汇编语言上机大作业报告

一、设计题目

四则运算

从键盘输入一个简单的表达式,如"S=4+6*9-1+8/5",按回车键结束输入,则屏幕显示 S=58.6,小数点保留 1 位。假设输入的表达式中只含个位十进制数和"+"、"-"、"*"、"/"运算符,且同一运算符最多出现 2 次。

二、设计说明

1. 整个程序的功能应该能完成的工作

这个程序应该能正确处理数字和数学表达式的输入。我的设想是使其进一步处理最多 12 位十进制小数的输入,以及带有括号、四则运算算式的正确处理,并给出可以精确到小数点后五位的正确结果。

该程序应完成工作:

- 1) 公式的输入,包括处理数字输入、符号输入,以及正确处理输入公式的句法。
- **2)** 公式的计算。其中包括正确处理各种符号运算的优先级和结合性、中间数的临时保存、小数的正确处理等。
- 3) 结果的正确显示。

2. 多个任务(子程序)的划分、调用关系

- 1) 十进制浮点数字的输入
- 2) 十进制浮点数字的输出
- 3) 符号的输入
- 4) 对于输入的公式的综合处理(转化为程序内部各数字/数据(称作 token) 一定的存储格式)。调用 1)与 3)
- 5) 对于处理后的公式的运算
- 6) 主程序。调用 4), 再调用 5), 最后调用 2)。

3. 各子程序的功能与联系

(1)处理十进制小数的转换: ID.ASM

包含过程: INPUTDECIMAL,负责将内存中的十进制小数字符串转换为双精度浮点数,仍然存在内存中。

输入参数包括字符串的起始位置 DS:SI, 和输出位置 ES:BX。

(2)处理十进制小数的输出: OD.ASM

包含过程: OUTPUTDECIMAL,负责将内存中的双精度浮点数以十进制小数形式输出。

输入参数包括字符串的起始位置 DS:SI。

(3)处理算式: PARSEEXP.ASM

包含过程: PARSEEXP,负责将内存中的一串算式字符串进行解析运算,并将最终结果存在内存中。

也同时输出算式的逆波兰符号形式,方便调试找出错误。

会调用 ID.ASM、OD.ASM 中的功能。

输入参数包括字符串的起始位置 DS:SI 和输出位置 DS:BX。

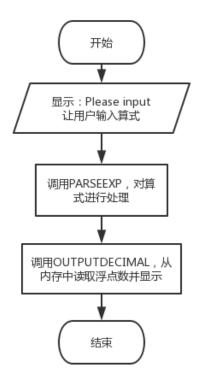
(4)主程序: MAIN.ASM

包含过程 MAIN, 是程序的入口。

功能包括显示输入提示、让用户输入算式,以及调用 PARSEEXP 处理算式,最后用 OUTPUTDECIMAL 输出计算结果。

4. 程序框图

主程序的大致框图如下:



5. 子程序说明和流程图

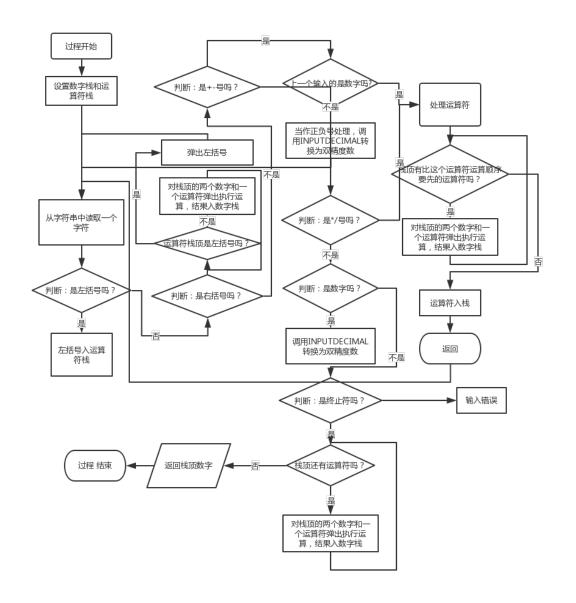
(1) PARSEEXP

逐字符进行读取,并根据读取到的字符判断算式中出现的 token 属于什么类型。若是数字,则调用 INPUTDECIMAL 将其处理成双精度浮点数 ;若是运算符,则对应处理(见下)。

对于算式的处理和运算,采用了调度场算法(Shunting yard algorithm)。 采用两个堆栈,一个放数字,一个放运算符。当算法执行到将运算符放置到数 字上的步骤时,立即进行运算。这样,分析完算式后,数字栈顶便是结果。

运算用 Intel 处理器的 FLD、FADD 等指令完成。

流程见图, 省略了对部分错误状况的处理, 详见源码。



(2) INPUTDECIMAL

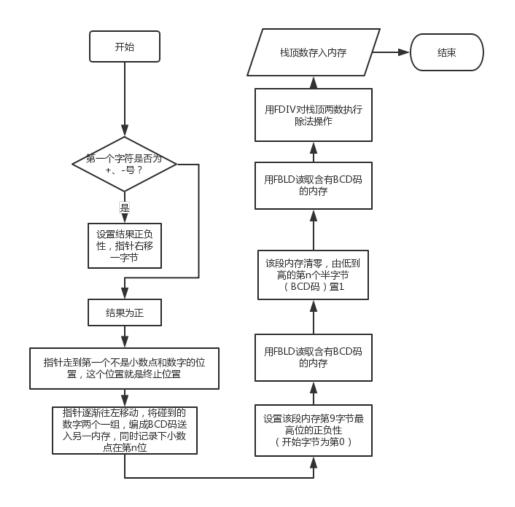
利用了英特尔处理器的 FBLD 指令,将十进制的小数转换为浮点数。

FBLD 指令可以将一段长度为 10 字节空间中的前 9 字节中的 18 位 BCD 码 从十进制整数转换为 FPU 寄存器中的 REAL10 浮点数,后一字节最高位决定符号位。

如一系列从低位到高位排列的字节: 21 43 65 87 00 00 00 00 00 80 可以转换为-87654321.0,存储在 FPU 的栈顶。

对字符串倒序、去掉小数点转 BCD 码后,用 FBLD 转换为大浮点数;再根据小数点位置,用 FBLD 构造一个是 10 的若干次幂的浮点数,除前者,得到字符串表示的浮点数。再用 FSTP 将其作为双精度数存储于内存,该功能即得以实现。

大致框图见图。



(3) OUTPUTDECIMAL

利用的是 FBLD 的逆向指令 FBSTP。将双精度数用 FLD 载入 FPU 的寄存器后,乘以 100000,用 FBSTP 转换为包含小数点后五位的 BCD 数码,再从右往左解析输出,在第五位加小数点即可。

6. 程序清单

共四个源码文件:

- MAIN.ASM
- ID.ASM
- OD.ASM
- PARSEEXP.ASM

附件见下。

源码见附录 (全文末端)











三、调试说明

1. 调试情况

本项目调试工作较重,一方面因为本人经验不足,一方面因为 16 位汇编本身难以调试的特性。在调试时,主要遇到以下问题:

(1)浮点数指令工作不正常

在用 FADD 等指令进行浮点运算时,发现加载的浮点数并无任何问题,但算出结果储存后是 FFF80000...(NaN)。最后发现,将各个子程序中分散的 FINIT 指令集中在 MAIN 程序中解决了此问题。可能 FINIT 执行多次会带来未定义的操作。

(2)在子程序中用地址表失效

在子程序 PARSEEXP 中,判断操作符类型时,采用了地址表。但无论是在 CS 段中定义还是在 DS 段中定义,地址表都无法正常工作。最后发现,地址表中的标号会在汇编时转换为偏移地址,但其对应的段地址和远调用时认定的子程序段地址并不一致,所以用地址表时会跳到不正确的地方造成 NTVDM 崩溃。最后,将地址表改为简单的条件跳转后问题解决。

其他问题皆是一些一眼能看出问题所在、由于粗心造成的小问题。

(3)程序调试技巧

在调试中我积累了一些经验:

- 程序源码的组织一定要经过深思熟虑,否则会在调试时造成更大麻烦;
- 善用 DEBUG 的一些指令以及其参数,如-P+数字可以一次执行多个指令,对于跳跃到问题代码很有帮助。

2. 链接的要求

对四个文件分别汇编,生成.OBJ 文件后,在控制台中输入 LINK

MAIN+PARSEEXP+ID+OD 回车,可以执行链接操作生成 MAIN.EXE。

3. 测试数据

```
Z:\INST\work-big>MAIN
Please input an expression.
S=3+4
3 4 +
Result: S = 7

Z:\INST\work-big>MAIN
Please input an expression.
S=3.14+90.52
3.14 90.52 +
Result: S = 93.66

CL 命令提示付
S=2048.3204-(-326.1)*((12-8)/0.002)
2048.3204 -326.1 12
Frame: Uncorported Taken
```

```
Z:\INST\work-big>MAIN
Please input an expression.
S=2048.3204-(-326.1)*((12-8))
2048.3204 -326.1 12 8 - * -
Result: S = 3352.7204
```

```
Z:\INST\work-big>MAIN
Please input an expression.
S=0
O
Result: S = 0
```

```
Z:\INST\work-big>MAIN
Please input an expression.
S=123456789012+111111111111
123456789012 111111111111 +
Result: S = 234567900123
```

```
Z:\INST\work-big>MAIN
Please input an expression.
S=0.00002+1/0.00002+50000
0.00002 1 0.00002 / + 50000 +
Result: S = 100000.00002
```

以下是各种输入错误的情形。

```
Z:\INST\work-big>MAIN
Please input an expression.
S=(
Error: Mismatched Parenthesis
Result: S = <Error>
```

```
Z:\INST\work-big>MAIN
Please input an expression.
S=
Result: S = <Error>
Z:\INST\work-big>
```

```
Z:\INST\work-big>MAIN
Please input an expression.
S=++9
Error: Incomplete Number or Unexpected Arrangement of +/
Result: S = <Error>
Z:\INST\work-big>_
```

```
Z:\INST\work-big>MAIN
Please input an expression.
S=3.14159+5*9)
3.14159 5 9 * +
Error: Mismatched Parenthesis
Result: S = <Error>
Z:\INST\work-big>_
```

```
Z:\INST\work-big>MAIN
Please input an expression.
S=3.14_
3.14
Error: Unexpected Token
Result: S = <Error>
Z:\INST\work-big>
```

Z:\INST\work-big>MAIN

Please input an expression.

S=6. 284-9+

6.284 9 -

Error: Missing Operand Somewhere

Result: S = <Error>

Z:\INST\work-big>

4. 结果分析

就本人测试过的数据来看,正确输入的算式都得到了正确的结果。对于输入错误的算式而言,本程序并没有覆盖到太多情况,比如+.+6 判断为 0+6=6等。

综上所述, 本程序基本达到了题目要求。

四、使用说明

1. 程序环境、适用范围

本程序应运行于:

- Intel 的 16 位处理器 (或模拟其工作的模拟器上), 如 8086, 并且应 支持浮点运算功能;
- MS-DOS, 或 Windows 7 及以下 x86 操作系统的命令提示符中(在 NTVDM 中运行)

本程序的测试环境为 DOSBox,以及 VMWare Workstation 下的 Windows 7 x86 虚拟机的 cmd.exe。

本程序适用干:

简单的、输入式的、包括+、-、*、/、()的数学运算式,结果精度不超过 5 位小数。

2. 程序运行

(1)使用方法

[1] 启动 MS-DOS 操作系统(或 DOSBox),或 Windows x86 系统的命令提示符(CMD.exe)。

- [2] 用 cd 命令和盘符命令,将当前路径定位到本程序可执行文件的目录下。
- [3] 输入 MAIN.EXE, 按回车键。
- [4] 此时程序提示用户输入"S="。用户应该输入一个算式,之后按回车键。
- [5] 程序应能给出 S 的最终结果。

(2)调试方法

- [1] 启动 MS-DOS 操作系统(或 DOSBox),或 Windows x86 系统的命令提示符(CMD.exe)。
- [2] 用 cd 命令和盘符命令,将当前路径定位到本程序可执行文件的目录下。
- [3] 输入 DEBUG MAIN.EXE。注意,操作系统的 Path 环境变量的条目路径下应能找到 16 位调试工具 DEBUG.EXE。
- [4] 使用命令 R、P、T、D、G等,对程序进行调试。

(3)输入信息的类型及格式

用户应输入一个长度小于等于 79 的字符串, 其每个字符只能是 0 到 9、+-*/、()、小数点、空格之一。输入的字符串应该是符合简单数学的格式要求的算式。

其中,输入数时,应注意数字的有效位数不得超过 **12** 位,得数的绝对值不应该超过 **1.0**×**10**¹²,不应出现除以 **0** 的情况。

(4)出错信息的含义

- "Error: Mismatched Parenthesis"表示你可能输入了一个没有匹配的括号。
- "Error: Missing Operand Somewhere"表示有一个地方没有操作数,可能是在某两个符号之间没有输入数字。
- "Error: Standard Syntax Error (No Operator between two operands)"发生的原因是两个数字挨在了一起。
- "Error: Unexpected Token"表示用户输入了在要求限制之外的符号。
- "Error: Unknown Error"是未知错误,可能有程序本身的缺陷产生。
- "Error: Incomplete Number or Unexpected Arrangement of +/-"表示某个数字未输入完整,也有可能是+、-号输入语法错误。

五、总结

1. 基本情况

(1)本人自学的内容和时间

- 7月25日-26日: 浮点操作的基本方法
- 8月2日: 浮点指令
- 8月14日-16日:宏

(2)参考书目

- 《汇编语言程序设计》, 张光长
- 《Intel 汇编语言程序设计(第五版)》,电子工业出版社,[美] Kip R.Irvine
- 《Intel® 64 and IA-32 Architectures Software Developer's Manual》, VOL 1, VOL 2A

(3)上机

在家里完成上机设计。在 7 月 25 日进行初步设想和试验程序设计,在 8 月 13 日-16 日完成全部程序设计。

本程序的所有部分均为我独立完成。读取、显示小数的算法来源于我自己的思考,解析表达式的方法来源于维基百科"调度场算法"词条。

2. 心得体会

本次大作业花了我很大的心思。无论是上课的过程、知识的积累、资料的搜集、真正编写、调试一个大工程的过程,都让我有了提高。

首先,我了解了汇编这门语言。它与机器码有着直接、密不可分的关系,是软件层面上的一个基层。没有它,就没有各种高级语言构建的美妙代码。在了解计算机的任何一个层次时,除了本层次的内容,还要稍微了解构成这个层次的下一层次内容。这样,才能更好的理解本层次的知识的来源及其所以然。所以,了解汇编,学习汇编,是重要的。

汇编让我知道了计算机软件的大厦是如何构建起来的,也锻炼了我的编程能力。我进一步地用汇编的思维,全面地考虑、简化问题,破解难题;思考在系统的底层层面中,诸如字符串的分析、数据的存储应该如何实现。要是不操作内存的话,就只有有限的十六位寄存器可以用。空间如何分配?代码如何简约?弄不好的话,不但代码可读性降低了,在调试时更是会让人抓狂。所以,

要给这方面以足够的重视。我想在任何一门编程语言中,都是这样的。代码简约、构造美观、可靠是重要的。

3. 存在的问题

由于学习汇编的时间不多,我对于很多已学过的指令,不能形成长期记忆; 另外,诸如 32 位汇编、宏、Windows API 等更多的汇编知识,我没有进一步的 了解,只能通过现学的知识来构造程序,这是一个不足之处;

同时,我在写本次大作业时,由于匆匆忙忙,对于一些小细节没有把握好,比如之前提到的错误覆盖不够全,没有处理浮点异常如除以零的情况,某些地方的代码不够精炼等。

改进方法包括完善一些小细节,优化代码;同时,可以加入更多的运算符, 甚至加入定义函数、使用函数的功能。

程序清单

文件名称	MAIN.ASM
子过程名称	MAIN
描述	主要过程,程序入口,负责输入输出及运算表达式子过程 调用

COUNT = 80D

KEYBOARD STRUCT

maxInput BYTE COUNT

inputCount BYTE ?
buffer BYTE COUNT DUP (?)

KEYBOARD ENDS

STACK SEGMENT STACK

BYTE 80H DUP (?)

STACK ENDS

DATA SEGMENT

expData KEYBOARD <>

pmtinstr DB "Please input an expression.",0DH, 0AH,

"S=\$"

pmtsstr DB "Result: S = \$"

result DQ ?

errorstr DB "<Error>\$"

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

EXTERN PARSEEXP: FAR, INPUTDECIMAL: FAR,

OUTPUTDECIMAL: FAR

mWriteStr MACRO buffer

PUSH DX PUSH AX

MOV DX, Offset buffer

DX

MOV AH, 09H INT 21H POP AX

ENDM

mWriteChr MACRO chr

POP

PUSH DX PUSH AX

ENDM	MOV MOV INT POP POP	DL, chr AH, 02H 21H AX DX
MAIN	PROC PUSH MOV PUSH	FAR DS AX, ØH AX
	MOV MOV	AX, DATA DS, AX
	FINIT	
	mWriteStr	pmtinstr
	MOV MOV INT	AH, ØAH DX, Offset expData 21H
	mWriteChr mWriteChr	0DH 0AH
	MOV MOV CBW ADD	BX, Offset expData.buffer AL, expData.inputCount BX, AX
	MOV	Byte Ptr[BX], '\$'
	MOV MOV CALL	SI, Offset expData.buffer BX, Offset result PARSEEXP
	mWriteStr	pmtsstr
	MOV	SI, BX
	CMP JNZ ADD CMP JNZ ADD CMP JNZ ADD CMP JNZ ADD	Word Ptr[BX], 0FFFH NO_ERROR BX, TYPE Word Word Ptr[BX], 0FFFFH NO_ERROR BX, TYPE Word Word Ptr[BX], 0FFFFH NO_ERROR BX, TYPE Word

CMP Word Ptr[BX], 0FFFFH

JNZ NO_ERROR JMP ERROR_OUT

NO_ERROR:

CALL OUTPUTDECIMAL

JMP FINAL

ERROR_OUT:

mWriteStr errorstr

FINAL:

RETF

MAIN ENDP CODE ENDS END MAIN

	1						
文件名称	PAR	SEEXP.AS	SM				
子过程名称	PAR	SEEXP					
描述	负责	数学表	达式字	符串的读取和原	解析计	算	
PUBLIC PA	ARSEEXP			OP_RBRA	EQU	1	
	EGMENT			OP ADD	EQU	2	
-	DW	0, 0,		OP_SUB	EQU	3	
EVAL_ADD, EVAL_SU			DIV	OP MUL	EQU	4	
;地址表在用作 PROC f				OP DIV	EQU	5	
				OP_BOTTOM_SP	EQU	0FFFFH	
ORIG_SS D	W	;		OP_BOTTOM_BP	EQU	0FFH, 0FFH,	
ORIG_SP D	W	?		OFFH, OFFH, OF	-		
				PAEX DEBUG	EQU	1	
ERR_INPUT_STR_MISN	MAT_PAR	DB "Er	ror:	PALA_DLBOG	LQU	1	
Mismatched Parenth	hesis", 0d	h, 0ah,	'\$'	DAEY CODE	CECME	NIT	
ERR_INPUT_STR_NO_0	OPERAND	DB "Er	ror:	PAEX_CODE	SEGME		
Missing Operand So					ASSUM	E CS:PAEX_CODE,	
ERR_INPUT_STR_SYN				DS:PAEX_DATA			
Standard Syntax Er							
two operands)", 00							
ERR_INPUT_STR_UNE			ror:				
Unexpected Token"					EXTER		
ERR INPUT STR UNK			ror:	FAR, OUTPUTDEC	IMAL: F	AR	
Unknown Error #1"			101.	PARSEEXP	PROC	FAR	
ERR_INPUT_STR_UNK			ror:	:输入:DS:ST.表ì	大式的首	地址: DS:BX. 结果 64bit	
Unknown Error #2"			1.01.	;输入: DS:SI, 表达式的首地址; DS:BX, 结果 64bit			
ERR_INPUT_STR_UNK		,♪ DB "Er	non:	浮点数的首地址			
					PUSH	AX	
Unknown Error #3"			DD		PUSH	CX	
ERR_INPUT_STR_INCO			DB		PUSH	DX	
"Error: Incomplete			ectea		PUSH	ВР	
Arrangement of +/-		an, '\$'			PUSH	ES	
PAEX_DATA E	NDS				PUSHF		
					PUSH	BX	
PAEX_STACK SI	EGMENT	PRIVATE			. 05	2/1	
;堆栈结构					. +4++1		
	地 ム・	. 今世/2	- }			段和栈顶指针的保存不能	
;堆栈不与其他堆栈段	组合,則 80	H 子下作/:	J灰直区	再用堆栈了,要存	在本程序	有的数据区	
算符的堆栈区域,由9	SP 控制,以ī	两个字节为	5单位,		MOV	AX, SS	
					MOV	BX, Offset	
内容与数字的对应见了	N田 OP_削缀	的太正义	;	ORIG_SS	1101	<i>b</i> , 0115cc	
;后 80H 字节作为放置	双精度浮点	数的堆栈区	区域,由	0110_55	MOV	CX, PAEX DATA	
					MOV	ES, CX	
BP 控制, 增长方向与 9	SP 怕反 ,以	ハハチャ	刈甲 似。		MOV	ES:[BX], AX	
;初始时, SP与BP指	(向的区域均)	以 FF 字节	填满,		MOV	AX, SP	
						BX, Offset	
这样若用户输入的表达	区式刀垒 (或	仔仕错误,	为 行设	ORTC CD	MOV	bx, orrset	
置),返回时将返回栈顶	页浮点数 FFF	FFFFFFF	FFFFF,	ORIG_SP	MOV	FC.[DV] AV	
					MOV	ES:[BX], AX	
在 MAIN 中进行判定是					CLT		
	YTE	80H	DUP		CLI	A.V.	
(?)				D.E.V. CT.10V	MOV	AX,	
SP_BOTTOM EC	QU	\$		PAEX_STACK			
WC	ORD	OP_BOTT	OM_SP		MOV	SS, AX	
Al	LIGN	QWORD					
BP_BOTTOM EC	QU	\$			MOV	SP, Offset	
				SP_BOTTOM		;指示存储操作符用的	
B,	YTE	8H	DUP	_			
(0FFH)				堆栈头指针			
•	YTE	80H	DUP		MOV	BP, Offset	
(?)		'		BP_BOTTOM		;指示存储数字用的头	
• •	NDS			_		22-12-12-12-12-12-12-12-12-12-12-12-12-1	
277761				指针			
OP_LBRA EG	QU	0			STI		
J	~~	-					

				REPE	AT 2	;两次	
	;对与 SP 增长方向相反的 BP 定义压栈、弹出的			检测栈顶是否有数字,没有即错误;有即弹出数字,			
宏				再执行 FLD 压)		.,	
	PUSHBP	MACRO	STH	1337113	CMP	BP, BP_BOTTOM	
		ADD MOV	BP, TYPE STH [BP], STH		JBE	_	
	ENDM		[5,], 5,	ERR_INPUT_NO	_OPERAND		
	POPBP	MACRO	STH		POPBPFLD		
		MOV SUB	STH, [BP] BP, TYPE STH	ENDM			
	ENDM	305	bi , 1112 5111			16.3	
	PUSHBPFST	MACRO			PUSH	BX ;将之	
QWo	nd	ADD	BP, TYPE	前备份的返回地	妣再次压入,之后	的 EVAL_ONCE 过	
QWO	ıu	FST	QWord Ptr	程不会改变栈。			
SS:	[BP]				CALL	EVAL_ONCE	
	ENDM PUSHBPFSTP	MACRO			;POP ;PUSH	BX BX	
	FUSITOFFSTF	ADD	BP, TYPE		RETN	5A	
QWo	rd		,	5144			
· · ·	[pp]	FSTP	QWord Ptr	EVAL_ADD	: FADD		
33:	[BP] ENDM			IF	PAEX_DEBUG		
	POPBPFLD	MACRO			OUTCHAR	'+'	
66	[DD]	FLD	QWord Ptr	ENDI		\/A1	
55:	[BP]	SUB	BP, TYPE	EVAL SUB	<pre>JMP AFTER_E :</pre>	VAL	
Qwo	rd	302	5.,2	_	FSUBR		
	ENDM			IF	PAEX_DEBUG	1_1	
	40 . I . 			ENDI	OUTCHAR F		
	;输出字符	MACRO	CUD		JMP AFTER_E	VAL	
	OUTCHAR	MACRO PUSH	CHR DX	EVAL_MUL			
		PUSH	AX	IF	FMUL PAEX DEBUG		
		MOV	DL, CHR	Ξ.	OUTCHAR	'*'	
		MOV INT	AH, 02H 21H	ENDI			
		POP	AX	EVAL DIV	JMP AFTER_E	VAL	
		POP	DX	LVAL_DIV	FDIVR		
	ENDM			IF	PAEX_DEBUG		
		MOV	CX,	FNDT	OUTCHAR	'/'	
0H	;cx	在本过程中作	为一个全局量,指	ENDI	г JMP AFTER E	VAL	
示刚		个 TOKEN 是否为	5一个数字。		_		
75 45.	31332 VA3133		为是否存在"1 2"	EVAL_ONC		AV 02	
\ 14	*+A-53-84-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-				CMP JE	AX, 02 EVAL_ADD	
			到底表示加号减号		CMP	AX, 03	
还是	正负号提供依				JE	EVAL_SUB	
RFΔI	D TOKEN LOOP	JMP			CMP JE	AX, 04 EVAL MUL	
11271	5_10KEN_200.				CMP	AX, 05	
	;弹出栈顶的两	两个数和一个操	作符,执行浮点运		JE	EVAL_DIV	
算,	运算压入数栈。	若开启了 DEBI	JG, 按逆波兰符号	EDD TNDIT IIN	JMP		
	i出符号	** *		ERR_INPUT_UN AFTER_EV	-		
14H!	EVAL STACKT	OP:		_			
	_	POP	BX ;因为	IF	PAEX_DEBUG		
要田	到压在返回地		所以先弹出返回地 1111年	ENDI	OUTCHAR F		
	存在 BX 里面。	I HIM 1			PUSHBPFSTP	;运算结果再次	
놰,	∵江 ▷∧ 王凪。	POP	AX	压入栈).—) [-A/N] 3/V	
		. 0.		// \ \ \ \	RETN		

READ_TOKEN_	LOOP:	;主过程。不断移	AFTER_UNTIL	XOR	CX, CX
动指针 SI,检测每	一个字节表示的	り符号,采取操作		ADD JMP	SP, TYPE Word
	MOV INC	DL, [SI] SI	READ_TOKEN_LOOP	-	
	CMP JE	DL, ''	GET_PLUSSIGN:	测到 , 是 时	则下 CX,判断到
READ_TOKEN_LOOP					
	CMP	DL, '('	底是按正号(跳到		
	JE	GET_LBRA		OR	CX, CX
	CMP JE	DL, ')'		JZ	
	CMP	GET_RBRA DL, '+'	GET_FLOAT_SIGNE	D	;之前没有输入
	JE	GET PLUSSIGN	过数字,所以+当正	号处理	
	CMP	DL, '-'		;否则当加号	k##
	JE	,		-	
GET_MINUSSIGN				JMP	GET_ADDOP
	CMP	DL, '*'	<pre>GET_ADDOP:</pre>		
	JE	GET_MULOP	021_78501 .	MOV	DX,
	CMP	DL, '/'	OP_ADD		· 号和减号优先级
	JE	GET_DIVOP	_		
	CMP JE	DL, '.'	相同,故某些部分	重合。DX 指示的	自前运算的是加号
GET_FLOAT_SIGNE	-		还是减号		
GET_TECAT_SIGNE	CMP	DL, '\$'		JMP	
	JE	GET END SIGN	START_ADDSUBOP		
	CMP	DL, '0'	<pre>GET_SUBOP:</pre>		
	ЈВ	-		MOV	DX, OP_SUB
ERR_INPUT_UNEX_	TOKEN		START_ADDSU		
	CMP	DL, '9'		CMP	SP,
	JA		SP_BOTTOM	;栈	里没东西的话,直
ERR_INPUT_UNEX_			接将+号进栈		
CET FLOAT NOCTO	JMP			JE	
GET_FLOAT_NOSIG	IN		PUSH_OPSTACK_AD	DSUB	
CET LDDA.		. 松测点(1)		MOV	BX,
GET_LBRA:		;检测到(时,压	SP	;否	则先看一下栈顶
入栈即可。			的二等效 加田化生		
	MOV	AX, OP_LBRA	的运算符,如果优先	级(以问儿元级	的左结节性尤异)
	PUSH	AX	高于+号,先计算。		
	XOR	CX, CX		MOV	AX,
DEAD TOKEN LOOD	JMP		SS:[BX]	;当	然,因为只有
READ_TOKEN_LOOP			 +-*/, 并且都是左续	结合 所以不是	碰到(的话都会失
CET DDDA.					
GET_RBRA:		;检测到)时,不	计算的。		
断执行操作符栈顶的	的操作符,将括	号内的得数算出		SHR	AX, 1H
来,直到碰到(。之	左子子上			CMP	AX,
LOOP_OP_UNT			2		;第二优先级
20000	CMP	SP, SP_BOTTOM	*/, 对应4和5		
	JE	, _		JE	
ERR_INPUT_MISMA	T_PAR		EVAL_STACKTOP_A		
	MOV	BX, SP	_	CMP	AX,
	MOV	DX, SS:[BX]	1		;第一优先级
	CMP	DX, OP_LBRA	, ⊽ dicho ≇ ⊓o		,
ACTED UNITE 155	JE ^		+-, 对应 2 和 3	75	
AFTER_UNTIL_LBR	А		EVAL STACKTOD A	JE DDSLIR	
	CALL		EVAL_STACKTOP_A	CMP	AX,
EVAL STACKTOP	<i></i>		a	J	•
	JMP		0	75	;左括号
LOOP_OP_UNTIL_L	BRA?		DIICH UDCTACA VA	JE DSUR	
			PUSH_OPSTACK_AD	טטכט	

	JMP			CALL	
ERR_INPUT_UNKNO	_	;循环直到可以	EVAL_STACKTOP	JMP	
将+号进栈即可。			START_MULDIVOP	JIIIF	
PUSH_OPSTACK_ADDSUB:					
	PUSH XOR	DX CX, CX	GET_FLOAT_S	IGNED: MOV	DX,
	JMP	ch, ch	SI		是 SIGNED,那么
READ_TOKEN_LOOP		_	为了防止只出现一个		
EVAL_ST	ACKTOP_ADDSU CALL	В:	为了的正六山地	I + 5 以・5 戸凪 JMP	以为数于四周次
EVAL_STACKTOP	CALL		GET_FLOAT	3111	;需要将 SI 复制
CT.DT .DDC.UDOD	JMP		一份,然后比对读耳	7数据之后的 0	
START_ADDSUBOP			有走动,则判定为		Λ ήμ эт。 1 эт /文
GET_MINUSSI	GN:		何になり、火リナリたノジキ GET_FLOAT_N		
	OR	CX, CX	GET_TEGAT_N	MOV.	DX, SI
GET FLOAT SIGNE	JZ D			DEC	
021_1 20/11_310H2	JMP	GET_SUBOP	DX		;由于在主过程
			时,读取后 SI 马上	加一,此时复制	制到的 DX 是往右
GET MULOP:			偏移一位的,如果是	没带符号的数字	字的话要修正过来
_	除和加减基本相	1同,只是优先级	<pre>GET_FLOAT:</pre>		
不同而已。	STADAWAL TO	41.37 7 (2010) 0-22		OR JNZ	CX, CX
، ۱۹۳۱۱۱۰۰	MOV	DX, OP_MUL	ERR_INPUT_SYNTA	-	
	JMP	27., 0		_	
START_MULDIVOP				DEC	
GET_DIVOP:	MOV	DX, OP DIV	SI	400	;同上,修正过来
		DA, 0DI	Qword	ADD	BP, TYPE
START_MULDI		CD CD DOTTON	Quo. u		
	CMP JE	SP, SP_BOTTOM		MOV	BX, BP
PUSH_OPSTACK_MU				PUSH MOV	ES AX, SS
	MOV	BX, SP		MOV	ES, AX
	MOV SHR	AX, SS:[BX] AX, 1H		CALL	
	CMP	AX,	INPUTDECIMAL		;备份、设置相应
2		;第二优先级	的段寄存器,调用等	俞入数字的过程	!
*/, 对应4和5				POP	ES
	JE			NOT	CX
EVAL_STACKTOP_M		AV		CMP	DX, SI
1	CMP	AX, ;第一优先级	EDD TNDUT TNCOM	JAE	
		, 另 1/6元级	ERR_INPUT_INCOM	PLETE_NUMBER	
+-, 对应 2 和 3	JE		IF PAE	X_DEBUG	
PUSH_OPSTACK_MU		;与加减不		DUCH	DC
		۱۰ کیا//۱۱۸ ات⊸ و		PUSH PUSH	DS SI
同,此处可以直接特	分兆陈亏压伐。 CMP	AX,		PUSH	AX
0	CMF	, ;左括号		MOV	AX, SS
0	JE	,在旧与		MOV MOV	DS, AX SI, BP
PUSH_OPSTACK_MU				CALL	, -·
EDD TAIDUT LINUS	JMP		OUTPUTDECIMAL		;按逆波兰符号
ERR_INPUT_UNKNO	WN_2 STACK_MULDIV	•	法输出数字		
. 03/1_0/	PUSH	DX		POP	AX
	XOR	CX, CX		POP	SI
READ TOKEN LOOP	JMP			POP	DS
	ACKTOP_MULDI	v:		OUTCHAR	1 1
			ENDIF		

```
OR
                                                                         DL, DL
                                                                         MOV FF
                JMP
                                                            JNZ
READ_TOKEN_LOOP
                                                            REP
                                                                         MOVSB
                                                            JMP
                                                                         AFTER_MOV
                                                MOV_FF:
                                                            REP
                                                                         STOSB
    GET_END_SIGN:
                                                AFTER_MOV:
                XOR
                            DL,
                                                            POP
                                                                         ES
            ;获取到 endsign $后,设置一下 DL
DL
                                                            POP
                                                                         DS
表示是否出现错误
    GET_END_SIGN_LOOP:
                CMP
                            SP,
                                                            POPF
            ;要执行操作符栈存留的所有运算
SP BOTTOM
                                                            POP
                                                                         ES
                            END_PARSING
                JΕ
                                                            POP
                                                                         ВР
                            BX, SP
AX, SS:[BX]
                MOV
                                                            POP
                                                                         \mathsf{DX}
                MOV
                                                            POP
                                                                         \mathsf{CX}
                CMP
                            ΑX,
                                                            POP
                                                                         AX
            ;若碰到左括号,说明有括号没有闭
OP_LBRA
                                                            RET
合。
                                                ERR_INPUT_SHOW
                                                                         MACRO
                JE
ERR INPUT MISMAT PAR
                                            ERR_STR
                                                        ;定义错误宏。错误时统一将 DL 设置
                                            成 FF 返回
                CALL
                                                    ERR_INPUT_&ERR_STR :
EVAL_STACKTOP
                                                        PAEX_DEBUG
                                                    ΙF
                JMP
                                                                         0DH
                                                            OUTCHAR
GET_END_SIGN_LOOP
                                                            OUTCHAR
                                                                         0AH
                                                    ENDIF
    END_PARSING:
                                                            PUSH
                                                                         DS
    IF PAEX_DEBUG
                                                                         AX, PAEX_DATA
                                                            MOV
                OUTCHAR
                            0DH
                                                            MOV
                                                                         DS, AX
                            0AH
                OUTCHAR
    ENDIF
                                                            MOV
                                                                         DX, Offset
                                            ERR_INPUT_STR_&ERR_STR
                                 ;将堆栈段
CLI
                                                            MOV
                                                                         AH, 09H
                                                            TNT
                                                                         21H
寄存器复位
                                                            POP
                                                                         DS
                            BX, Offset
                MOV
ORIG_SP
                                                            MOV
                                                                         DL,
                MOV
                            AX, ES:[BX]
                                            0FFH
                MOV
                            SP, AX
                                                        ;表示出现错误
                MOV
                            BX, Offset
                                                            JMP
                                                                         END_PARSING
ORIG_SS
                                                ENDM
                MOV
                            AX, ES:[BX]
                MOV
                            SS, AX
                                                ERR_INPUT_SHOW
                                                                         MISMAT_PAR
                STI
                                                ERR_INPUT_SHOW
                                                                         NO OPERAND
                CLD
                                                ERR_INPUT_SHOW ERR_INPUT_SHOW
                                                                         SYNTAX ERROR
                                                                         UNEX_TOKEN
                                                ERR_INPUT_SHOW
                POP
                                                                         UNKNOWN_1
                                                ERR_INPUT_SHOW
                                                                         UNKNOWN 2
                    ;此时可以 POP 出目标地址
BX
                                                ERR INPUT SHOW
                                                                         UNKNOWN 3
                PUSH
                            DS
                                                ERR INPUT SHOW
                PUSH
                            ES
                                            INCOMPLETE NUMBER
                MOV
                            AX, DS
                MOV
                            ES, AX
                                            PARSEEXP
                                                            ENDP
                MOV
                            AX,
                                            PAEX_CODE
                                                            ENDS
PAEX STACK
                                            END
                MOV
                            DS, AX
                MOV
                            SI, BP
                MOV
                            DI,
                ;将原数字栈的栈顶一个字节一
BX
个字节地 MOVSB 到目标地址, 共 8 次
                MOV
                            CX, 8H
                MOV
                            ΑL,
0FFH
                ;若出现异常 (DL 非 0), 返回
OFFFFFFFFFFFFF
```

文件名称	ID.ASM
子过程名称	INPUTDECIMAL
描述	负责将十进制小数字符串转换为双精度

		2 12 17 7 7 17 7 7	• • • • • • • • •	* * * * * * * * * * * * * * * * * * * *	714.5
PUBLIC	INPUTD	ECIMAL		PUSH	DX
DECM_DATA	SEGMI	ENT		PUSH	DI
BCDSPC				PUSHF	
DEBUGS		?		PUSH	ES
	AVE2 DQ			PUSH	DS
				PUSH	
	iING_ =			РОЗП	DA
	TR DW	<i>!</i>			11
DECM_DATA	ENDS		40 == 10 HI -A-		AX, DECM_DATA ;先将 DS
			和 ES 设置成		
DECM_CODE		SEGMENT		MOV	ES, AX
ISTHISDIGITV	ALID:			MOV	BX, SI ; 判断是
:判断这个位:	是否还是	该小数的内容。判断依据:	否有符号"+		艮据情况将 DH 置 1(NEGATIVE)
		小数点(1),也有可能不出现(0),	或 0	, , , , , 1 1	KATHACA, DIT E. I(NEGATIVE)
记录在 AH 中	1			MOV	DL, [BX]
; 存储小数	的字符在	内存中的起始位置,记录在 BX		CMP	
中				JE	SIGNPOSITIVE
· 返回 DI. 指	示该位是	是否是小数的内容;返回 AH,指		CMP	DL, '-'
示小数点是否		E 1 7 7 11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		JE	
		廿小数之后的终止位置的下一个		MOV	
•	小以果是	1小数之后的公正位直的下一个		PUSH	•
字节	_			JMP	
		DL, [BX]		JIVIP	GOTOENDOPSTK
		DL, '0'	CICNIDOC	ITIVE.	
	JAE	DIGITNEXT	SIGNPOS		DII 011
DIGITNO	TNUM:			MOV	DH, OH
	CMP	DL, '.'		PUSH	DX
	JE	DIGITDOT		INC	BX
INVALID:	:MOV	DL, 0H		INC	SI
	RET			JMP	GOTOENDOFSTR
DIGITDO	T:		SIGNNEG	ATIVE:	
	:若是第	一次碰到点,CX+1,因为这个		MOV	DH, 1H
点不算数字,				PUSH	DX
W.1 77 X 1 1	INC	CX		INC	BX
	OR			INC	SI
		AH, AH			
	JNZ	INVALID			
		AH, 1H		· 抬针;	走到最后一个符号
	JMP	VALID	GOTOENI		
			GOTOLIVI	MOV	CX, 18D
DIGITNE				XOR	AH, AH
	CMP	DL, '9'	GOTOENI		•
	JA	DIGITNOTNUM	GOTOLINI	CALL	ISTHISDIGITVALID
VALID:	MOV	DL, 1H			
	RET			OR	DL, DL
				JZ	HEREISENDOFSTR
INPUTDECIMA	AL PROC I	FAR		INC	BX
		首地址; ES:BX 64 位浮点数的目		LOOP	GOTOENDOFSTR_LOOP
标内存首地均			HEREISEN	IDOESTD	
;输出: 在 ES	:BX 输出		HENEISEN	PUSH	 DS
	ASSUMI	E DS:DECM_DATA,			
CS:DECM_CO	DE			MOV	AX, DECM_DATA
_	PUSH	AX		MOV	DS, AX
	PUSH	BX		MOV	ENDOFSTR, BX
	PUSH	CX		POP	DS
		-			

```
MOV
                    AX, Offset BCDSPC
                                                                 GIVING_MSBSIGN
                                                          JΖ
           MOV
                    DI, AX
                                                          OR
                                                                  AL, 80H
           XOR
                   AX, AX
                                                  GIVING_MSBSIGN:
           XOR
                   DH, DH
                                                          STOSB
                                                          MOV
                                                                   AX, DECM_DATA
                                                          MOV
           MOV
                    CX, 18D + 1D
                                     :最多支
                                                                   DS, AX
持 18 个有效位, 当然精确后没有
           CLD
                                                  ;开始将 10 字节的 BCD 转换为实数
           JMP
                   SENDTOSPACE_LOOP?
                                                          FBLD
                                                                  TByte Ptr BCDSPC
   SENDTOSPACE_LOOP:
                                                  IF DEBUGGING EQ 1
                                                          PUSH
           DEC
           MOV
                    DL, [BX]
                                                          MOV
                                                                   AX, DECM_DATA
                                                          MOV
                                                                   DS, AX
           CMP
                    DL, '.'
                                                          FST
                                                                 QWord Ptr DEBUGSAVE
           JE
                   SENDTOSPACE_LOOP?_DOT
                                                          POP
                                                                  DS
           AND
                    DL, 0FH
           OR
                   AH, AH
                                                  ENDIF
           JNZ
                   UPPERPART
                                                  ;TODO:
                                                         除法
           MOV
                    AL, DL
                                                          MOV
                                                                   CX, 10D
           NOT
                    AΗ
           JMP
                   SENDTOSPACE_LOOP?
                                                          MOV
                                                                   DI, Offset BCDSPC
   UPPERPART:
                                                          XOR
                                                                  AL, AL
                                                          REP
                                                                  STOSB
           PUSH
                   CX
           MOV
                    CL, 4H
           SHL
                   DL, CL
                                                          TEST
                                                                  DH, 1H
           POP
                   \mathsf{CX}
                                                          JΖ
                                                                 DH_EVEN
           OR
                   AL, DL
                                                          MOV
                                                                   AL, 10H
           STOSB
                                                          JMP
                                                                  GEN_DIVISOR
           ;载入完一个字节后,送入 SPACE,相
                                                  DH_EVEN:
关指针增
                                                          MOV
                                                                   AL, 01H
                                                  GEN_DIVISOR:
           XOR
                   AX, AX
                                                          SHR
                                                                  DH, 1H
                                                          MOV
                                                                  DI, Offset BCDSPC
   SENDTOSPACE_LOOP?:
                                                          PUSH
                                                                  AX
           CMP
                    BX, SI
                                                          MOV
                                                                   AL, DH
           LOOPNE SENDTOSPACE_LOOP
                                                          CBW
           JMP
                   PADDING_ZEROS
                                                          ADD
                                                                  DI, AX
                                                          POP
                                                                  \mathsf{AX}
   SENDTOSPACE_LOOP?_DOT:
                                                          STOSB
           ;碰到小数点后,记录一下出现的位置
于 DH, 即小数点后有多少位(没出现时, DH 为 0)
           MOV
                    DH, CL
                                                          FBLD
                                                                  TByte Ptr BCDSPC
           SUB
                   DH. 18D
                                                  IF DEBUGGING_ EQ 1
           NEG
                   DH
                                                          PUSH
           CMP
                    BX, SI
                                                          MOV
                                                                   AX, DECM_DATA
                   SENDTOSPACE LOOP
           JNE
                                                          MOV
                                                                   DS, AX
                                                                   QWord Ptr DECM_DATA:
                                                          FST
   PADDING ZEROS: ;若有 AL 剩余,填入 SPACE;
                                              DEBUGSAVE2
高位补 0 补到 10 个字节
                                                          POP
                                                                  DS
           OR
                   AH, AH
                  PADDING START
           JΖ
                                                  ENDIF
                    AL, OFH
           AND
                                                          FDIV
                                                                             ;此时 ST(0)存有
           STOSB
                                              目标数
           DEC
                   CX
   PADDING START:
                                                          POP
                                                                  ВХ
           SAR
                   CX, 1H
                                                          POP
                                                                  DS
                                                          POP
                                                                  ES
           XOR
                   AL, AL
                                                                 QWord Ptr ES:[BX]
                                                          FSTP
                   STOSB
           REP
                                    ;此时 BH
           POP
                    BX
                                                          PUSH
                                                                  DS
记录该数正负性
                                                          MOV
                                                                  AX, DECM_DATA
```

OR

BH, BH

MOV DS, AX MOV SI, ENDOFSTR POP DS POPF POP DI POP DX POP CX POP ВХ POP АХ RET INPUTDECIMAL ENDP DECM_CODE ENDS

END

文件名称	OD.ASM
子过程名称	OUTPUTDECIMAL
描述	负责输出双精度小数

PUBLIC	OUTPUTD	OUTPUTDECIMAL		;开始找 BCD 的	的最高位、最低位
OUTD_DATA	SEGMENT		有效数字		
PRECISIO	UQ3 NC	6 ;一定要是	BCDSPC	MOV	BX, Offset
偶数 MUST BE	EVEN			MOV	CX, 0H
BIGDECM	-	1000000.0			
BCDSPC	DT	;		MOV	MOSTSIG, ØD
MOSTSIG		,		MOV	LESTSIG, 17D
LESTSIG		;	_,,_,_		
OUTD_DATA	ENDS		EVENBIT		CV 111
OUTD CODE	CECMENT			SHR	CX, 1H DI, CX
OUTD_CODE	SEGMENT	CS:OUTD_CODE,		MOV AND	DI, CX DI, 0FH
DS:OUTD_DATA		C3.001D_C0DE,		MOV	DL, [BX][DI]
D3.001D_DA17	٦			SHL	CX, 1H
				TEST	DL, 0FH
OUTPUTD	ECIMAL	PROC FAR		JNZ	CHKUPDATESIG
	;输出一个小数		CHKEND_		
			_	INC	CX
	;输入: SI, 6	4bit 小数的偏移			
地址。			ODDBIT:		
-0-11	. 工学投业共用	∃ .		TEST	DL, 0F0H
	;正常输出范围	₫.		JNZ	CHKUPDATESIG
-100000~-0.0			CHKEND_		
0.00000001~				INC	CX
	;输出格式: 聶	是多 8 位小数,如		CMP	CX, 17D
25.00152225	. 33.441.			JBE	EVENBIT
	PUSH	AX		JMP	AFTERSETSIG
	PUSH	BX	CHKUPDA	TECTC.	
	PUSH	CX	CHRUPDA	CMP	MOSTSIG, CL
	PUSH	DX		JNB	MOSTSIG, CL
	PUSHF		CHKUPDATELE		
			Cintol Bittell	MOV	MOSTSIG, CL
	PUSH	DS	CHKUPDA	TELESTSIG:	
				CMP	LESTSIG, CL
	FLD	QWord Ptr		JNA	
[SI]			CHKUPDATELE	STSIG_END	
	MOV	AX, OUTD_DATA		MOV	LESTSIG, CL
	MOV	DS, AX	CHKUPDA	TELESTSIG_EN	
	FMUL	QWord Ptr		TEST	CL, 1H
		_		JZ	CHKEND_EVEN
BIGDECM	;非一日力,	就可以先化成整		JMP	CHKEND_ODD
数			;此时 MO	STSIG和LESTS	SIG 分别储存小数
	FBSTP	BCDSPC ;	的最高位和最		
计每组型 pcp	. 5511	, ,			
计算得到 BCD			AFTERSE	CMP	MOSTSTG GD
				JAE	MOSTSIG, 6D
			AFTER_CORRE		

```
MOV
                          MOSTSIG, 6D
                                                         PUSH
                                                                      DX
                                                                      DL, '.'
    AFTER_CORRECT_MOSTSIG:
                                                         MOV
                                                         MOV
                                                                      AH, 02H
             CMP
                           LESTSIG, 6D
             JBE
                                                         INT
                                                                      21H
AFTER_CORRECT_LESTSIG
                                                         POP
                                                                      DX
                           LESTSIG, 6D
             MOV
                                                DIRECT_PRINTDIGIT:
    AFTER_CORRECT_LESTSIG:
                                                         PRINTDIGIT
                                                                      DL
    ;开始打印
                                                         PUSH
                                                                      CX
    PRINTDIGIT
                          MACRO
                                                         MOV
                                                                      CL, 4H
digit
                                                                      DL, CL
                                                         ROR
             PUSH
                          \mathsf{DX}
                                                                      \mathsf{CX}
                                                         POP
                          DL, digit
DL, 0FH
             MOV
             AND
                                                         DEC
                                                                      CL
                          DL, 30H
             OR
                                                                      CL, LESTSIG
                                                         CMP
             MOV
                          AH, 02H
                                                         JL
                           21H
             INT
                                           PRINTDIGIT_END
             POP
                          DX
                                                PRINTDIGIT_EVEN:
    ENDM
                                                         PRINTDIGIT
    ;先打印符号
                                                         DEC
                                                                      CL
             MOV
                          DL, 9[BX]
                                                         CMP
                                                                      CL, LESTSIG
                          DL, 80H
             TEST
                                                         JNL
                           PRINTDIGITS
             JΖ
                                           PRINTDIGIT_ODD
             PUSH
                          DX
                          DL, '-'
             MOV
                                                PRINTDIGIT_END:
                           AH, 02H
             MOV
             INT
                           21H
                                                         POP
                                                                      DS
             POP
                          DX
                                                         POPF
                                                         POP
                                                                      DX
    PRINTDIGITS:
                                                         POP
                                                                      CX
                          CX, CX
             XOR
                                                         POP
                                                                      BX
             MOV
                          CL, MOSTSIG
                                                         POP
                                                                      AX
                          CL, 1H
             TEST
                                                         RET
             JNZ
PRINTDIGIT_ODD
                                                OUTPUTDECIMAL
                                                                      ENDP
                          CX, 1H
DI, CX
CX, 1H
             ROR
             MOV
                                           OUTD_CODE
                                                                      ENDS
             ROL
                                           END
             AND
                          DI, 0FH
             MOV
                          DL, [BX][DI]
             JMP
PRINTDIGIT_EVEN
    PRINTDIGIT ODD:
             PUSH
                           \mathsf{CX}
                           CX, 1H
             SHR
                           DI, CX
             MOV
                          DI, 0FH
             AND
                          DL, [BX][DI]
             MOV
                          CL, 4H
             MOV
             ROR
                          DL, CL
             POP
                          \mathsf{CX}
             CMP
                          CL, PRECISION
- 1D
             JNZ
DIRECT_PRINTDIGIT
```