

Experiment 4 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  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
-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  NV 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  NV 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  NV 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.

code:

DATA SEGMENT

X DW 4

Y DW 2

Z DW 14H

V DW 18H

DATA ENDS

CODE SEGMENT

ASSUME CS:CODE,DS:DATA

START:

MOV AX,DATA

MOV DS,AX

MOV AX,X

IMUL Y

MOV CX,AX

MOV BX,DX

MOV AX,Z

CWD

ADD CX,AX

ADC BX,DX

SUB CX,16

SBB BX,0

MOV AX,V

CWD

SUB AX,CX

SBB DX,BX

IDIV X

MOV AH,4CH

INT 21H

CODE ENDS

END START

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)

MOV AX, DATA

MOV DS,AX

MOV AL, '*'

LEA DI, STR1

MOV CX,STR1 - BUFF

CLD

REP STOSB

(2)

MOV AX, DATA

MOV DS,AX

MOV ES,AX

CLD extension

LEA YES, BUFF

LEA DI, STR1

MOV CX, STR1 - BUFF

REP. MOVSB

(3)

MOV AX,DATA

MOV DS,AX

MOV ES,AX

STD

LEA SI, STR1 - 1

LEA DI, LEN - 1

MOV CX, STR1 - BUFF

REP. MOVSB

(4)

MOV AX,DATE

MOV DS,AX

MOV ES,AX

CLD

LEA SI,BUFF

LEA DI,STR1

MOV CX,STR1-BUFF

REPE CMPSB

(5)

MOV AX,DATA

MOV ES,AX

MOV BX,0

CLD

MOV AL,'\$'

LEA SI,BUFF

MOV CX,STR1-BUFF

NEXT:REPNE SCASB

JCXZ NO-FOUND

INC BX

JMP NEXT