

# Assembly Language Practice

Anyi Rao 141180092

April 15th 2016

**Abstract:** An assembly (or assembler) language, often abbreviated asm, is a low-level programming language for a computer. in which there is a very strong (generally one-to-one) correspondence between the language and the architecture's machine code instructions.

**Keywords:**MASM, LINK, DEBUG

## 1 汇编总体框架

汇编程序的最小配置，以第一个实验20个无符号8位二进制数求和为例，需要有堆栈段，数据段，代码段和程序

```
STACK SEGMENT STACK
    DW 128 DUP(?)
STACK ENDS

DATA SEGMENT
    ARRAY DB 01,02,03,04,05,06,07,08,09,10,15,16,17,18,19,20,21,22,23,24
    PRINT DB ? ;print ASCII
    RESULT DW ? ;result
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA, SS:STACK
main PROC FAR
CODE ENDS
    END main
```

Figure 1: 汇编总体框架

## 2 20个无符号8位二进制数相加

### 2.1 操作实现

20个无符号8位二进制数相加，先全部求和，得到SUM

通过对这20个数之和做初步估计，需要三个字节空间来储存，故print+3

然后对SUM除以CX=10，得到的商放入AX，余数放入DX。然后对余数和30（ASCII的0）进行或操作，得到ASCII码，存入【SI】指向的内存。

通过DOS功能调用，将这些数打印出来

```
16 main PROC FAR
17     PUSH DS
18     SUB AX, AX
19     PUSH AX
20     MOV AX, DATA
21     MOV DS, AX           ;initialization
22     MOV AX, 0
23     LEA BX, ARRAY       ;get the initail address
24     MOV CX, 20         ;time of loop is 20
25
26 SUM:  ADD AL, [BX]      ;add to get the sum
27     JNC OVER
28     INC AH
29     CLC                ;clear carry
30 OVER: INC BX
31     LOOP SUM
32     MOV RESULT, AX
33     MOV CX, 0010       ;inord to div 10 to get the remainder
34     LEA SI, PRINT+3   ;the result is within 3 byte space
35 LOr:  CMP AX, CX       ;loop to get the remainder of high bit
36     JB Lre
37     XOR DX, DX
38     DIV CX
39     OR DL, 30H
40     MOV [SI], DL      ;put into memory
41     DEC SI
42     JMP LOr
43 Lre:  OR AL, 30H       ;The last remainder
44     MOV [SI], AL      ;put into memory
45     MOV AH, 02H
46     MOV CX, 4         ;time of loop to print is 4
47     LEA DI, PRINT
48 LOPR: LOOP PRINT
49     MOV DL, [DI]
50     INT 21H
51     INC DI
52     LOOP LOPR
53
54     MOV AX, 4C00H
55     INT 21H
```

Figure 2: 无符号二进制数运算的汇编主体程序

## 2.2 DEBUG

从调试中，数据dec压入memory SI 17-15。打印时，数据出来，inc 15-17

可以看到在memory 077A:0015-007A0017中正是存放的数的ASCII码

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
077C:0039 B804 MDU [SI],AL DS:0015=FA
-d10
077A:0010 15 16 17 18 00 FA 35 30-00 00 00 00 00 00 00 00 .....50.....
077A:0020 1E 2B C0 50 B8 7A 07 8E-D8 B8 00 00 8D 1E 00 00 .+.P.z.....
077A:0030 B9 14 00 02 07 73 03 FE-C4 F8 43 E2 F6 A3 15 00 ....s...C....
077A:0040 B9 0A 00 8D 36 17 00 3B-C1 72 0C 33 D2 F7 F1 80 ...6...;r.3...
077A:0050 CA 30 88 14 4E EB F0 0C-30 88 04 B4 02 B9 04 00 .0.N...0.....
077A:0060 8D 3E 14 00 8A 15 CD 21-47 E2 F9 B8 00 4C CD 21 .>....!G...L.!
077A:0070 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
077A:0080 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
-t
AX=0032 BX=0014 CX=000A DX=0035 SP=00FC BP=0000 SI=0015 DI=0000
DS=077A ES=075A SS=076A CS=077C IP=003B NU UP EI PL NZ NA PO NC
077C:003B B402 MDU AH,02
-d10
077A:0010 15 16 17 18 00 32 35 30-00 00 00 00 00 00 00 .....250.....
077A:0020 1E 2B C0 50 B8 7A 07 8E-D8 B8 00 00 8D 1E 00 00 .+.P.z.....
077A:0030 B9 14 00 02 07 73 03 FE-C4 F8 43 E2 F6 A3 15 00 ....s...C....
077A:0040 B9 0A 00 8D 36 17 00 3B-C1 72 0C 33 D2 F7 F1 80 ...6...;r.3...
077A:0050 CA 30 88 14 4E EB F0 0C-30 88 04 B4 02 B9 04 00 .0.N...0.....
077A:0060 8D 3E 14 00 8A 15 CD 21-47 E2 F9 B8 00 4C CD 21 .>....!G...L.!
077A:0070 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
077A:0080 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....

```

Figure 3: DEBUG内存段显示

## 3 6位BCD数减法

### 3.1 已知两个6位的BCD码，完成减法运算，被减数123456，减数789123

SBB AL,【DI】 AL 减去DI 所指向memory的数，结果存放在AL中

### 3.2 DEBUG

offset0000-0005存放的是789123 0006-000B 存放的是123456 000C-0012存放的是结果-665667

```

1  STACK SEGMENT STACK
2      DW 128 DUP(?)
3  STACK ENDS
4  DATA SEGMENT
5      NUM1 DB '789123' ;subtrahend
6      NUM2 DB '123456' ;minuend
7      SF DB '-' ;minus sign
8      RESULT DB ?
9  DATA ENDS
10
11
12  CODE SEGMENT
13      ASSUME CS:CODE, DS:DATA, SS:STACK
14  MAIN PROC FAR
15      PUSH DS
16      SUB AX, AX
17      PUSH AX
18      MOV AX, DATA
19      MOV DS, AX
20      LEA SI, NUM1+5 ;the num is 6bit so offset is 5
21      LEA DI, NUM2+5
22      LEA BX, RESULT+5
23      MOV CX, 6
24      CLC
25      MOV AL, [SI]
26
27  1sub: MOV AH, [SI-1] ;loop to subtract
28      SBB AL, [DI]
29      AAS ; ASCII adjution
30      OR AX, 3030H ; get the ASCII
31      MOV [BX], AL
32      MOV AL, AH
33      DEC SI
34      DEC DI
35      DEC BX
36      LOOP 1sub
37      MOV AH, 02H
38      MOV CX, 7 ;time of loop is 7
39      LEA DI, SF ;the result is minus
40  LOPR: MOV DL, [DI] ;LOOP PRINT
41      INT 21H
42      INC DI
43      LOOP LOPR
44
45      MOV AX, 4C00H
46      INT 21H
47  MAIN ENDP

```

Figure 4: BCD运算的汇编主体程序

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
077B:002F B90700    MOV     CX,0007
077B:0032 BD3E0C00    LEA    DI,[000C1
077B:0036 BA15    MOV     DL,[DI]
077B:0038 CD21    INT     21
077B:003A 47    INC     DI
077B:003B E2F9    LOOP   0036
077B:003D B8004C    MOV     AX,4C00
077B:0040 CD21    INT     21
077B:0042 56    PUSH   SI
077B:0043 B8BE05    MOV     AX,05BE
-g2d

AX=3131 BX=000C CX=0000 DX=0000 SP=00FC BP=0000 SI=FFFF DI=0005
DS=077A ES=075A SS=076A CS=077B IP=002D  NU UP EI PL NZ NA PE NC
077B:002D B402    MOV     AH,02
-d0
077A:0000 37 38 39 31 32 33 31 32-33 34 35 36 2D 36 36 35 789123123456-665
077A:0010 36 36 37 50 B8 7A 07 8E-D8 8D 36 05 00 8D 3E 0B 667P.z....6...>.
077A:0020 00 8D 1E 12 00 B9 06 00-F8 8A 04 8A 64 FF 1A 05 .....d...
077A:0030 3F 0D 30 30 88 07 8A C4-4E 4F 4B E2 EE B4 02 B9 ?.00...NOK....
077A:0040 07 00 8D 3E 0C 00 8A 15-CD 21 47 E2 F9 B8 00 4C ...>.....!G...L
077A:0050 CD 21 56 B8 BE 05 50 E8-C3 71 83 C4 02 8B F0 0B .!U...P..q.....
077A:0060 F6 74 61 8D 86 7E FF 8B-F8 80 3C 3B 74 05 80 3C .ta..~....<;t..<
077A:0070 00 75 45 8D 86 7E FF 3B-C7 73 42 8B C7 2B C5 04 .uE..~.;.sB..+..

```

Figure 5: DEBUG BCD运算

## 4 非数值运算代码转换

### 4.1 BCD9649转换

编程将组合的BCD码9649转换成二进制数，考虑采用  $((A*10) + B)*10 + C)*10 + D$  的算法。

通过将9649分别和0f000h, 0f00h, 00f0h, 000fh或操作得到9649的千位百位十位个位。将上述得到的数右移相应的位数，在分别乘以1000,100,10,1得到十进制的9649，而计算机是通过2进制来储存的。

其中：乘以10的实现通过调用子程序multi，将数调入子程序，得到2倍的数和8倍的数，再相加即得到10倍的数。

最后通过左移时的进位位和30h，得到ASCII码后功能调用显示出来

### 4.2 DEBUG BCD9649代码转换

DEBUG 分别展示了 \*如何得到9649的千位 \*得到9649的16进制数25B1 \*通过SHL得到2进制表达并通过dos功能调用将其输出

```

main:  mov AX, DATAS
       mov DS, AX
       mov ax, [num1]
       and ax, 0f000h
       mov cl, 12
       shr ax, cl
       mov dx, ax
       call multi

       mov ax, dx
       mov bx, [num1]
       and bx, 0f00h
       mov cl, 8
       shr bx, cl
       add ax, bx
       mov dx, ax
       call multi

       mov ax, dx
       mov bx, [num1]
       and bx, 00f0h
       mov cl, 4
       shr bx, cl
       add ax, bx
       mov dx, ax
       call multi

       mov ax, dx
       mov bx, [num1]
       and bx, 000fh
       add ax, bx           ;get 9649(decimal) 25b1(hex)

       mov num2, ax
       mov cx, 16
print: shl num2, 1
       mov dl, 0           ;initial
       adc dl, 30h         ;carry is from shl ASCII 30h=0 31h=1
       mov ah, 2           ;function display on screen by DL
       int 21h

loop  print
      mov ah, 4CH
      INT 21H

```

(a) 主体程序

```

multi PROC ;multi 10(decimal)
  add dx, dx
  mov cx, dx ;cx=2dx
  add dx, dx ;dx=4dx
  add dx, dx ;dx=8dx
  add dx, cx ;dx=cx+dx=2dx+8dx=10dx
  ret
multi ENDP

```

(b) 调用程序将数乘以10

Figure 6: 9649转换汇编源代码

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\PROJECT>debug 9649.EXE
-t
076B:0000 B86A07      MOV     AX,076A
076B:0003 8ED8             MOV     DS,AX
076B:0005 A10000      MOV     AX,[0000]
076B:0008 2500F0      AND     AX,F000
076B:000B B10C             MOV     CL,0C
076B:000D D3E8             SHR     AX,CL
076B:000F 8BD0             MOV     DX,AX
076B:0011 EB4E00      CALL   0062
076B:0014 8BC2             MOV     AX,DX
076B:0016 8D130000     MOV     BX,[0000]
076B:001A B1E3000F     AND     BX,0F00
076B:001E B108             MOV     CL,08
-gB
AX=9649 BX=0000 CX=007D DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0008  NU UP EI PL NZ NA PO NC
076B:0008 2500F0      AND     AX,F000
-t
AX=9000 BX=0000 CX=007D DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=000B  NU UP EI NG NZ NA PE NC
076B:000B B10C             MOV     CL,0C
-t

```

(a) 得到9649的千位

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
076B:0044 83E30F      AND     BX,+0F
076B:0047 03C3      ADD     AX,BX
076B:0049 A30200      MOV     [0002],AX
076B:004C B91000      MOV     CX,0010
-g44
AX=25A8 BX=9649 CX=078B DX=25A8 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0044  NU UP EI PL NZ NA PO NC
076B:0044 83E30F      AND     BX,+0F
-t
AX=25A8 BX=0009 CX=078B DX=25A8 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0047  NU UP EI PL NZ NA PE NC
076B:0047 03C3      ADD     AX,BX
-t
AX=25B1 BX=0009 CX=078B DX=25A8 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0049  NU UP EI PL NZ AC PE NC
076B:0049 A30200      MOV     [0002],AX      DS:0002=0000
-t
AX=25B1 BX=0009 CX=078B DX=25A8 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=004C  NU UP EI PL NZ AC PE NC
076B:004C B91000      MOV     CX,0010

```

(b) 得到9649的16进制数25B1

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
AX=0230 BX=0009 CX=000C DX=2530 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=004F  NU UP EI PL NZ NA PE NC
076B:004F D1260200     SHL     WORD PTR [0002],1      DS:0002=5B10
-t
AX=0230 BX=0009 CX=000C DX=2530 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0053  OU UP EI NG NZ AC PO NC
076B:0053 B200             MOV     DL,00
-t
AX=0230 BX=0009 CX=000C DX=2500 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0055  OU UP EI NG NZ AC PO NC
076B:0055 80D230      ADC     DL,30
-t
AX=0230 BX=0009 CX=000C DX=2530 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0058  NU UP EI PL NZ NA PE NC
076B:0058 B402             MOV     AH,02
-t
AX=0230 BX=0009 CX=000C DX=2530 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=005A  NU UP EI PL NZ NA PE NC
076B:005A CD21             INT     21

```

(c) SHL得到2进制表达并输出

Figure 7: DEBUG BCD9649代码转换

```

STACK SEGMENT stack
    DW 50 DUP (?)
STACK ENDS

DATA SEGMENT
    DUP DB >,0Dh,0Ah,'$';CR (carriage return) LF (line feed)
    INP DB OFFh,0,255 DUP (?)
DATA ENDS

CODE SEGMENT
MAIN PROC FAR
    ASSUME SS:STACK,CS:CODE,DS:DATA
    PUSH DS
    SUB AX,AX
    PUSH AX
    MOV AX,DATA
    MOV DS,AX
    MOV DX,OFFSET INP
    MOV AH,10
    INT 21h
    MOV BL,INP+1
    MOV BH,0
    MOV INP[BX+2],'$'
    MOV DL,0Ah ;line feed
    MOV AH,2
    INT 21h
    MOV DX,OFFSET INP+2
    MOV AH,9
    INT 21h
    ret
MAIN ENDP
CODE ENDS
END MAIN

```

(a) 主体程序

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Microsoft (B) Micro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.
Object filename: DISPLAY.OBJ
Source listing: INUL.LST
Cross-reference: INUL.CRF
51554 + 464990 Bytes symbol space free
0 Warning Errors
0 Severe Errors
C:\PROJECT>link DISPLAY.OBJ
Microsoft (B) Overlay Linker Version 3.69
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
Run File: DISPLAY.EXE
List File: INUL.LIB
Libraries: I.LIB
C:\PROJECT>DISPLAY.EXE
hello 123
C:\PROJECT>

```

(b) 运行结果

Figure 8: 输入字符串再显示

## 5 DOS功能调用

前面的程序都通过DOS功能调用，进行了结果的显示。考虑到直接输入一个字符，再回显与前面的程序类似，故做了一个输入字符串再回显的。区别在于9号功能是以dollar符号结束的。

在数据段定义字节变量INP,第一个单元表示最多接受的字符，第二个单元表示输入字符的个数（便于之后找到放dollar符号的位置），第三个单元用来转载字符。只要把10号功能里的回车变成dollar，即可用9号功能从INP单元开始的字符送回显示器显示，直到dollar结束。回车的0DH存放单元的偏移地址等于INP的偏移地址、立即数2与输入字符个数之和。若将输入字符个数+1单元的内容送给BX，则INP【BX+2】表示的偏移地址就是dollar符号存入的单元地址

## 6 Analysis

### 6.1 段结构伪指令

Format:段名 SEGMENT [定位类型][组合类型][‘类别’]  
段体



段名 ENDS

添加组合类型STACK 把不同模块中的同名段组合成一个堆栈段。可以解决link时，没有堆栈段的警告

## 6.2 DEBUG循环程序

对有loop循环结构的程序，DEBUG 调试时，反汇编会把loop之后的程序直接反汇编在一个循环的后面 如果用DEBUG环境下的G命令，无法实现

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
077B:0021 0D3030      OR      AX,3030
-t
AX=3736 BX=000E CX=0002 DX=0000 SP=00FC BP=0000 SI=0001 DI=0007
DS=077A ES=075A SS=076A CS=077B IP=0024  NU UP EI PL NZ NA PE NC
077B:0024 8807          MOV     [BX],AL          DS:000E=00
-u
077B:0024 8807          MOV     [BX],AL
077B:0026 B4C4          MOV     AL,AH
077B:0028 4E           DEC     SI
077B:0029 4F           DEC     DI
077B:002A 4B           DEC     BX
077B:002B E2EE          LOOP    001B
077B:002D B402          MOV     AH,02
077B:002F B90700       MOV     CX,0007
077B:0032 BD3E0C00     LEA    DI,[000C]
077B:0036 8A15          MOV     DL,[DI]
077B:0038 CD21          INT     21
077B:003A 47           INC     DI
077B:003B E2F9          LOOP    0036
077B:003D B8004C       MOV     AX,4C00
077B:0040 CD21          INT     21
077B:0042 56           PUSH   SI
077B:0043 B8BE05       MOV     AX,05BE

```

Figure 9: 循环结构反汇编显示的结果

一次把部分的循环（大于一次又小于全部次数）做完。

在给的图片显示的例子中，只能做到要么运行一次循环（G 2B），要么运行全部循环（G 2D）