PUBLIC DISPBX

Data segment

N1 db 18

N2 db 36

N3 db 54

Data ends

CODE SEGMENT

ASSUME CS:CODE

DISPBX PROC FAR

MOV CH,4

MOV CL,4

ROTATE: ROL BX,CL

MOV AL,BL

AND AL,0FH

ADD AL,30H

CMP AL,3AH

JL PRINTIT

ADD AL,7

PRINTIT:MOV DL,AL

MOV AH,2

INT 21H

DEC CH

JNZ ROTATE

RET

Main:

Mov ax, data

Mov ds, ax

Mov bl, n1

Mov ch, 4

Mov bh, 0 ;将高位清零，低位赋值，以达到输出

Mov cl, 4

DISPBX ENDP

CODE ENDS

END

Code segment

Assume cs:code, ds:data

Rotate1:

Rol bx, cl

Mov al, bl

And al, 0fh

Add al, 30h

未完待续（现在在手里的就是个没用的程序）

子程序设计：

Call 返回地址进栈，转入子程序执行

Ret 从栈上面去地址，返回程序执行

Ppt3：

段内间接调用

目的的偏移地址存放在寄存器或者内存单元中

Call dx

汇编格式： call reg/mem

Ppt4：

段间间接远调用

目的的偏移地址……

Return：

Ppt6：

格式：ret/ret【imm16】

段内近返回：ret

段间远返回：retf

模块化程序设计方法：

子程序参数传递方法：

根据子程序的功能，

比如显示bx中的数值

那么参数传递的方法就是把要显示的数字传入bx中

（约定参数传递）

有三种方法：内存（变量），寄存器，堆栈

例子：

写个程序，求n个16位补码的累加和，十进制方式显示

三个子程序

Mov cx, len

Mov s, 0

Begin: add s, [SI]

Inc si / add si, 2

Loop begin

用栈来传递参数

根据子程序的要求，依次序将参数压入栈，之后弹出，

但是需要注意的是，pop和push的方向是相反的。

F proc Push bp

Mov bp, sp

Mov ax, [bp + 4] ;现在bp是sp了

Add ax, [bp + 6]

Sub ax, [bp + 8]

Pop bp

Ret

ENDP

MAIN: Mov ax, 5

Push ax

Mov ax, 4

Push ax

Mov ax, 3

Push ax

Call F

Add sp, 6

Mov r1, ax

Mov ax, 50

Push ax

Mov ax, 40

Push ax

Mov ax, 30

Push ax

Call F

Add sp, 6

Mov r2, ax