# 实验五 控制器部件实验

## 一、实验要求

1、实验之前认真预习，明确实验的目的和具体实验内容，写出实验用到的数据和控制信号的取值，做好实验之前的必要准备。

2、想好实验的操作步骤，明确通过实验到底可以学习哪些知识，想一想怎么样有意识地提高教学实验的真正效果。

3、在教学实验过程中，要爱护教学实验设备和用到的辅助仪表，记录实验步骤中的数据和运算结果，仔细分析遇到的现象与问题，找出解决问题的办法，有意识地提高自己创新思维能力。

4、实验之后认真写出实验报告，重点在于预习时准备的内容，实验数据，运算结果的分析讨论，实验过程、遇到的现象和解决问题的办法，自己的收获体会，对改进教学实验安排的建议等。善于总结和发现问题，写好实验报告是培养实际工作能力非常重要的一个环节，应给予足够的重视。

## 二、实验目的

1. 深入理解计算机控制器的功能、组成知识；
2. 深入地学习计算机各类典型指令的执行流程；
3. 对指令格式、寻址方式、指令系统、指令分类等建立具体的总体概念。

## 三、实验注意事项

控制器设计是学习计算机总体组成和设计的最重要的部分。要在 TEC-XP+教学计算机完成这项实验，必须比较清楚地懂得：

1. TEC-XP+教学机的组合逻辑控制器主要由 MACH 器件组成；
2. TEC-XP+教学机上已实现了 29 条基本指令的控制信号由 MACH 给出的；
3. 应了解监控程序的 A 命令只支持基本指令，扩展指令应用 E 命令将指令代码写入到相应的存储单元中；不能用 T、P 命令单步调试扩展指令，只能用 G 命令执行有扩展指令的程序；
4. 要明白 TEC-XP+教学机支持的指令格式及指令执行流程分组情况；理解 TEC-XP+教学机中已经设计好并正常运行的各类指令的功能、格式和执行流程，也包括控制器设计与实现中的具体线路和控制信号的组成；

5. 要明确自己要实现的指令格式、功能、执行流程设计中必须遵从的约束条件。

## 四、实验内容

1. 完成控制器部件的教学实验，主要内容是由学生观察、验证指令的的功能、格式和执行流程，并在教学计算机上实现、调试正确。

2. 首先是看懂 TEC-XP+教学计算机的功能部件组成和线路逻辑关系，然后分析教学计算机中已经设计好并正常运行的几条典型指令(例如，ADD、SHR、OUT、MVRD、JRC、CALA、RET 等指令)的功能、格式和执行流程。

3. 学习扩展指令的功能、格式和执行流程，并在教学计算机上实现、调试正确。例如 ADC、JRS、JRNS、LDRA、STOR、JMPR 等指令，可以从《TEC-XP+教学计算机系统技术说明》第二章给出的 19 条扩展扩展指令中任意选择。

4. 单条运行指令，查看指令的功能、格式和执行流程。

先将教学机左下方的 5个拨动开关置为11110，再按一下“RESET”按键，然后通过 16 位的数据开关(SW15~SW0)置入指令，按“START”按键单步送脉冲，通过指示灯观察控制信号的变化。

5. 用监控程序的 A、E(扩展指令必须用 E 命令置入)命令编写一段小程序，观察运行结果。

实验时将教学机左下方的 5 个拨动开关置为**00110**，运行编写的小程序。观察终端显示的结果，检验设计的指令是否正确。若与预定结果不符，可查看指令的功能、格式、执行、流程设计的是否正确。

## 五、实验步骤

1.接通教学机电源；

2.将 TEC-XP+教学机左下方的 5 个拨动开关置为 **11110**【单步、手动、硬布线、联机】；

3.按一下“RESET”按键；

4.通过 16 位的数据开关 SW15 ~SW0 置入 16 位的指令操作码；

5.在单步方式下，通过指示灯观察各类基本指令的节拍。

《1》选择基本指令的 A 组指令中的 **ADD** 指令，观察其节拍流程：

【1】置拨动开关 **SW=00000000 00000001**；(表示指令 ADD R0，R1)

【2】按 RESET 按键；节拍指示灯 T3~T0 显示 1000；(本拍在第 1 次复位后才会执行)

【3】按 START 按键；节拍指示灯 T3~T0 显示 0000；(公共节拍，在手动置指令方式下无意义)

【4】按 START 按键；节拍指示灯 T3~T0 显示 0010；(公共节拍，将指令编码写入 IRH、IRL)

【5】按 START 按键；节拍指示灯 T3~T0 显示 0011；(执行 ADD 指令，R0←R0+R1 操作)

可以看到，A 组指令(包括 ADD、SUB、CMP、AND、XOR、SHR、SHL、INC、DEC、TEST、OR、MVRR、JR、JRC、JRNC、JRZ、JRNZ)的执行除公共节拍外，只需一步完成。

6. 单步方式下，通过指示灯观察各类基本指令的控制信号。

《1》选择基本指令的A组指令中的SHR指令，观察其执行过程中控制信号的变化，分析其作用。

【1】置拨动开关SW=0000101100010000；（表示指令SHR R1）【SSHSCI简化为SSI】

【2】先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号如下表：

| 节拍 | 指令 | 编码 | MRW | A | B | SSI | I8-I6 | I5-I3 | I2-0 | SST | DC1 | DC2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000 | | | 100 | 0101 | 0101 | 001 | 011 | 001 | 001 | 000 | 000 | 111 |
| 0000 | | | 100 | 0101 | 0101 | 001 | 010 | 000 | 011 | 000 | 000 | 011 |
| 0010 | | | 001 | 0000 | 0000 | 000 | 001 | 000 | 000 | 000 | 000 | 001 |
| 0011 | SHR | 00001011 | 100 | 0000 | 0001 | 000 | 101 | 000 | 011 | 101 | 000 | 000 |

7. 验证几条扩展指令的控制信号如下表：

【1】选择扩展指令 ADC、STC、JRS、LDRX、STRX 和 JMPR，其节拍和设计的控制信号为：

| 节拍 | 指令 | 编码 | MRW | A | B | SSI | I8-I6 | I5-I3 | I2-0 | SST | DC1 | DC2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000 | | | 100 | 0101 | 0101 | 001 | 011 | 001 | 001 | 000 | 000 | 111 |
| 0000 | | | 100 | 0101 | 0101 | 001 | 010 | 000 | 011 | 000 | 000 | 011 |
| 0010 | | | 001 | 0000 | 0000 | 000 | 001 | 000 | 000 | 000 | 000 | 001 |
| 0011 | ADC | 00100000 | 100 | SR | DR | 010 | 011 | 000 | 001 | 001 | 000 | 000 |
| | JRS | 01100100 | 100 | 0101 | 0101 | 000 | 0S1 | 000 | 101 | 000 | 010 | 000 |
| | STC | 01101101 | 100 | 0000 | 0000 | 000 | 001 | 000 | 000 | 100 | 000 | 000 |
| | JMPR | 01100000 | 100 | SR | 0101 | 000 | 011 | 000 | 100 | 000 | 000 | 000 |
| 0110 | LDRX | 11100101 | 100 | 0101 | 0101 | 001 | 010 | 000 | 011 | 000 | 000 | 011 |
| | STRX | 11100110 | 100 | 0101 | 0101 | 001 | 010 | 000 | 011 | 000 | 000 | 011 |
| 0111 | LDRX | 11100101 | 001 | SR | 0000 | 000 | 001 | 000 | 101 | 000 | 000 | 011 |
| | STRX | 11100110 | 001 | SR | 0000 | 000 | 001 | 000 | 101 | 000 | 000 | 011 |
| 0101 | LDRX | 11100101 | 001 | 0000 | DR | 000 | 011 | 000 | 111 | 000 | 000 | 000 |
| | STRX | 11100110 | 000 | 000 | DR | 000 | 001 | 000 | 011 | 000 | 001 | 000 |

8. 单步方式下，通过指示灯观察上面扩展的几条扩展指令的控制信号是否与设计的一致。

《1》观察 A 组指令中的 **ADC** 指令：

【1】置拨动开关 **SW=00100000 00010000**；

【2】先按“RESET”按键；再连续按“START”按键，观察每一步的节拍及控制信号如下表：

| 节拍 | 指令 | 编码 | MRW | A | B | SSI | I8-I6 | I5-I3 | I2-0 | SST | DC1 | DC2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000 | | | 100 | 0101 | 0101 | 001 | 011 | 001 | 001 | 000 | 000 | 111 |
| 0000 | | | 100 | 0101 | 0101 | 001 | 010 | 000 | 011 | 000 | 000 | 011 |
| 0010 | | | 001 | 0000 | 0000 | 000 | 001 | 000 | 000 | 000 | 000 | 001 |
| 0011 | ADC | 00100000 | 100 | SR | DR | 010 | 011 | 000 | 001 | 001 | 000 | 000 |

9.用教学机已实现的基本指令和扩展的几条指令编写程序并运行，测试扩展的几条指令是否正确。

**《1》测试 ADC 指令:**

在命令行提示符状态下输入：A 2000↙

屏幕将显示：

2000：

从地址 2000H 开始输入下列程序：

2000: MVRD R0,0101 ；给 R0 赋值 0101

2002: MVRD R1,1010 ；给 R1 赋值 1010

2004: ↙

在命令行提示符状态下输入：

A 2006↙

2006: RET

2007: ↙

扩展指令 STC、ADC 不能用 A 命令键入，必须用 E 命令在相应的内存地址键入操作码所有扩展指令都必须用 E 命令键入。

**用 E 命令输入 STC、ADC R0,R1 的代码**，在命令行提示符状态下输入：

E 2004↙

2004：6D00

2005：2001

2006: ↙

用 G 命令运行前面刚键入源程序，在命令行提示符状态下输入：

G 2000↙

用 R 命令察看寄存器的内容，在命令行提示符状态下输入

R↙

运行结果应为 R0=？ R1=？。

**《2》测试 JMPR 指令：**

在命令行提示符状态下输入：

A 2020↙

屏幕将显示：

2020：

从地址 2020 开始输入下列程序：

2020: MVRD R2,000D ；给 R2 赋值 000D，000D 为回车键的 ASCII 码值

2022: IN 81 ；判键盘上是否按了一个键,

2023: SHR R0 ；即串行口是否有了输入的字符

2024: SHR R0

2025: JRNC 2022 ；没有输入则循环测试

2026: IN 80 ；输入字符到 R0 低位字节

2027：MVRD R1，00FF

2029：AND R0，R1 ；清零 R0 的高位字节内容

202A: CMP R0, R2 ；判断输入字符是否为回车

202B: JRZ 2030 ；若是转向程序结束地址

202C: OUT 80 ；若否输出键入字符

202D: MVRD R3，2022

202F：↙

在命令行提示符状态下输入：

A 2030↙

2030: RET

2031: ↙

**用 E 命令输入 JMPR R3 的代码**，在命令行提示符状态下输入：

E 202F↙

202F:6003

2030: ↙

用 G 命令运行前面刚键入源程序，在命令行提示符状态下输入：

G 2020↙

光标闪烁等待键盘输入，若输入非回车字符，则在屏幕上回显；若输入回车字符，则程序执行结束。

**《3》测试 JRS 指令：**

在命令行提示符状态下输入：

A 2100↙

屏幕将显示：

2100：

从地址 2100H 开始输入下列程序：

2100：MVRD R1，0000 ；给 R1 赋值 0000

2102：MVRD R2，4040 ；给 R2 赋值 4040

2104：MVRD R3，01FF ；给 R3 赋值 01FF

2106：ADD R2，R3 ；R2 和 R3 相加

***2107：JRS 210E ；判第一位，若为 1，向后跳 6 个单元**

2108：MVRD R0，0030 ；给 R0 赋字符“0”

210A：OUT 80 ；输出该字符

210B：INC R3 ；R3 加 1

210C：INC R1 ；R1 加 1

210D：JR 2106 ；跳到 2106 循环执行

210E：MVRD R0，0031 ；给 R0 赋字符“1”

2110：OUT 80 ；输出该字符

2111：RET

**注：*表示扩展指令 JRS 应用 E 命令键入**，在命令行提示符状态下输入：

E 2107↙

2107:6406 ；06 为偏移量，该值是要转向的地址值减去 JRS 下一条指令的地址得出的。

用 G 命令运行前面刚键入源程序，在命令行提示符状态下输入：

G 2100↙

屏幕显示字符 0001。

用 R 命令看寄存器的内容，在命令行提示符状态下输入：

R↙

屏幕回显 15 个寄存器的值，其中 R1 的值表示 R3 加 1 的次数。

可改变 R2、R3 的值观察程序运行结果。以加强对该条指令的理解。

**《4》测 LDRX、STRX 指令:**

**例 1：测 LDRX 指令**

1〉在命令行提示符状态下输入：

A 2080

屏幕将显示：

2080：

从地址 2080H 开始输入下列程序：

2080：MVRD R2，2000 ；给寄存器 R2 赋值 2000

***2082：LDRX R1，0020[R2] ；将寄存器 R2 的内容与偏移量相加，相加的和为内存单元 2020，将该单元的内容赋给 R1**

***2084：JMPR R1 ；跳转到寄存器 R1 所示的内存单元**

2085：MVRD R0，0030 ；将字符‘0’的 ASCII 码值赋给 R0

2087：OUT 80 ；输出该字符

2088：RET

2089：↙

**注：扩展指令 LDRX、JMPR 必须用 E 命令键入，形式为：E 2082**

**2082 原值：E512 (空格) 原值：0020(空格)原值：6001↙**

2〉在命令行提示符状态下输入：

E 2020

屏幕将显示：

2020 内存单元原值：-

在光标处输入 2100

3〉在命令行提示符状态下输入：

A 2100

屏幕将显示：

2100：

从地址 2100H 开始输入下列程序：

2100：MVRD R0，0036 ；将字符‘6’的 ASCII 码值赋给 R0

2102：OUT 80 ；输出该字符

2103：RET

2104：↙

4〉在命令行提示符状态下输入：

G 2080↙

屏幕回显数字 6。

**例 2：测 STRX 指令**

1〉在命令行提示符状态下输入：

A 2000

屏幕将显示：

2000：

从地址 2000H 开始输入下列程序：

2000:MVRD R1,6666

2002:MVRD R2,2000

***2004:STRX R1，0080[R2]**

2006:RET

2007: ↙

扩展指令 STRX 是按如下格式输入的:

在命令行提示符状态下输入：

E 2004↙

2004 内存单元原值：E612(空格) 内存单元原值：0060↙

2〉在命令行提示符状态下输入：

G 2000↙ ;执行输入的程序。

3〉在命令行提示符状态下输入：

D 2060↙

可以观察到 2060 内存单元的值为 6666，表明寄存器 R1 中的内容已放入该内存单元，内存单元的值是由寄存器 R2 中的内容和偏移量 0060 相加得到的。

## 六、思考题

1、简述计算机控制器的原理和作用。

附表（p110）：

2.1.1 基本指令汇总表

| 指令格式 | 汇编语言 | 操作数 | CZVS | 类型 | 功能说明 |
|---|---|---|---|---|---|
| 00000000 DRSR | ADD DR, SR | 2 | **** | A | DR←DR+SR |
| 00000001 DRSR | SUB DR, SR | 2 | **** | A | DR←DR-SR |
| 00000010 DRSR | AND DR, SR | 2 | •**• | A | DR←DR and SR |
| 00000011 DRSR | CMP DR, SR | 2 | **** | A | DR-SR |
| 00000100 DRSR | XOR DR, SR | 2 | •**• | A | DR←DR xor SR |
| 00000101 DRSR | TEST DR, SR | 2 | **** | A | DR and SR |
| 00000110 DRSR | OR DR, SR | 2 | •**• | A | DR←DR or SR |
| 00000111 DRSR | MVRR DR, SR | 2 | •••• | A | DR←SR |
| 00001000 DR0000 | DEC DR | 1 | **** | A | DR←DR-1 |
| 00001001 DR0000 | INC DR | 1 | **** | A | DR←DR+1 |
| 00001010 DR0000 | SHL DR | 1 | *••• | A | DR, C←DR*2 |
| 00001011 DR0000 | SHR DR | 1 | *••• | A | DR, C←DR/2 |
| 01000001 OFFSET | JR ADR | 1 | •••• | A | 无条件跳转到 ADR |
| 01000100 OFFSET | JRC ADR | 1 | •••• | A | C=1 时跳转到 ADR |
| 01000101 OFFSET | JRNC ADR | 1 | •••• | A | C=0 时跳转到 ADR |
| 01000110 OFFSET | JRZ ADR | 1 | •••• | A | Z=1 时跳转到 ADR |
| 01000111 OFFSET | JRNZ ADR | 1 | •••• | A | Z=0 时跳转到 ADR |
| 10000000 0000000 | JMPA ADR | 1 | •••• | B | 无条件跳到 ADR |
| ADR(16 位) | | | | | |
| 10000001 DRSR | LDRR DR, [SR] | 2 | •••• | B | DR←[SR] |
| 10000010 I/O PORT | IN I/O PORT | 1 | •••• | B | R0←[I/O PORT] |
| 10000011 DRSR | STRR [DR], SR | 2 | •••• | B | [DR]←SR |
| 10000100 00000000 | PSHF | 0 | •••• | B | FLAG 入栈 |
| 10000101 0000SR | PUSH SR | 1 | •••• | B | SR 入栈 |
| 10000110 I/O PORT | OUT I/O PORT | 1 | •••• | B | [I/O PORT]←R0 |
| 10000111 DR0000 | POP DR | 1 | •••• | B | DR←出栈 |
| 10001000 DR0000 | MVRD DR, DATA | 2 | •••• | B | DR←DATA |
| 10001100 00000000 | POPF | 0 | •••• | B | FLAG←出栈 |
| 10001111 00000000 | RET | 0 | •••• | B | 子程序返回 |
| 11001110 00000000 | CALA ADR | 1 | •••• | D | 调用首地址为 ADR 的子程序 |

注：① 表中 CZVS 一列，* 表示对应的状态位在该指令执行后会被重置；
• 表示对应状态位在该指令执行后不会被修改。

② 运算器芯片中有 16 个通用寄存器(累加器)R0~R15，其中：
R4 用作 16 位的堆栈指针 SP； R5 用作 16 位的程序计数器 PC；
其余寄存器用作通用寄存器，即多数双操作数指令和单操作数指令中的 DR、SR。

附表（p111）

2.1.2 扩展指令汇总表

| 指令格式 | 汇编语言 | 数 | CZVS | 类 | 功能说明 |
|---|---|---|---|---|---|
| 00100001 DRSR | SBB DR, SR | 2 | **** | A | DR←DR-SR-C |
| 00101010 DR0000 | RCL DR | 1 | *••• | A | DR 带进位 C 循环左移 |
| 00101011 DR0000 | RCR DR | 1 | *••• | A | DR 带进位 C 循环右移 |
| 00101100 DR0000 | ASR DR | 1 | *••• | A | DR←DR 算术右移 |
| 00101101 DR0000 | NOT DR | 1 | **** | A | DR←/DR |
| 01100000 0000SR | JMPR SR | 1 | •••• | A | 跳转到 SR 指明的地址 |
| 01100100 OFFSET | JRS ADR | 1 | •••• | A | S=1 时跳转到 ADR |
| 01100101 OFFSET | JRNS ADR | 1 | •••• | A | S=0 时跳转到 ADR |
| 01101100 00000000 | CLC | 0 | 0••• | A | C=0 |
| 01101101 00000000 | STC | 0 | 1••• | A | C=1 |
| 01101110 00000000 | EI | 0 | •••• | A | 开中断, INTE←1 |
| 01101111 00000000 | DI | 0 | •••• | A | 关中断, INTE←0 |
| 11101111 00000000 | IRET | 0 | **** | D | 中断返回 |
| 11100000 0000SR | CALR SR | 1 | •••• | B | 调用 SR 指明的子程序 |
| 11100100 DR0000 | LDRA DR, [ADR] | 2 | •••• | B | DR←[ADR] |
| 11100101 DRSR | LDRX DR, OFFSET[SR] | 2 | •••• | B | DR←[DATA+SR] |
| ADR(16 位) | | | | | |
| 11100110 DRSR | STRX DR, OFFSET[SR] | 2 | •••• | B | [DATA+SR]←SR |
| ADR(16 位) | | | | | |
| 11100111 0000SR | STRA [ADR], SR | 1 | •••• | B | [ADR]←SR |
| ADR(16 位) | | | | | |

注：① 表中 CZVS 一列，* 表示对应的状态位在该指令执行后会被重置；
• 表示对应状态位在该指令执行后不会被修改。

② 扩展指令的功能、格式、操作码和操作数地址字段的确定，留给同学自己设计。表中给出的只是可能的一种选择，但同学们一定要认识到，这里的基本指令和扩展指令共同构成教学计算机的完整的指令系统，彼此需要协调，至少不能有冲突。

附（p5）：

## 表 1.1 TEC-XP16 教学计算机系统教学实验 简明操作卡

一. 微指令格式：

| 下址 | CI | SCC | MRW | I2~I0 | I8~I6 | I5~I3 | B 口 | A 口 | SST | SSH SCI | DC2 | DC1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 4 | 4 | 3 | 3 | 3 | 3 | 4 | 4 | 3 | 3 | 3 | 3 |

二. 控制信号的控制功能：

| CI3~0 | 功能 |
|---|---|
| 0000(0#) | 初始化 |
| 0010(2#) | MAPROM |
| 0011(3#) | 条件微转移 |
| 1110(14#) | 顺序执行 |

/MIO、REQ、/WE

| MRW | 功能 |
|---|---|
| 000 | 内存写 |
| 001 | 内存读 |
| 010 | I/O 写 |
| 011 | I/O 读 |
| 1XX | 无读写 |

| 编码 | I8~6 | | | I5~3 | I2~0 | |
|---|---|---|---|---|---|---|
| | REG | Q | Y | 功能 | R | S |
| 000 | | F→Q | F | R+S | A | Q |
| 001 | | | F | S－R | A | B |
| 010 | F→B | | A | R－S | 0 | Q |
| 011 | F→B | | F | R∨S | 0 | B |
| 100 | F/2→B | Q/2→Q | F | R∧S | 0 | A |
| 101 | F/2→B | | F | /R∧S | D | A |
| 110 | 2F→B | 2Q→Q | F | R⊕S | D | Q |
| 111 | 2F→B | | F | /(R⊕S) | D | 0 |

| SST | C | Z | V | S |
|---|---|---|---|---|
| 000 | C | Z | V | S |
| 001 | CY | F=0 | OVR | F15 |
| 010 | 内部总线 | | | |
| 011 | 0 | Z | V | S |
| 100 | 1 | Z | V | S |
| 101 | RAM0 | Z | V | S |
| 110 | RAM15 | Z | V | S |
| 111 | Q0 | Z | V | S |

| DC2 | 译码信号 | 操作 |
|---|---|---|
| 000 | NC | NC |
| 001 | /GIR | 指令寄存器接收 |
| 010 | /GARL | AR 低位接收 |
| 011 | /GARH | AR 高位接收 |
| 100 | /INTR | 恢复中断优先级 |
| 101 | /INTN | 新中断优先级 |
| 110 | /EI | 开中断，置 INTE 为 1 |
| 111 | /DI | 关中断，清 INTE 为 0 |

| DC1 | 译码信号 | 操作 |
|---|---|---|
| 000 | /SWTOIB | 开关到内部总线 |
| 001 | /RTOIB | ALU 输出到内部总线 |
| 010 | /ETOIB | 16 扩展符号到内部总线 |
| 011 | /FTOIB | 状态到内部总线 |
| 100 | /STOIB | 8 扩展符号 到内部总线 |
| 101 | /INTVH | 中断向量高位到内部总线 |
| 110 | /INTVL | 中断向量低位到内部总线 |

| SSH SCI | Cin / Shift |
|---|---|
| 000 | Cin = 0 |
| 001 | Cin = 1 |
| 010 | Cin = C |
| | |
| 100 | 逻辑移位 |
| 101 | 循环移位 |