MOV R0,30H
MOV A,31H
MOV 30H,A
MOV 31H,R0
```
(2)
```
MOV A,R7
RR A
MOV R7,A
```
(3)
```
MOV A,R3
MOV B,#4
MUL AB
MOV R3,A
```
(4)
```
MOVX A,@DPTR
DEC A
MOVX @DPTR, A
```
(5)
```
PUSH 1EH 或 MOV PSW,#10H
MOV A,R6
PUSH A
```
(6)
```
MOV A,R4
RRC A
MOV R4,A
MOV A,R5
RRC A
MOV R5,A
MOV A,R4
RL A
RRC A
MOV R4,A
```

2.4 (注意每条语句在原始条件下单独执行)
（1）（R0）=20H
（2）（25H）=F8H
（3）（A）=13H P=1

（4）（A）=38H　　P=1
（5）（A）=33H　　P=0
（6）（A）=03H　　P=0
（7）（A）=36H　　（34H）=03H
（8）（A）=08H　　（36H）=F3H
（9）（SP）=72H　（72H）=05H
（10）（DPH）=00H　（SP）　=70H
（11）0CCH, 1,0,1,0
（12）E7H,0,0,0,0
（13）FCH,0,0,0,0
（14）2CH,01H
（15）01H
（16）02H,1
（17）52H,0,0
（18）02H,36H,1
（19）5BH,1
（20）77H,0
（21）03H,0
（22）00H,0
（23）0FCH,0
（24）0FCH,0
（25）06H,1
（26）81H,1
（27）1
（28）0A0H
（29）88H
（30）00H
（31）0
（32）1202H
（33）0900H
（34）1280H
（35）1148H
（36）1800H,73H,03H,12H
（37）80H,6FH

2.5

（1）R3->R4
（2）R5 取反
（3）R4 除以 R5，商放 R4，余数放 R5
（4）P1.1 与 P1.2 与!P1.3，结果放 P1.6
（5）位存储 00H 和位存储 01H 进行或操作，位存储 02H 和 03H 进行或操作，两次结果进行与操作，放 P1.7
（6）R4 和 R3 连在一起左移一位（不循环）
（7）R5*3，结果放 R5
（8）00H-FFH 置零

（9）AB 交换

第四章

1.1 用伪指令将下列常数依次定义在 1600H 为首地址的程序存储器中。

```
        org 1600H
LABLE: db 0,1,4,9,16,25,36,49,64,81
```

1.2 用伪指令将下列常数依次定义在 1700H 为首地址的程序存储器中，要求数据类型一致 。

```
        org 1700H
LABLE: dw 0,1,8,27,64,125,216,343,512,729
```

1.3 用伪指令将字节型变量 X1、X2 定义在内部数据存储器 30H、31H 单元中，将字节型变量 Y1、Y2 定义在外部数据存储器 2000H、2001H 中。

```
X1 data 30H
X2 data 31H
Y1 xdata 2000H
Y2 xdata 2001H
```

1.4 用伪指令将 ASCII 码字符串“Beijing,how are you!”定义在 1200H 为首地址的程序存储器中。

```
        org 1200H
LABLE: db ‘Beijing, how are you!’
```

1.5 用伪指令将逻辑变量 A1、A2 定义在 00H、01H 位单元中。

```
A1 bit 00H
A2 bit 01H
```

2.1

```
        MOV R2,#200
        MOV DPTR,#1400H
        MOV P2,#15H
        MOV R0,#00H
LOOP:   MOVX A,@DPTR
        MOV P2，#15H
        MOVX @R0,A
        INC DPTR
        INC   R0
        DJNZ R2,LOOP
        SJMP $
        END
```

2.2

```
        MOV R2,#20
        MOV DPTR,#2400H
        MOV R0,#30H
LOOP:   MOV A,@R0
        JNB ACC.7 NEXT
        MOVX @ DPTR,A
        INC DPTR
NEXT:   INC R0
```

```
        DJNZ R2,LOOP
        SJMP $
        END
```

2.3

```
        MOV R2,#25
        MOV R0,#40H
        MOV R1,#3FH
LOOP:   MOVX A,@R0
        MOVX @R1,A
        INC R0
        INC R1
        DJNZ R2,LOOP
        SJMP $
        END
```

2.4

```
        x data 30h
        y data 31h
        MOV A ,x
        MOV B,#5
        MUL AB
        MOV y,A
        MOV A,x
        MOV B,A
        MUL AB
        ADD A,#6
        ADD A,y
        MOV y,A
        SJMP $
        END
```

2.5

```
入口参数:   x bit 00H
            y bit 01H
            z bit 02H
出口参数:  L bit 03H

        ORG 0000H
        LJMP MAIN
        ORG 0040H
MAIN:   MOV c,x
        ANL c,/y
        ANL c,z
        CPL c
        MOV L,c
        SJMP $
```

```
        END
```

2.6

（1）地址为 1000H 的存储单元连续 10 个单元，最大值放到寄存器 B

（2）计算 R3 中 1 的个数

（3）统计 50H 块的数目，FF 结束

2.7

入口参数：子程序名 DISP

转换的 BCD 在累加器 ACC 中

出口参数：转换后的七段显示码在累加器 ACC 中

```
DISP:   PUSH PSW
        PUSH DPH
        PUSH DPL
        MOV DPTR, #TABLE
        MOVC A,@A+DPTR
        MOVC A，@A+PC
        POP DPL
        POP DPH
        POP PSW
        RET
TABLE: DB 3fh,06h,6bh,4fh,66h
        DB 6ph,7dh,07h,7fh,6fh
```

2.8

```
MAP:    PUSH    PSW                         ;保护现场。
        PUSH    B;
        PUSH    ACC;
        MOV     DPTR,   #TABLE              ;设置数据表格指针。
        CLR F0                              ;清 F0。
LOOP:   CLR     A                           ;清 ACC。
        MOVC    A,@A+DPTR                   ;从表格中取数。
STORE1: JNZ     CONTI                       ;判断查表是否结束。
        SJMP STOP                           ;查表结束，转入 STOP。
CONTI:  CJNE A,40H,NEXT                     ;若 x≠ai，转入 NEXT。
        INC DPTR                            ；若 x=ai, 则调整指针取 bi。
STORE2: CLR     A;
        MOVC    A,@A+DPTR;
        MOV         50H，A
        SETB    F0                          ;使 F0 等于 1, 表示查表成功。
STOP:   POP             ACC                 ;恢复现场。
        POP PSW                             ;
        RET                                 ;子程序返回。
NEXT:   INC  DPTR                           ;调整指针，准备继续检索。
        INC  DPTR                           ;
        SJMP    LOOP                        ;
TABLE:  DB 3fh, 00h,06h,01h,6bh,03h,4fh,02h,66h,06h
```

```
                DB 60h,0eh,7dh,0ah,07h,08h,7fh,0ch,6fh,04h,00h
```

2.9

入口参数：子程序名 COMADD

加数在 R2R3

被加数在 R4R5

出口参数：和在 R6R7

和的进位位在 00H

均在 0 区寄存器

```
COMADD:     PUSH   PSW
            PUSH   ACC
            CLR    C
            MOV    A,R3
            ADD    A,R5
            MOV    R7，A
            MOV    A,R2
            ADDC A,R4
            MOV    R6，A
            MOV 00H,C
            POP    ACC
            POP    RET
            RET
```

2.10

```
    D10ms:  MOV R7,#40
    LOOP1:  MOV R6,#248
            NOP
    LOOP2:  DJNZ R6,LOOP2
            DJNZ R7,LOOP1
            RET
```

2.11

```
    BCDCON:     PUSH PSW
                PUSH ACC
                MOV PSW,#08H
                MOV R0,#40H
                MOV R1,#50H
                MOV R6,#5
    LOOP:       CLR A
                XCHD A,@R0
                SWAP A
                INC R0
                XCHD A,@R0
                MOV @R1,A
                INC R0
                INC R1
                DJNZ R6,LOOP
```

```
        POP ACC
        POP PSW
        RET
```

2.12

```
SORT:   PUSH PSW
        PUSH ACC
        MOV PSW, #08h
        MOV B,#14
START:  MOV R7,B
        MOV R0,#30
        CLR 00H
LOOP:   MOV A,@R0
        MOV 0BH,A
        CPL A
        INC A
        MOV 09H,A
        INC R0
        MOV A,@R0
        CPL A
        INC A
        MOV 0AH,A
        MOV A,0BH
        CJNE 09H,0AH,COMP
COMP:   JC NEXT
        XCH A,@R0
        DEC R0
        XCH A,@R0
        SETB 00H
NEXT:   DJNE R7,LOOP
        DEC B
        JB 00H,START
        POP ACC
        POP PSW
        RET
```

2.13

```
RL:     PUSH PSW
        PUSH ACC
        CLR C
        CLR A
        MOV R0,#39H
        MOV R6,#10
LOOP:   MOV A,@R0
        RLC A
        MOV @R0,A
```

```
        DEC R0
        DJNZ R6, LOOP
        POP ACC
        POP PSW
        RET
```

第五章

1.1 中断申请信号如何才能被 CPU 检测到?

答：中断允许寄存器的设置为‘1’，即允许中断

1.2 中断响应时 CPU 需要做哪些事情?

答：保护正在执行程序的断点地址，转入特定的中断入口地址。

1.3 中断入口与中断服务子程序入口有什么区别?

答：中断入口地址是 CPU 在固定的程序存储单元地址中存放中断源的入口地址。

1.4 89C52 单片机中有哪些中断源？它们的中断申请标志分别是什么?

答：中断源有 INT0、INT1、T0、T1、RI/TI、T2，标志为 IE0、IE1、TF0、TF1、RI/TI、TF2。

2.1

（1）答：为中断 1 服务，应为在同级中断中中断 1 的级别比 T1 的高。

（2）答：能响应外部中断 1、定时器 T1、以及定时器 T2，应为它们是高级中断，而外部中断 0 是低级中断

2.2

```
        ORG 0000H
        SJMP MAIN
        ORG 0013H
        SJMP INT1
        ORG 0100H
MAIN:   MOV SP, #60H
        SETB EA
        SETB EX1
        SETB IT1
        CLR PX1
        MOV P1, #0FFH
        SJMP $
        ORG 1100H
INT1:   PUSH ACC
        PUSH PSW
        PUSH DPH
        PUSH DPL
        MOV A,P1
        MOV DPTR，#2400h
        MOVX @DPTR,A
        POP DPL
        POP DPH
        POP PSW
        POP ACC
        RETI
```

2.3

```
        ORG 0000H
        SJMP MAIN
        ORG 0003H
        MOV A,P1
        CPL A
        MOV P1, A
        RETI
        ORG 0100H
MAIN:   MOV SP, #60H
        SETB EA
        SETB EX0
        SETB IT0
        SETB PX0
        MOV P1, 0FFH
        SJMP $
        END
```

第六章

1.4

T1 工作在模式 1 时，定时器最长的定时时间为：

$$T_{max} = \frac{12}{f_{osc}} \times 2^{16} = \frac{12}{6 \times 10^6} \times 2^{16} = 131.072(ms)$$

T1 工作在模式 2 时，定时器最长的定时时间为：

$$T_{max} = \frac{12}{f_{osc}} \times 2^{8} = \frac{12}{6 \times 10^6} \times 2^{8} = 0.512(ms)$$

2.1

$$a = 2^{16} - \frac{f_{osc}}{12} \times T = 65536 - \frac{6 \times 10^6}{12} \times 5 \times 10^{-3} = 63036 = F63CH$$

根据题意，对 T1 初始化编程如下：

```
MOV     TMOD,     #10H;       将 T1 设置为模式 1 定时器方式，内部启动。
MOV     TH1,#F6H;             设置计数器初值。
MOV     TL1, #3CH;            设置计数器初值。
SETB    TR1;                  启动计数器工作。
```

2.2

$$a = 2^{8} - \frac{f_{osc}}{12} \times T = 256 - \frac{6 \times 10^6}{12} \times 0.25 \times 10^{-3} = 131 = 83H$$

根据题意，对 T0 初始化编程如下：

```
MOV     TMOD,     #0AH;       将 T0 设置为模式 2 定时器方式，外部启动。
MOV     TH0,#83H;             设置计数器初值。
MOV     TL0, #83H;            设置计数器初值。
SETB    TR0;
```

2.3

$$a = 2^{16} - 500 = 65536 - 500 = 65036 = FE0CH$$

系统对外部计数信号的频率最高为：

$$f = \frac{1}{24} f_{osc} = \frac{1}{24} \times 24MHz = 1MHz$$

根据题意，对 T0 初始化编程如下：

```
MOV     TMOD,     #05H;       将 T0 设置为模式 1 计数器方式，内部启动。
MOV     TH0,#FEH;             设置计数器初值。
MOV     TL0, #0CH;            设置计数器初值。
SETB    TR0;                  启动计数器工作，准备对外部信号计数。
```

2.4

$$a = 2^{16} - \frac{f_{osc}}{12} * T = 65536 - \frac{12*10^6}{12} * 50*10^{-3} = 15536 = 3CB0H$$

初始化编程：

```
MOV T2MOD,#00H;       将 T2 设置为自动装入初值的定时器/计数器模式
MOV T2CON,#00H;       T2 处于定时器模式，加 1 计数。
MOV TH2,#3CH;         设置计数器初值
MOV TL2,#0B0H;        设置计数器初值
MOV RCAP2H,#3CH;      设置寄存器初值
MOV RCAP2L,#0B0H;     设置寄存器初值
SETB EA;              中断系统开放
SETB ET2;             开放 T2 中断源
SETB TR2;             启动计数器工作
```

2.5

方法一：

进行 100μs 定时，T1 设置在模式 1 的定时器方式，其初值计算如下：

$$a = 2^{16} - \frac{f_{osc}}{12} \times T = 65536 - \frac{12 \times 10^6}{12} \times 100 \times 10^{-6} = 65436 = FF9CH$$

```
        ORG     0000H
        LJMP    MAIN
        ORG     001BH
        LJMP    INT_T1
        ORG     0100H
MAIN:   MOV     SP, #60H;       设置堆栈。
        MOV     TMOD,  #10H;  将 T1 设置为模式 1 定时器方式，内部启动。
        MOV     TH1, #FFH;  设置计数器初值。
        MOV     TL1, #9CH;  设置计数器初值。
        SETB    EA;           中断系统开放。
        SETB    ET1;          开放 T1 中断源。
```

```
        SETB    TR1;        启动计数器工作。
       SJMP $
        ORG     1000H
INT_T1: CPL     P1.0;       每隔 100μs 翻动 P1.0 的电平。
        MOV     TH1, #FFH;  重新装入初值。
        ORL     TL1, #9CH;  重新装入初值。
        RETI;       中断返回。
        END;                                        执行其他程序。
```

方法二：

进行 100μs 定时，T1 设置在模式 2 的定时器方式，其初值计算如下：

$$a = 2^8 - \frac{f_{osc}}{12} \times T = 256 - \frac{12 \times 10^6}{12} \times 100 \times 10^{-6} = 156 = 9CH$$

```
        ORG     0000H
        LJMP    MAIN
        ORG     001BH
        CLR     TF1
        CPL     P1.0;                   每隔 100μs 翻动 P1.0 的电平。
        RETI;                           中断返回。
        ORG     0100H
MAIN:   MOV     SP, #60H;               设置堆栈。
        MOV     TMOD,   #20H;           将 T1 设置为模式 2 定时器方式, 内部启动。
        MOV     TH1, #9CH;              设置计数器初值。
        MOV     TL1, #9CH;              设置计数器初值。
        SETB    EA;                     中断系统开放。
        SETB    ET1;                    开放 T1 中断源。
        SETB    TR1;                    启动计数器工作。
       SJMP $
        END;                            执行其他程序。
```

方法三：

进行 200μs 方波，T2 设置信号发生器方式，其初值计算如下：

$$a = 2^{16} - \frac{f_{osc}}{4 \times f} = 65536 - \frac{12 \times 10^6}{4 \times 5 \times 10^3} = 64936 = FDA8H$$

```
    ORG 0000
    LJMP MAIN
    ORG 0100H
MAIN:
    MOV T2MOD,#00H
    MOV T2CON,#02H
    MOV TH2,#0FDH
    MOV TL2,#0A8H
    MOV RCAP2H,#0FDH
    MOV RCAP2L,#0A8H
    SETB TR2
    SJMP $
```

```
    END
```

2.6

```
counter_H    DATA    30H; 定义软件计数器。
counter_L    DATA    31H; 定义软件计数器。
level_F      BIT     00H;      定义电平标志位。
             ORG     0000H
             LJMP    MAIN
             ORG     000BH;    T0 中断入口。
             LJMP    INT_T0;   转入中断服务子程序。
             ORG     0100H
MAIN:    MOV     SP, #60H;     设置堆栈。
             MOV     TMOD, #01H;   将 T0 设置为模式 1 定时器方式, 内部启动。
             MOV     TH0, #0D8H;   设置计数器初值。
             MOV     TL0, #0F0H;   设置计数器初值。
             MOV     counter_H,#10;  设置软件计数器，初值为 10。
             MOV     counter_L,#90;  设置软件计数器，初值为 90。
             SETB    level_F;      设置信号电平状态标志位，初值为高电平。
             SETB    EA;           中断系统开放。
             SETB    ET0;          开放 T0 中断源。
             SETB  TR0;            启动计数器工作。
             SJMP $
             ORG     1000H
INT_T0:      JNB     level_F,  NEXT;     如果 level_F=0，转入 NEXT。
             DJNZ    counter_H, STOP; 计数器 counter_H 减 1。
             CPL     P1.2;         如果计数器 counter_H 为零, 将 P1.2 取反。
             MOV     counter_H, #20;     计数器 counter_H 归零后重装初值。
             CLR     level_F;      将标志位 level_F 清 0。
             SJMP    STOP;
NEXT:        DJNZ    counter_L, STOP; 计数器 counter_L 减 1。
             CPL     P1.2;         如果计数器 counter_L 为零，将 P1.2 取反。
             MOV     counter_L, #80;     计数器 counter_L 归零后重装初值。
             SETB    level_F;
STOP:        MOV     TH0, #0D8H;     重新装入初值。
             ORL     TL0, #0F0H;   重新装入初值。
             RETI;
             END
```

2.7

P1.2 信号周期为 20ms，频率为 50Hz。

a1=2^16-1000=FC18H—— TH0=0FCH,TL0=18H

```
        a2=2^8-10=F6H——TH1=TL1=0F6H
        ORG 0000H
        LJMP MAIN
        ORG 000BH
        LJMP INT0
        ORG 001BH
        CPL P1.2
        MOV TH1,#0F6H
        MOV TL1,#0F6H
        ORG    0100H
INT0:   CPL P1.1
        MOV TH0,#0FCH
        MOV TL0,#18H
        RETI
        ORG 1000H
MAIN:MOV SP,   #60H
        MOV TMOD,#61H
    MOV TH0,#0FCH
    MOV TL0,#18H
    SETB ET0
    SETB ET1
    SETB EA
    SETB TR0
    MOV TH1,#0F6H
    MOV TL1,#0F6H
    SETB TR1
    SETB ET1
    SJMP $
    END
```

2.8

```
            ORG 0000H
START:    LJMP MAIN
            ORG      001BH
            LJMP     IT1P
            ORG 0100H
MAIN:    MOV SP, #60H
            MOV TMOD, #20H
            MOV TL1, #9CH
            MOV TH1, #9CH
            MOV P1, #0FFH
WAIT:      MOV A, P1
            JB    P1.4, WAIT
            SETB EA
```

```
        SETB ET1
        SETB TR1
        SJMP $
;以下是中断服务子程序
        ORG 1200H
IT1P:   CPL P2.0
        RETI
        END
```

2.9

```
    Flag_1 BIT 00H;          1
    Flag_2 BIT 01H;          2
    ORG 0000H;               3
    LJMP MAIN;               4
    ORG 0013H;               5
    LJMP INT_1;              6
    ORG   001BH;             7
    LJMP INT_T1;             8
    ORG 0040H;               9
MAIN: MOV SP, #80H;          10
    CLR Flag_1;              11
    CLR Flag_2;              12
    MOV TMOD, #60H;          13
    MOV TH1,#0FBH;           14
    MOV TL1, #0FBH;          15
    SETB ET1 ;               16
    SETB TR1 ;               17
    MOV IE, # 84H;           18
    MOV IP, # 00H;           19
WAIT: SJMP WAIT;             20
INT_1: SETB Flag_1;          21
    CLR Flag_2;              22
    RETI;                    23
INT_T1: SETB Flag_2;         24
    CLR Flag_1;              25
    RETI;                    26
    END
```

第七章

2.1

根据题意，T1 初值计算如下：

$a^1=2^8-2^{SMOD}\times f_{osc}/（384\times BR）$

$=2^8-2^1\times 11.0592\times 10^6/（384\times 19000）$

$=253=FDH$

初始化程序如下：

```
CLR TCLK;
CLK RCLK;
MOV    TMOD,  #20H;
MOV    TH1,#0FDH;
MOV    TL1, #0FDH;
ORL PCON,  #80H;
SETB   TR1;
MOV    SCON,  #0D0H;
```

2.2

根据题意，T1 初值计算如下：

$a^1=2^8-2^{SMOD}\times f_{osc}/（384\times BR）$

$=2^8-2^1\times 11.0592\times 10^6/（384\times 4800）$

$=244=F4H$

发送程序：

```
M1T:    PUSH   PSW
        PUSH   ACC
        PUSH   DPL
        MOV    PSW,    #08H
        CLR TCLK
        CLK RCLK
        MOV    TMOD,  #20H
        MOV    TH1,#0F4H
        MOV    TL1, #0F4H
        MOV    SCON,  #40H
        ORL PCON,  #80H
        SETB   TR1
        MOV    DPTR,   #2300H
        MOV    R7,  #64H
LOOP:   MOVX   A,  @DPTR;  PUSH    DPH
        MOV    C,  P
        MOV    ACC.7   C
        MOV    SBUF,   A
        WAIT:   JBC  TI,  NEXT
        SJMP   WAIT
        NEXT:   INC  DPTR
        DJNZ   R7,  LOOP
        POP DPL
```

```
        POP DPH
        POP ACC
        POP PSW
        RET
```
接收程序:
```
M1R:    PUSH    PSW
        PUSH    ACC
        PUSH    DPH
        PUSH    DPL
        MOV     PSW,    #08H
        CLR TCLK
        CLK RCLK
        MOV     TMOD,   #20H
        MOV     TH1,#0F4H
        MOV     TL1, #0F4H
        MOV     SCON,   #50H
        ORL PCON,   #80H
        SETB    TR1
        MOV     DPTR,   #1600H
        MOV     R7,  #64H
WAIT:   JBC  RI,   FETCH
        SJMP    WAIT
FETCH:  MOV     A,   SBUF
        JNB P,   RIGHT
        CLR F0
        RIG: MOVX   @DPTR,A
        INC  DPTR
        DJNZ    R7   WAIT
        SETB    F0;
        SJMP    RETURN
RETURN:  POP    DPL
        POP DPH
```

```
        POP ACC
        POP PSW
        RET
```

2.3

a1=2^8-2^SMOD×fosc/（384×BR） =2^8-2^1×11.0592×10^6/（384×4800）
=244=F4H

发送：

```
          ORG     0100H
M1T:      PUSH    PSW;
          PUSH    ACC;
          PUSH  DPH;
          PUSH DPL;
          MOV     PSW,  #08H;
          CLR    TCLK;
          CLK    RCLK;
          MOV    TMOD,#20H;
          MOV    TH1,#0F4H;
          MOV    TL1,#0F4H;
          MOV    SCON,#40H;
          ORL    PCON,#80H;
          SETB    TR1;
          MOV    DPTR,#2300H;
          MOV    R7,#64H;
LOOP:     MOVX    A,   @DPTR;
          MOV          C,    P   ;
          MOV    ACC.7,    C;
          MOV    SBUF,A;
WAIT:     JBC          TI,    NEXT;
          SJMP    WAIT;
NEXT:     INC            DPTR;
          DJNZ    R7,   LOOP;
          POP     DPL;
          POP     DPH;
          POP    ACC;
          POP    PSW;
          RET;
```

接收：

```
M1R:        PUSH    PSW;
                PUSH    ACC;
                  PUSH    DPH;
                  PUSH      DPL;
                  MOV       PSW,#08H;
                  CLR       TCLK;
```

```
                    CLK         RCLK;
                    MOV       TMOD,#20H;
                    MOV         TH1,#0F4H;
                    MOV       TL1,#0F4H;
                    MOV       SCON,#50H;
                     ORL      PCON,#80H;
                    SETB       TR1;
                    MOV      DPTR,#1600H;
                    MOV         R7,#64H;
WAIT:       JBC           RI,    FETCH;
                     SJMP      WAIT;
FETCH:      MOV    A,    SBUF;
                     JNB          P ,    RIGHT;
                   CLR    F0;
                        SJMP    RETURN;
RIGHT:      MOVX   @DPTR,    A;
                      INC    DPTR;
                      SETB      F0;
                         DJNZ    R7      WAIT;
RETURN:   POP    DPL;
                     POP         DPH;
                     POP        ACC;
                        POP      PSW;
                        RET;
```

2.4

a1=2^8-2^SMOD×fosc/（384×BR）=2^8-2^0×11.0592×10^6/（384×9600）
=253=FDH

发送：

```
                      ORG      0100H
M1T:         PUSH  PSW;
                   PUSH    ACC;
                   PUSH    DPH;
                   PUSH    DPL;
                   MOV    PSW,  #08H;
                   CLR    TCLK;
                   CLK    RCLK;
                   MOV    TMOD,#30H;
                   MOV    TH1,#0FDH;
                   MOV    TL1,#0FDH;
                   MOV    SCON,#0C0H;
                   ANL     PCON,#7FH;
                   SETB    TR1;
                   MOV    DPTR,#1700H;
                   MOV     R6,#00H;
```

```
LOOP:     MOVX   A,  @DPTR;
                 MOV          C,   P   ;
                  CPL         C
              MOV     TB8,   C;
              MOV     SBUF,A;
WAIT:     JBC     TI,   NEXT;
              SJMP   WAIT;
NEXT:     INC           DPTR;
              DJNZ  R6,   LOOP;
              POP  DPL;
              POP  DPH;
              POP  ACC;
              POP  PSW;
              RET;
```

接收：

```
ORG     0100H
M1R:        PUSH   PSW;
                PUSH   ACC;
                PUSH   DPH;
                PUSH   DPL;
                MOV    PSW,#08H;
                CLR    TCLK;
                CLK    RCLK;
                MOV    TMOD,  #20H;
                MOV    TH1,#0FDH;
                MOV    TL1,#0FDH;
                MOV    SCON,#0D0H;
                ANL    PCON,#7FH;
                SETB   TR1;
                MOV    DPTR,  #2400H;
              MOV  R6,  #00H;
WAIT:       JBC    RI,    FETCH;
                SJMP   WAIT;
FETCH:     MOV    A,  SBUF;
                 JNB     P,  NP;
                    JNB     RB8，RIGHT
ERROR:     CLR      F0;
                    SJMP     RETURN；
NP：            JNB      RB8 ，ERROR
RIGHT:    MOVX   @DPTR,   A;
                INC    DPTR;
                SETB   F0;
                   DJNZ   R6   WAIT;
RETURN:  POP    DPL;
```

```
POP     DPH;
POP    ACC;
POP    PSW;
  RET;
```