专业：信息工程
姓名：李坤林
学号：3200101135
日期：11/10/2022
地点：外经贸楼实验室

浙江大学实验报告

课程名称：微机原理与接口技术　指导老师：黄凯　实验名称：内存操作和数制及代码转换

一、实验目的和要求

内存操作：

① 掌握数据传送指令；
② 掌握各种数据传送指令的寻址方式；
③ 熟练运用“伟福 WAVE”环境对汇编程序进行调试。

数制及代码转换：

① 了解微机系统中的数制与代码表示方法；
② 掌握计算机中使用的各种代码转换方法；
③ 掌握实现分支、循环的指令及其程序的编写方法；

二、实验内容和原理

内存操作：

①下列程序的功能是给外部 RAM8000～80FFH 的 256 个单元的内容赋值，赋值的内容取决于程序中 A 的赋值。在 WAVE 环境运行该程序，并观察寄存器及内存单元的变化。

②下列程序将 3000H 起始的 256 个字节存储块移动到 4000H 起始的 256 个字节存储块，在 WAVE 环境运行如下程序，观察寄存器及存储单元的变化。

③在 WAVE 环境运行如下程序，观察寄存器及内存单元的变化，将变化结果注释于 右侧，并说明程序完成什么功能？将程序中 MOV A,@R0 改成 MOVX A,@R0，将 MOV @R1,A 改成 MOVX @R1,A，运行如下程序，观察寄存器及内存单元的变化。

④在 WAVE 环境修改内部 RAM 30H ～3FH 的内容分别为#00H-#0FH，设计程序实现将内部 RAM30H-3FH 到 40H-4FH 的数据块拷贝。

⑤在 WAVE 环境修改内部 RAM 30H ～3FH 的内容分别为#00H-#0FH，设计程序实现将片内 30H～3FH 单元的内容复制到片外 1030H～103FH 中。

⑥在 WAVE 环境修改内部 RAM 30H ～3FH 的内容分别为#00H-#0FH，设计程序实现将内部 RAM30H ～3FH 内容逆序拷贝到外部数据 XRAM：0000H ～000FH 中。使用单步、断点方式调试程序，查看特殊功能寄存器、内部数据 RAM、外部数据空间的变化。

数制及代码转换：

⑦以下程序完成单字节的 ASCII 码到十六进制数转换，完成空白处程序填写，并在 WAVE 环境运行程序，观察寄存器及内存单元的变化。

⑧以下程序完成单字节的 BCD 码到十六进制数转换，,在 WAVE 环境运行程序，观 察寄存器及内存单元的变化。

⑨以下程序将单字节十六进制数 A 的值转换为十进制数，存放在 30H～32H 中，完成 空白处程序填写，并在 WAVE 环境运行程序，观察寄存器及内存单元的变化

⑩ 设一串字母的 ASCII 存于 30H 起始的单元中，设计程序判断字母是否为大写字母， 是则将大写字母的 ASCII 字符转换成小写字母的 ASCII 字符，为小写则不转换。

⑪ 将单字节十六进制数 D8H 转换为十进制数，存放在 30H ~ 33H 中。

⑫ 设计程序，将十六进制数 614EH 转换成 ASCII 码，使用单步、断点方式调试程序， 查看结果。

三、主要仪器设备

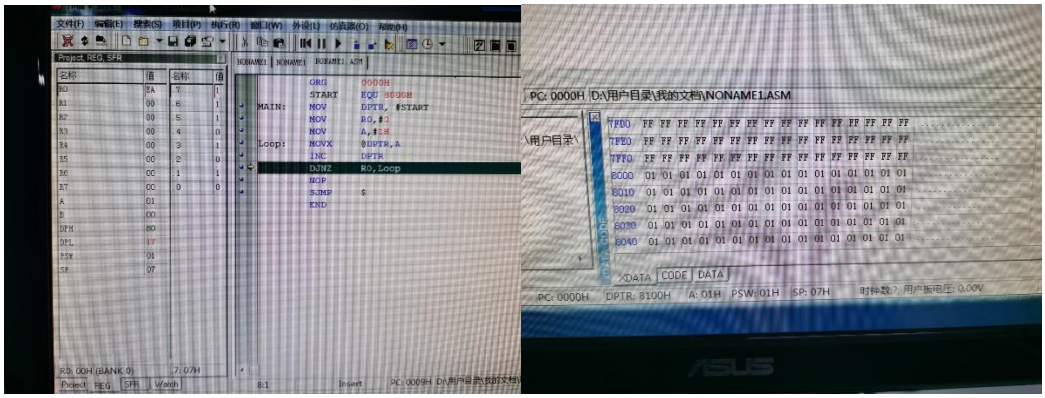
计算机一台

四、操作方法和实验步骤

1. Code1

   按 F8 能进行逐步操作，观察寄存器和内存的变化，R0 和 DPL 一直在变化

   观察到外部 RAM 的 8000~80FFH 的单元被赋值了 01，跟 A 的值一致



2. Code2

   将断点设置在 Loop1 处，全速执行后，由地址 3000-30F0 共 256 个地址被填充为 A 的值 01H（即 01）

3000	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3010	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3020	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3030	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3040	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3050	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3060	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3070	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3080	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3090	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30A0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30B0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30C0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30D0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30E0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30F0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01

   取消断点，全速执行后，地址 4000-40F0 共 256 个地址被填充为地址 3000-30F0 中的值 01

3000	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3010	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3020	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3030	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3040	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3050	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3060	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3070	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3080	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
3090	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30A0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30B0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30C0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30D0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30E0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
30F0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01

4000	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4010	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4020	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4030	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4040	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4050	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4060	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4070	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4080	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
4090	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
40A0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
40B0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
40C0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
40D0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
40E0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
40F0	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01

3. Code3

```
        ORG     0000H
        MOV     R0,#30H
        MOV     R1,#50H
        MOV     R2,#20H
L1:     MOV     A,@R0
        MOV     @R1,A
        INC     R0
        INC     R1
        DJNZ    R2,L1
```

单步执行，三条 MOV 完成 30-R0，50-R1，20-R2，立即数直接进寄存器；

名称	值	名称	值
R0	30	.7	0
R1	50	.6	0
R2	20	.5	1
R3	00	.4	1

```
        ORG     0000H
        MOV     R0,#30H
        MOV     R1,#50H
        MOV     R2,#20H
L1:     MOV     A,@R0
```

间接寻址，R0，R1 不断加；R2 不断自减，自减后 R2 若不为零，跳回 L1 处继续循环，直至 R2 变 0。

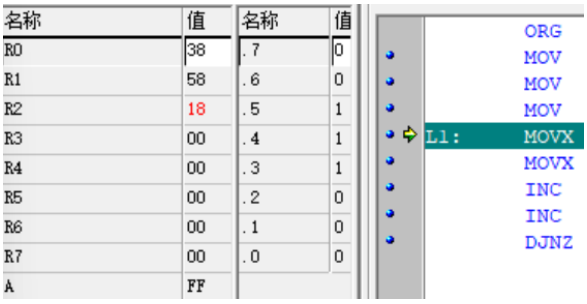
R0	32	.7	0
R1	52	.6	0
R2	1E	.5	1
R3	00	.4	1
R4	00	.3	0
R5	00	.2	0
R6	00	.1	1

```
        MOV     R0,#30H
        MOV     R1,#50H
        MOV     R2,#20H
L1:     MOV     A,@R0
        MOV     @R1,A
        INC     R0
        INC     R1
        DJNZ    R2,L1
```

将程序中 MOV A,@R0 改成 MOVX A,@R0，将 MOV @R1,A 改成 MOVX @R1,A

```
        ORG     0000H
        MOV     R0,#30H
        MOV     R1,#50H
        MOV     R2,#20H
L1:     MOVX    A,@R0
        MOVX    @R1,A
        INC     R0
        INC     R1
        DJNZ    R2,L1
```

前三个 MOV 操作一致，修改为 MOVX 后，累加器 A 与外部 RAM 进行数据传递，所以 A 和外部 RAM 值均可看到变化。



名称	值	名称	值
R0	38	.7	0
R1	58	.6	0
R2	18	.5	1
R3	00	.4	1
R4	00	.3	1
R5	00	.2	0
R6	00	.1	0
R7	00	.0	0
A	FF		

```
        ORG
        MOV
        MOV
        MOV
L1:     MOVX
        MOVX
        INC
        INC
        DJNZ
```

4. Code4

```
MAIN:   MOV     R0,#30H
        MOV     R5,#16
        MOV     A, #0H
LOOP:   MOV     @R0,A
        INC     A
        INC     R0
        DJNZ    R5,LOOP
        MOV     R1,#30H
        MOV     R7,#16
LOOP1:  MOV     A, @R1
        MOV     @R0, A
        INC     R0
        INC     R1
        DJNZ    R7, LOOP1
        SJMP    $
        END
```

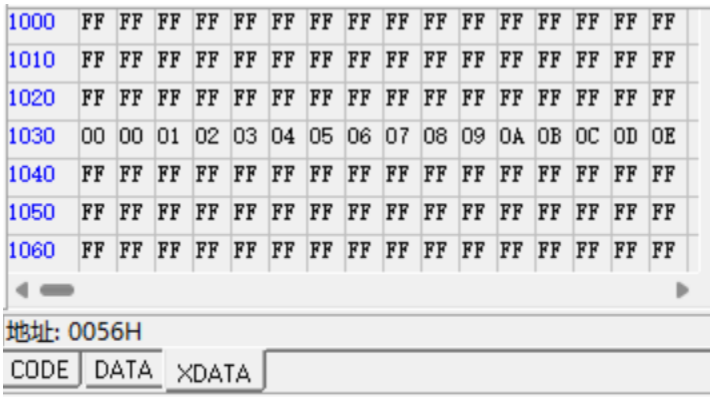
30	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F

仿照上面的例子，进行赋值，右下角查看内部数值，30H-3FH 先被赋值，40H-4FH 紧接着被赋值。

5. Code5

```
LOOP:   MOV     @R0, A
        INC     A
        INC     R0
        DJNZ    R5, LOOP
        MOV     R1, #30H
        MOV     R2, #10H
        MOV     R3, #30H
        MOV     R7, #16
LOOP1:  MOV     A, @R1
        MOV     DPH, R2
        MOV     DPL, R3
        MOVX    @DPTR, A
        INC     R1
        INC     R3
        DJNZ    R7,LOOP1
        SJMP    $
        END
```

30	00	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00



在 XDATA 中可见赋值成功

6. Code6（扩展实验）

采用 Ri 只能寻址到 256 低字节位置，而 DPTR 可以对 XRAM 到 64K。低 256 效果一样，三者都能进行访问。

将 P2 口清零，执行到 MOVX 前，R0、R1、DPTR 立即数赋值

```
;Extended questions
            ORG     0000H
            AJMP    MAIN
            ORG     0100H
MAIN:       MOV     P2, #00H
            MOV     A,  #10H
            MOV     R0, #0FDH
            MOV     R1, #0FEH
            MOV     DPTR, #00FFH
            MOVX    @R0, A
            INC     A
            MOVX    @R1, A
            INC     A
            MOVX    @DPTR,A
            SJMP    $
            END
```

R0	FD
R1	FE
R2	00
R3	00
R4	00
R5	00
R6	00
R7	00
A	10
B	00
DPH	00
DPL	FF
PSW	01
SP	07

三步 MOVX 分别将 A 此时的 10、11、12（三次自增）通过间接寻址传给 XRAM，地址是 0FD、E、F。

Lab2

1. Code1

```
RESULT  EQU     30H
ORG     0000H
MOV     A, #41H
CLR     C
SUBB    A, #37H
MOV     RESULT, A
LJMP    $
END
```

空格内填 ***#37H***

A	0A
B	00
DPH	00
DPL	00
PSW	40
SP	07

30	0A	00	00	00	00
40	00	00	00	00	00
50	00	00	00	00	00
60	00	00	00	00	00

CODE DATA XDATA

2. Code2

```
RESULT  EQU     30H
ORG     0000H
MOV     A, #23H
MOV     R0, A
MOV     A, #0F0H
SWAP    A
MOV     B, #0AH
MUL     AB
MOV     RESULT, A
MOV     A, R0
MOV     B, #0FH
ADD     A, RESULT
MOV     RESULT, A
SJMP    $
END
```

单步执行，R0 变 23，A 变 23；继续单步，A 变 F0；swap 后变 0F；给 B 赋值后 AB 相乘，15×10=150，十六进制是 0096H，A 储存低八位；

00	23	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	B9	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

R0	23
R1	00
R2	00
R3	00
R4	00
R5	00
R6	00
R7	00
A	23
B	00
DPH	00
DPL	00
PSW	01
SP	07

R0	23
R1	00
R2	00
R3	00
R4	00
R5	00
R6	00
R7	00
A	F0
B	00
DPH	00
DPL	00
PSW	00
SP	07

R0	23
R1	00
R2	00
R3	00
R4	00
R5	00
R6	00
R7	00
A	0F
B	00
DPH	00
DPL	00
PSW	00
SP	07

R0	23
R1	00
R2	00
R3	00
R4	00
R5	00
R6	00
R7	00
A	0F
B	0A
DPH	00
DPL	00
PSW	00
SP	07

A	96
B	00
DPH	00
DPL	00
PSW	00
SP	07

A 最后 A+result 就是 23+96 得 B9

A	B9
B	0F
DPH	00
DPL	00
PSW	01
SP	07

30	B9	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00

CODE DATA XDATA

3. Code3

第一空填 100 的 16 进制，DIV AB 后，A 存入百分位 1，B 是余数 23 的十六进制 17H，

A	7B
B	64
DPH	00
DPL	00
PSW	00
SP	07

A	01
B	17
DPH	00
DPL	00
PSW	01
SP	07

A	02
B	03
DPH	00
DPL	00
PSW	01
SP	07

30	01	02	03	00	00	00
40	00	00	00	00	00	00

A 存到地址 30 处，接着把十位个位求出来存入地址

4. Code4

```
;this is for string process
      ORG    0000H
      MOV    30H, #42H
      MOV    31H, #63H
      AJMP   MAIN
      ORG    0100H
MAIN: MOV    R0, #30H
      MOV    R1, #02H
LOOP: MOV    A, @R0
      CLR    C
      SUBB   A, #5BH
      JNC    LOOP1
      MOV    A, @R0
      ADD    A, #20H
      MOV    @R0, A
LOOP1:INC    R0
      DJNZ   R1, LOOP
      SJMP   $
      END
```

30	42	63	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	62	63	00	00	00	00	00	00	00	00	00	00	00	00	00	00

5. Code5

```
;D8H
        RESULT EQU 30H
        ORG    0000H
        AJMP   MAIN
        ORG    0100H
MAIN:   MOV    A, #0D8H
        MOV    B, #64H
        DIV    AB
        MOV    RESULT, #0       ;THOUSAND->0
        MOV    RESULT+1, A      ; /100
        MOV    A, B
        MOV    B, #0AH
        DIV    AB
        MOV    RESULT+2, A      ; /10
        MOV    RESULT+3, B
        SJMP   $
        END
```

30	00	02	01	06	00	00	00	00	00	00	00	00	00	00	00	00

6. Code6

```
;BC614EH
        ORG     0000H
        MOV     R7, #24
        MOV     R1, #0BCH
        MOV     R2, #61H
        MOV     R3, #4EH
        CLR     A
LOOP:   CLR     C
        MOV     A, R3
        RLC     A
        MOV     R3, A
        MOV     A, R2
        RLC     A
        MOV     R2, A
        MOV     A, R1
        RLC     A
        MOV     R1, A
        MOV     R0, #43H
        MOV     R6, #8
LOOP1:  MOV     A, @R0
        ADDC    A, @R0
        DA      A
        MOV     @R0, A
        DEC     R0
        DJNZ    R6,LOOP1
        DJNZ    R7,LOOP
        SJMP    $
        END
```

40	12	34	56	78	00	00	00	00	00	00	00	00	00	00	00	00