

《微机原理实验》实验报告

实验名称: 多位 16 进制加法运算实验 指导教师: 肖山林
 姓名: 徐睿琳 学号: 23342107 专业/班级: 微电子/三班 分组序号: A412-A05
 实验日期: 2025.11.13 实验地点: 教学楼 A412 是否调课/补课: 否 成绩: _____

目录

1 实验内容与设计	2
1.1 多位十六进制加法运算实验	2
1.1.1 实验电路图	2
1.1.2 实验设计	2
1.2 多位十六进制减法运算实验	3
1.2.1 实验设计	3
1.3 多位十六进制乘法运算实验	3
1.3.1 实验设计	3
2 实验结果	4
2.1 多位十六进制加法运算实验	4
2.1.1 基础实验	4
2.1.2 扩展一: 观察进位 (CF) 和零 (ZF)	6
2.1.3 扩展二: 观察溢出 (OF) 和符号 (SF)	7
2.1.4 扩展三: 观察 OF 和 CF 同时置位	7
2.1.5 扩展四: 实现 ADC (带进位加法)	8
2.1.6 扩展五: 实现 INC (不影响 CF)	8
2.2 多位十六进制减法运算实验	8
2.2.1 基础实验	8
2.2.2 扩展一: SUB 指令, 相减为零、借位情况	9
2.2.3 扩展二: SUB 溢出情况	11
2.2.4 扩展三: 掌握 SBB 指令	12
2.3 多位十六进制乘法运算实验	13
2.3.1 基础实验	13
2.3.2 扩展一: IMUL (有符号乘法)	13
3 思考与总结	14
附录 A 实验汇编代码	14

1 实验内容与设计

1.1 多位十六进制加法运算实验

1.1.1 实验电路图

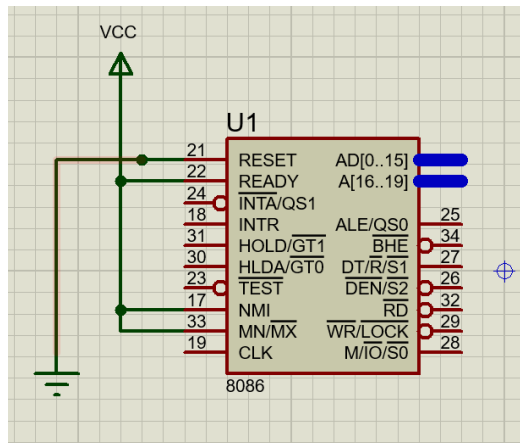


Figure 1: 实验电路图

1.1.2 实验设计

实验	实验目的	DATA 段数据	CODE 段指令	预期标志位结果
基础实验	观察“无特殊情况”的加法	NUM1 DW 1111H	ADD AX, [SI+0]	AX = 6666H
		NUM2 DW 2222H	ADD AX, [SI+2]	CF=0, ZF=0
		NUM3 DW 3333H	ADD AX, [SI+4]	OF=0, SF=0
扩展一	观察进位 (CF) 和零 (ZF)	NUM1 DW 0F000H	ADD AX, [SI+4]	AX = 0000H
		NUM2 DW 00FFFFH		CF = 1 (进位)
		NUM3 DW 00001H		ZF = 1 (结果为零)
扩展二	观察溢出 (OF) 和符号 (SF)	NUM1 DW 6000H	ADD AX, [SI+4]	AX = C000H
		NUM2 DW 0000H		OF = 1 (溢出)
		NUM3 DW 6000H		SF = 1 (结果为负)
扩展三	观察 OF 和 CF 同时为 1	NUM1 DW 8000H	ADD AX, [SI+4]	AX = 0000H
		NUM2 DW 0000H		CF = 1 (无符号进位)
		NUM3 DW 8000H		OF = 1 (有符号溢出) ZF = 1
扩展四	实现 ADC (带进位加法)	NUM1 DW 0FFFFH	ADD AX, [SI+2]	ADD 后 CF=1
		NUM2 DW 00001H	ADC AX, [SI+4]	ADC 后 AX = 0002H
		NUM3 DW 00001H		(因为 0+1+CF)
扩展五	实现 INC (不影响 CF)	NUM1 DW 0FFFFH	MOV AX, [SI+0]	AX = 0000H
			INC AX	ZF = 1
				CF = 0 (或保持不变)

1.2 多位十六进制减法运算实验

1.2.1 实验设计

实验	实验目的	DATA 段数据	CODE 段指令	预期标志位结果
基础实验	实现基础 SUB 指令，无特殊情况	N1 DW 3333H	MOV AX, [N1]	AX = 2222H
		N2 DW 1111H	SUB AX, [N2]	CF=0 (无借位) ZF=0, SF=0, OF=0
扩展一	SUB 指令，相减为零、借位情况	N1 DW 5555H	MOV AX, [N1]	第一次 SUB 后:
		N2 DW 5555H	SUB AX, [N2]	AX = 0000H
		N3 DW 0001H	SUB AX, [N3]	ZF = 1 (零标志) 第二次 SUB 后: AX = FFFFH CF = 1 (借位标志)
扩展二	SUB 溢出情况	N1 DW 8000H	MOV AX, [N1]	(负 - 正 = 正)
		N2 DW 0001H	SUB AX, [N2]	AX = 7FFFH OF = 1 (溢出标志) SF = 0 (结果为正) CF = 0 (无借位)
扩展三	掌握 SBB 指令	N1 DW 1000H	MOV AX, [N1]	SUB 后: CF = 1
		N2 DW 2000H	SUB AX, [N2]	SBB 后:
		N3 DW 0001H	SBB AX, [N3]	AX = AX-N3-CF AX = F000-1-1 AX = EFFE H

1.3 多位十六进制乘法运算实验

1.3.1 实验设计

实验名称	实验目的	DATA 段数据	CODE 段指令	预期标志位/寄存器结果
基础实验	MUL (16 位)	N1 DW 8000H	MOV AX, [N1]	(DX:AX = AX * BX)
		N2 DW 0010H	MOV BX, [N2]	AX = 0000H
			MUL BX	DX = 0008H DX != 0, 因此: CF = 1, OF = 1
扩展一	IMUL (16 位)	N1 DW 4000H (+16384)	MOV AX, [N1]	(DX:AX = AX * BX)
		N2 DW 0003H (+3)	MOV BX, [N2]	AX = C000H
			IMUL BX	DX = 0000H (结果 +49152 无法存入 AX) CF = 1, OF = 1

2 实验结果

2.1 多位十六进制加法运算实验

2.1.1 基础实验

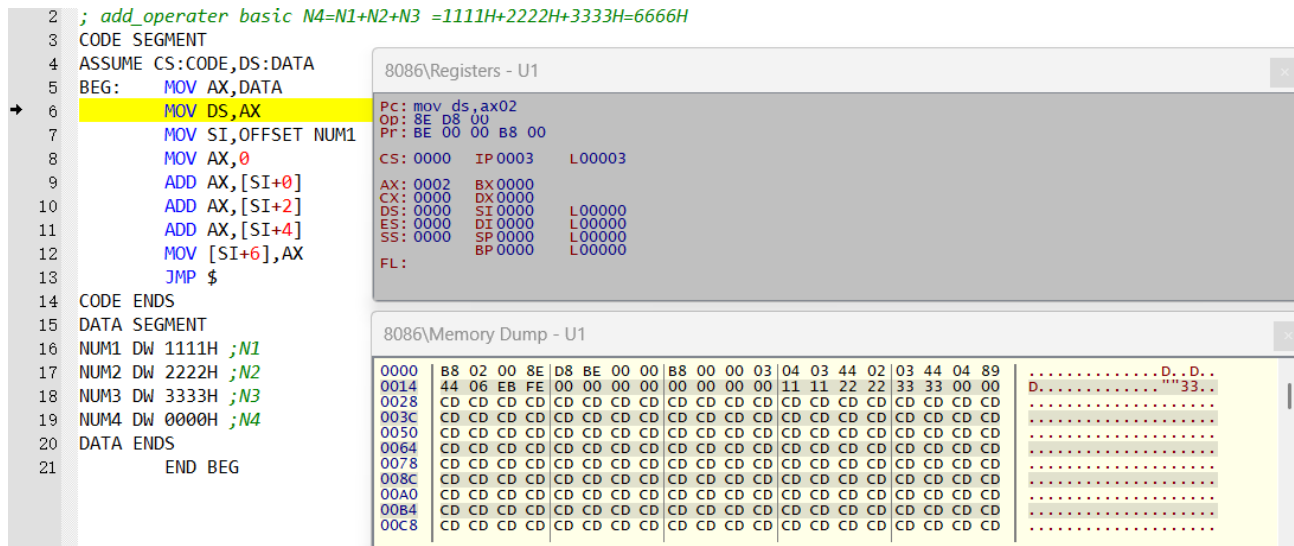


Figure 2: MOV AX, DATA 后断点

程序刚刚执行完指令 `MOV AX, DATA`, `DATA` 是数据段的段地址, 汇编器分配给它的值是 `0002H`。AX 已经准备好了数据段的起始地址 `0002H`, 但地址尚未交付给 `DS` 寄存器。需要下一步指令: `MOV DS,AX` 将 AX 中的 `0002H` 移入 `DS`, 从而建立起正确的数据段寻址环境。

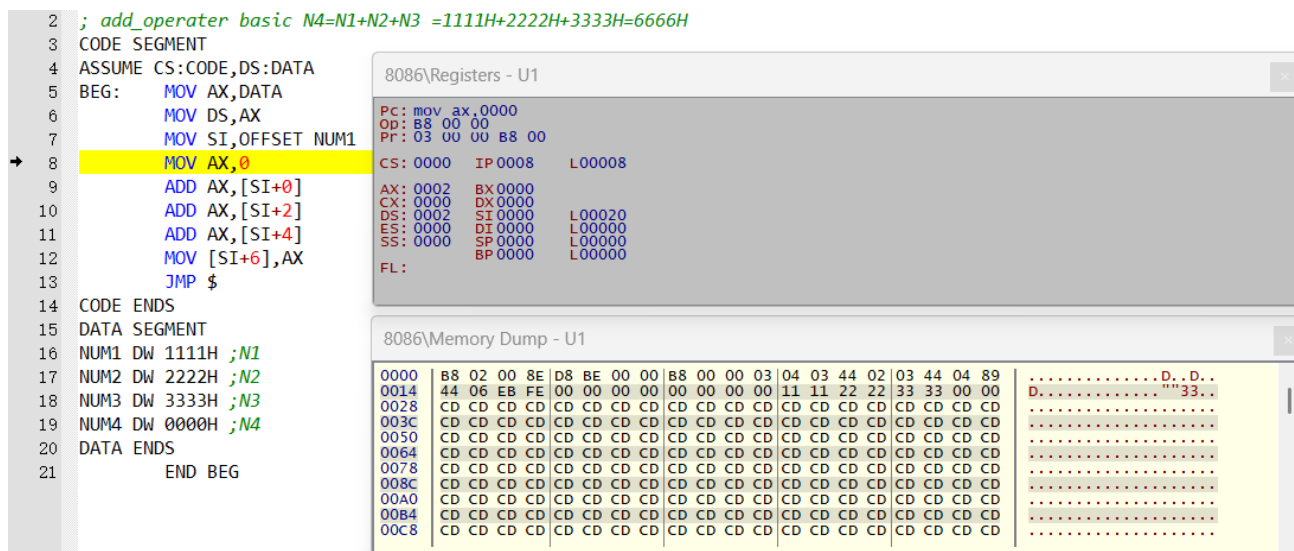
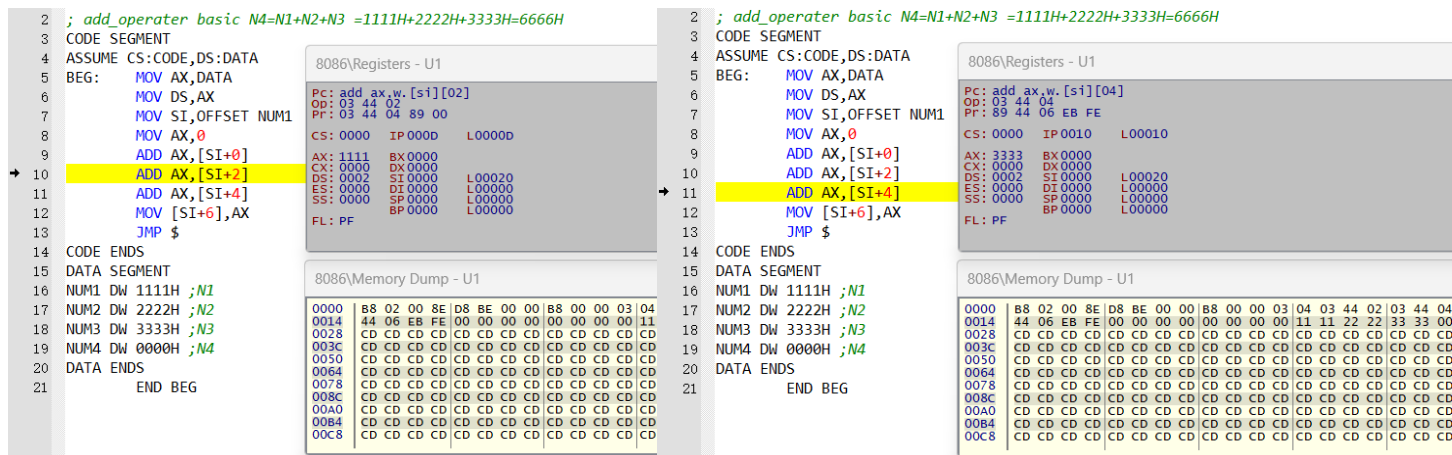


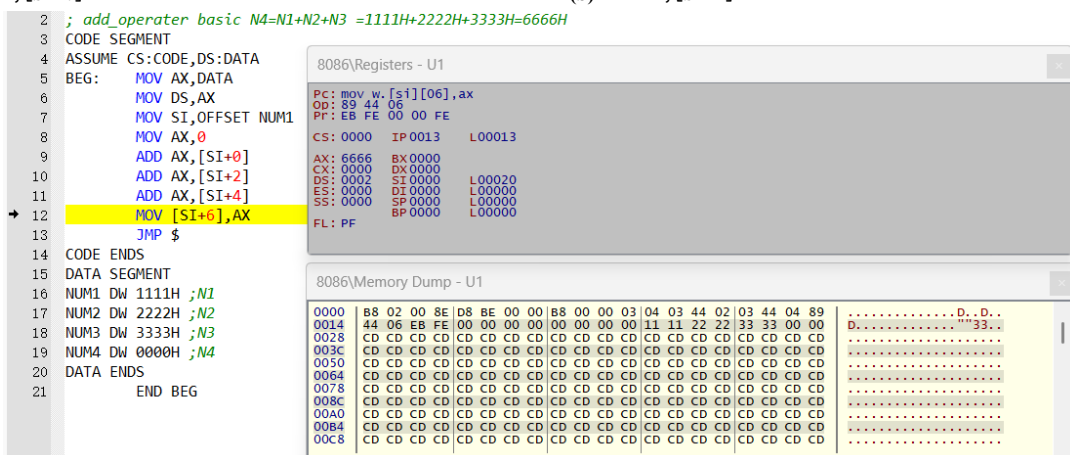
Figure 3: MOV SI,OFFSET NUM1 后断点

`SI` 寄存器现在存储了变量 `NUM1` 在数据段内的起始偏移地址 `0002H`, 这为后续通过 `DS:SI` 访问数据做好了源指针准备。



(a) ADD AX, [SI+0]

(b) ADD AX, [SI+1]



(c) ADD AX, [SI+2]

Figure 4: 加法运算过程

如图可以看到 AX 寄存器的数值逐步累加，最终结果为 6666H，符合预期。同时，标志寄存器中的 PF 被置位，表示结果为偶数个 1，这与 6666H 的二进制表示相符。

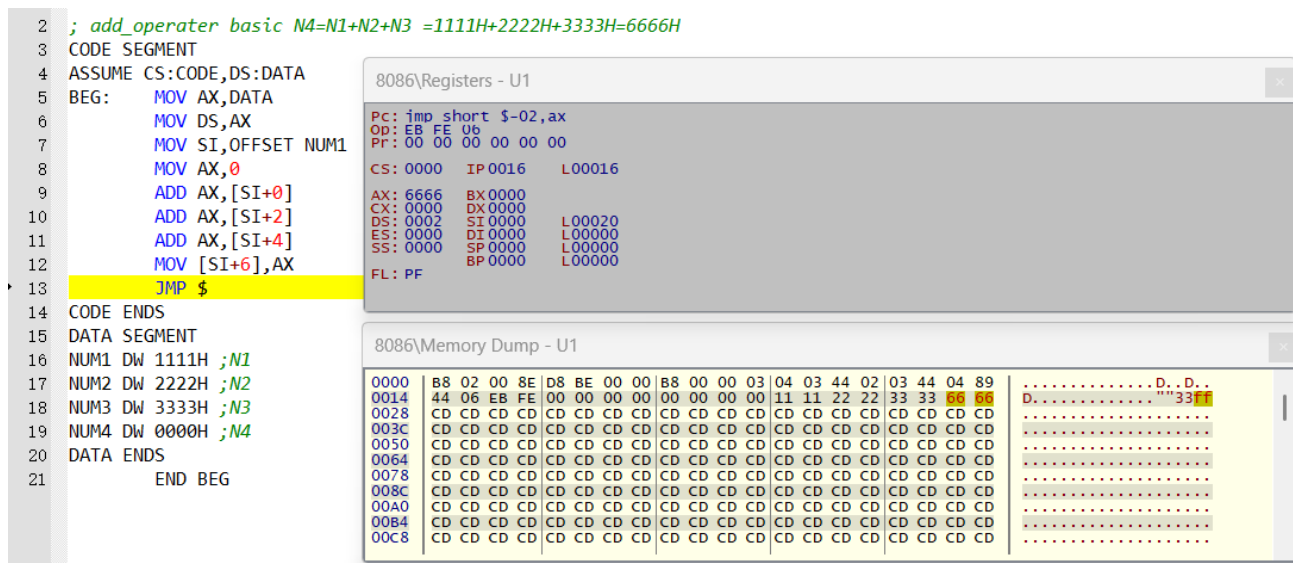


Figure 5: MOV [SI+6], AX 后断点

如图可以看到最终结果 6666H 已经成功存储在变量 NUM4 对应的内存地址中，验证了加法运算的正确性。

2.1.2 扩展一：观察进位 (CF) 和零 (ZF)

注意：由于部分扩展实验为课后本地实验，Debug.exe 无法直接运行，因此以下扩展实验的结果均通过 8086 Emulator 仿真软件进行验证，截图均来自该软件。且由于运行环境不同，部分代码与预期代码略有差异，但均实现了相同的功能。

Reg	H	L	Segments		Pointers	
A	ff	ff	SS	0000	SP	0000
B	0f	ff	DS	0000	BP	0000
C	00	00	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	1	0	0	1	0

Figure 6: F000H + OFFFH 后断点

如图可以看到 SF 和 PF 被置位，表示结果为负数且二进制表示中有偶数个 1。

Reg	H	L	Segments		Pointers	
A	00	00	SS	0000	SP	0000
B	00	01	DS	0000	BP	0000
C	00	00	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	0	1	1	1	1

Figure 7: FFFFH + 0001H 后断点

如图可以看到 ZF,PF,CF 被置位，表示结果为零，且发生了进位，符合预期。而且 AF（辅助进位标志）也被置位，表示低四位发生了进位。

2.1.3 扩展二：观察溢出 (OF) 和符号 (SF)

Reg	H	L	Segments		Pointers	
A	c0	00	SS	0000	SP	0000
B	00	00	DS	0000	BP	0000
C	60	00	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
1	0	0	0	1	0	0	1	0

Figure 8: 6000H + 6000H 后断点

如图可以看到 OF,SF,PF 被置位，表示结果为负数且二进制表示中有偶数个 1。并且发生了溢出，符合预期。

2.1.4 扩展三：观察 OF 和 CF 同时置位

Reg	H	L	Segments		Pointers	
A	00	00	SS	0000	SP	0000
B	00	00	DS	0000	BP	0000
C	80	00	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
1	0	0	0	0	1	0	1	1

Figure 9: 8000H + 8000H 后断点

如图可以看到 OF, ZF,PF,CF 被置位，表示结果为零，且同时发生了进位和溢出，符合预期。

2.1.5 扩展四：实现 ADC (带进位加法)

Reg	H	L	Segments		Pointers	
A	10	01	SS	0000	SP	0000
B	00	01	DS	0000	BP	0000
C	00	01	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	0	0	0	0	0

Figure 10: ADC 后断点

2.1.6 扩展五：实现 INC (不影响 CF)

Reg	H	L	Segments		Pointers	
A	00	00	SS	0000	SP	0000
B	00	01	DS	0000	BP	0000
C	00	01	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	0	1	1	1	1

Figure 11: ADC 后断点

如图，和之前 FFFFH + 0001H 的结果一致。

2.2 多位十六进制减法运算实验

2.2.1 基础实验

除减法运算外其余操作与加法运算类似，以下仅展示减法运算过程截图。



Figure 12: SUB AX, [SI+0] 后断点

如图可见，最终减法结果为 0000H，使 ZF 和 PF 置位，符合预期。

2.2.2 扩展一：SUB 指令，相减为零、借位情况

Reg	H	L	Segments		Pointers	
A	55	55	SS	0000	SP	0000
B	55	55	DS	0000	BP	0000
C	00	01	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	0	0	0	0	0

Figure 13: 相减之前的寄存器状态

Reg	H	L	Segments		Pointers	
A	00	00	SS	0000	SP	0000
B	55	55	DS	0000	BP	0000
C	00	01	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	0	1	0	1	0

Figure 14: AX - BX 后断点（相减为 0 的情况）

Reg	H	L	Segments		Pointers	
A	ff	ff	SS	0000	SP	0000
B	55	55	DS	0000	BP	0000
C	00	01	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	1	0	1	1	1

Figure 15: AX - CX 后断点（即 0-1，产生借位）

如图可见，最终结果为 FFFFH，使 CF, SF, PF 置位，符合预期。

2.2.3 扩展二：SUB 溢出情况

Reg	H	L	Segments		Pointers	
A	80	00	SS	0000	SP	0000
B	00	01	DS	0000	BP	0000
C	00	00	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	0	0	0	0	0

Figure 16: 相减之前的寄存器状态

Reg	H	L	Segments		Pointers	
A	7f	ff	SS	0000	SP	0000
B	00	01	DS	0000	BP	0000
C	00	00	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
1	0	0	0	0	0	1	1	0

Figure 17: 8000H - 0001H 后断点

如图可见，我们通过使用最小的负数 8000H 减去正数 0001H，得到了结果 7FFFH，这个结果是一个正数。然而，由于 8000H 代表的是 -32768，而 0001H 代表的是 +1，因此实际的数学运算是 $-32768 - 1 = -32769$ 。这个结果超出了 16 位有符号整数的表示范围（-32768 到 +32767），因此发生了溢出，导致 OF 标志被置位。同时，结果 7FFFH 是一个正数，所以 SF 标志被清零，符合预期。

2.2.4 扩展三：掌握 SBB 指令

Reg	H	L	Segments		Pointers	
A	10	00	SS	0000	SP	0000
B	20	00	DS	0000	BP	0000
C	00	01	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	0	0	0	0	0

Figure 18: 运算之前的寄存器状态 (AX,BX,CX)

Reg	H	L	Segments		Pointers	
A	f0	00	SS	0000	SP	0000
B	20	00	DS	0000	BP	0000
C	00	01	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	1	0	0	1	1

Figure 19: 通过 1000H - 2000H 产生借位

Reg	H	L	Segments		Pointers	
A	ef	fe	SS	0000	SP	0000
B	20	00	DS	0000	BP	0000
C	00	01	ES	0000	SI	0000
D	00	00			DI	0000

Flags:								
OF	DF	IF	TF	SF	ZF	AF	PF	CF
0	0	0	0	1	0	1	0	0

Figure 20: 利用上一步的结果 (F000H) 使用 SBB 指令减去 0001H (包含借位的计算)

本次实验结果 AX 为 $EFFEH$ ，准确证明了‘SBB’指令的作用，即 $F000H - 0001H - CF(1)$ 的结果；标志位显示 $SF = 1$ ($EFFEH$ 为负数) 且 $CF = 0$ ，表明运算成功使用了前一步的借位，但最终没有产生新的借位。

2.3 多位十六进制乘法运算实验

2.3.1 基础实验

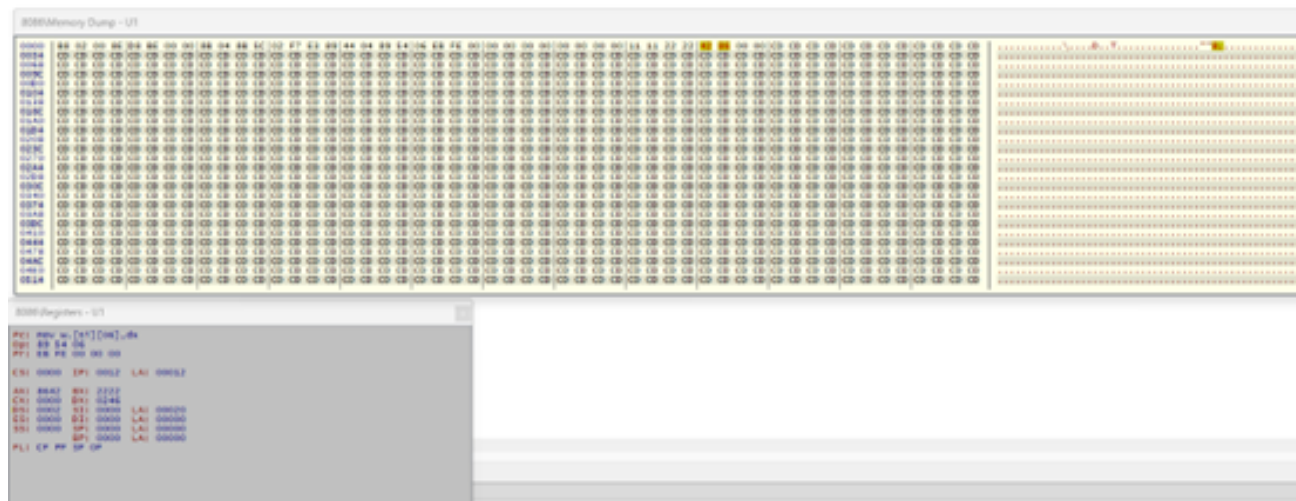


Figure 21: 乘法运算结果

经验算，乘法结果正确，其中，结果低 4 位被存储在 AX 中，高 4 位存储在 DX 中。由于结果超出 16 位， DX 不为 0，因此 CF 和 OF 被置位，符合预期。

2.3.2 扩展一：IMUL（有符号乘法）

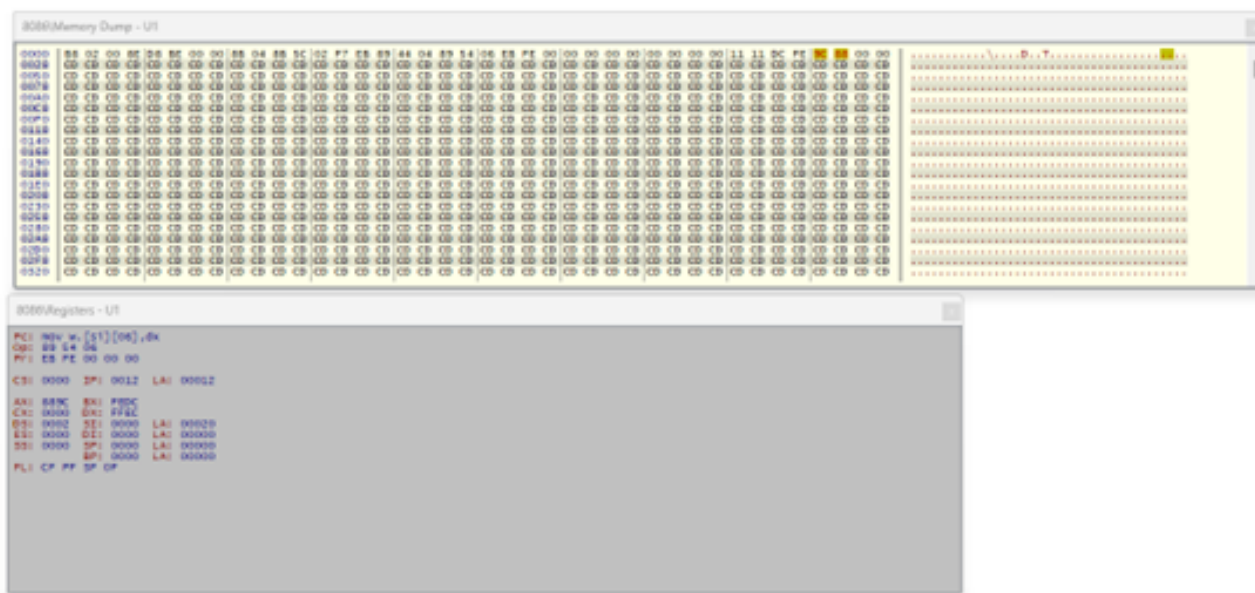


Figure 22: 有符号乘法运算结果

3 思考与总结

在本次实验中，我们通过多个实验步骤深入理解了 16 位十六进制数的加法、减法、乘法运算及其标志位的变化规律。通过使用汇编语言编写程序并进行调试，我们验证了各种运算指令的行为，并观察到了标志位（如 CF、ZF、SF、PF、OF）在不同情况下的变化。

在加法运算实验中，我们首先进行了基础实验，成功实现了三个 16 位十六进制数的加法，并将结果存储在指定内存位置。随后，通过扩展实验，我们观察了进位（CF）和零（ZF）标志位的变化，验证了当结果超过 16 位时 CF 被置位，以及当结果为零时 ZF 被置位的情况。此外，我们还探讨了溢出（OF）和符号（SF）标志位的行为，特别是在加法结果超出有符号数范围时 OF 被置位的现象。

在减法运算实验中，我们同样从基础实验开始，成功实现了两个 16 位十六进制数的减法。通过扩展实验，我们观察了借位（CF）和零（ZF）标志位的变化，验证了当被减数小于减数时 CF 被置位的情况。同时，我们还探讨了溢出（OF）标志位在减法中的行为，特别是在有符号数范围内进行减法时 OF 被置位的现象。此外，我们学习了 SBB 指令的使用，理解了如何在有借位情况下进行减法运算。

在乘法运算实验中，我们通过基础实验实现了 16 位有符号数的乘法，并观察了结果存储在 AX 和 DX 寄存器中的情况。通过扩展实验，我们验证了当乘法结果超出 16 位时 CF 和 OF 标志位被置位的现象。

附录 实验汇编代码

```

1  CODE SEGMENT
2  ASSUME CS:CODE,DS:DATA
3  BEG:  MOV AX,DATA
4        MOV DS,AX
5        MOV SI,OFFSET NUM1
6        MOV AX,0
7        ADD AX,[SI+0]
8        ADD AX,[SI+2]
9        ADD AX,[SI+4]
10       MOV [SI+6],AX
11       JMP $
12 CODE ENDS
13 DATA SEGMENT
14 NUM1 DW 1111H ;N1
15 NUM2 DW 2222H ;N2
16 NUM3 DW 3333H ;N3
17 NUM4 DW 0000H ;N4
18 DATA ENDS
19 END BEG

```

Listing 1: 加法运算-基础实验

```

1  OPR1: DW 0x0000
2  OPR2: DW 0xF000
3  OPR3: DW 0x0FFF
4  OPR4: DW 0x0001
5  RESULT: DW 0
6
7  start:
8  MOV AX, word OPR1
9  MOV BX, word OPR2
10 CLC
11 ADD AX, BX
12
13 MOV BX, word OPR3
14 ADD AX, BX
15

```

```
16 MOV BX, word OPR4
17 ADD AX, BX
18
19 MOV DI, OFFSET RESULT
20 MOV word [DI], AX
21 print reg
```

Listing 2: 加法运算-扩展一

```
1 OPR1: DW 0x6000
2 OPR2: DW 0x0000
3 OPR3: DW 0x6000
4 RESULT: DW 0
5
6 start:
7 MOV AX, word OPR1
8 MOV BX, word OPR2
9 MOV CX, word OPR3
10
11 CLC
12 ADD AX, BX
13 ADD AX, CX
14
15 MOV DI, OFFSET RESULT
16 MOV word [DI], AX
17
18 print reg
```

Listing 3: 加法运算-扩展二

```
1 CODE SEGMENT
2 ASSUME CS:CODE
3 BEG:
4     MOV AX, 0
5     ADD AX, 8000H
6     ADD AX, 8000H
7     JMP $
8 CODE ENDS
9     END BEG
```

Listing 4: 加法运算-扩展三

```
1 CODE SEGMENT
2 ASSUME CS:CODE
3 BEG:
4     MOV AX, FFFFH
5     ADD AX, 0001H
6     ADC AX, 0001H
7     JMP $
8 CODE ENDS
9     END BEG
```

Listing 5: 加法运算-扩展四

```
1 CODE SEGMENT
2 ASSUME CS:CODE
3 BEG:
4     MOV AX, FFFFH
5     INC AX
6     JMP $
7 CODE ENDS
```

```
8 | END BEG
```

Listing 6: 加法运算-扩展五

```
1 | CODE SEGMENT
2 | ASSUME CS:CODE,DS:DATA
3 | BEG: MOV AX,DATA
4 |     MOV DS,AX
5 |     MOV SI,OFFSET NUM1
6 |     MOV AX,6666H
7 |     SUB AX,[SI+0]
8 |     SUB AX,[SI+2]
9 |     SUB AX,[SI+4]
10 |    MOV [SI+6],AX
11 |    JMP $
12 | CODE ENDS
13 | DATA SEGMENT
14 | NUM1 DW 1111H ;N1
15 | NUM2 DW 2222H ;N2
16 | NUM3 DW 3333H ;N3
17 | NUM4 DW 0000H ;N4
18 | DATA ENDS
19 | END BEG
```

Listing 7: 减法运算-基础实验

```
1 | OPR1: DW 0x5555
2 | OPR2: DW 0x5555
3 | OPR3: DW 0x0001
4 | RESULT: DW 0
5 |
6 | start:
7 | MOV AX, word OPR1
8 | MOV BX, word OPR2
9 | MOV CX, word OPR3
10 |
11 | CLC
12 | SUB AX, BX           ; AX=0000H, ZF=1
13 | SUB AX, CX           ; AX=FFFFH, CF=1
14 |
15 | MOV DI, OFFSET RESULT
16 | MOV word [DI], AX
17 |
18 | print reg
```

Listing 8: 减法运算-扩展一

```
1 | OPR1: DW 0x8000
2 | OPR2: DW 0x0001
3 | RESULT: DW 0
4 |
5 | start:
6 | MOV AX, word OPR1
7 | MOV BX, word OPR2
8 |
9 | CLC
10 | SUB AX, BX
11 |
12 | MOV DI, OFFSET RESULT
13 | MOV word [DI], AX
14 |
15 | print reg
```

Listing 9: 减法运算-扩展二

```

1  OPR1: DW 0x1000
2  OPR2: DW 0x2000
3  OPR3: DW 0x0001
4  RESULT: DW 0
5
6  start:
7  MOV AX, word OPR1
8  MOV BX, word OPR2
9  MOV CX, word OPR3
10
11 CLC
12 SUB AX, BX           ; 产生CF=1
13
14 SBB AX, CX           ; AX = F000H - 0001H - 1 = EFFEh
15
16 MOV DI, OFFSET RESULT
17 MOV word [DI], AX
18
19 print reg

```

Listing 10: 减法运算-扩展三

```

1  CODE SEGMENT
2  ASSUME CS:CODE,DS:DATA
3  BEG:  MOV AX,DATA
4        MOV DS,AX
5        MOV SI,OFFSET NUM1
6        MOV AX,[SI+0]
7        MOV BX,[SI+2]
8        MUL BX
9        MOV [SI+4],AX
10       MOV [SI+6],DX
11       JMP $
12 CODE ENDS
13 DATA SEGMENT
14 NUM1 DW 1111H ;N1
15 NUM2 DW 2222H ;N2
16 NUM3 DW 0000H ;N3
17 NUM4 DW 0000H ;N4
18 DATA ENDS
19 END BEG

```

Listing 11: 乘法运算-基础实验

```

1  CODE SEGMENT
2  ASSUME CS:CODE,DS:DATA
3  BEG:  MOV AX,DATA
4        MOV DS,AX
5        MOV SI,OFFSET NUM1
6        MOV AX,[SI+0]
7        MOV BX,[SI+2]
8        IMUL BX
9        MOV [SI+4],AX
10       MOV [SI+6],DX
11       JMP $
12 CODE ENDS
13 DATA SEGMENT
14 NUM1 DW 1111H ;N1

```

```
15 NUM2 DW 0FEDCH ;N2
16 NUM3 DW 0000H ;N3
17 NUM4 DW 0000H ;N4
18 DATA ENDS
19     END BEG
```

Listing 12: 乘法运算-扩展一