

安徽大学人工智能学院

实验报告



课程名称: 《计算机组成原理与汇编语言》

专 业: 人工智能

学 号: WA2214014

姓 名: 杨跃浙

指导老师: 杜库

实验项目	实验 7-第七次上机实验			实验次序	07
实验地点	笃行南楼 A104	参与人员	杨跃浙	实验日期	05.22

一、实验目的

学习标志寄存器 FLAG (也称为 psw, 程序状态字)。

8086/8088 CPU 中有一个 16 位的标志寄存器,包含了 9 个标志,主要用于反映处理器的状态和运算结果的某些特征。各标志在标志寄存器中的位置如下所示。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				O	D	I	T	S	Z		A		P		C
				F	F	F	F	F	F		F		F		F

汇编编程时的条件转移语句(如 jz, jnz, jb, jg 等)就是根据标志位的状态来判断的。其中标志位状态的含义可见下表 (除 TF 标志位) :

标志名		标志为 1	标志为 0
OF	溢出 (是/否)	OV	NV
DF	方向 (减量/增量)	DN	UP
IF	中断 (允许/关闭)	EI	DI
SF	符号 (负/正)	NG	PL
ZF	零 (是/否)	ZR	NZ
AF	辅助进位 (是/否)	AC	NA
PF	奇偶 (偶/奇)	PE	PO
CF	进位 (是/否)	CY	NC

本次实验了解 OF, SF, ZF, CF 等标志位。

OF :溢出标志。运算时超过了机器的表示范围,则标志为 1,表示溢出。例如 2

个正数相加,结果符号位为 1 (负数)则溢出;两个负数相加,结果符号位为 0 (正数)则溢出。

SF:符号位。运算后的结果最高位 (符号位) 为 1,则 SF 为 1,否则为 0。

ZF:零标志位。运算结果为 0,则 ZF 为 1

CF :进位标志。运算时最高位有进位或借位,则 CF 为 1。

二、实验环境

Windows 2011, DOSBox

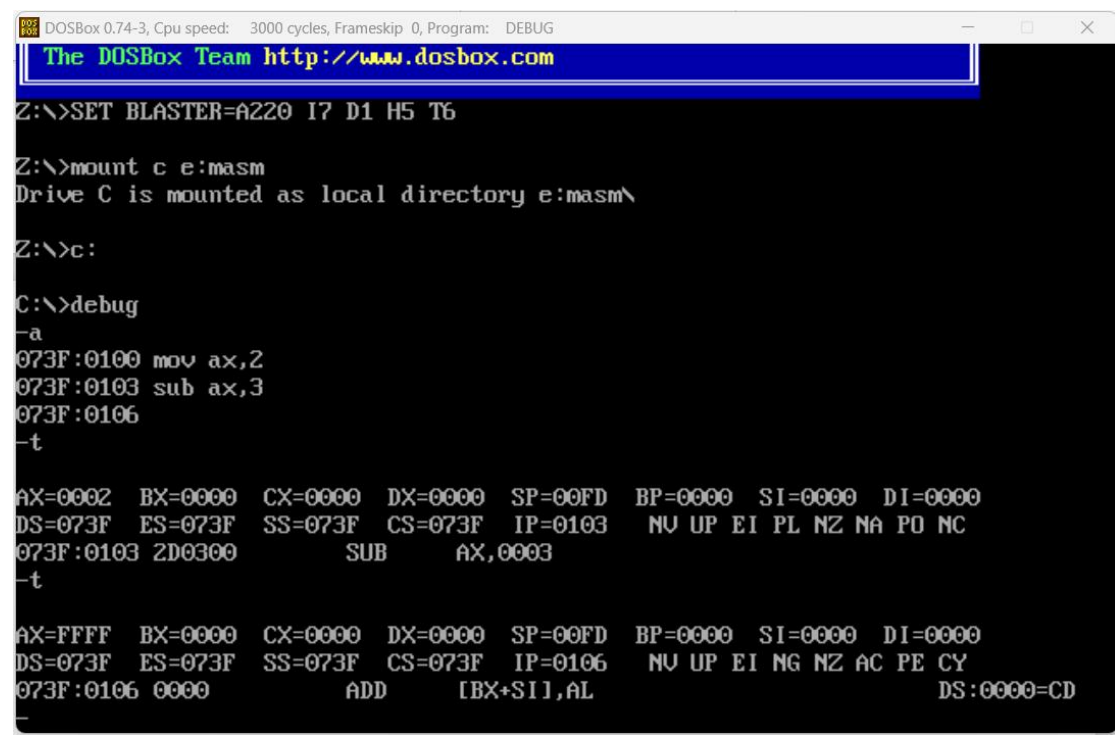
三、实验内容

(1)、在 debug 中先 A 输入下列指令，再 T 单步执行，观察 PSW 标志位

MOV AX, 2 ;MOV 指令不影响标志位

SUB AX, 3 ; 注意 sf(NG) , cf(CY)

说明：因为 $2-3 = -1$ ，有借位，补码表示结果为 FFFF，所以 sf 为 NG(符号位为 1)，cf 为 CY(有借位)



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount c e:masm
Drive C is mounted as local directory e:masm\
Z:\>c:
C:\>debug
-a
073F:0100 mov ax,2
073F:0103 sub ax,3
073F:0106
-t
AX=0002 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0103  NV UP EI PL NZ NA PO NC
073F:0103 2D0300      SUB     AX,0003
-t
AX=FFFF BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NV UP EI NG NZ AC PE CY
073F:0106 0000      ADD     [BX+SI],AL      DS:0000=CD
```

ADD AX, 5 ; 注意 sf(PL), cf(CY)

说明: 因为 $FFFFH+5H=4H$, 结果为正, sf 为 PL(符号位为 0), cf 为 CY(有进位)

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
073F:0106
-t
AX=0002 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0103  NU UP EI PL NZ NA PO NC
073F:0103 2D0300 SUB AX,0003
-t
AX=FFFF BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NU UP EI NG NZ AC PE CY
073F:0106 0000 ADD [BX+SI],AL DS:0000=CD
-a
073F:0106 add ax,5
073F:0109
-r
AX=FFFF BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NU UP EI NG NZ AC PE CY
073F:0106 050500 ADD AX,0005
-t
AX=0004 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0109  NU UP EI PL NZ AC PO CY
073F:0109 0000 ADD [BX+SI],AL DS:0000=CD
```

SUB AX, AX ; 注意 zf (ZR)

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
AX=FFFF BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NU UP EI NG NZ AC PE CY
073F:0106 0000 ADD [BX+SI],AL DS:0000=CD
-a
073F:0106 add ax,5
073F:0109
-r
AX=FFFF BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NU UP EI NG NZ AC PE CY
073F:0106 050500 ADD AX,0005
-t
AX=0004 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0109  NU UP EI PL NZ AC PO CY
073F:0109 0000 ADD [BX+SI],AL DS:0000=CD
-a
073F:0109 sub ax,ax
073F:010B
-t
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010B  NU UP EI PL ZR NA PE NC
073F:010B 0000 ADD [BX+SI],AL DS:0000=CD
```

说明：因为 $ax-ax=0$ ，所以 zf 标志位 $ZR(1)$ ，表示结果为 0

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
DS=073F ES=073F SS=073F CS=073F IP=0109  NU UP EI PL NZ AC PO CY
073F:0109 0000 ADD [BX+SI],AL DS:0000=CD
-a
073F:0109 sub ax,ax
073F:010B
-t
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010B  NU UP EI PL ZR NA PE NC
073F:010B 0000 ADD [BX+SI],AL DS:0000=CD
-a
073F:010B add ax,8899
073F:010E add ax,99aa
073F:0111
-t
AX=8899 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010E  NU UP EI NG NZ NA PE NC
073F:010E 05AA99 ADD AX,99AA
-t
AX=2243 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0111  OV UP EI PL NZ AC PO CY
073F:0111 0000 ADD [BX+SI],AL DS:0000=CD

```

ADD AX, 8899 ; debug 中都是 16 进制，8899H

ADD AX, 99AA ; 注意 OF(OV)，溢出了

(2)、在 debug 中单步执行下列指令，执行前先写出 CF, ZF, SF, OF 的值，再和实验结果对照。

参考如下

Mov ax, 7896				
ADD AL, AH	; CF=1	ZF=0	SF=0	OF=0
ADD AH, AL	; CF=0	ZF= 0	SF=1	OF=1
ADD AL, F2	; CF=1	ZF=1	SF=0	OF=0
ADD AX, 1234	; CF=0	ZF=0	SF=1	OF=0

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\>debug
-a
073F:0100 mov ax,7896
073F:0103 add al,ah
073F:0105 add ah,al
073F:0107 add al,f2
073F:0109 add ax,1234
073F:010C
-t
AX=7896 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0103  NV UP EI PL NZ NA PO NC
073F:0103 00E0          ADD     AL,AH
-t
AX=780E BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0105  NV UP EI PL NZ NA PO CY
073F:0105 00C4          ADD     AH,AL
-t
AX=860E BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0107  OV UP EI NG NZ AC PO NC
073F:0107 04F2          ADD     AL,F2
-
-t
AX=8600 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0109  NV UP EI PL ZR AC PE CY
073F:0109 053412        ADD     AX,1234
-t
```

(3)、带进位的加法 ADC 。要实现双字加法 12348686H + 22339876H ,

MOV AX, 8686

MOV DX, 1234 ;先将一个 32 位的数高 16/低 16 位分别放入 DX:AX 中

ADD AX,9876 ;先加低 16 位, 将产生进位, CF=1

ADC DX,2233 ;再加高 16 位, ADC 是带进位的加法, 将加上 cf 位

相加后的 32 位结果保存在 dx:ax 中, 结果为 34681EFC

注: ADC 将考虑进位位 CF, 连同上次 ADD 的进位一起相加, 实现双字加法,

结果存放于 DX:AX 中, 为 34681EFC H 。本例演示了 CF 位的作用。

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
-q
C:\>debug
-a
073F:0100 mov ax,8686
073F:0103 mov dx,1234
073F:0106 add ax,9876
073F:0109 adc dx,2233
073F:010D
-t
AX=8686 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0103  NU UP EI PL NZ NA PO NC
073F:0103 BA3412      MOV     DX,1234
-t
AX=8686 BX=0000 CX=0000 DX=1234 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NU UP EI PL NZ NA PO NC
073F:0106 057698      ADD     AX,9876
-t
AX=1EFC BX=0000 CX=0000 DX=1234 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0109  NU UP EI PL NZ NA PE CY
073F:0109 81D23322     ADC     DX,2233
-
```

2.编程统计给定的 9 个成绩中 ≥ 60 分的人数。示范如下

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

Object filename [7_1.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

51690 + 464854 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link 7_1.obj

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [7_1.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>7_1.exe
6
C:\>_
```

```
5_3.asm 7_1.asm
1 data segment
2     scores db 59, 62, 70, 48, 91, 85, 66, 55, 75
3     count db 0
4 data ends
5
6 code segment
7 assume cs:code, ds:data
8
9 start:
10     mov ax, data
11     mov ds, ax
12
13     lea bx, scores
14     mov cx, 9
15
16 count_loop:
17     mov al, [bx]
18     cmp al, 60
19     jb next_score
20     inc count
21
22 next_score:
23     inc bx
24     loop count_loop
25
26     mov ah, 02h
27     mov dl, count
28     add dl, '0'
29     int 21h
30
31     mov ax, 4c00h
32     int 21h
33
34 code ends
35 end start
36
```

3. 比较 5 个字节长的字符串 A 和 B，若两个串相等则字节标志单元 FLG 置 1，否则置 0。
代码可定义如下：


```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

Object filename [7_2.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:

51766 + 464778 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>

C:\>link 7_2.obj

Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [7_2.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>7_2.exe
not equal
C:\>_
```

```
5_3.asm 7_1.asm 7_2.asm
1  data segment
2      A  db 'abcde'
3      B  db 'abxde'
4      FLG db 0
5      yes db 'equal$'
6      no  db 'not equal$'
7  data ends
8
9  code segment
10 assume cs:code, ds:data
11
12 start:
13     mov ax, data
14     mov ds, ax
15
16     lea si, A
17     lea di, B
18     mov cx, 5
19     repe cmpsb
20
21     jz equal
22     mov FLG, 0
23     lea dx, no
24     jmp exit
25
26 equal:
27     mov FLG, 1
28     lea dx, yes
29
30 exit:
31     mov ah, 9
32     int 21h
33
34     mov ah, 4ch
35     int 21h
36 code ends
37 end start
38
```

