

Experiment 3 Arithmetic and bit string processing program

Purpose:

Master the design of programs such as multi-digit arithmetic operations, shift operations, and string operations, learn to use basic programming methods such as branching and looping, and use Debug proficiently.

Experiment content:

- 【1】** Store 16 ASCII codes of hexadecimal numbers in advance in the data segment, and the first address is ASC. Input a hexadecimal number from the keyboard to BX, use the ASC[BX] addressing mode to find the ASCII code of the corresponding digit, and take it out for display.

code:

DATA SEGMENT

ASC DB '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'

DATAS ENDS

CODES SEGMENT

ASSUME CS:CODES,DS:DATAS

START:

MOV AX,DATAS

MOV DS,AX

MOV AH,1H

INT 21H

MOV BL,AL

MOV BH,0

CMP BX,40H

JNS ALP

JS NUM

NUM: SUB BX,30H

JMP OVER

ALP: SUB BX,37H

JMP OVER

OVER: MOV CL,ASC[BX]

MOV DL,CL

MOV AH,2

int 21h

MOV AH,4CH

INT 21H

CODES ENDS

END START

(As shown in the figure below, input the hexadecimal number 7 to BX from the keyboard)

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
AX=076A BX=0000 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076B IP=0003  NV UP EI PL NZ NA PO NC
076B:0003 8ED8      MOV     DS,AX
-p
AX=076A BX=0000 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0005  NV UP EI PL NZ NA PO NC
076B:0005 B401      MOV     AH,01
-p
AX=016A BX=0000 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0007  NV UP EI PL NZ NA PO NC
076B:0007 CD21      INT     21
-p
7
AX=0137 BX=0000 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0009  NV UP EI PL NZ NA PO NC
076B:0009 8ADB      MOV     BL,AL
-p
AX=0137 BX=0037 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=000B  NV UP EI PL NZ NA PO NC
076B:000B B700      MOV     BH,00
-

```

As shown in the figure below, use the ASC [bx] addressing mode to find the ASCII code of the corresponding digit

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
AX=0137 BX=0037 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0012  NV UP EI NG NZ NA PO CY
076B:0012 7800      JS      0014
-p
AX=0137 BX=0037 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0014  NV UP EI NG NZ NA PO CY
076B:0014 83EB30     SUB     BX,+30
-p
AX=0137 BX=0007 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0017  NV UP EI PL NZ NA PO NC
076B:0017 EB07      JMP     0020
-p
AX=0137 BX=0007 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0020  NV UP EI PL NZ NA PO NC
076B:0020 8A8F0000   MOV     CL,[BX+0000]      DS:0007=37
-p
AX=0137 BX=0007 CX=0037 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0024  NV UP EI PL NZ NA PO NC
076B:0024 8AD1      MOV     DL,CL
-

```

As shown in the figure below, take out the found ASCII code and display it

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: ...
AX=0137 BX=0007 CX=003E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0020 NU UP EI PL NZ NA PO NC
076B:0020 8A8F0000 MOV CL,[BX+0000] DS:0007=37
-p
AX=0137 BX=0007 CX=0037 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0024 NU UP EI PL NZ NA PO NC
076B:0024 8AD1 MOV DL,CL
-p
AX=0137 BX=0007 CX=0037 DX=0037 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0026 NU UP EI PL NZ NA PO NC
076B:0026 B402 MOV AH,02
-p
AX=0237 BX=0007 CX=0037 DX=0037 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0028 NU UP EI PL NZ NA PO NC
076B:0028 CD21 INT 21
-p
?
AX=0237 BX=0007 CX=0037 DX=0037 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=002A NU UP EI PL NZ NA PO NC
076B:002A B44C MOV AH,4C
-

```

【2】 Program with 16-bit instructions to handle 32-bit addition, subtraction, multiplication, and division.

Require:

- (1) All variables are defined as word type, where negative numbers are expected. Some variables can also use registers, which are temporarily given under Debug. The program must be executed under Debug in order to verify the results.
- (2) Track the program and record the ZF, SF, CF, OF flags after each instruction is executed. Answer the reason why the ZF, SF, CF, OF flags are set after each instruction is executed.

The code of this question intends to calculate $(4 * X + YZ) / 7$ the value of code:

```

data segments
x dw 8;
y dw 4
z dw 6
v dd ?
data ends
code segments
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov ax,4

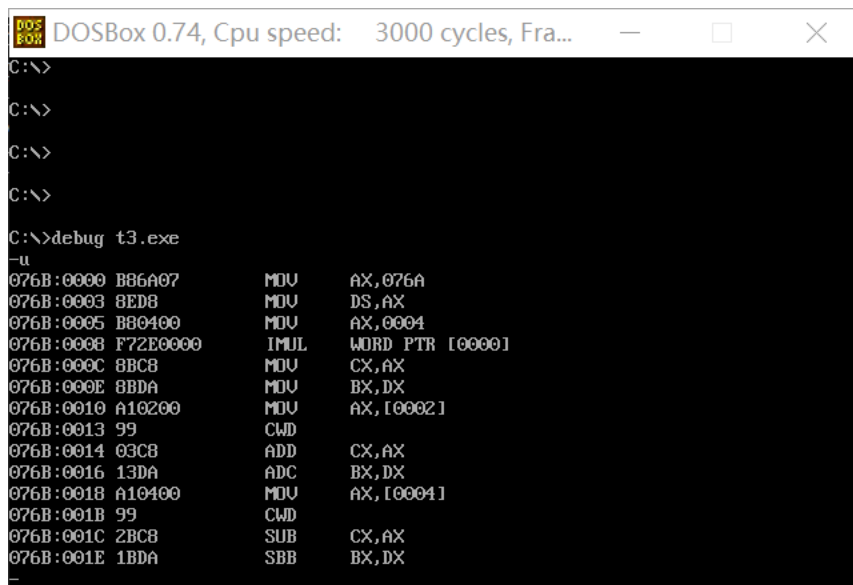
```

```

imul x
mov cx, ax
mov bx, dx
mov ax, y
cld
add cx, ax
adc bx, dx
mov ax, z
cld
sub cx, ax
sbb bx, dx
mov ax, cx
mov dx, bx
mov cx, 7
idiv cx
lea bx, v
mov [bx], dx
mov [bx+2], ax
mov ah, 4ch
int 21h
code ends
end start

```

实验结果：



The screenshot shows a DOSBox 0.74 window with the title bar "DOSBox 0.74, Cpu speed: 3000 cycles, Fra...". The command prompt shows the execution of "debug t3.exe". The assembly code is displayed in a table format with addresses, hex values, and assembly instructions.

Address	Hex Value	Instruction
076B:0000	B86A07	MOV AX, 076A
076B:0003	8ED8	MOV DS, AX
076B:0005	B80400	MOV AX, 0004
076B:0008	F72E0000	IMUL WORD PTR [0000]
076B:000C	8BC8	MOV CX, AX
076B:000E	8BDA	MOV BX, DX
076B:0010	A10200	MOV AX, [0002]
076B:0013	99	CWD
076B:0014	03C8	ADD CX, AX
076B:0016	13DA	ADC BX, DX
076B:0018	A10400	MOV AX, [0004]
076B:001B	99	CWD
076B:001C	2BC8	SUB CX, AX
076B:001E	1BDA	SBB BX, DX

```

DOSBox 0.74, Cpu speed: 3000 cycles, Fra...
Libraries [LIB]:
LINK : warning L4021: no stack segment

C:\>debug t3.exe
~p
AX=076A BX=0000 CX=0046 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076B IP=0003  NU UP EI PL NZ NA PO NC
076B:0003 8ED8      MOV     DS,AX
~p
AX=076A BX=0000 CX=0046 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0005  NU UP EI PL NZ NA PO NC
076B:0005 B80400     MOV     AX,0004
~p
AX=0004 BX=0000 CX=0046 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0008  NU UP EI PL NZ NA PO NC
076B:0008 F72E0000    IMUL    WORD PTR [0000]      DS:0000=0000
~p
AX=0020 BX=0000 CX=0046 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=000C  NU UP EI PL NZ NA PO NC
076B:000C 8BC8      MOV     CX,AX

```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Fra...
AX=0004 BX=0000 CX=0007 DX=0002 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0029  NU UP EI PL ZR NA PE NC
076B:0029 8D1E0600    LEA     BX,[0006]      DS:0006=0000
~p
AX=0004 BX=0006 CX=0007 DX=0002 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=002D  NU UP EI PL ZR NA PE NC
076B:002D 8917      MOV     [BX],DX      DS:0006=0000
~p
AX=0004 BX=0006 CX=0007 DX=0002 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=002F  NU UP EI PL ZR NA PE NC
076B:002F 894702     MOV     [BX+02],AX   DS:0008=0000
~p
AX=0004 BX=0006 CX=0007 DX=0002 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0032  NU UP EI PL ZR NA PE NC
076B:0032 B44C      MOV     AH,4C
~p
AX=4C04 BX=0006 CX=0007 DX=0002 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0034  NU UP EI PL ZR NA PE NC
076B:0034 CD21      INT     21

```

As shown in the figure, this code handles the formula of $(4*8+4-6)/7$, and the answer should be quotient 4 and remainder 2.

As shown in the screenshot, after Debug, AL=04, DL=02, it is in compliance.

During the above process,

ZF is the zero flag. If ZR is displayed, the operation result is zero; if NZ is displayed, the operation result is non-zero.

SF is a symbol flag. Displaying NG indicates that the highest bit of the operation result is '1'.

CF is the carry flag. Displaying CY means that there is a carry in the addition operation of two unsigned numbers, or there is a borrow in the subtraction operation, and the high bits need to be supplemented; displaying NC means that there is no carry or borrow.

OF is the overflow flag. When OV is displayed, it means that the operation result of two signed numbers exceeds the range that can be expressed, and the result is wrong; if NV is displayed, it means that there is no overflow, and the result is correct.

【3】 The data section has the following definition:

```
BUFF DB 'ABCD$EFGHIJK$ '
```

```
STR1 DB 12 DUP(?)
```

```
LEN DB ?
```

Write a program with string instructions to complete the following operations:

- (1) Put all ' * ' symbols on the string STR1.
- (2) Transfer the character string in BUFF to STR1 from left to right.
- (3) Transfer the character string in BUFF to STR1 from right to left.
- (4) Compare whether the two strings of BUFF and STR1 are equal, if they are equal, DX=1, otherwise DX=0.
- (5) Find whether there is a character \$ in BUFF, and count the number of occurrences of the character \$ into the BX register.

code:

(1)

```
DATA SEGMENT
```

```
BUFF DB 'ABCD$EFGHIJK$'
```

```
STR1 DB 12 DUP(?)
```

```
LEN DB ?
```

```
DATA ENDS
```

```
CODE SEGMENT
```

```
ASSUME CS:CODE, DS:DATA
```

```
START:
```

```

mov ax, data
mov es, ax
lea di, STR1
mov al, 2AH ;2ah '*'
mov cx, 12
cld
rep stosb

```

```

MOV AH, 4CH
INT 21H

```

```

CODE ENDS
END START

```



```

DATA SEGMENT
BUFF DB 'ABCD$EFGHIJK$'
      STR1 DB 12 DUP(?)
      LEN DB ?
DATA ENDS

```

```

CODE SEGMENT
ASSUME CS:CODE,DS:DATA

```

```

START:
    mov ax, data
    mov es, ax
    lea di, STR1
    mov al, 2AH
    mov cx, 12
    cld
    rep stosb

```

```

MOV AH, 4CH
INT 21H

```

```

CODE ENDS
END START

```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Fra...
0761:0070 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
-q
C:\>
C:\>
C:\>
C:\>debug f2.exe
-p
AX=076A BX=0000 CX=0040 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076C IP=0003 NU UP EI PL NZ NA PO NC
076C:0003 BEC0 MOV ES,AX
-d 076a:0
076A:0000 41 42 43 44 24 45 46 47-48 49 4A 4B 24 00 00 00 ABCD$EFGHIJK$...
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076A:0020 B8 6A 07 8E C0 8D 3E 0D-00 8E D8 8D 36 00 00 8D .j.....>.....6...
076A:0030 0E 0D 00 8D 16 00 00 2B-C8 FC F3 A4 B4 4C CD 21 .....*....L.!
076A:0040 00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04 .RP..H...P.t....
076A:0050 3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A =..t.....^.&.G.*
076A:0060 E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83 .eP.....RP..H.
076A:0070 C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6 ..P....P..s.....

```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Fra...
AX=076A BX=0000 CX=000D DX=0000 SP=0000 BP=0000 SI=0000 DI=000D
DS=076A ES=076A SS=0769 CS=076C IP=0019 NU UP EI PL NZ NA PO NC
076C:0019 FC CLD
-P
AX=076A BX=0000 CX=000D DX=0000 SP=0000 BP=0000 SI=0000 DI=000D
DS=076A ES=076A SS=0769 CS=076C IP=001A NU UP EI PL NZ NA PO NC
076C:001A F3 REPZ
076C:001B A4 MOUSB
-P
AX=076A BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=000D DI=001A
DS=076A ES=076A SS=0769 CS=076C IP=001C NU UP EI PL NZ NA PO NC
076C:001C B44C MOV AH,4C
-d 076a:0
076A:0000 41 42 43 44 24 45 46 47-48 49 4A 4B 24 41 42 43 ABCD$EFGHIJK$ABC
076A:0010 44 24 45 46 47 48 49 4A-4B 24 00 00 00 00 00 00 D$EFGHIJK$.....
076A:0020 B8 6A 07 8E C0 8D 3E 0D-00 8E D8 8D 36 00 00 8D .j....>....6...
076A:0030 0E 0D 00 8D 16 00 00 2B-CA FC F3 A4 B4 4C CD 21 .....*....L.!
076A:0040 00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04 .RP..H...P.t....
076A:0050 3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A =.t.....^.&.G.*
076A:0060 E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83 .eP.....RP..H.
076A:0070 C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6 ..P....P..s.....

```

(3)

DATA SEGMENT

BUFF DB 'ABCD\$EFGHIJK\$'

STR1 DB 12 DUP(?)

LEN DB ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE, DS:DATA

START:

MOV AX, DATA

MOV ES, AX

LEA DI, LEN-1

MOV DS, AX

LEA SI, STR1-1

MOV CX, STR1-BUFF

STD

REP MOVSB

MOV AH, 4CH

INT 21H

CODE ENDS

END START

```

DOSBox 0.74, Cpu speed: 3000 cycles, Fra...
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.

Run File [F3.EXE]:
List File [NUL.MAP]:
Libraries [LIB]:
LINK : warning L4021: no stack segment

C:\>debug f3.exe
~p
AX=076A BX=0000 CX=0040 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075A ES=075A SS=0769 CS=076C IP=0003 NU UP EI PL NZ NA PO NC
076C:0003 BEC0 MOV ES,AX
~d 076a:0
076A:0000 41 42 43 44 24 45 46 47-48 49 4A 4B 24 00 00 00 ABCD$EFGHIJK$...
076A:0010 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
076A:0020 B8 6A 07 8E C0 8D 3E 18-00 8E D8 8D 36 0C 00 8D .j....>....6...
076A:0030 0E 0D 00 8D 16 00 00 2B-CA FD F3 A4 B4 4C CD 21 .....*....L.!
076A:0040 00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04 .RP..H...P.€. ...
076A:0050 3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A =.t.....^.&.G.*
076A:0060 E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83 .eP.....RP..H.
076A:0070 C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6 ..P...P..s....

```

(4)

DATA SEGMENT

BUFF DB 'ABCD\$EFGHIJK\$'

STR1 DB 12 DUP(?)

LEN DB ?

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

```

mov ax, data
mov es, ax
lea di, STR1
mov ds, ax
lea si, buff
lea cx, STR1
lea dx, buff
sub cx, dx
cld
repe cmpsb
jz yes
mov dx, 0
yes:
mov dx, 1

```

MOV AH, 4CH

INT 21H