

山东大学 计算机科学与技术 学院

汇编语言 课程实验报告

学号：202200130053	姓名：陈红瑞	班级：3 班
实验题目：实验 2：例 2.4，2.5		
实验学时：2	实验日期：20241021	
实验目的：继续熟悉 MASM、LINK、DEBUG、EDIT、TD 等汇编工具。掌握汇编语言分支程序的设计思路。理解条件转移指令和无条件转移指令的机理（哪个是根据状态寄存器，哪个是位移，哪个是段首址加有效地址）。了解段内短转移、段内近转移、段间转移、直/间接等的含义及其跳转范围。		
实验环境：Windows11、DOSBox-0.74、Masm64		
源程序清单：  1. 2_4.asm （示例 2.4）  2. 2_5.asm （示例 2.5）		
编译及运行结果：  1. 例 2.4 编译和运行的结果如下图：		

```

C:\>masm 2_4:
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

51688 + 464856 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link 2_4:
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

LINK : warning L4021: no stack segment

C:\>2_4
month?
1
JAN
month?

```

根据输入的数字来判断是哪一个月。如果没有输出，直接按下回车，会结束程序。

代码的数据段部分的定义如图，这里定义了偏移量为 3，并定义了一个表 montab，包含了所有月份的字符串

	segment	para	'data'
three	db	3	
mess	db	'month?',13,10,'\$'	
monin	label	byte	
max	db	3	
act	db	?	
mon	db	3 dup(?)	
alfmon	db	'???' ,13,10,'\$'	
montab	db	'JAN','FEB','MAR','APR','MAY','JUN'	
	db	'JUL','AUG','SEP','OCT','NOV','DEC'	
	ends		

下面对代码进行调试，首先通过 INT 指令来将 month? 输出。

```

C:\>debug 2_4.exe
-u
076E:0000 1E          PUSH    DS
076E:0001 2BC0          SUB     AX,AX
076E:0003 50          PUSH    AX
076E:0004 B86A07        MOV     AX,076A
076E:0007 8ED8          MOV     DS,AX
076E:0009 8EC0          MOV     ES,AX
076E:000B 8D160100      LEA     DX,[0001]
076E:000F B409          MOV     AH,09
076E:0011 CD21          INT     21
076E:0013 8D160A00      LEA     DX,[000A]
076E:0017 B40A          MOV     AH,0A
076E:0019 CD21          INT     21
076E:001B B20D          MOV     DL,0D
076E:001D B402          MOV     AH,02
076E:001F CD21          INT     21

```

然后，通过 INT 指令执行 0A H 对应的操作，即读取字符。

```

DOS
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
AX=096A BX=0000 CX=00B1 DX=000A SP=FFFC BP=0000 SI=0000 DI=0000
DS=076A ES=076A SS=0769 CS=076E IP=0017 NU UP EI PL ZR NA PE NC
076E:0017 B40A          MOV     AH,0A
-t
AX=0A6A BX=0000 CX=00B1 DX=000A SP=FFFC BP=0000 SI=0000 DI=0000
DS=076A ES=076A SS=0769 CS=076E IP=0019 NU UP EI PL ZR NA PE NC
076E:0019 CD21          INT     21
-t
AX=0A6A BX=0000 CX=00B1 DX=000A SP=FFF6 BP=0000 SI=0000 DI=0000
DS=076A ES=076A SS=0769 CS=F000 IP=14A0 NU UP DI PL ZR NA PE NC
F000:14A0 FB          STI
-t
AX=0A6A BX=0000 CX=00B1 DX=000A SP=FFF6 BP=0000 SI=0000 DI=0000
DS=076A ES=076A SS=0769 CS=F000 IP=14A1 NU UP EI PL ZR NA PE NC
F000:14A1 FE38          ???     [BX*SI]          DS:0000=03
-t
3
AX=0A6A BX=0000 CX=00B1 DX=000A SP=FFF6 BP=0000 SI=0000 DI=0000
DS=076A ES=076A SS=0769 CS=F000 IP=14A5 NU UP EI PL ZR NA PE NC
F000:14A5 CF          IRET

```

当输入完成后，在数据段的 000ch 位置存储的是输入的数字，下面需要将这个数字分别存入寄存器的低位和高位。

```

-d0c
076A:0000                                     31 30 0D 3F                                     10.?
076A:0010 3F 3F 0D 0A 24 4A 41 4E-46 45 42 4D 41 52 41 50  ??..$JANFEBMARAP
076A:0020 52 4D 41 59 4A 55 4E 4A-55 4C 41 55 47 53 45 50  RMAYJUNJULAUSEP
076A:0030 4F 43 54 4E 4F 56 44 45-43 00 00 00 00 00 00 00  OCTNOUDEC.....
076A:0040 1E 2B C0 50 B8 6A 07 8E-D8 8E C0 8D 16 01 00 B4  .+.P.j.....
076A:0050 09 CD 21 8D 16 0A 00 B4-0A CD 21 B2 0D B4 02 CD  !.....!.....
076A:0060 21 B2 0A B4 02 CD 21 80-3E 0B 00 00 74 42 B4 30  ?!...!.>...tB.0
076A:0070 80 3E 0B 00 02 74 06 A0-0C 00 EB 08 90 A0 0D 00  .>...t.....
076A:0080 8A 26 0C 00 35 30 30 80-FC 00 74 04          .&...500...t.

```

下面将寄存器 AX 的高位设置为 30h，用于将输入部分将 ASCII 转化为数

字。

```
-t
AX=020A BX=0000 CX=00B1 DX=000A SP=FFFC BP=0000 SI=0000 DI=0000
DS=076A ES=076A SS=0769 CS=076E IP=002E  NU UP EI PL NZ NA PO NC
076E:002E B430          MOV     AH,30
-u
076E:002E B430          MOV     AH,30
076E:0030 803E0B0002    CMP     BYTE PTR [000B],02
076E:0035 7406          JZ      003D
076E:0037 A00C00        MOV     AL,[000C]
076E:003A EB08          JMP     0044
076E:003C 90            NOP
076E:003D A00D00        MOV     AL,[000D]
076E:0040 8A260C00      MOV     AH,[000C]
076E:0044 353030        XOR     AX,3030
076E:0047 80FC00        CMP     AH,00
076E:004A 7404          JZ      0050
076E:004C 2AE4          SUB     AH,AH
```

如图，下面根据变量 act 的值来确定是否需要将 AH 中的值设置为 1，然后通过异或运算将 ASCII 值转化为数字，其中 AH 存储十位，AL 存储个位，然后判断高位是否为 0，即这个数字是否为 1-9，如果不是就将高位置为 0 并在低位加 10，然后将低位的值减 1 再乘 3 作为表中的偏移量，根据这个偏移量再读取 3 个字符，并将结果输出。

```
-t
AX=300A BX=0000 CX=00B1 DX=000A SP=FFFC BP=0000 SI=0000 DI=0000
DS=076A ES=076A SS=0769 CS=076E IP=003D  NU UP EI PL ZR NA PE NC
076E:003D A00D00        MOV     AL,[000D]          DS:000D=30
-t
AX=3030 BX=0000 CX=00B1 DX=000A SP=FFFC BP=0000 SI=0000 DI=0000
DS=076A ES=076A SS=0769 CS=076E IP=0040  NU UP EI PL ZR NA PE NC
076E:0040 8A260C00      MOV     AH,[000C]          DS:000C=31
-t
AX=3130 BX=0000 CX=00B1 DX=000A SP=FFFC BP=0000 SI=0000 DI=0000
DS=076A ES=076A SS=0769 CS=076E IP=0044  NU UP EI PL ZR NA PE NC
076E:0044 353030        XOR     AX,3030
-t
AX=091B BX=0000 CX=0000 DX=000F SP=FFF6 BP=0000 SI=0033 DI=0012
DS=076A ES=076A SS=0769 CS=F000 IP=14A1  NU UP EI PL NZ AC PE NC
F000:14A1 FE38          ???     [BX+SI]          DS:0033=4E
-t
DCT
```

## 2. 例 2.5

例 2.5 的编译和运行结果如图。

```

C:\>masm 2_5;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.

51690 + 464854 Bytes symbol space free

0 Warning Errors
0 Severe Errors

C:\>link 2_5;
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

LINK : warning L4021: no stack segment

C:\>2_5
Error code is not in valid range(1-83)

```

下面对这个程序进行调试。

代码中先将 076A, 即数据段的首地址作为 DS 的值, 然后将 AX 的初始值 FFFFH 直接与 0053H 进行比较, 这个值大于 83, 因此, 执行的是 0\_0\_R 部分, 会输出 Error code is not in valid range(1-83)。

```

AX=FFFF BX=0000 CX=03F6 DX=0000 SP=FFFA BP=0000 SI=076A DI=0000
DS=076A ES=075A SS=0769 CS=07A5 IP=000A NU UP EI PL ZR NA PE NC
07A5:000A 3D5300      CMP     AX,0053
-u
07A5:000A 3D5300      CMP     AX,0053
07A5:000D 7F05        JG      0014
07A5:000F 3D0000      CMP     AX,0000
07A5:0012 7F06        JG      001A
07A5:0014 8D160000    LEA     DX,[0000]
07A5:0018 EB26        JMP     0040
07A5:001A 3D2300      CMP     AX,0023
07A5:001D 7F08        JG      0027
07A5:001F 8D1E5A03    LEA     BX,[035A]
07A5:0023 4B          DEC     AX
07A5:0024 EB14        JMP     003A
07A5:0026 90          NOP
07A5:0027 3D4F00      CMP     AX,004F

```

下面对寄存器的内容进行修改, 如这里将 ax 改为 0050h, 再将这个值与 83 比较, 输出的结果变为 File exists.

```

-rax
AX FFFF
:0050
-r
AX=0050 BX=0000 CX=03F6 DX=0000 SP=FFFA BP=0000 SI=076A DI=0000
DS=076A ES=075A SS=0769 CS=07A5 IP=000A NU UP EI PL ZR NA PE NC
07A5:000A 3D5300      CMP     AX,0053

```

```

AX=0050 BX=0000 CX=03F6 DX=0000 SP=FFFC BP=0000 SI=076A DI=0000
DS=076A ES=075A SS=0769 CS=07A5 IP=0009  NU UP EI PL ZR NA PE NC
07A5:0009 50          PUSH    AX
-g45
File exists

```

下面对 2.5 的代码进行修改，使其能够实现从键盘输入一个数字并进行错误输入。

首先，同 2.4 中对数据段的定义，这里定义了一个用于存储输入的变量，其中 MAX 和 ACT 分别表示支持的最大输入字符数和实际输入字符数，然后将数字读取到 NUM 中。

	NUMIN	LABEL	BYTE
61	NUMIN	LABEL	BYTE
62	MAX	DB	3
63	ACT	DB	?
64	NUM	DB	3 DUP(?)
65	;		

下面再增加输入部分的代码，首先获取到 NUMIN 部分的地址，然后用 INT 指令将输入存储到缓冲区，并读取换行符。并且通过将输入的长度与 0 比较，如果没有输入就直接退出。

		INPUT:
LEA	DX, NUMIN	
MOV	AH, 0AH	;get input
INT	21H	
MOV	DL, 13	
MOV	AH, 02	
INT	21H	
MOV	DL, 10	
MOV	AH, 02	; get LF CR
INT	21H	
CMP	ACT, 0	; have no input
JE	EXIT	

下面执行对输入数字的转换，首先将 AX 中存储两位数的 ASCII 码，如果没有十位，就默认为 0，然后分别将两个数字存入 AX 的高位和低位，并转换为二进制数字类型。

并将高位  $\times 10$  加到低位，将这个结果作为 AX 最终的值并用于后面的判断。

```

MOV     AH, 30H
CMP     ACT, 2
JE      TWO
MOV     AL, NUM
JMP     CONU
TWO:    MOV     AL, NUM + 1
        MOV     AH, NUM
CONU:   XOR     AX, 3030H      ;ascii to binary
        MOV     BX, AX
        XOR     AX, 0         ;CLEAR AX
        MOV     AL, 10
        MUL     BH            ;TIMES 10
        ADD     AL, BL        ;CONVERT

```

修改后的程序的运行结果如图。

```

C:\>2_5
10
Invalid environment

```

### 问题及收获：

1. 掌握了汇编代码的分支结构的实现方法，并通过例 2.4 了解了对于输入，循环结构的实现方式。
2. 借助例 2.4，可以实现对例 2.5 的程序进行改写，实现了将输入的数据计算和转换为二进制数的方法，可以将两位十进制输入转换为二进制数字并用于后面的计算。