**1. 实验目的**

熟悉8086 寻址方式及基本指令的功能，进一步理解和巩固课堂学习内容。

掌握汇编语言程序设计上机过程，掌握汇编语言程序结构，为后续汇编语言程序设计打好基础。

熟悉Microsoft 的 DEBUG 或 Borland 的 Turbo DEBUG 调试工具的使用方法。

**2. 实验内容**

1、读懂下列源程序，使用 EDIT 生成名为 EX11.ASM 的源程序，汇编生成 EX11.OBJ 文件和 EX11.LST 文件，

连接生成 EX11.EXE 文件；用 EDIT 打开 EX11.LST 文件，了解 LST 文件包含的信息；使用 DEBUG 调试工具单步执行 EX11.EXE 程序，注意观察 IP 值的变化，并回答下列问题。

（1）程序读完后，代码段寄存器 CS 的内容为\_\_076B\_\_H，代码段第一条可执行指令

MOV AX, DATA 对应的机器代码为\_\_\_\_\_B06A07H\_\_\_\_\_H，它是\_\_3\_字节指令，单步执行该指令时 IP 值的变化情况，该指令寻址操作数 DATA 的寻址方式为\_立即数\_\_\_寻址方式，

其值为\_\_\_\_\_\_076AH\_\_\_\_。

（2）执行完 MOV DS, AX 指令后，数据段寄存器 DS 的内容为\_076A\_\_H，源程序在数据段中定义的数据

DATA、68H 和 88H 被装入内存单元中的地址分别为：\_\_\_\_\_\_076A\_\_\_\_\_ H，

\_\_\_\_076A1\_\_\_\_\_\_H，\_\_\_076A2\_\_\_\_\_\_\_\_H。

1. 程序中第一条 ADD AL, [BX] 指令对应的机器代码为\_\_\_\_\_0207\_\_\_\_\_H，它是\_2\_\_字节指令，注意观察执行该指令时 IP 值的变化情况；该指令中源操作数的寻址方式为\_寄存器间接\_\_\_寻址方式，

其值为逻辑地址 DS:（BX）为\_\_076A:0001\_\_\_\_\_\_\_\_H，其物理地址为\_\_076A1\_\_\_\_\_\_\_\_H；执行完该指令后，

（AL）=\_\_\_\_\_\_\_\_FA\_\_H，CF=\_0\_\_，OF=\_\_0\_，ZF=\_0\_\_，SF=\_\_1\_，AF=\_0\_\_，

操作数为无符号字节，计算结果是否正确\_\_\_是\_\_\_\_\_；若寄存器操作数为带符号字节，计算结果是否正确\_\_否\_\_。

（4）执行完第二条 ADD AL, [BX] 指令后，数据寄存器（AL）=\_\_\_\_\_72\_\_\_\_\_H，CF=\_\_1\_，OF=\_1\_\_，ZF=\_\_0\_，SF=\_\_0\_，AF=\_1\_\_，PF= 1若寄存器操作数为无符号字节，计算结果

是否正确\_\_\_\_\_\_\_否\_；若寄存器操作数为带符号字节，计算结果是否正确\_\_\_\_\_否\_\_\_。

（5）指令 MOV SUM, AL 的寻址方式：寄存器寻址。该指令执行完后，注意观察（DS:）0003H单元中内容的变化，该单元内容变为\_\_\_\_\_\_72\_\_\_\_H。

DATA SEGMENT

NUM DB 82H, 68H, 88H

SUM DB ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START: MOV AX, DATA

MOV DS, AX

MOV BX, OFFSET NUM

MOV AL, [BX]

INC BX

ADD AL, [BX]

INC BX

ADD AL, [BX]

MOV SUM, AL

MOV AH, 4CH

INT 21H

CODE ENDS

END START

2、读懂下列源程序；编排、平编、连接生成 EX12.ASM、EX12.OBJ、EX12.LST、EX12.EXE 文件；使用 DEBUG 单步执行 EX12.EXE 文件，并回答下列问题。

（1）代码中的前三条指令 MOV AX, STACK、MOV SS, AX 和 MOV SP, LENGTH STL 的功能为堆栈初始化，

其中执行完前三条指令后，堆栈段和堆栈段寄存器中内容逻辑地址分别是：SS:（SP）=\_\_\_\_\_\_076A:0100\_\_\_\_，SP 为\_\_00FE\_\_\_\_\_\_\_\_。

PUSH 指令将寄存器中内容压入堆栈，SP=\_\_\_\_00fE\_\_\_\_\_\_，其内容为\_\_\_\_\_\_\_4AHa05h\_\_\_；

执行完 PUSH BX 指令后，堆栈顶内容为：SS:（SP）=\_\_00feh\_\_\_\_\_\_\_\_，其内容为\_\_\_\_0100\_\_\_\_\_\_。执行完 POP BX 指令后，堆栈顶内容 SS:（SP）=\_\_\_00FEh\_\_\_\_\_\_\_，其内容为\_\_\_\_0100h\_\_\_\_\_\_；

寄存器 BX=\_\_\_\_\_\_0100\_\_\_\_H；

(2)执行完 AND AX, BX 指令后，AX=\_\_\_a7ad\_\_\_\_\_\_\_H，CF=\_0\_\_，OF=\_1\_\_，ZF=\_0\_\_，SF=\_\_1\_，AF=\_0\_\_，PF=0\_\_\_；（注意移 PF 位与计算结果中 8 位有关系）；

若寄存器操作数为无符号数，计算结果是否正确\_\_\_\_是\_\_\_\_；若寄存器操作数为带符号数，计算结果是否正确\_\_\_\_否\_\_\_\_？

（3）执行 AND AX, BX 指令后（AX）=\_\_\_\_4802\_\_\_\_\_\_H，CF=\_0\_\_，OF=\_\_0\_，ZF=\_\_0\_，SF=\_\_0\_，AF=\_0\_\_，PF=\_\_0\_。

STACK SEGMENT stack

STL DW 100H DUP(?)

STACK ENDS

CODE SEGMENT

ASSUME CS:CODE, SS:STACK

START: MOV AX, STACK

MOV SS, AX

MOV SP, LENGTH STL

MOV AX, 0101101100011010B

MOV BX, 0100100100011101B

PUSH AX

PUSH BX

POP AX

POP BX

AND AX, BX

MOV AH, 4CH

INT 21H

CODE ENDS

END START

3、指出下列指令的错误原因，上机运行，观察汇编程序（MASM.EXE）给出的出错信息，改正后再上机验证。

(1) MOV BP, BL  
错误原因：BP 是 16 位寄存器，BL 是 8 位寄存器，位宽不匹配。  
改正：MOV BP, BX 或 MOV BL, AL

(2) MOV [BX],[BP]  
错误原因：两个操作数都是内存地址，指令不允许。  
改正：MOV AX, [BP]  
MOV [BX], AX

(3) MOV [BX], 20H  
错误原因：将立即数送入内存单元，需要明确操作数位宽。  
改正：MOV BYTE PTR [BX], 20H

(4) INC [SI]  
错误原因：对内存操作数 INC，必须指定操作数大小（字节/字）。  
改正：INC BYTE PTR [SI]

(5) ADD AX, [SI+DI]  
错误原因：SI+DI 不是 8086 合法的有效地址计算方式。  
改正：使用 8086 支持的方式，如 ADD AX, [BX+SI]

(6) SHL AX, 3  
错误原因：SHL 指令的移位数为立即数时，只允许 1，若大于 1 必须在 CL 中。  
改正：MOV CL, 3  
SHL AX, CL

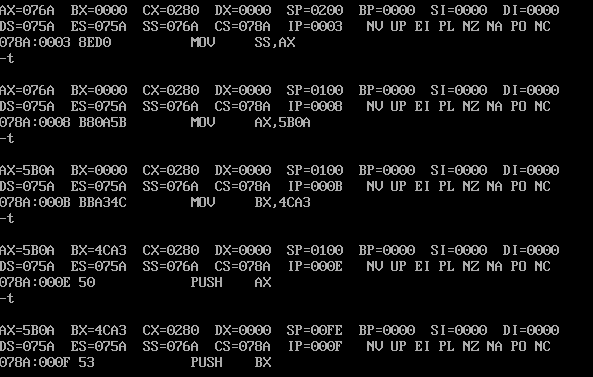
(7) MOV [BX], AL  
错误原因：缺少明确的内存操作数大小声明（有时汇编器需要明确指出）。  
改正：MOV BYTE PTR [BX], AL

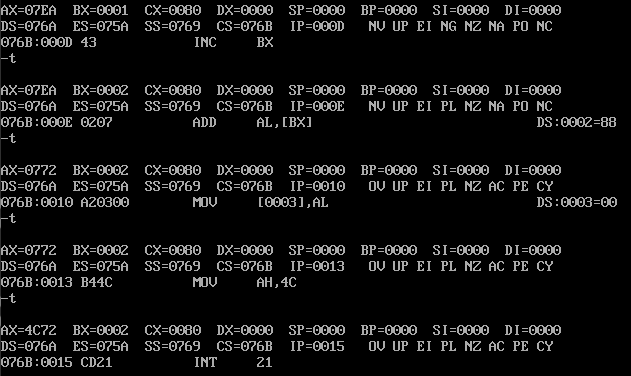
(8) POP AL  
错误原因：POP 只能操作 16 位寄存器或内存单元，不能是 8 位寄存器。  
改正：POP AX

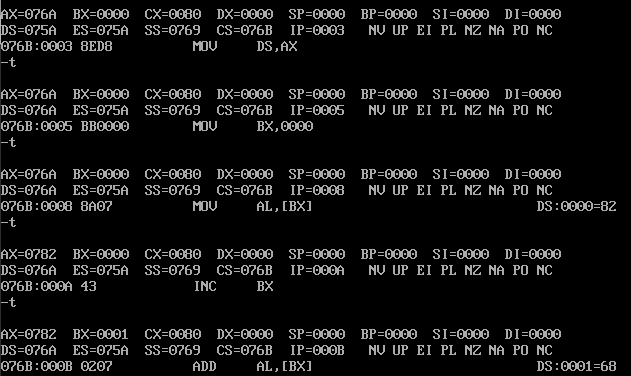
(9) MOV CS, AX  
错误原因：CS 段寄存器不能被直接赋值。  
改正：不能直接赋值，需通过远跳转或调用指令修改 CS。

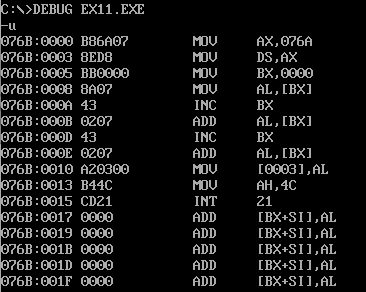
(10) MOV DS, 1000H  
错误原因：立即数 1000H 不能直接送段寄存器，需先送入通用寄存器。  
改正：MOV AX, 1000H  
MOV DS, AX

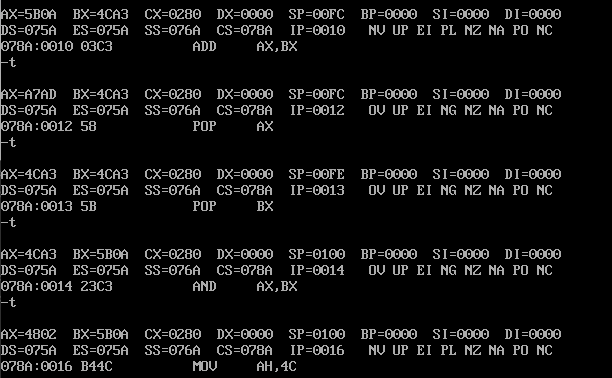
**3.运行结果**











1. **实验总结**

