

学生学号	0122015710114	实验课成绩	
------	---------------	-------	--

# 武汉理工大学

## 学生实验报告书

实验课程名称	微机原理与通信接口 A
开 课 学 院	信息工程学院
指导教师姓名	张琪
学 生 姓 名	胡姗
学生专业班级	信息 2001

2022    --    2023    学 年    第   二   学 期

## 实验教学管理基本规范

实验是培养学生动手能力、分析解决问题能力的重要环节；实验报告是反映实验教学水平与质量的重要依据。为加强实验过程管理，改革实验成绩考核方法，改善实验教学效果，提高学生质量，特制定实验教学管理基本规范。

- 1、本规范适用于理工科类专业实验课程，文、经、管、计算机类实验课程可根据具体情况参照执行或暂不执行。
- 2、每门实验课程一般会包括许多实验项目，除非常简单的验证演示性实验项目可以不写实验报告外，其他实验项目均应按本格式完成实验报告。
- 3、实验报告应由实验预习、实验过程、结果分析三大部分组成。每部分均在实验成绩中占一定比例。各部分成绩的观测点、考核目标、所占比例可参考附表执行。各专业也可以根据具体情况，调整考核内容和评分标准。
- 4、学生必须在完成实验预习内容的前提下进行实验。教师要在实验过程中抽查学生预习情况，在学生离开实验室前，检查学生实验操作和记录情况，并在实验报告第二部分教师签字栏签名，以确保实验记录的真实性。
- 5、教师应及时评阅学生的实验报告并给出各实验项目成绩，完整保存实验报告。在完成所有实验项目后，教师应按学生姓名将批改好的各实验项目实验报告装订成册，构成该实验课程总报告，按班级交课程承担单位（实验中心或实验室）保管存档。
- 6、实验课程成绩按其类型采取百分制或优、良、中、及格和不及格五级评定。

**附表：实验考核参考内容及标准**

	观测点	考核目标	成绩组成
实验预习	1. 预习报告 2. 提问 3. 对于设计型实验，着重考查设计方案的科学性、可行性和创新性	对实验目的和基本原理的认识程度，对实验方案的设计能力	20%
实验过程	1. 是否按时参加实验 2. 对实验过程的熟悉程度 3. 对基本操作的规范程度 4. 对突发事件的应急处理能力 5. 实验原始记录的完整程度 6. 同学之间的团结协作精神	着重考查学生的实验态度、基本操作技能；严谨的治学态度、团结协作精神	30%
结果分析	1. 所分析结果是否用原始记录数据 2. 计算结果是否正确 3. 实验结果分析是否合理 4. 对于综合实验，各项内容之间是否有分析、比较与判断等	考查学生对实验数据处理和现象分析的能力；对专业知识的综合应用能力；事实求实的精神	50%

实验课程名称： 微机原理与通信接口 A

实验项目名称	实验一、DEBUG 调试			实验成绩	
实 验 者	胡 姗	专业班级	信息 2001	组 别	
同 组 者	无			实验日期	2023 年 5 月 16 日

**第一部分：实验预习报告**（包括实验目的、意义，实验基本原理与方法，主要仪器设备  
备及耗材，实验方案与技术路线等）

### 一、实验目的

1. 在 DEBUG 环境下，通过跟踪调试，理解和熟悉数据传输、数据处理、过程控制、CPU 制指令的功能，了解寻址方式、堆栈等基本概念。
2. 了解计算机的开发环境，理解 DEBUG 的使用方法，熟悉调试环境。

### 二、实验基本原理

DEBUG 是 DOS 操作系统支持的一种系统软件，是 MS 公司献给用户的一个通用软件工具，主要用于 8088/8086 汇编语言程序的调试。它不仅为用户和系统管理员提供了一个可控制的程序调试与开发环境，以便动态地监视管理被调试程序的执行，帮助人们查出程序在逻辑功能上的深层次错误和不完善地方，验证程序的正确性。还提供了一个观察研究窗口，为分析、解剖、开发程序提供了有力的实验手段。因此，熟练地掌握 DEBUG 的使用很必要也很重要。

几种常用的 DEBUG 命令的功能及使用格式：

子命令及其功能	命令格式
Assemble ——对指令语句进行汇编	A[(地址)]
Dump ——显示内存单元内容	D[(地址或地址范围)]
Enter ——显示和修改存储单元内容	E[(地址 字符串)]
Fill —— 向内存区填充数值或字符代码	F[(地址 字符串)]
Load —— 装载文件或扇区数据	L[(地址)]
Perform ——单步执行	P[(地址)]
Register —— 显示修改寄存器及标志位	R[(寄存器名)]
Go ——运行调试的程序	G[(始地址 断点地址)]
Trace —— 跟踪执行单条或多条指令	T[(地址 条数)]
Uassemble ——对指令代码反汇编	U[(地址)]
Quit ——返回 DOS	Q

由 DOS 进入 DEBUG 调试环境。

1. C: \>DEBUG↵ 将调试程序装入内存

注意：当机器控制权由 DOS 成功地转移给调试程序 DEBUG 后，将显示 “—”，它是 DEUBG 的状态提示符，表示可以接受调试子命令了。

2. —R↵ 显示 CPU 中各寄存器当前初始内容

AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000

DS=11D8 ES=11D8 SS=11D8 CS=11D8 IP=0100 NV UP EI PL NZ NA PO NC

11D8:0100 7303 JNB 0105

说明：

(1) 此时，调试工作区的四个段值相同，指向同一起点，表明四个段共用 64KB 空间；

(2) SS: SP 指向堆栈栈顶单元，是本段的最高可用地址，表明堆栈自动由栈顶使用，栈区由底往上（低址方向）生长；

(3) CS: IP 为调试工作区地址，IP 指向将要执行的指令单元（IP 默认值为 0100）；

(4) FLAG 寄存器中八个标志位状态如下表所示

标志位含义	1 对应符号	0 对应符号
OF 溢出	OV 有溢出	NV 无溢出
DF 方向	DN 递减	UP 递增
IF 中断	EI 允许中断	DI 禁止中断
SF 符号	NG 负	PL 正
ZF 全零	ZR 全零	NZ 非零
AF 辅助进位	AC 有半位进位	NA 无半位进位
PF 奇偶	PE 偶	PO 奇
CF 进位	CY 有进位	NC 无进位

(5) 每条指令执行结束，DEBUG 均显示各寄存器内容，便于及时观察结果。

3. 在 “—” 提示符后输入所需 DEBUG 命令进行程序调试。

4. 切换 DOS —Q↵

**第二部分：实验过程记录**（可加页）（包括实验原始数据记录，实验现象记录，实验过程发现的问题等）

### 1、寻址方式

—ACS: 0100

: 0100 DB 02, 4A, 58, 1F, 7D, 31

MOV AL, [0100]

MOV AX, [0101]

MOV SI, 0102

MOV AX, [SI]

MOV BP, 0103

MOV AX, [BP]

MOV SI, 2

MOV AX, 0100[SI]

MOV BX, 4

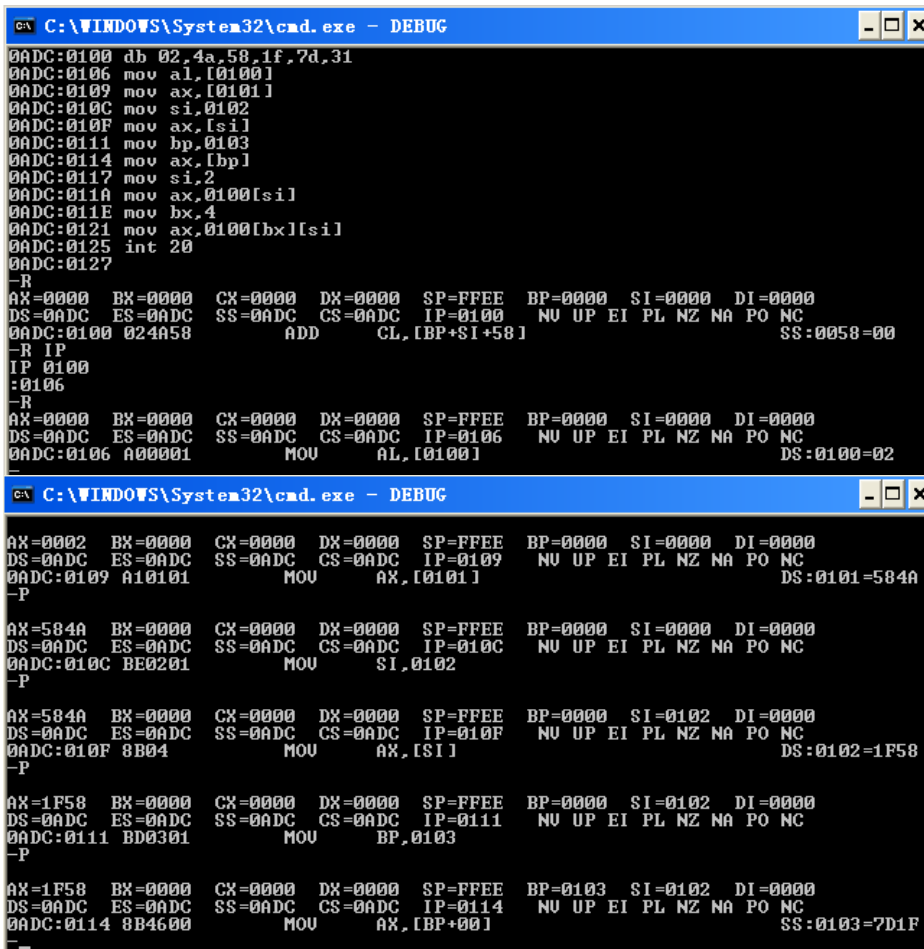
MOV AX, 0100[BX][SI]

INT 20

(1) 程序中寻址方式包括：

直接寻址、立即数寻址、寄存器间接寻址、寄存器相对寻址和相对基址加变址寄存器寻址。

(2) 用 P 命令单步执行指令，记录 AX 的变化；



```
C:\WINDOWS\System32\cmd.exe - DEBUG
0ADC:0100 db 02,4a,58,1f,7d,31
0ADC:0106 mov al,[0100]
0ADC:0109 mov ax,[0101]
0ADC:010C mov si,0102
0ADC:010F mov ax,[si]
0ADC:0111 mov bp,0103
0ADC:0114 mov ax,[bp]
0ADC:0117 mov si,2
0ADC:011A mov ax,0100[si]
0ADC:011E mov bx,4
0ADC:0121 mov ax,0100[bx][si]
0ADC:0125 int 20
0ADC:0127
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0100  NU UP EI PL NZ NA PO NC
0ADC:0100 024A58      ADD     CL,[BP+SI+58]      SS:0058=00
-R IP
IP 0100
:0106
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0106  NU UP EI PL NZ NA PO NC
0ADC:0106 A00001      MOV     AL,[0100]      DS:0100=02
-P
AX=0002 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0109  NU UP EI PL NZ NA PO NC
0ADC:0109 A10101      MOV     AX,[0101]      DS:0101=584A
-P
AX=584A BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=010C  NU UP EI PL NZ NA PO NC
0ADC:010C BE0201      MOV     SI,0102
-P
AX=584A BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0102 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=010F  NU UP EI PL NZ NA PO NC
0ADC:010F 8B04      MOV     AX,[SI]      DS:0102=1F58
-P
AX=1F58 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0102 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0111  NU UP EI PL NZ NA PO NC
0ADC:0111 BD0301      MOV     BP,0103
-P
AX=1F58 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0103 SI=0102 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0114  NU UP EI PL NZ NA PO NC
0ADC:0114 8B4600      MOV     AX,[BP+00]      SS:0103=7D1F
```

```

C:\WINDOWS\System32\cmd.exe - DEBUG
AX=7D1F BX=0000 CX=0000 DX=0000 SP=FFEE BP=0103 SI=0102 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0117
0ADC:0117 BE0200 MOV SI,0002
-P
AX=7D1F BX=0000 CX=0000 DX=0000 SP=FFEE BP=0103 SI=0002 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=011A
0ADC:011A 8B840001 MOV AX,[SI+0100] DS:0102=1F58
-P
AX=1F58 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0103 SI=0002 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=011E
0ADC:011E BB0400 MOV BX,0004
-P
AX=1F58 BX=0004 CX=0000 DX=0000 SP=FFEE BP=0103 SI=0002 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0121
0ADC:0121 8B800001 MOV AX,[BX+SI+0100] DS:0106=00A0
-P
AX=00A0 BX=0004 CX=0000 DX=0000 SP=FFEE BP=0103 SI=0002 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0125
0ADC:0125 CD20 INT 20
-

```

(2) 比较 BX 和 BP 寻址的区别。

BX 寄存器通常用于基址寻址。在基址寻址中，BX 寄存器存储了一个内存段的基地址，通过将 BX 寄存器的值与一个偏移量相加，可以计算出要访问的内存地址。例如，MOV AX, [BX] 指令将寄存器 BX 中的值作为基地址，加载该地址处的内容到 AX 寄存器中。

BP 寄存器通常用于堆栈相关操作，如访问函数的参数和局部变量。在堆栈帧（Stack Frame）中，BP 寄存器常用作堆栈基址指针，指向当前函数的栈帧底部。通过使用 BP 寄存器进行堆栈帧访问，可以轻松访问函数参数、局部变量和其他相关数据。例如，MOV AX, [BP] 指令将 BP 寄存器的值作为基址，加载该地址处的内容到 AX 寄存器中。

## 2、指针传送

—ACS: 0100

```

: 0100 MOV AX, DS
      ADD AX, 0010
      MOV [0202], AX
      LDS SI, [0200]
      MOV AX, [SI]
      ADD AL, AH
      DAA
      INC SI
      INC SI
      MOV AX, [SI]
      ADC AL, AH
      DAA
      INT 20

```

(1) 向 DS: 0200 开始的 11 个单元填充 04, 01, 1F, 27, 38, 81, 17, 25, 79, 28, 76 的指令如下：

—F DS: 0200 0210 04, 01, 1F, 27, 38, 81, 17, 25, 79, 28, 76

(2) 执行 LDS 后, DS= 0AEC , SI= 0104 , 则 LDS 功能是 从指定的存储单元开始, 由 4 个连续的存储单元中取出前两个字节送到偏移地址, 取出后两个字节送到 DS 中。

(3) 用 P 命令单步执行指令:

第一个 DAA 执行前 AL= B9 , 执行后 AL= 19

第二个 DAA 执行前 AL= 3D , 执行后 AL= 43

```

C:\WINDOWS\System32\cmd.exe - DEBUG
AX=0ADC BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0102  NU UP EI PL NZ NA PO NC
0ADC:0102 051000      ADD     AX,0010
-P
AX=0AEC BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0105  NU UP EI PL NZ NA PO NC
0ADC:0105 A30202      MOV     [0202],AX      DS:0202=271F
-P
AX=0AEC BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0108  NU UP EI PL NZ NA PO NC
0ADC:0108 C5360002    LDS     SI,[0200]      DS:0200=0104
-P
AX=0AEC BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0104 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=010C  NU UP EI PL NZ NA PO NC
0ADC:010C 8B04      MOV     AX,[SI]      DS:0104=8138
-P
AX=8138 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0104 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=010E  NU UP EI PL NZ NA PO NC
0ADC:010E 00E0      ADD     AL,AH
-P
AX=81B9 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0104 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=0110  NU UP EI NG NZ NA PO NC
0ADC:0110 27      DAA
-P
AX=8119 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0104 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=0111  NU UP EI PL NZ NA PO CY
0ADC:0111 46      INC     SI
-P
AX=8119 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0105 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=0112  NU UP EI PL NZ NA PE CY
0ADC:0112 46      INC     SI
-P
AX=8119 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0105 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=0112  NU UP EI PL NZ NA PE CY
0ADC:0112 46      INC     SI
-P
AX=8119 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0106 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=0113  NU UP EI PL NZ NA PE CY
0ADC:0113 8B04      MOV     AX,[SI]      DS:0106=2517
-P
AX=2517 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0106 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=0115  NU UP EI PL NZ NA PE CY
0ADC:0115 10E0      ADC     AL,AH
-P
AX=253D BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0106 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=0117  NU UP EI PL NZ NA PO NC
0ADC:0117 27      DAA
-P
AX=2543 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0106 DI=0000
DS=0AEC ES=0ADC SS=0ADC CS=0ADC IP=0118  NU UP EI PL NZ AC PO NC
0ADC:0118 CD20      INT     20

```

### 3、字符串传送、查找

—ACS: 0100

: 0100 DB "IT'S A TEST PROGRAM"

MOV SI, 0100

MOV DI, 0300

MOV CX, 0013

REP MOVSB

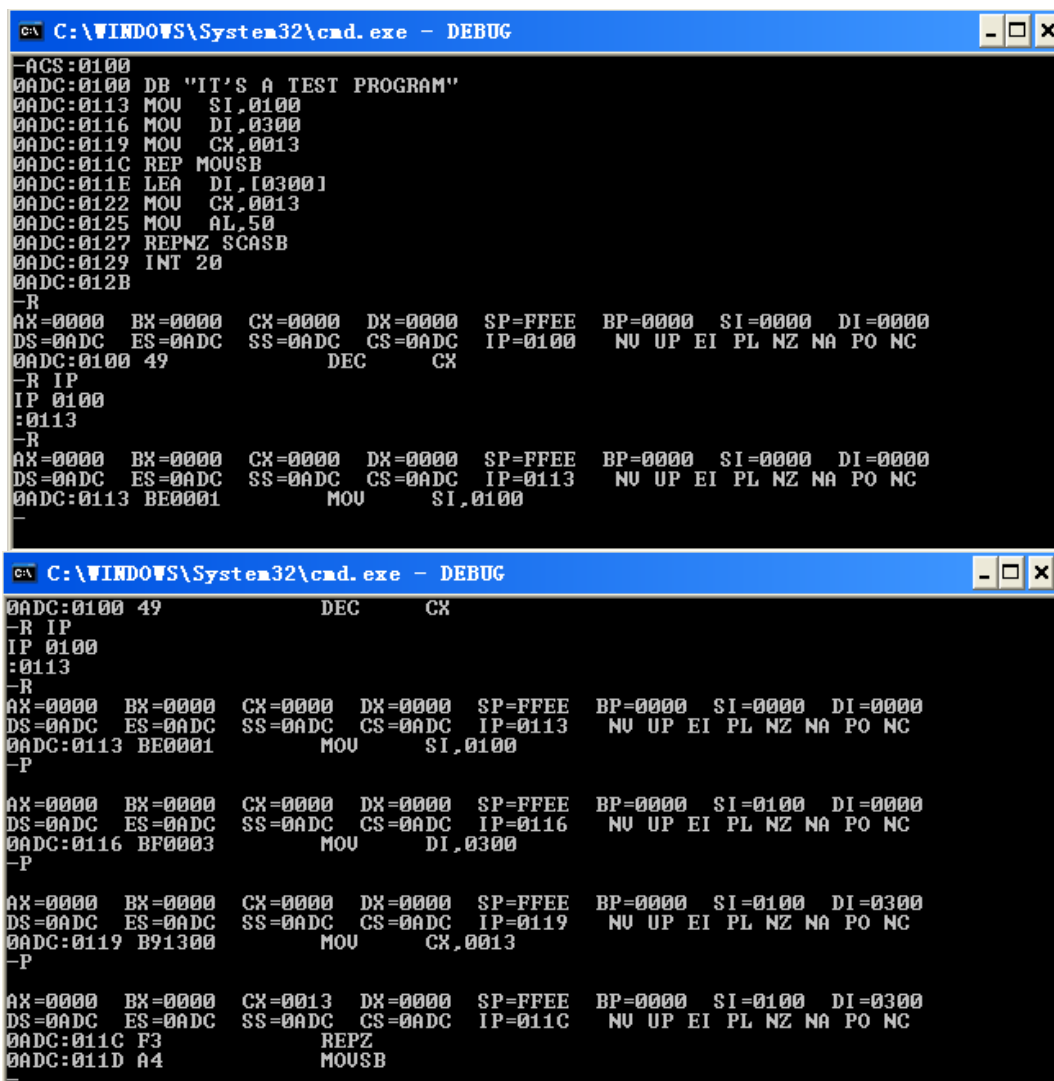
LEA DI, [0300]

MOV CX, 0013

MOV AL, 50

REPNZ SCASB

INT 20



```
C:\WINDOWS\System32\cmd.exe - DEBUG
-ACS:0100
0ADC:0100 DB "IT'S A TEST PROGRAM"
0ADC:0113 MOV SI,0100
0ADC:0116 MOV DI,0300
0ADC:0119 MOV CX,0013
0ADC:011C REP MOVSB
0ADC:011E LEA DI,[0300]
0ADC:0122 MOV CX,0013
0ADC:0125 MOV AL,50
0ADC:0127 REPNZ SCASB
0ADC:0129 INT 20
0ADC:012B
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0100  NU UP EI PL NZ NA PO NC
0ADC:0100 49          DEC     CX
-R IP
IP 0100
:0113
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0113  NU UP EI PL NZ NA PO NC
0ADC:0113 BE0001     MOV     SI,0100
-
0ADC:0100 49          DEC     CX
-R IP
IP 0100
:0113
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0113  NU UP EI PL NZ NA PO NC
0ADC:0113 BE0001     MOV     SI,0100
-P
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0100 DI=0000
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0116  NU UP EI PL NZ NA PO NC
0ADC:0116 BF0003     MOV     DI,0300
-P
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0100 DI=0300
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0119  NU UP EI PL NZ NA PO NC
0ADC:0119 B91300     MOV     CX,0013
-P
AX=0000 BX=0000 CX=0013 DX=0000 SP=FFEE BP=0000 SI=0100 DI=0300
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=011C  NU UP EI PL NZ NA PO NC
0ADC:011C F3          REPZ
0ADC:011D A4          MOUSB
-
```



```

C:\WINDOWS\System32\cmd.exe - DEBUG
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0113 DI=0313
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=011E  NU UP EI PL NZ NA PO NC
0ADC:011E 8D3E0003 LEA DI,[0300] DS:0300=5449
-P
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0113 DI=0300
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0122  NU UP EI PL NZ NA PO NC
0ADC:0122 B91300 MOV CX,0013
-P
AX=0000 BX=0000 CX=0013 DX=0000 SP=FFEE BP=0000 SI=0113 DI=0300
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0125  NU UP EI PL NZ NA PO NC
0ADC:0125 B050 MOV AL,50
-P
AX=0050 BX=0000 CX=0013 DX=0000 SP=FFEE BP=0000 SI=0113 DI=0300
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0127  NU UP EI PL NZ NA PO NC
0ADC:0127 F2 REPNZ
0ADC:0128 AE SCASB
-P
AX=0050 BX=0000 CX=0006 DX=0000 SP=FFEE BP=0000 SI=0113 DI=0300
DS=0ADC ES=0ADC SS=0ADC CS=0ADC IP=0129  NU UP EI PL ZR NA PE NC
0ADC:0129 CD20 INT 20

```

(1) 程序执行首地址 IP= 0113

(2) 程序完成 将字符串从 0100 传送到 0300、查找字符串中 ASCII 码为 0x50 的字符位置  
功能

(3) LEA DI, [0300]等价于 MOV DI, 0300 还是 MOV DI, [0300]?  
等价于 **MOV DI,0300**。

该指令功能为 将 4 位 16 进制偏移地址送到指定的寄存器。

比较 LEA 与 LDS 的差别。

**LEA 直接将操作数的偏移地址装入目的寄存器；LDS 将操作数的 4 个相继字节中的前 2 个装入指定通用寄存器，后 2 个装入 ES 段寄存器。**

(4) 执行完 REPNZ SCASB 后 DI= 030D , CX= 0006

教师签字\_\_\_\_\_

### 第三部分 结果与讨论（可加页）

#### 一、实验结果分析（包括数据处理、实验现象分析、影响因素讨论、综合分析和结论等）

在本次实验中，我使用了 DEBUG 调试器来进行汇编语言的调试和探索。通过这次实验，熟悉了数据传输、数据处理、过程控制和 CPU 指令的功能：DEBUG 提供了一种直观的方式来查看和修改寄存器中的数据以及内存中的内容。通过单步执行指令并观察其结果，深入了解了不同指令的功能和操作。

学习了解到寻址方式和堆栈的基本概念：DEBUG 允许我们在不同寻址方式下查看和修改内存数据，从而理解了直接寻址、寄存器间接寻址、基址寻址等寻址方式的概念和用法。

理解了计算机的开发环境和 DEBUG 的使用方法：DEBUG 是一个简单的调试工具，主要用于汇编语言的调试。

总结而言，本次实验通过 DEBUG 调试器的使用，深入理解了数据传输、数据处理、过程控制和 CPU 指令的功能，同时加深了对寻址方式和堆栈的理解。