

计算机组成原理 Homework13 (11.25)

中国人民大学 信息学院 崔冠宇 2018202147

1. 以二进制计算的方式实现两无符号数 X 和 Y 相加的运算过程，要求通过键盘输入 X 和 Y，两数的大小均限制在 65535 之内（可以用 16 位二进制表示），将输入的数字字符串转换为对应的二进制数，然后相加。将得到的和（二进制）转换为十进制数字字符串显示输出。实现要求：

- 主程序：
 - 调用输入子程序获取第一个数据 X
 - 调用输入子程序获取第二个数据 Y
 - 计算 $Z = X + Y$
 - 调用输出子程序将 Z 转换十进制数字字符串输出
- 输入子程序：从键盘获取十进制字符串之后，转换为二进制数
- 输出子程序：将 AX 中的二进制转换为十进制字符串，输出到屏幕上

解：需要用到循环结构和子程序调用。

```
1 INPUT:                ; 100 输入子程序
2 MOV DX, 200            ; 缓冲区首地址
3 MOV BYTE PTR [200], 5  ; 放置最大长度
4 MOV AH, A
5 INT 21                 ; 系统调用 10 - 读取字符串
6 MOV CX, 0
7 MOV CL, [201]          ; 获取输入长度
8 MOV SI, 202            ; 字符串开头
9 MOV AX, 0              ; sum = 0
10 LIN:                  ; 119
11 MOV BX, A              ; BX <- 0xA (10)
12 MUL BX                ; sum = sum * 10, (DX, AX)=AX * 0xAH
13 MOV BL, [SI]           ; BL <- char
14 SUB BL, 30             ; BL <- BL - 48, ASCII 转数字 c
15 ADD AX, BX             ; sum += c
16 INC SI
17 LOOP LIN              ; 119
18 MOV BP, SP
19 MOV DI, [BP + 2]
20 MOV [DI], AX           ; 将二进制结果放到返回地址
```

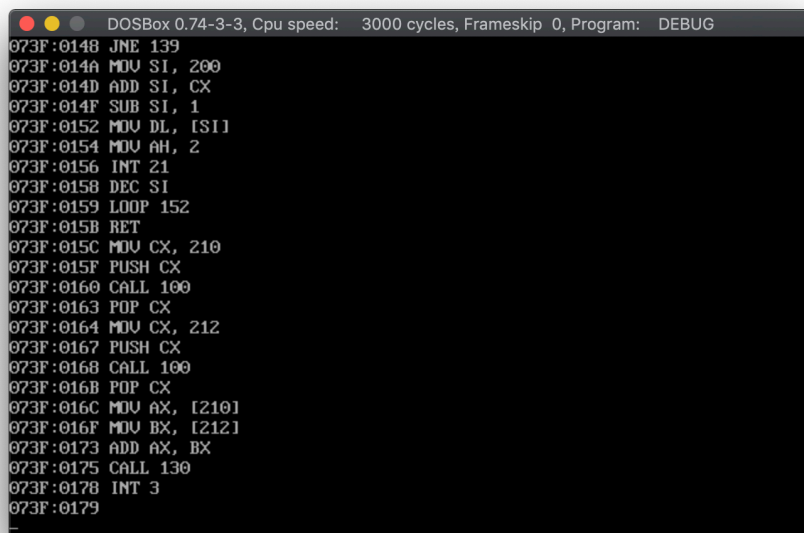
```

21 RET
22
23 OUTPUT: ; 130 AX 输出子程序. 先转成ASCII
24 MOV DI, 200
25 MOV BX, A ; BX <- 0xA (10)
26 MOV CX, 0 ; CX <- 0, 记录结果长度
27 LOUT1: ; 139
28 MOV DX, 0 ; DX <- 0, 32位除法准备
29 DIV BX ; c = num % 10, num /= 10
30 ADD DL, 30H ; c = c + 30H, 转成 ASCII
31 MOV [DI], DL ; [DI] <- c
32 INC CX ; 长度加一
33 INC DI ; 移到下一个内存单元
34 CMP AX, 0 ; num = 0, 结束
35 JNE LOUT1 ; 139
36 MOV SI, 200
37 ADD SI, CX
38 SUB SI, 1 ; 从 200 + CX - 1 开始逆序输出
39 LOUT2: ; 152
40 MOV DL, [SI]
41 MOV AH, 2
42 INT 21 ; 调用2号系统调用
43 DEC SI
44 LOOP LOUT2 ; 152 循环显示一个字符
45 RET
46
47 MAIN: ; 15C 主程序
48 MOV CX, 210 ; 子程序结果存放地址
49 PUSH CX ; 入栈
50 CALL INPUT ; 100 调用
51 POP CX
52 MOV CX, 212
53 PUSH CX
54 CALL INPUT ; 100
55 POP CX
56 MOV AX, [210] ; 第一次结果
57 MOV BX, [212] ; 第二次结果

```

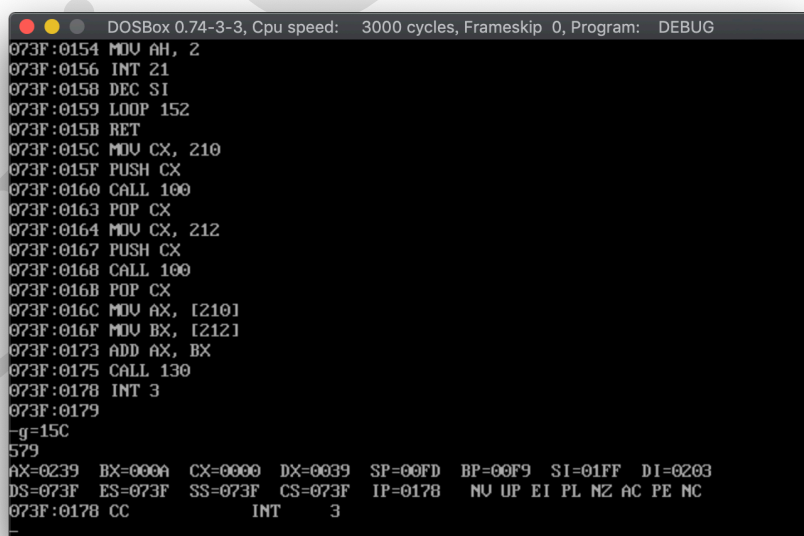
```
58 ADD AX, BX
59 CALL OUTPUT          ; 130
60 INT 3
```

在 DEBUG 环境中，首先用 -a100 键入代码并用 -g=15C 命令执行，程序运行时要求输入两个十进制数字。部分截图如下所示：



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
073F:014B JNE 139
073F:014A MOV SI, 200
073F:014D ADD SI, CX
073F:014F SUB SI, 1
073F:0152 MOV DL, [SI]
073F:0154 MOV AH, 2
073F:0156 INT 21
073F:0158 DEC SI
073F:0159 LOOP 152
073F:015B RET
073F:015C MOV CX, 210
073F:015F PUSH CX
073F:0160 CALL 100
073F:0163 POP CX
073F:0164 MOV CX, 212
073F:0167 PUSH CX
073F:0168 CALL 100
073F:016B POP CX
073F:016C MOV AX, [210]
073F:016F MOV BX, [212]
073F:0173 ADD AX, BX
073F:0175 CALL 130
073F:0178 INT 3
073F:0179
```

图 1: 键入代码



```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
073F:0154 MOV AH, 2
073F:0156 INT 21
073F:0158 DEC SI
073F:0159 LOOP 152
073F:015B RET
073F:015C MOV CX, 210
073F:015F PUSH CX
073F:0160 CALL 100
073F:0163 POP CX
073F:0164 MOV CX, 212
073F:0167 PUSH CX
073F:0168 CALL 100
073F:016B POP CX
073F:016C MOV AX, [210]
073F:016F MOV BX, [212]
073F:0173 ADD AX, BX
073F:0175 CALL 130
073F:0178 INT 3
073F:0179
-g=15C
579
AX=0239 BX=000A CX=0000 DX=0039 SP=00FD BP=00F9 SI=01FF DI=0203
DS=073F ES=073F SS=073F CS=073F IP=0178 NU UP EI PL NZ AC PE NC
073F:0178 CC          INT 3
```

图 2: 输入 123、456 后程序输出 579

根据程序运行结果可见代码功能正确。

2. 以 BCD 码方式实现两无符号数 X 和 Y 相加的运算过程，要求通过键盘输入 X 和 Y，两数的大小均限制在 10 位之内，将输入的数字字符串转换为对应的二进制 BCD 码，然后相加。将得到的和（二进制 BCD 码）转换为十进制数字字符串显示输出。实现要求：

- 主程序：
 - 调用输入子程序获取第一个数据 X 的 BCD 码
 - 调用输入子程序获取第二个数据 Y 的 BCD 码
 - 计算 $Z = X + Y$ (BCD 运算)
 - 调用输出子程序将 Z 的 BCD 码转换十进制数字字符串输出
- 输入子程序：从键盘获取十进制字符串之后，转换为二进制 BCD 码
- 输出子程序：将 Z 的 BCD 码转换为十进制字符串，输出到屏幕上

解：需要用到循环结构和子程序调用（为简便起见，使用了非压缩 BCD 码存储和运算）。

```
1 INPUT:                ; 100 输入子程序
2 MOV DX, 200            ; 缓冲区首地址
3 MOV BYTE PTR [200], A  ; 放置最大长度
4 MOV AH, A
5 INT 21                ; 系统调用 10 - 读取字符串
6 MOV CX, 0
7 MOV CL, [201]         ; 获取输入长度
8 MOV SI, 202           ; 字符串开头
9 LIN1:                 ; 116 循环将 ASCII 转非压缩BCD
10 MOV AL, [SI]
11 SUB AL, 30            ; AL <- AL - 48, ASCII 转数字 c
12 MOV [SI], AL
13 INC SI
14 LOOP LIN1             ; 116
15 MOV BP, SP
16 MOV DI, [BP + 2]      ; 获取栈内参数 —— BCD放置地址
17 MOV AX, 202
18 MOV BX, 0
19 MOV BL, [201]
20 ADD AX, BX
21 SUB AX, 1            ; 202 + len - 1, 最后一个字符开始
22 MOV SI, AX
23 MOV CX, 0
```

```

24 MOV CL, [201]
25 LIN2:                ; 13C 将 BCD 倒序放置
26 MOV AL, [SI]
27 MOV [DI], AL
28 INC DI
29 DEC SI
30 LOOP LIN2            ; 13C
31 MOV BX, 0            ; 将长度放到 BX, 作为返回值
32 MOV BL, [201]
33 RET
34
35 OUTPUT:              ; 14C 输出子程序.
36 MOV BP, SP
37 MOV SI, [BP + 2]     ; 获得最高位地址
38 LOUT1:               ; 151
39 MOV DL, [SI]
40 DEC SI
41 CMP DL, 0            ; 忽略前导零
42 JE LOUT1             ; 151
43 INC SI
44 MOV CX, [BP + 2]
45 SUB CX, A            ; CX <- [BP + 2] - A
46 LOUT2:               ; 160 此时 DL 已经有一个非零值
47 MOV DL, [SI]         ; 读入一位 BCD
48 ADD DL, 30           ; 转成 ASCII
49 MOV AH, 2
50 INT 21              ; 调用 2 号系统调用
51 DEC SI
52 CMP SI, CX
53 JGE LOUT2            ; 160 循环显示一个字符
54 RET
55
56 MAIN:                ; 16F 主程序
57 MOV CX, 10
58 MOV DI, 300
59 LMAIN1:              ; 175
60 MOV BYTE PTR [DI], 0

```

```

61 MOV BYTE PTR [DI+10], 0 ; 将目的地址清零
62 INC DI
63 LOOP LMAIN1 ; 175
64 MOV DX, 300 ; BCD存放地址
65 PUSH DX ; 入栈
66 CALL INPUT ; 100 调用
67 POP DX
68 PUSH BX ; 将 BCD1 的长度入栈暂存
69 MOV DX, 310
70 PUSH DX
71 CALL INPUT ; 100
72 POP DX
73 POP AX ; 弹出 BCD1 的长度
74 MOV DI, 300 ; 回到两 BCD 码结尾
75 MOV SI, 310
76 MOV CX, A ; CX <- A 循环次数
77 CLC ; 清空 CF
78 LMAIN2: ; 19B
79 MOV AL, [SI]
80 ADC AL, [DI] ; 加法
81 AAA ; 非压缩 BCD 调整
82 MOV [DI], AL
83 INC DI
84 INC SI
85 LOOP LMAIN2 ; 19B
86 PUSH DI ; 结果最高位的地址入栈
87 CALL OUTPUT ; 14C 调用
88 POP DI
89 INT 3

```

在 DEBUG 环境中，首先用 -a100 键入代码并用 -g=16F 命令执行，程序运行时要求输入两个十进制数字。部分截图如下所示：

```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
075A:017E CALL 100
075A:0181 POP DX
075A:0182 PUSH BX
075A:0183 MOV DX, 310
075A:0186 PUSH DX
075A:0187 CALL 100
075A:018A POP DX
075A:018B POP AX
075A:018C MOV DI, 300
075A:018F MOV SI, 310
075A:0192 MOV CX, A
075A:0195 CLC
075A:0196 MOV AL, [SI]
075A:0198 ADC AL, [DI]
075A:019A AAA
075A:019B MOV [DI], AL
075A:019D INC DI
075A:019E INC SI
075A:019F LOOP 196
075A:01A1 PUSH DI
075A:01A2 CALL 14C
075A:01A5 POP DI
075A:01A6 INT 3
075A:01A7
```

图 3: 设置内存单元的值

```
DOSBox 0.74-3-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
075A:015A MOV CX, [BP+2]
075A:015D SUB CX, A
075A:0160
-a 16A
075A:016A CMP SI, CX
075A:016C
-g=16F
019
AX=0239 BX=0003 CX=0300 DX=0339 SP=FFF6 BP=FFF2 SI=02FF DI=030A
DS=075A ES=075A SS=075A CS=075A IP=01AB NU UP EI NG NZ NA PE CY
075A:01AB CC INT 3
-g=16F 159
999
AX=0200 BX=0003 CX=0000 DX=0301 SP=FFF2 BP=FFF2 SI=0302 DI=030A
DS=075A ES=075A SS=075A CS=075A IP=0159 NU UP EI PL NZ NA PD NC
075A:0159 47 INC DI
-a 159
075A:0159 INC SI
075A:015A
-g=16F
1019
AX=0239 BX=0003 CX=0300 DX=0339 SP=FFF2 BP=FFEE SI=02FF DI=030A
DS=075A ES=075A SS=075A CS=075A IP=01AB NU UP EI NG NZ NA PE CY
075A:01AB CC INT 3
```

图 4: 输入 20、999 后程序输出 1019

根据程序运行结果可见代码功能正确。