Assembly Language Practice

Anyi Rao 141180092

Apirl 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ACSII码，存入【SI】指向的内存。

通过DOS功能调用，将这些数打印出来

```
main  PROC FAR
      PUSH DS
      SUB AX,AX
      PUSH AX
      MOV AX,DATA
      MOV DS,AX       ;initializion
      MOV AX,0
      LEA BX,ARRAY    ;get the initail address
      MOV CX,20       ;time of loop is 20

SUM:  ADD AL,[BX]     ;add to get the sum
      JNC OVER
      INC AH
      CLC             ;clear carry
OVER: INC BX
      LOOP SUM
      MOV RESULT,AX
      MOV CX,0010     ;inord to div 10 to get the remainder
      LEA SI,PRINT+3  ;the result is within 3 byte space
LOre:  CMP AX,CX      ;loop to get the remainder of high bit
      JB Lre
      XOR DX,DX
      DIV CX
      OR DL,30H
      MOV [SI],DL     ;put into memory
      DEC SI
      JMP LOre
Lre:  OR AL,30H       ;The last remainder
      MOV [SI],AL     ;put into memory
      MOV AH,02H
      MOV CX,4        ;time of loop to print is 4
      LEA DI,PRINT
LOPR:                 ;LOOP PRINT
      MOV DL,[DI]
      INT 21H
      INC DI
      LOOP LOPR

      MOV AX,4C00H
      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ACSII码

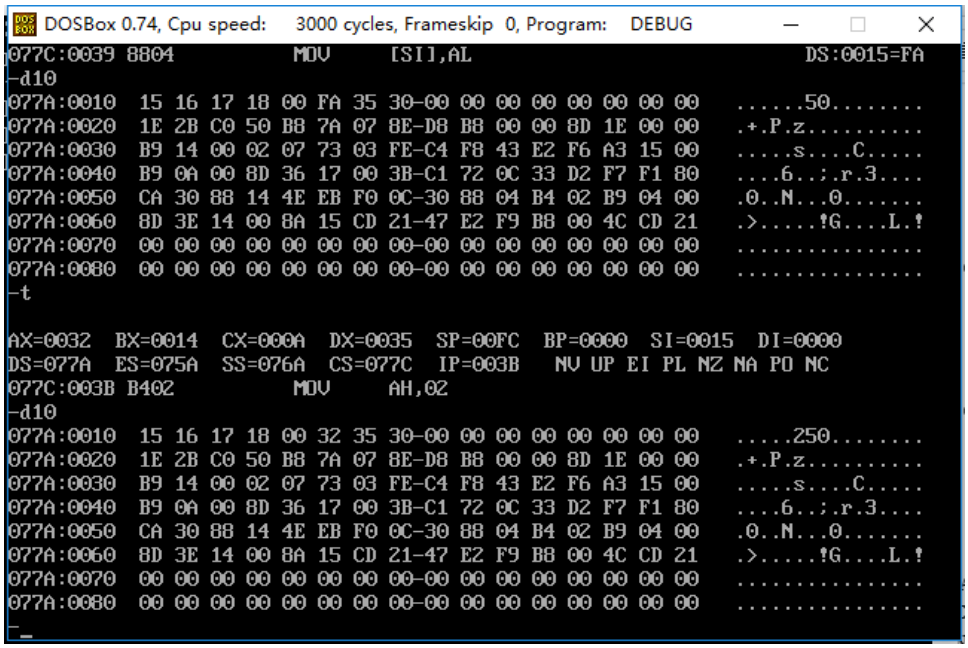


Figure 3: DEBUG内存段显示

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 lsub:MOV AH,[SI-1] ;loop to subtract
28       SBB AL,[DI]
29       AAS           ; ASCII adjustion
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 lsub
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

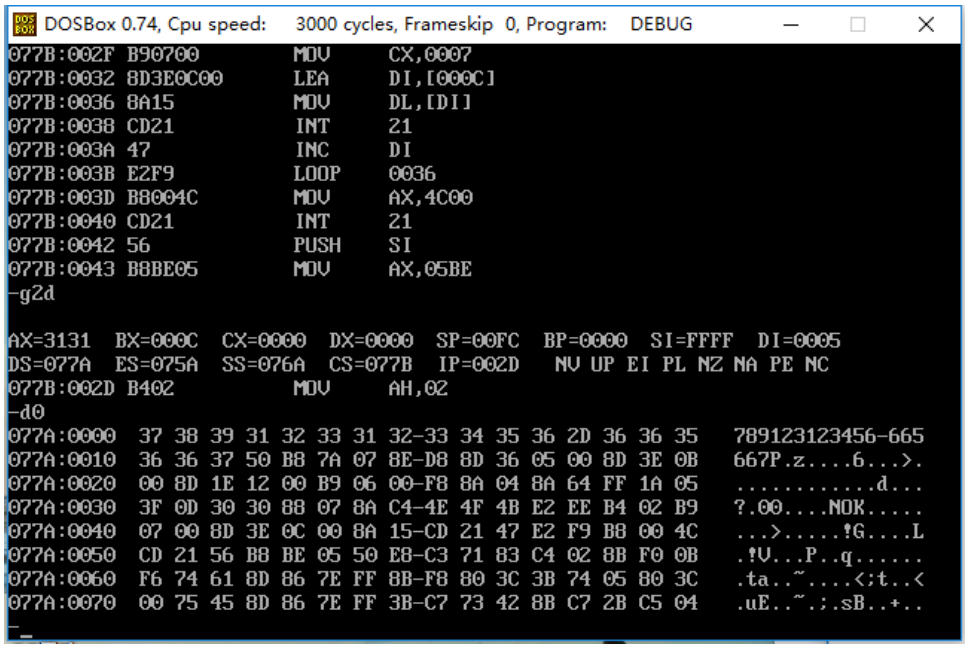


Figure 5: DEBUG BCD运算

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

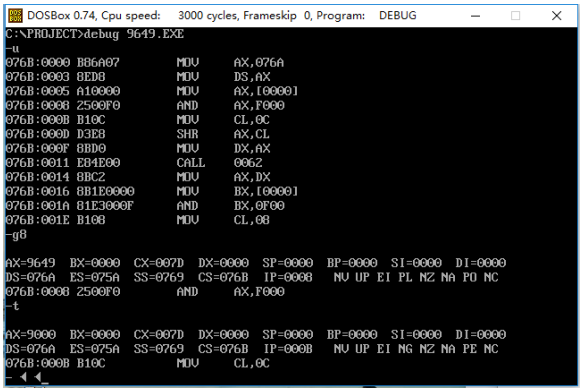
```
main:  mov AX,DATAS
       mov DS,AX
       mov ax,[num1]
       and ax,0f000h
       mov cl,12
       shr ax,cl
       mov dx,ax
       call multi

       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

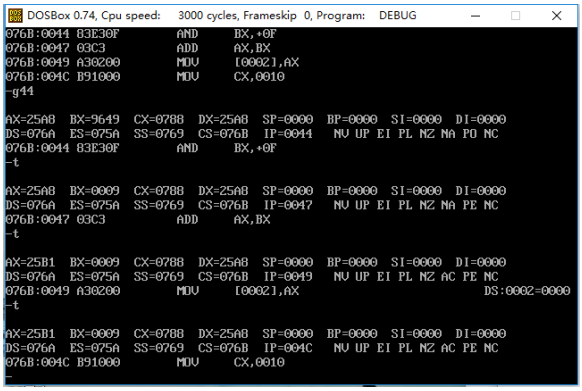
```
multi PROC    ;multi 10(decimal)
  add dx,dx
  mov cx,dx   ;cx=2dx
  add dx,dx   ;dx=4dx
  add dx,dx   ;dx=8dx
  add dx,cx   ;dx=cx+dx=2dx+8dx=10dx
  ret
multi ENDP
```

(b) 调用程序将数乘以10

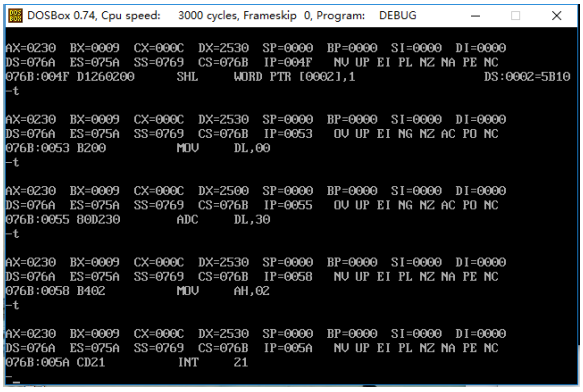
Figure 6: 9649转换汇编源代码



(a) 得到9649的千位

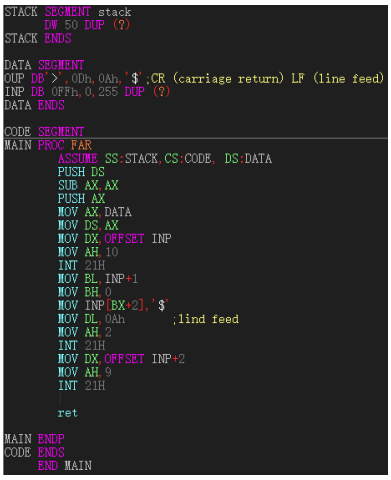


(b) 得到9649的16进制数25B1



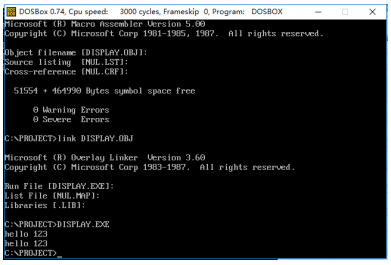
(c) SHL得到2进制表达并输出

Figure 7: DEBUG BCD9649代码转换



```
STACK SEGMENT stack
      DW 50 DUP (?)
STACK ENDS

DATA SEGMENT
OUP DB '$',0Dh,0Ah,'$' ;CR (carriage return) LF (line feed)
INP DB 0FFh,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        ;lind 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 (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

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

  51554 + 464990 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\PROJECT>link DISPLAY.OBJ

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

Run File [DISPLAY.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\PROJECT>DISPLAY.EXE
hello 123
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

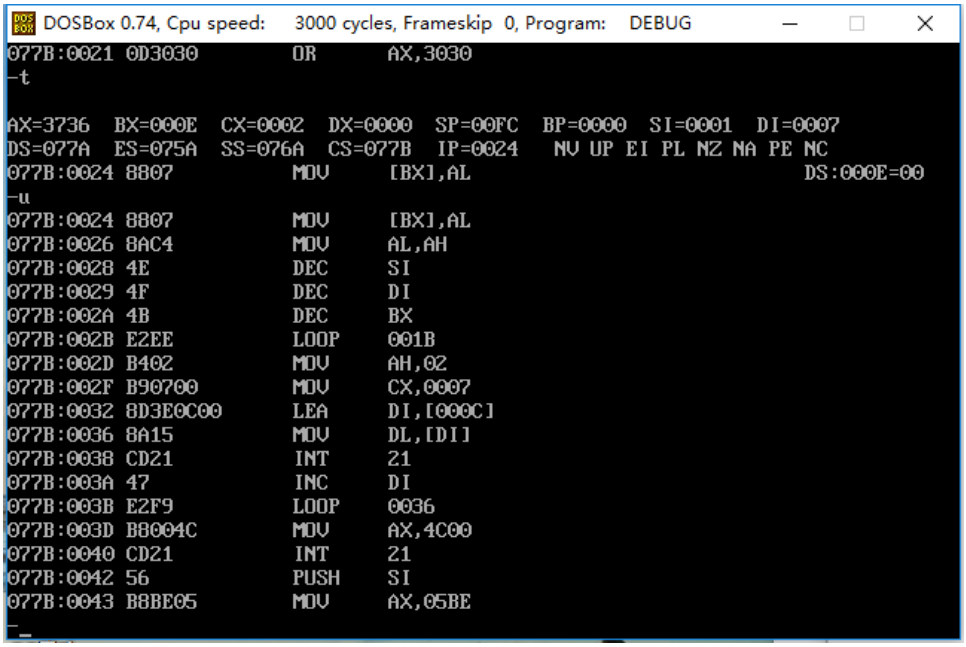


Figure 9: 循环结构反汇编显示的结果

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)