

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 4：例 2.6，2.7		
实验学时：2	实验日期：20241104	
实验目的： 全面掌握汇编语言中的过程及其使用，进一步实践编程方法论。掌握通过全局变量、栈和寄存器传递过程参数与返回值的方法。掌握过程的模块化设计，及其嵌套与测试方法。掌握字节、字与双字数据的基本计算方法，以及部分系统调用。		
实验环境：Windows11、DOSBox-0.74、Masm64		
源程序清单： 1. 2_6.asm （示例 2.6） 2. 2_7.asm （示例 2.7）		
编译及运行结果： 1. 例 2.6 程序的数据段定义如图，这里将输入的的成绩的值存入到 grade 中，并将后面计算出的名次信息存储到 rank 中，count 变量用来计算输入的的成绩数量。mess1 用于在输入成绩之前输出提示内容，mess2 用于当输入非法内容后结束输入时输出的语句。mess3 是在显示名次的时候输出的内容。		

```

1 ;PROGRAM TITLE GOES HERE -- RANK
2 ;*****
3 dataarea      segment                ;define data segment
4     grade     dw      50 dup(?)
5     rank      dw      50 dup(?)
6     count     dw      ?
7     mess1     db      'Grade? $'
8     mess2     db      13,10,'Input Error!',13,10,'$'
9     mess3     db      'Rank: $'
10 dataarea     ends
11 ;*****

```

在代码段，main 部分的代码如图，程序从 start 开始，依次执行 input, rankp 和 output 三个部分。

```

main          proc      far                ;main part of program
              assume cs:program,ds:dataarea
start:
;set up stack for return
              push      ds                ;save old data segment
              sub       ax,ax             ;put zero in AX
              push      ax                ;save it on stack
;set DS register to current data segment
              mov       ax,dataarea       ;dataarea segment addr
              mov       ds,ax             ;into DS register
;MAIN PART OF PROGRAM GOES HERE
              call      input
              call      rankp
              call      output
              ret
main          endp

```

Input 部分用于数据输入。如下图，进入到 input 部分后，这里先使用 int 21h 的调用，其中 09h 表示输出，按照其对应数据段的地址，可以看出这里要输出的是提示内容。这里还调用的另外一部分的函数。

```

- u
0779:0013 8D16CA00      LEA     DX,[00CA]
0779:0017 B409          MOV     AH,09
0779:0019 CD21          INT     21
0779:001B BE0000      MOV     SI,0000
0779:001E C706C8000000    MOV     WORD PTR [00C8],0000
0779:0024 E87C00      CALL    00A3
0779:0027 FF06C800      INC     WORD PTR [00C8]
0779:002B 80FA2C      CMP     DL,2C
0779:002E 7407          JZ      0037
0779:0030 80FA0D      CMP     DL,0D
- dca
076A:00C0                                47 72 61 64 65 3F      Grade?
076A:00D0 20 24 0D 0A 49 6E 70 75-74 20 45 72 72 6F 72 21    $..Input Error!
076A:00E0 0D 0A 24 52 61 6E 6B 3A-20 24 00 00 00 00 00    ..$Rank: $.
076A:00F0 1E 2B C0 50 B8 6A 07 8E-D8 E8 07 00 E8 41 00 E8  .+.P.j.....A..
076A:0100 6B 00 CB 8D 16 CA 00 B4-09 CD 21 BE 00 00 C7 06    k.....!.....
076A:0110 C8 00 00 00 E8 7C 00 FF-06 C8 00 80 FA 2C 74 07    ....i.....t.
076A:0120 80 FA 0D 74 13 75 09 89-9C 00 00 83 C6 02 EB E4    ...t.u.....
076A:0130 8D 16 D2 00 B4 09 CD 21-89 9C 00 00 E8 A0 00 C3    .....!.....
076A:0140 8B 3E C8 00 BB 00 00 8B-87 00                    .>.....
- ▲ ▲

```

进入到 decibin 部分的代码，这里先用 int 21h 来实现 DOS 调用，01h 表示获取键盘输入字符。然后进行计算，将其转换为二进制，如果输入的是两位，就需要将高位转换为数字并乘 10 并与个位数字相加。这里先输入了一个数字 9，则会在 AX 中最终计算出 9 并暂存在 BX 中，下面再重复这个过程。

```

- u
0779:00A3 BB0000      MOV     BX,0000
0779:00A6 B401          MOV     AH,01
0779:00A8 CD21          INT     21
0779:00AA 8AD0          MOV     DL,AL
0779:00AC 2C30          SUB     AL,30
0779:00AE 7C10          JL      00C0
0779:00B0 3C09          CMP     AL,09
0779:00B2 7F0C          JG      00C0
0779:00B4 98          CBW
0779:00B5 93          XCHG    BX,AX
0779:00B6 B90A00      MOV     CX,000A
0779:00B9 F7E1          MUL     CX
0779:00BB 93          XCHG    BX,AX
0779:00BC 03D8          ADD     BX,AX
0779:00BE EBE6          JMP     00A6
0779:00C0 C3          RET
0779:00C1 53          PUSH    BX
0779:00C2 51          PUSH    CX

```

```

AX=0000 BX=0009 CX=01EC DX=0039 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=00B6  NU UP EI PL ZR NA PE NC
0779:00B6 B90A00      MOV     CX,000A
-t

AX=0000 BX=0009 CX=000A DX=0039 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=00B9  NU UP EI PL ZR NA PE NC
0779:00B9 F7E1      MUL     CX
-t

AX=0000 BX=0009 CX=000A DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=00BB  NU UP EI PL ZR NA PE NC
0779:00BB 93      XCHG    BX,AX

```

这里，我输入了一个逗号，由于在 cmp 指令中判断出不是 0-9 的字符，此时这一轮的输入结束，decibin 函数返回。此时内存中 count 对应的值加 1。下面再判断刚才输入的非数字字符是否为逗号，由于刚才输入的是逗号，这里会将数据存入内存，并回到调用 decibin 的部分。

```

AX=012C BX=0063 CX=000A DX=0000 SP=FFF2 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=F000 IP=14A5  NU UP EI PL NZ AC PE NC
F000:14A5 CF      IRET
-t

AX=012C BX=0063 CX=000A DX=0000 SP=FFF8 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=00AA  NU UP EI PL NZ AC PE NC
0779:00AA 8AD0      MOV     DL,AL
-t

AX=012C BX=0063 CX=000A DX=002C SP=FFF8 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=00AC  NU UP EI PL NZ AC PE NC
0779:00AC 2C30      SUB     AL,30
-t

AX=01FC BX=0063 CX=000A DX=002C SP=FFF8 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=00AE  NU UP EI NG NZ NA PE CY
0779:00AE 7C10      JL      00C0
-t

AX=01FC BX=0063 CX=000A DX=002C SP=FFF8 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=00C0  NU UP EI NG NZ NA PE CY
0779:00C0 C3      RET

```

```

-t
AX=01FC BX=0063 CX=000A DX=002C SP=FFFA BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=002B NU UP EI PL NZ NA PO CY
0779:002B 80FA2C CMP DL,2C
-dc8
076A:00C0 01 00 47 72 61 64 65 3F ..Grade
076A:00D0 20 24 0D 0A 49 6E 70 75-74 20 45 72 72 6F 72 21 $..Input Error
076A:00E0 0D 0A 24 52 61 6E 6B 3A-20 24 00 00 00 00 00 00 ..$Rank: $. ....
076A:00F0 1E 2B C0 50 B8 6A 07 8E-D8 E8 07 00 E8 41 00 E8 ..+.P.j.....A.
076A:0100 6B 00 CB 8D 16 CA 00 B4-09 CD 21 BE 00 00 C7 06 k.....!....
076A:0110 C8 00 00 00 E8 7C 00 FF-06 C8 00 80 FA 2C 74 07 .....!. ....,t
076A:0120 80 FA 0D 74 13 75 09 89-9C 00 00 83 C6 02 EB E4 ...t.u.....
076A:0130 8D 16 D2 00 B4 09 CD 21-89 9C 00 00 E8 A0 00 C3 .....!.....
076A:0140 8B 3E C8 00 BB 00 00 8B .>.....

AX=01FC BX=0063 CX=000A DX=002C SP=FFFA BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=0037 NU UP EI PL ZR NA PE NC
0779:0037 899C0000 MOV [SI+0000],BX DS:00
-t
AX=01FC BX=0063 CX=000A DX=002C SP=FFFA BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=003B NU UP EI PL ZR NA PE NC
0779:003B 83C602 ADD SI,+02
-d0
076A:0000 63 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 C.....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
- ▲ ▲

```

下面再输入两个相同的数字 96，并在最后输入一个非法符号，此时 decibin 部分会返回，然后继续将刚才输入的数字部分存储到内存中，并将 count 的值加 1，由于输入中包含了非法字符，因此这里会结束输入部分，并输出回车和换行，然后将 Input Error 输出。

```

Input Error!

Rank: 001,003,003,

Program terminated normally
-t
AX=01F0 BX=0060 CX=000A DX=0020 SP=FFFA BP=0000 SI=0004 DI=0000
DS=076A ES=075A SS=0769 CS=0779 IP=002B  NU UP EI PL NZ NA PO CY
0779:002B 80FA2C          CMP     DL,2C
-d0
076A:0000 62 00 60 00 60 00 00 00-00 00 00 00 00 00 00 00  b.'.'....
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ....
076A:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ....
076A:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ....
076A:0040 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ....
076A:0050 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ....
076A:0060 00 00 00 00 01 00 03 00-03 00 00 00 00 00 00 00  ....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ....

```

下面对 rankp 部分的代码进行调试。先进入到 rankp 部分的调试。这里首先从内存中读取到 count 的值，然后通过 bx 来依次读取内存中输入的成绩的值。并记录 count 的值作为循环的次数。其中 DI 用于存储外部循环的次数。

```

-u
0779:0050 8B3EC800      MOV     DI,[00C8]
0779:0054 BB0000      MOV     BX,0000
0779:0057 8B870000      MOV     AX,[BX+0000]
0779:005B C78764000000  MOV     WORD PTR [BX+0064],0000
0779:0061 8B0EC800      MOV     CX,[00C8]
0779:0065 8D360000      LEA     SI,[0000]
0779:0069 3B04          CMP     AX,[SI]
0779:006B 7F04          JG      0071
0779:006D FF876400      INC     WORD PTR [BX+0064]

```

```

-dc8
076A:00C0          03 00 47 72 61 64 65 3F          ..Gr
076A:00D0 20 24 0D 0A 49 6E 70 75-74 20 45 72 72 6F 72 21  $..Input Er
076A:00E0 0D 0A 24 52 61 6E 6B 3A-20 24 00 00 00 00 00 00  ..$Rank: $.
076A:00F0 1E 2B C0 50 B8 6A 07 8E-D8 E8 07 00 E8 41 00 E8  .+.P.j.....
076A:0100 6B 00 CB 8D 16 CA 00 B4-09 CD 21 BE 00 00 C7 06  k.....!
076A:0110 C8 00 00 00 E8 7C 00 FF-06 C8 00 80 FA 2C 74 07  ....!.....
076A:0120 80 FA 0D 74 13 75 09 89-9C 00 00 83 C6 02 EB E4  ...t.u.....
076A:0130 8D 16 D2 00 B4 09 CD 21-89 9C 00 00 E8 A0 00 C3  ....!.....
076A:0140 8B 3E C8 00 BB 00 00 8B          .>.....

```

然后依次读取每一个成绩，对于每一个成绩，都需要与当前的所有成

绩进行比较，当有成绩的值大于等于自己时，名次的值加 1，这里的比较包括自己，名次的值初始为 0.

```

AX=0062 BX=0000 CX=0003 DX=000D SP=FFFA BP=0000 SI=0000 DI=0003
DS=076A ES=075A SS=0769 CS=0779 IP=0071 NU UP EI PL NZ NA PO NC
0779:0071 83C602          ADD     SI,+02
-d64
076A:0060                01 00 00 00-00 00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0080 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0090 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:00A0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:00B0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:00C0 00 00 00 00 00 00 00 00-03 00 47 72 61 64 65 3F .....
076A:00D0 20 24 0D 0A 49 6E 70 75-74 20 45 72 72 6F 72 21 $. Input
076A:00E0 0D 0A 24 52                                     $B

```

当执行完一轮的循环后，将 DI 的值减一，并与 0 进行比较，并在 DI 非 0 的情况下执行循环，读取下一个数。这里由于名次的初始值为 0，在有相同的值的情况下，也会将名次的值相加，因此当出现并列的时候，如这里一共有 3 个数 98, 96, 96, 96 的名次为 3. 因此，在执行完这部分的函数后，内存中的名次部分的变量的值为 1, 3, 3.

```

-d64
076A:0060                01 00 03 00-00 00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0080 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0090 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:00A0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:00B0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:00C0 00 00 00 00 00 00 00 00-03 00 47 72 61 64 65 3F .....
076A:00D0 20 24 0D 0A 49 6E 70 75-74 20 45 72 72 6F 72 21 .....
076A:00E0 0D 0A 24 52
- ▲

```

```

AX=0060 BX=0006 CX=0000 DX=000D SP=FFFA BP=0000 SI=0000 DI=0003
DS=076A ES=075A SS=0769 CS=0779 IP=007C NU UP EI PL NZ NA PO NC
0779:007C C3          RET
-d64
076A:0060                01 00 03 00-03 00 00 00 00 00 00 00 00 .....
076A:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0080 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0090 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:00A0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:00B0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:00C0 00 00 00 00 00 00 00 00-03 00 47 72 61 64 65 3F .....
076A:00D0 20 24 0D 0A 49 6E 70 75-74 20 45 72 72 6F 72 21 .....
076A:00E0 0D 0A 24 52

```

下面再调试 output 部分的代码。如图，根据访问的内存地址，这里先通 int 21h 中的功能 09h 来实现将 Rank: 输出, 然后再访问变量 count, 用于表示循环次数, 然后从 rank 中依次访问名次的值, 调用 binidec 来实现将二进制数字转换为十进制字符, 这里依次调用 dec_div 函数, 分别除以 100, 10, 1.

```
-u
0779:007D 8D16E300      LEA     DX,[00E3]
0779:0081 B409          MOV     AH,09
0779:0083 CD21          INT     21
0779:0085 BE0000       MOV     SI,0000
0779:0088 8B3EC800       MOV     DI,[00C8]
0779:008C 8B9C6400       MOV     BX,[SI+0064]
0779:0090 E82E00       CALL    00C1
0779:0093 B22C          MOV     DL,2C
0779:0095 B402          MOV     AH,02
0779:0097 CD21          INT     21
0779:0099 83C602       ADD     SI,+02
0779:009C 4F           DEC     DI
-de3
076A:00E0          52 61 6E 6B 3A-20 24 00 00 00 00 00 00      Rank: $
076A:00F0 1E 2B C0 50 B8 6A 07 8E-D8 E8 07 00 E8 41 00 E8 .+.P.j....
076A:0100 6B 00 CB 8D 16 CA 00 B4-09 CD 21 BE 00 00 C7 06 k.....
076A:0110 C8 00 00 00 E8 7C 00 FF-06 C8 00 80 FA 2C 74 07 .....!....
076A:0120 80 FA 0D 74 13 75 09 89-9C 00 00 83 C6 02 EB E4 ...t.u....
076A:0130 8D 16 D2 00 B4 09 CD 21-89 9C 00 00 E8 A0 00 C3 .....?...
076A:0140 8B 3E C8 00 BB 00 00 8B-87 00 00 C7 87 64 00 00 .>.....
076A:0150 00 8B 0E C8 00 8D 36 00-00 3B 04 7F 04 FF 87 64 .....6...
076A:0160 00 83 C6          ...
```



```

-dc8
076A:00C0          03 00 47 72 61 64 65 3F          ..Gr
076A:00D0  20 24 0D 0A 49 6E 70 75-74 20 45 72 72 6F 72 21  $..Input Er
076A:00E0  0D 0A 24 52 61 6E 6B 3A-20 24 00 00 00 00 00 00  ..$Rank: $..
076A:00F0  1E 2B C0 50 B8 6A 07 8E-D8 E8 07 00 E8 41 00 E8  .+.P.j.....
076A:0100  6B 00 CB 8D 16 CA 00 B4-09 CD 21 BE 00 00 C7 06  k.....?..
076A:0110  C8 00 00 00 E8 7C 00 FF-06 C8 00 80 FA 2C 74 07  ....i.....
076A:0120  80 FA 0D 74 13 75 09 89-9C 00 00 83 C6 02 EB E4  ...t.u.....
076A:0130  8D 16 D2 00 B4 09 CD 21-89 9C 00 00 E8 A0 00 C3  ....?.....
076A:0140  8B 3E C8 00 BB 00 00 8B          .>.....
-t

AX=0960  BX=0006  CX=0000  DX=00E3  SP=FFFA  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=0779  IP=0088  NU UP EI PL ZR NA PE NC
0779:0088  8B3EC800          MOV     DI,[00C8]          DS:00C
-t

AX=0960  BX=0006  CX=0000  DX=00E3  SP=FFFA  BP=0000  SI=0000  DI=0003
DS=076A  ES=075A  SS=0769  CS=0779  IP=008C  NU UP EI PL ZR NA PE NC
0779:008C  8B9C6400          MOV     BX,[SI+0064]      DS:006

```

进入到 dec_div 函数，这里 cx 存储的是除数，将计算的结果存入 DX 中，并通过 INT 21H 的功能 02H 来实现将这个十进制数字输出。如，这里先将百位输出 0，再将十位输出 0，最后将个位输出 1。

```

-u
0779:00DC  8BC3          MOV     AX,BX
0779:00DE  BA0000        MOV     DX,0000
0779:00E1  F7F1          DIV     CX
0779:00E3  8BDA          MOV     BX,DX
0779:00E5  8AD0          MOV     DL,AL
0779:00E7  80C230        ADD     DL,30
0779:00EA  B402          MOV     AH,02
0779:00EC  CD21          INT     21
0779:00EE  C3           RET
0779:00EF  B20A          MOV     DL,0A
0779:00F1  B402          MOV     AH,02
0779:00F3  CD21          INT     21
0779:00F5  B20D          MOV     DL,0D
0779:00F7  B402          MOV     AH,02
0779:00F9  CD21          INT     21
0779:00FB  C3           RET
-t

AX=0001  BX=0001  CX=0064  DX=00E3  SP=FFEE  BP=0000  SI=0000  DI=00
DS=076A  ES=075A  SS=0769  CS=0779  IP=00DE  NU UP EI PL ZR NA PE N
0779:00DE  BA0000        MOV     DX,0000

```

```

0779:00EA B402          MOV     AH,02
-t
AX=0200 BX=0001 CX=0064 DX=0030 SP=FFEE BP=0000 SI=0000 DI=0003
DS=076A ES=075A SS=0769 CS=0779 IP=00EC  NU UP EI PL NZ NA PE NC
0779:00EC CD21          INT     21
-t
AX=0200 BX=0001 CX=0064 DX=0030 SP=FFEB BP=0000 SI=0000 DI=0003
DS=076A ES=075A SS=0769 CS=F000 IP=14A0  NU UP DI PL NZ NA PE NC
F000:14A0 FB          STI
-t
AX=0200 BX=0001 CX=0064 DX=0030 SP=FFEB BP=0000 SI=0000 DI=0003
DS=076A ES=075A SS=0769 CS=F000 IP=14A1  NU UP EI PL NZ NA PE NC
F000:14A1 FE38        ???     [BX+SI]          DS:0
-t
0
AX=0230 BX=0001 CX=0064 DX=0030 SP=FFEB BP=0000 SI=0000 DI=0003
DS=076A ES=075A SS=0769 CS=F000 IP=14A5  NU UP EI PL NZ NA PE NC
F000:14A5 CF          IRET

```

整个程序的运行结果如图。

```

C:\>2_6
Grade? 99,98,90,97,97

Rank: 001,002,005,004,004,
C:\>

```

2. 例 2.7

程序数据段的定义如图。这里 hrspar 用于存储小时数的输入信息，ratepar 用于存储每小时的工资数的输入信息。messg1, messg2, messg3 分别用于在对数据进行输入和输出之前显示的提示内容。

```

;*****
datasg      segment      para 'data'
    hrspar      label      byte      ;Hours parameter list
    maxhlen      db      6      ;-----
    acthlen      db      ?
    hrsfld      db      6 dup(?)
;
    ratepar      label      byte      ;Rate parameter list
    maxrlen      db      6      ;-----
    actrlen      db      ?
    ratefld      db      6 dup(?)
;
    messg1      db      'Hours worked? ', '$'
    messg2      db      'Rate of pay? ', '$'
    messg3      db      'Wage = ', '$'
    ascwage      db      14 dup(30h), 13, 10, '$'
;
    messg4      db      13, 10, 'Overflow! ', 13, 10, '$'
    adjust      dw      ?
    binval      dw      0
    binhrs      dw      0
    binrate      dw      0
    col         db      0
    decind      db      0
    mult10      dw      01
    nodec       dw      0
    row         db      0
    shift       dw      ?
    tenwd       dw      10
    tempdx      dw      ?
    tempax      dw      ?
datasg      ends
;*****

```

下面对代码进行调试。

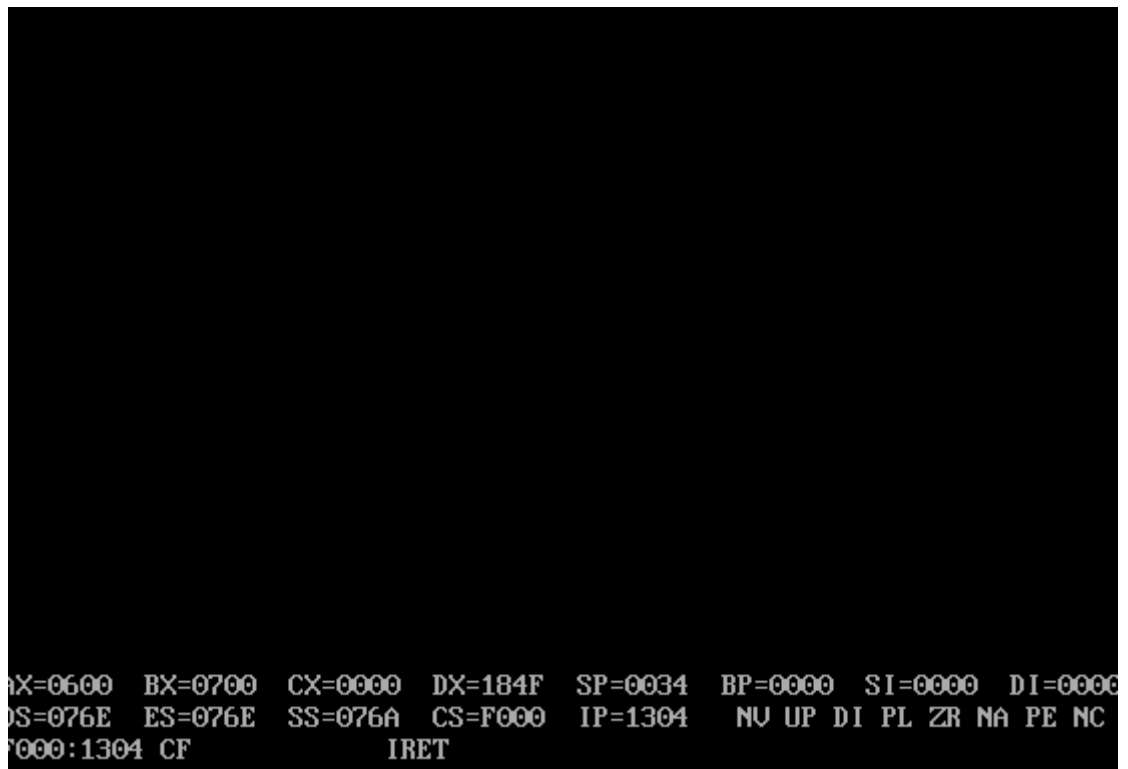
这里先调用两个函数，q10scr 和 q20curs，分别用于清除屏幕和设置光标的位置。

```

AX=0600 BX=0000 CX=02C0 DX=0000 SP=003C BP=0000
DS=076E ES=076E SS=076A CS=0775 IP=000E NU UP DI PL ZR NA PE NC
0775:000E E8E601 CALL 01F7
-u
0775:000E E8E601 CALL 01F7
0775:0011 E8ED01 CALL 0201
0775:0014 E81F00 CALL 0036
0775:0017 803E010000 CMP BYTE PTR [0001],00
0775:001C 7411 JZ 002F
0775:001E E84600 CALL 0067
0775:0021 E85F00 CALL 0083
0775:0024 E87200 CALL 0099
0775:0027 E8EB00 CALL 0115
0775:002A E82A01 CALL 0157
0775:002D EBE5 JMP 0014

```

在清除屏幕的部分，执行完清除屏幕的中断指令后，显示的结果如图。这里使用的是 INT 10H 的 DOS 调用，其中 AH 为 06H 时表示初始化屏幕和滚屏，这里通过将 CX, DX 设置对应的数字来控制滚动的矩形区域的大小。



```

AX=0600 BX=0700 CX=0000 DX=184F SP=0034 BP=0000 SI=0000 DI=0000
DS=076E ES=076E SS=076A CS=F000 IP=1304 NU UP DI PL ZR NA PE NC
F000:1304 CF IRET

```

然后调用 q20curs 函数，用于设置光标位置，执行完成后的结果如图。这里使用了 INT 10H 的 DOS 调用，AH 为 02h 时，用于设置光标的位置。

置，其中 DH 和 DL 分别表示光标要定位到的行号和列号。

```

AX=0200 BX=0700 CX=0000 DX=184F SP=003A BP=0000 SI=0000 DI=0000
AX=0200 BX=0000 CX=0000 DX=0000 SP=0034 BP=0000 SI=0000 DI=0000
DS=076E ES=076E SS=076A CS=F000 IP=1304 NU UP DI PL ZR NA PE NC
F000:1304 CF IRET
- ▲ -
AX=0200 BX=0000 CX=0000 DX=184F SP=003A BP=0000 SI=0000 DI=0000
DS=076E ES=076E SS=076A CS=0775 IP=0205 NU UP EI PL ZR NA PE NC
0775:0205 8A366300 MOV DH,[0063] DS:00
-t
AX=0200 BX=0000 CX=0000 DX=004F SP=003A BP=0000 SI=0000 DI=0000
DS=076E ES=076E SS=076A CS=0775 IP=0209 NU UP EI PL ZR NA PE NC
0775:0209 8A165D00 MOV DL,[005D] DS:00
-t
AX=0200 BX=0000 CX=0000 DX=0000 SP=003A BP=0000 SI=0000 DI=0000
DS=076E ES=076E SS=076A CS=0775 IP=020D NU UP EI PL ZR NA PE NC
0775:020D CD10 INT 10
-t
AX=0200 BX=0000 CX=0000 DX=0000 SP=0034 BP=0000 SI=0000 DI=0000
DS=076E ES=076E SS=076A CS=F000 IP=1300 NU UP DI PL ZR NA PE NC
F000:1300 FE38 ??? [BX+SI] DS:00
-t

```

下面再调用输入函数，如图，这里先输出 Hours worked?，用于提示接下来的小时数的输入。然后再调用 INT 21H 的 09H 功能，将接收输入到缓冲区，并将输入的内容，最大长度，输入长度通过一组连续的变量来存储。这里判断输入的长度为 0 时，就直接返回。然后再定位光标的位置，输出 Rate of pay?。然后再使用 INT 21H 中断来实现输入每小时工资数，最后返回。

```

0775:0036 8D161000 LEA DX,[0010] DS:0010=6F4
-u
0775:0036 8D161000 LEA DX,[0010] 000
0775:003A B409 MOV AH,09 NC
0775:003C CD21 INT 21 DS:0000=06
0775:003E 8D160000 LEA DX,[0000]
0775:0042 B40A MOV AH,0A
0775:0044 CD21 INT 21
0775:0046 803E010000 CMP BYTE PTR [0001],00
0775:004B 7501 JNZ 004E
0775:004D C3 RET

```

```

-d10
076E:0010 48 6F 75 72 73 20 77 6F-72 6B 65 64 3F 20 24 52 Hours worked?
076E:0020 61 74 65 20 6F 66 20 70-61 79 3F 20 24 57 61 67 ate of pay? $
076E:0030 65 20 3D 20 24 30 30 30-30 30 30 30 30 30 30 e = $000000000
076E:0040 30 30 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77 000..$.0verf
076E:0050 21 20 0D 0A 24 00 00 00-00 00 00 00 00 00 01 ! ..$.
076E:0060 00 00 00 00 00 00 0A 00-00 00 00 00 00 00 00 .....
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6 .+.P.n.
076E:0080 01 EB ED 01 E8 1F 00 80-3E 01 00 00 74 11 E8 46 .....>...t

```

```

0775:0056 8D161F00 LEA DX,[001F]
-g5e:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6
Rate of pay? E8 ED 01 E8 1F 00 80-3E 01 00 00 74 11 E8 46
AX=0900 BX=0000 CX=0000 DX=001F SP=003A BP=0000 SI=000
DS=076E ES=076E SS=076A CS=0775 IP=005E NU UP EI PL ZR
0775:005E 8D160800 LEA DX,[0008]
-▲076E ES=076E SS=076A CS=0775 IP=004E NU UP EI PL NZ
0775:004E C6065D0019 MOV BYTE PTR [005D],19
-u
0775:004E C6065D0019 MOV BYTE PTR [005D],19
0775:0053 E8AB01 CALL 0201
0775:0056 8D161F00 LEA DX,[001F]
0775:005A B409 MOV AH,09
0775:005C CD21 INT 21
0775:005E 8D160800 LEA DX,[0008]
0775:0062 B40A MOV AH,0A
0775:0064 CD21 INT 21
0775:0066 C3 RET
0775:0067 C70661000000 MOV WORD PTR [0061],0000
0775:006D 8A0E0100 MOV CL,[0001]
-g56

```

在主函数中，这里再次进行比较小时数部分的输入内容，当输入的内容为空时，就清除屏幕并返回。下面，在使用 d10hour 函数来对输入的小时数进行处理。这里将输入的小时数的部分进行处理。这里找到输入的数据中的最右边的一位数字开始，然后调用 m10asbi 函数来将输入的内容转换为二进制数字进行存储。

```

0775:006D 8A0E0100      MOV     CL,[0001]      DS:0001
-t
AX=0A00 BX=0000 CX=0001 DX=0008 SP=003A BP=0000 SI=0000 DI=0000
DS=076E ES=076E SS=076A CS=0775 IP=0071  NU UP EI PL NZ NA PO NC
0775:0071 2AED          SUB     CH,CH
-t
AX=0A00 BX=0000 CX=0001 DX=0008 SP=003A BP=0000 SI=0000 DI=0000
DS=076E ES=076E SS=076A CS=0775 IP=0073  NU UP EI PL ZR NA PE NC
0775:0073 8D360100      LEA     SI,[0001]      DS:0001
-d1
076E:0000      01 38 0D 00 00 00 00-06 01 39 0D 00 00 00 00      .8.....9..
076E:0010 48 6F 75 72 73 20 77 6F-72 6B 65 64 3F 20 24 52      Hours worked?
076E:0020 61 74 65 20 6F 66 20 70-61 79 3F 20 24 57 61 67      ate of pay? $
076E:0030 65 20 3D 20 24 30 30 30-30 30 30 30 30 30 30 30      e = $00000000
076E:0040 30 30 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77      000..$.0verf
076E:0050 21 20 0D 0A 24 00 00 00-00 00 00 00 00 19 00 01      ! ..$.
076E:0060 00 00 00 00 00 00 0A 00-00 00 00 00 00 00 00 00      .....
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6      .+.P.n.....
076E:0080 01

```

这里进入到 m10asbi 函数，这里依次设置变量的值。然后判断是否为小数点，如果是小数点，这里就需要在变量中将 decind 的值设置为 1，然后将内存中取出的一位数通过 AND 指令来从 ASCII 转换成数字，并乘上一个乘数，然后将计算完的结果保存到变量 binval 中。而这个乘数会在每次计算完后乘 10，后面依次执行这个过程，如果出现小数点的情况时，就需要计算小数部分的位数，这里使用 BX 来对读取的数字进行计数，当遇到小数点时就停止计数，并在返回之前将这个数字记录在变量 nodec 中。如果出现溢出的情况，这个最终计算的值会是 0。

```

AX=0A00 BX=0000 CX=0001 DX=0008 SP=0038 BP=0000 SI=0002 DI=0000
DS=076E ES=076E SS=076A CS=0775 IP=01B2  NU UP EI PL ZR NA PE NC
0775:01B2 8A04          MOV     AL,[SI]      DS:
-d02
076E:0000      38 0D 00 00 00 00 00-06 01 39 0D 00 00 00 00      8.....
076E:0010 48 6F 75 72 73 20 77 6F-72 6B 65 64 3F 20 24 52      Hours wo
076E:0020 61 74 65 20 6F 66 20 70-61 79 3F 20 24 57 61 67      ate of p
076E:0030 65 20 3D 20 24 30 30 30-30 30 30 30 30 30 30 30      e = $0000
076E:0040 30 30 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77      000..$.
076E:0050 21 20 0D 0A 24 00 00 00-00 00 00 00 00 19 00 01      ! ..$.
076E:0060 00 00 00 00 00 00 0A 00-00 00 00 00 00 00 00 00      .....
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6      .+.P.n..
076E:0080 01 E8

```

```

AX=0A00 BX=0000 CX=0001 DX=0008 SP=0038 BP=0000 SI=0002 DI=0000
DS=076E ES=076E SS=076A CS=0775 IP=01B2 NU UP EI PL ZR NA PE NC
0775:01B2 8A04          MOV     AL,[SI]
-d02
076E:0000          38 0D 00 00 00 00-06 01 39 0D 00 00 00 00      8.....
076E:0010 48 6F 75 72 73 20 77 6F-72 6B 65 64 3F 20 24 52    Hours wo
076E:0020 61 74 65 20 6F 66 20 70-61 79 3F 20 24 57 61 67    ate of p
076E:0030 65 20 3D 20 24 30 30 30-30 30 30 30 30 30 30 30    e = $000
076E:0040 30 30 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77    000..$.
076E:0050 21 20 0D 0A 24 00 00 00-00 00 00 00 19 00 01    ! ..$.
076E:0060 00 00 00 00 00 00 0A 00-00 00 00 00 00 00 00 00    .....
076E:0070 1E 2B C0 50 B8 6E 07 8E-DB 8E C0 B8 00 06 E8 E6    .+.P.n..
076E:0080 01 E8
..

```

当返回到主函数时，下面会调用 e10rate 函数，与前面的 d10hour 类似，也会调用 m10asbi 函数，只是换成了处理 rate 的值。

```

0775:0070 1E 2B C0 50 B8 6E 07 8E-DB 8E C0 B8 00 06 E8 E6
-t
AX=0008 BX=0001 CX=0000 DX=0000 SP=0038 BP=0000 SI=0002 DI=0000
DS=076E ES=076E SS=076A CS=0775 IP=0080
0775:0083 8A0E0900      MOV     CL,[0009]
-u
0775:0083 8A0E0900      MOV     CL,[0009]
0775:0087 2AED          SUB     CH,CH
0775:0089 8D360900      LEA     SI,[0009]
0775:008D 03F1          ADD     SI,CX
0775:008F E80D01      CALL    019F
0775:0092 A15700      MOV     AX,[0057]
0775:0095 A35B00      MOV     [005B],AX
0775:0098 C3          RET
0775:0099 B90700      MOV     CX,0007
0775:009C 8D3E3500      LEA     DI,[0035]
0775:00A0 B83030      MOV     AX,3030

```

下面再进入 f10wage 函数，这个函数用于将 hour 和 rate 的值相乘。这里先将存储结果的字符串进行初始化为 ASCII 中的 0，然后从内存中读取前面计算的小数部分的位数并判断是否溢出，如果溢出，就直接将 0 作为结果。这里我输入了两个整数，因此 nodec 中的值为 0，不会溢出。后面再判断是否有小数的舍入计算，如果有，就按照小数的位数来计算舍入计算时的加数的大小，用 adjust 来存储这个计算舍入的加数。


```

-u
0775:0099 B90700      MOV     CX,0007
0775:009C 8D3E3500     LEA     DI,[0035]
0775:00A0 B83030      MOV     AX,3030
0775:00A3 FC        CLD
0775:00A4 F3        REPZ
0775:00A5 AB      STOSW
0775:00A6 C70664000A00  MOV     WORD PTR [0064],000A
0775:00AC C70655000000  MOV     WORD PTR [0055],0000
0775:00B2 8B0E6100     MOV     CX,[0061]
0775:00B6 80F906     CMP     CL,06
-d35
076E:0030                30 30 30-30 30 30 30 30 30 30 30 30 30 30 30 30 00
076E:0040 30 30 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77 000..$.
076E:0050 21 20 0D 0A 24 00 00 09-00 08 00 09 00 19 00 0A ? ..$.
076E:0060 00 00 00 00 00 00 0A 00-00 00 00 00 00 00 00 00 .....
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6 .+.P.n.
076E:0080 01 E8 ED 01 E8 1F 00 80-3E 01 00 00 74 11 E8 46 .....
076E:0090 00 E8 5F 00 E8 72 00 E8-EB 00 E8 2A 01 EB E5 B8 .._..r.
076E:00A0 00 06 E8 C2 01 CB 8D 16-10 00 B4 09 CD 21 8D 16 .....
076E:00B0 00 00 B4 0A CD .....

```

```

AX=3030 BX=0001 CX=0000 DX=0000 SP=003A BP=0000
DS=076E ES=076E SS=076A CS=0775 IP=00B9 NU UP
0775:00B9 7753      JA     010E
-t
AX=3030 BX=0001 CX=0000 DX=0000 SP=003A BP=0000
DS=076E ES=076E SS=076A CS=0775 IP=00BB NU UP
0775:00BB 49      DEC     CX
-t
AX=3030 BX=0001 CX=FFFF DX=0000 SP=003A BP=0000
DS=076E ES=076E SS=076A CS=0775 IP=00BC NU UP
0775:00BC 49      DEC     CX
-t
AX=3030 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000
DS=076E ES=076E SS=076A CS=0775 IP=00BD NU UP
0775:00BD 7E17     JLE     00D6
-t
AX=3030 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000
DS=076E ES=076E SS=076A CS=0775 IP=00D6 NU UP
0775:00D6 A15900     MOV     AX,[0059]
-

```

在计算 wage 的过程中,这里通过内存中存储 hour 的变量与存储 rate 的变量相乘,并将结果加上 adjust 进行舍入计算。当计算完乘法后,如果 adjust 存在舍入部分,就需要将前面的 wage 的结果进行除法运算来确保小数部分只有两位。

```

AX=3030 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000 SI=0009 DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=00D6  NU UP EI NG NZ NA PO CY
0775:00D6 A15900      MOV     AX,[0059]      DS:0
-t

AX=0008 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000 SI=0009 DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=00D9  NU UP EI NG NZ NA PO CY
0775:00D9 F7265B00    MJL     WORD PTR [005B]      DS:0
-t

AX=0048 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000 SI=0009 DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=00DD  NU UP EI NG NZ NA PO NC
0775:00DD 03065500    ADD     AX,[0055]      DS:0

```

下面再调用 g10wage 函数，这里先设置小数点的位置，然后找到保存 wage 字符串的最右边的位置，将 wage 的值依次除以 10 并取出余数，将余数转换为 ASCII 字符保存，将计算好的商作为下一次计算的被除数，直到剩余的数字小于 10.。

```

AX=0048 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000 SI=0009 DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=0115  NU UP EI PL ZR NA PE NC
0775:0115 8D364000    LEA     SI,[0040]      DS:0
-t

AX=0048 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000 SI=0040 DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=0119  NU UP EI PL ZR NA PE NC
0775:0119 C6042E      MOV     BYTE PTR [SI],2E      DS:0
-t

AX=0048 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000 SI=0040 DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=011C  NU UP EI PL ZR NA PE NC
0775:011C 03366100    ADD     SI,[0061]      DS:0
-t
-d40
076E:0040 2E 30 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77  .00..$...00
076E:0050 21 20 0D 0A 24 00 00 09-00 08 00 09 00 19 00 0A  ? ..$. ....
076E:0060 00 00 00 00 0A 00 0A 00-00 00 48 00 00 00 00 00  .....
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6  .+.P.n....
076E:0080 01 E8 ED 01 E8 1F 00 80-3E 01 00 00 74 11 E8 46  .....>.
076E:0090 00 E8 5F 00 E8 72 00 E8-EB 00 E8 2A 01 EB E5 B8  ..._.r....
076E:00A0 00 06 E8 C2 01 CB 8D 16-10 00 B4 09 CD 21 8D 16  .....
076E:00B0 00 00 B4 0A CD 21 80 3E-01 00 00 75 01 C3 C6 06  .....!.>..

```

```

AX=0000 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000 SI=003F DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=013C  NU UP EI PL NZ AC PO NC
0775:013C A16A00      MOV     AX,[006A]      DS:0
-t

AX=0048 BX=0001 CX=FFFE DX=0000 SP=003A BP=0000 SI=003F DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=013F  NU UP EI PL NZ AC PO NC
0775:013F F7366600     DIV     WORD PTR [0066]      DS:0
-t

AX=0007 BX=0001 CX=FFFE DX=0002 SP=003A BP=0000 SI=003F DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=0143  NU UP EI PL NZ AC PO NC
0775:0143 A36A00      MOV     [006A],AX      DS:0
-t

AX=0007 BX=0001 CX=FFFE DX=0002 SP=003A BP=0000 SI=003F DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=0146  NU UP EI PL NZ AC PO NC
0775:0146 80CA30      OR      DL,30
-t

AX=0007 BX=0001 CX=FFFE DX=0032 SP=003A BP=0000 SI=003F DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=0149  NU UP EI PL NZ NA PO NC
0775:0149 8814      MOV     [SI],DL      DS:0
-t

```

下面再实现将 wage 的值输出。这里先通过 q20curs 函数设置光标的位置，然后从内存中读取出 wage = 这个字符串并将其输出。然后从内存中读取出前面计算的 wage 转换为 ASCII 字符后存储的位置，并将其输出。

```

-t
AX=0237 BX=0001 CX=FFFE DX=0032 SP=0038 BP=0000 SI=003E DI=0
AX=0237 BX=0001 CX=000A DX=0032 SP=003A BP=0000 SI=003E DI=0
DS=076E ES=076E SS=076A CS=0775 IP=0162 NU UP EI PL ZR NA PE
0775:0162 8D363500 LEA SI,[0035]
-t
AX=0237 BX=0001 CX=FFFE DX=0032 SP=0032 BP=0000 SI=003E DI=0
AX=0237 BX=0001 CX=000A DX=0032 SP=003A BP=0000 SI=0035 DI=0
DS=076E ES=076E SS=076A CS=0775 IP=0166 NU UP EI PL ZR NA PE
0775:0166 803C30 CMP BYTE PTR [SI],30
-u
0775:0166 803C30 CMP BYTE PTR [SI],30
0775:0169 7506 JNZ 0171
0775:016B C60420 MOV BYTE PTR [SI],20
0775:016E 46 INC SI
0775:016F E2F5 LOOP 0166
0775:0171 8D162D00 LEA DX,[002D]
0775:0175 B409 MOV AH,09
0775:0177 CD21 INT 21
0775:0179 33D2 XOR DX,DX
0775:017B 8D163500 LEA DX,[0035]
0775:017F B409 MOV AH,09
0775:0181 CD21 INT 21
0775:0183 803E630014 CMP BYTE PTR [0063],14

```

```

0775:018E EBOE JMP 019E
-g171
AX=0237 BX=0001 CX=0001 DX=0032 SP=003A BP=0000 SI=003E DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=0171 NU UP EI PL NZ NA PO NC
0775:0171 8D162D00 LEA DX,[002D] DS:002D=615
-d2d
076E:0020 57 61 67 Wag
076E:0030 65 20 3D 20 24 20 20 20-20 20 20 20 20 37 32 e = $ 72
076E:0040 2E 30 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77 .00..$.Overflow
076E:0050 21 20 0D 0A 24 00 00 09-00 08 00 09 00 32 00 0A ! ..$.2..
076E:0060 00 00 00 00 0A 00 0A 00-00 00 07 00 00 00 00 00 .....
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6 .+.P.n.....
076E:0080 01 E8 ED 01 E8 1F 00 80-3E 01 00 00 74 11 E8 46 .....>...t..F
076E:0090 00 E8 5F 00 E8 72 00 E8-EB 00 E8 2A 01 EB E5 B8 ...r.....*....
076E:00A0 00 06 E8 C2 01 CB 8D 16-10 00 B4 09 CD .....

```

下面运行整个程序。

这里先调用输入的函数，将 hours 和 rate 的值进行输入，输入完成后，会将其转换成二进制数字，并在 nodec 变量中保存小数部分的位数，如图，内存中 63H 和 13H 就是前面输入的数字。

```

Hours worked? 9.9 Rate of pay? 19
AX=0A00 BX=0000 CX=0000 DX=0008 SP=003C BP=0000 SI=0000 DI=0000
DS=076E ES=076E SS=076A CS=0775 IP=0017 NU UP EI PL ZR NA PE NC
0775:0017 803E010000 CMP BYTE PTR [0001],00 DS
-

```

```

-d0
076E:0000 06 03 39 2E 39 0D 00 00-06 02 31 39 0D 00 00 00 ..9.9.....1
076E:0010 48 6F 75 72 73 20 77 6F-72 6B 65 64 3F 20 24 52 Hours worke
076E:0020 61 74 65 20 6F 66 20 70-61 79 3F 20 24 57 61 67 ate of pay?
076E:0030 65 20 3D 20 24 30 30 30-30 30 30 30 30 30 30 30 e = $0000000
076E:0040 30 30 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77 000..$.Overf
076E:0050 21 20 0D 0A 24 00 00 13-00 63 00 13 00 19 00 64 ! ..$....c...
076E:0060 00 01 00 00 00 00 0A 00-00 00 00 00 00 00 00 00 .....Y...
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6 .+.P.n.....

```

再运行下一个函数，这里计算完成了 wage 的值，由于前面输入的是 9.9 和 19，这里计算出的结果是 0759，结果已经保存在内存中。

```

-g27
AX=0759 BX=0002 CX=FFFF DX=0000 SP=003C BP=0000 SI=0009 DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=0027 NU UP EI PL ZR NA PE NC
0775:0027 EBEB00 CALL 0115
-d0
076E:0000 06 03 39 2E 39 0D 00 00-06 02 31 39 0D 00 00 00 ..9.9.....19..
076E:0010 48 6F 75 72 73 20 77 6F-72 6B 65 64 3F 20 24 52 Hours worked?
076E:0020 61 74 65 20 6F 66 20 70-61 79 3F 20 24 57 61 67 ate of pay? $
076E:0030 65 20 3D 20 24 30 30 30-30 30 30 30 30 30 30 30 e = $000000000
076E:0040 30 30 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77 000..$.Overf
076E:0050 21 20 0D 0A 24 00 00 13-00 63 00 13 00 19 00 64 ! ..$....c...
076E:0060 00 01 00 00 0A 00 0A 00-00 00 59 07 00 00 00 00 .....Y...
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6 .+.P.n.....

```

然后再执行将 wage 的值转换为 ASCII 的函数，执行完的结果如图，转换完的结果显示为 188.10，已经在内存中可以找到。

```

-g2a
AX=0031 BX=0002 CX=FFFF DX=0000 SP=003C BP=0000 SI=003D DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=002A NU UP EI PL NZ NA PD NC
0775:002A E82A01 CALL 0157
-d0
076E:0000 06 03 39 2E 39 0D 00 00-06 02 31 39 0D 00 00 00 ..9.9.....19....
076E:0010 48 6F 75 72 73 20 77 6F-72 6B 65 64 3F 20 24 52 Hours worked? $R
076E:0020 61 74 65 20 6F 66 20 70-61 79 3F 20 24 57 61 67 ate of pay? $Wag
076E:0030 65 20 3D 20 24 30 30 30-30 30 30 30 30 31 38 38 e = $000000000188
076E:0040 2E 31 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77 .10..$.Overflow
076E:0050 21 20 0D 0A 24 00 00 13-00 63 00 13 00 19 00 64 ! ..$....c.....d
076E:0060 00 01 00 00 0A 00 0A 00-00 00 01 00 00 00 00 00 .....
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6 .+.P.n.....

```

最后，在右上角的位置，可以看到 wage 的输出结果。

```

0775:002A E82A01      CALL    0157      Wage =      188.10
-d0
AX=0931 BX=0002 CX=0002 DX=0035 SP=003C BP=0000 SI=003D DI=0043
DS=076E ES=076E SS=076A CS=0775 IP=002D  NU UP EI PL NZ NA PO CY k
0775:002D EBE5        JMP     0014      pay
-▲E:0030 65 20 3D 20 24 30 30 30-30 30 30 30 30 31 38 38 e = $000000
076E:0040 2E 31 30 0D 0A 24 0D 0A-4F 76 65 72 66 6C 6F 77 .10..$.0v
076E:0050 21 20 0D 0A 24 00 00 13-00 63 00 13 00 19 00 64 ! ..$....c
076E:0060 00 01 00 00 0A 00 0A 00-00 00 01 00 00 00 00 00 .....
076E:0070 1E 2B C0 50 B8 6E 07 8E-D8 8E C0 B8 00 06 E8 E6 .+.P.n....
-u
0775:002A E82A01      CALL    0157
0775:002D EBE5        JMP     0014
0775:002F B80006      MOV     AX,0600
0775:0032 E8C201      CALL    01F7
0775:0035 CB          RETF
0775:0036 8D161000     LEA     DX,[0010]
0775:003A B409        MOV     AH,09
0775:003C CD21        INT     21
0775:003E 8D160000     LEA     DX,[0000]
0775:0042 B40A        MOV     AH,0A
0775:0044 CD21        INT     21
0775:0046 803E010000   CMP     BYTE PTR [0001],00

```

问题及收获：

1. 通过这次实验，了解了模块化设计汇编程序的方法。
2. 这次实验除了使用前面已经使用过的 DOS 调用以外，还使用了 INT 10H 调用，其中 02H 用于设置光标的位置，DH 和 DL 中分别存储行号和列号，06H 用于滚动屏幕，在屏幕上面空出一个矩形区域。