**作业F** 等级：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学号：\_1929409021\_\_ 姓名:\_\_\_胡志彬\_\_\_\_\_

[F1]设寄存器EAX的内容是75H，在执行指令“CBW”后，寄存器EAX内容是\_0075H\_。接着执行指令“CWD”后，寄存器EAX的内容是\_00000075H\_。

接着执行指令“CWDE”后，寄存器EAX的内容是\_00000075H\_\_\_。

[F2]设寄存器EAX的内容是85H，在执行指令“CBW”后，寄存器EAX内容是FF85H\_\_\_。接着执行指令\_\_MOV BX, AX\_后，寄存器BX的内容是0FF85H。

接着执行指令\_MOVSX EBX, BX\_后，寄存器EBX的内容是0FFFFFF85H。

[F3]设寄存器EAX的内容是83838383H，在执行指令“CWD”后，寄存器EAX内容是83838383H。接着执行指令\_\_MOV CX, AX\_\_后，寄存器ECX的内容是8383H。

接着执行指令\_\_MOV CL, CH后，寄存器ECX的内容是83H。

**[S1]以下指令中，正确的是（）？ C, G, I, J**

（A）MUL AX, 5 （B）DIV 16

（C）NOT BYTE PTR [ECX] （D）AND AL, BX

（E）OR AL, EDX （F）XOR 0FH, AL

（G）TEST DX, 05H （H）SHL EAX, CX

（I）ROR BX, CL （J）IMUL AX, -5

**[S2]以下关于逻辑运算指令的叙述，正确的是（多选）？ A, B, D**

（A）如果有两个操作数，最多只能有一个是存储器操作数。

（B）如果有两个操作数，最多只能有一个是立即数。

（C）如果有两个操作数，最多只能有一个寄存器操作数。

（D）如果有两个操作数，尺寸必须一致。

**[S3]以下关于一般移位指令和循环移位指令的叙述，正确的是（多选）？ B, C, D**

（A）移位的位数只能是1、2、4、8。

（B）移位的位数可以由寄存器CL、CX或ECX指定。

（C）可以左移，也可以右移。

（D）实际移位的位数可以是0。

**[R1]假设执行如下程序片段，请写出每条逻辑运算指令执行后标志ZF、SF和PF的状态：**

MOV AL, 45H

AND AL, 0FH ;ZF/SF/PF = ZF=0（非零）, SF=0（正）, PF=1（偶数个1）

OR AL, 0C3H ;ZF/SF/PF = ZF=0, SF=1（负）, PF=0（奇数个1）

TEST AL, 88H ;ZF/SF/PF = ZF=1（结果为零）, SF=0, PF=1

XOR AL, AL ;ZF/SF/PF = ZF=1, SF=0, PF=1

**[R2]假设执行如下程序片段，请写出每条逻辑运算指令执行后标志CF、ZF、SF和PF的状态：**

MOV AL, 84H

SAR AL, 1 ;CF/ZF/SF/PF =*CF=0, ZF=0, SF=1, PF=0*

SHR AL, 1 ;CF/ZF/SF/PF =*CF=0, ZF=0, SF=0, PF=1*

ROR AL, 1 ;CF/ZF/SF/PF =*CF=1, ZF=0, SF=1, PF=1*

RCL AL, 1 ;CF/ZF/SF/PF =*CF=1, ZF=0, SF=0, PF=1*

SHL AL, 1 ;CF/ZF/SF/PF =*CF=0, ZF=0, SF=1, PF=0*

ROL AL, 1 ;CF/ZF/SF/PF =*CF=1, ZF=0, SF=1, PF=0*

**[R3]请阅读下列代码片段，说明其功能：**

movzx eax, byte ptr [esp+4]

mov ecx, eax

shl eax, 8

add eax, ecx

mov ecx, eax

shl eax, 10h

add eax, ecx

将输入字节（**[esp+4]**）扩展为 **0xXXXXXX** 格式的重复模式。

例如，输入 **0xAB** → 输出 **0xABABAB**。

步骤：

1. **movzx eax, byte ptr [esp+4]**（加载字节到 EAX，零扩展）
2. **shl eax, 8** + **add eax, ecx** → **EAX=AB00H + ABH = ABABH**
3. **shl eax, 16** + **add eax, ecx** → **ABAB0000H + ABABH = ABABABABH**

**[Q1]**指令“MOV AL,0”使寄存器AL清0。请至少写出另外四条可使寄存器AL清0的指令。

清零 AL 的其他指令：

1. **XOR AL, AL**
2. **SUB AL, AL**
3. **AND AL, 0**
4. **MOV AL, [0]**（假设内存地址 0 为 0）

**[Q2]**假设寄存器EBX内容为2。请至少写出四条使寄存器EBX内容为1的指令。

1. **MOV EBX, 1**
2. **XOR EBX, EBX** + **INC EBX**
3. **SUB EBX, EBX** + **ADD EBX, 1**
4. **SHR EBX, 1**（若原 EBX=2）

**[Q3]**请列举标志CF的主要用途。请至少给出使标志CF清0的四种方法。

**CF 用途**：

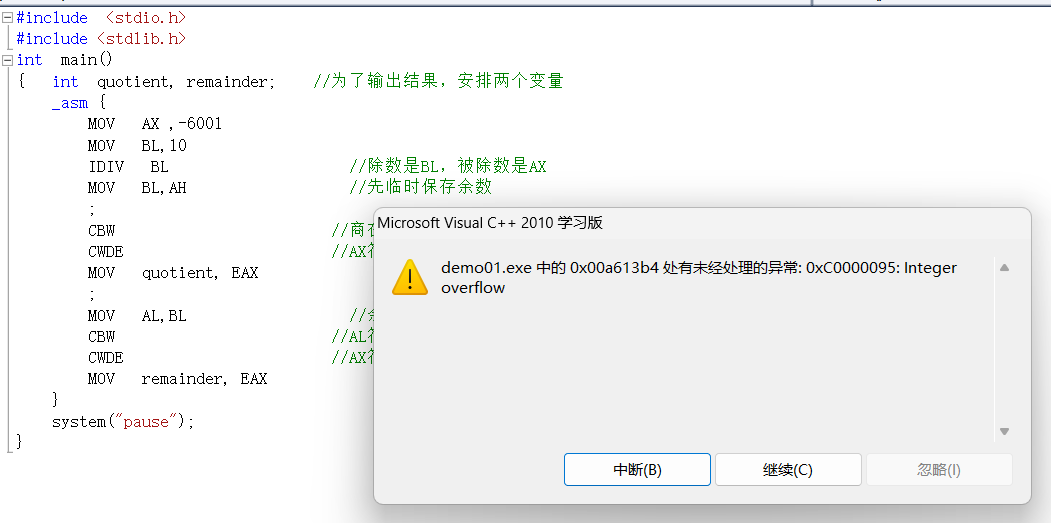
* 无符号数溢出标志（加法/减法）
* 移位/循环移位的进位位
* 比较结果（**CMP**）
* 多精度运算（如 **ADC**）

**清零 CF 的方法**：

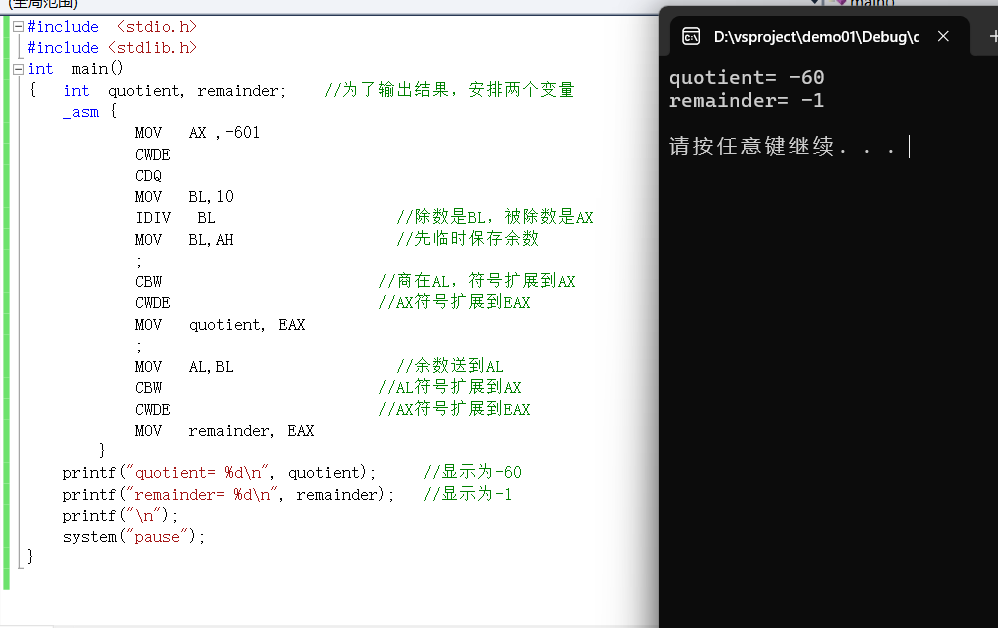
1. **CLC**
2. **AND AL, AL**（不影响 CF） + **CLC**
3. **XOR EAX, EAX**（同时清零 CF）
4. **TEST EAX, EAX**（不影响 CF） + **CLC**

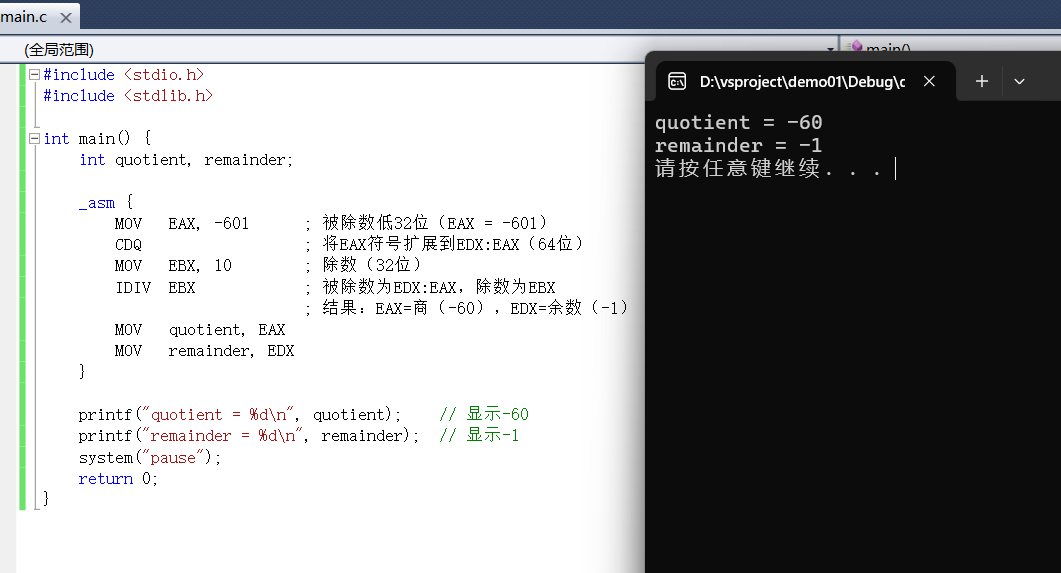
**[Q4]**参考3.2节的例9，（1）修改被除数的大小为-6001，观察除法操作溢出的结果；（2）尝试把被除数扩展到64位，观察除法操作的结果。请分别给出屏幕截图。

（1）



（2）





**[Q5]**参考3.2节的例12，删除函数cf311中的强制类型转换，然后编译生成目标代码，比较分析之。

unsigned int cf311(unsigned char x, unsigned char y) {

003A1360 push ebp

003A1361 mov ebp,esp

003A1363 sub esp,0C0h

003A1369 push ebx

003A136A push esi

003A136B push edi

003A136C lea edi,[ebp-0C0h]

003A1372 mov ecx,30h

003A1377 mov eax,0CCCCCCCCh

003A137C rep stos dword ptr es:[edi]

return (x + 22) / y; // 无强制类型转换

003A137E movzx eax,byte ptr [x]

003A1382 add eax,16h

003A1385 movzx ecx,byte ptr [y]

003A1389 cdq

003A138A idiv eax,ecx

}

003A138C pop edi

003A138D pop esi

003A138E pop ebx

003A138F mov esp,ebp

003A1391 pop ebp

003A1392 ret

unsigned int cf311(unsigned char x, unsigned char y) {

00211360 push ebp

00211361 mov ebp,esp

00211363 sub esp,0C0h

00211369 push ebx

0021136A push esi

0021136B push edi

0021136C lea edi,[ebp-0C0h]

00211372 mov ecx,30h

00211377 mov eax,0CCCCCCCCh

0021137C rep stos dword ptr es:[edi]

return (unsigned)(x + 22) / y; // 有强制类型转换

0021137E movzx eax,byte ptr [x]

00211382 add eax,16h

00211385 movzx ecx,byte ptr [y]

00211389 xor edx,edx

0021138B div eax,ecx

}

0021138D pop edi

0021138E pop esi

0021138F pop ebx

00211390 mov esp,ebp

00211392 pop ebp

00211393 ret

**两段汇编代码的对比分析**

有强制类型转换：0021138B div eax,ecx

无强制类型转换：003A138A idiv eax,ecx

**div（带强制转换）**：严格按无符号除法计算，逻辑正确。

**idiv（无强制转换）**：因编译器隐式处理符号性，可能产生意外行为（如 **cdq** 误用）。