4.2

如果AX大于BX，CX:1

如果AX等于BX，CX:0

如果AX等于BX，CX:-1

这里的比较是有符号数

4.3

简化后的程序：

MOV AX,X

CMP AX,Y

JC L1

JO L2

JE L3

JNS L4

L3:

ADD AX,Y

JC L5

4.5

(2)循环执行0FFFFH+1次

(3)如果漏掉了该指令循环的次数由CX寄存器原本的数据决定，故不能够确定循环数据

(4)如果漏掉了该指令那么会把BUF1中的第一个地址放到BUF2的前n个内存单元中

(5)如果把标号向上移一行，如果你为1，那么循环执行一次直接退出，没有影响，如果n不为1，那么程序会进入死循环，因为每循环一次，CX又被重新赋值，并不会发生改变。

4.11

源程序：

RADIX PROC

PUSH CX

PUCH EDX

XOR CX,CX

LOP:

XOR EDX,EDX

DIV EBX

PUSH DX

INC CX

OR EAX,EAX

JNZ LOP1

LOP2:

POP AX

CMP AL,10

JB L1

ADD AL,7

L1:

ADD AL,30H

MOV [SI],AL

INC SI

LOOP LOP2

POP EDX

POP CX

RET

RADIX ENDP

上述程序直接使用了寄存器EAX作为参数传递

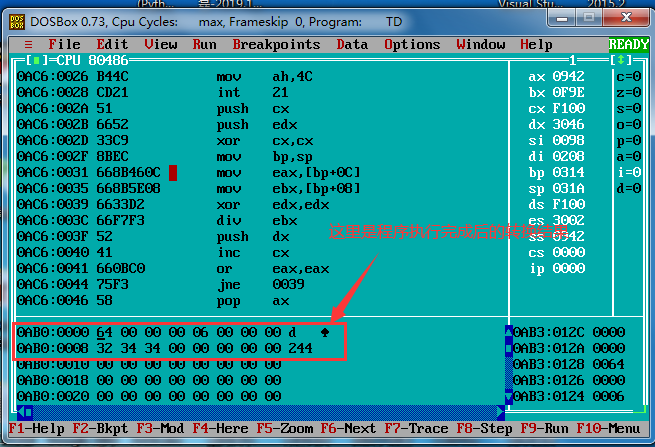
接下来我把程序改为堆栈传送参数：

其中我把N设置为100，转换为6进制，所以P的值为6，分配了20个字大小的数组用来存储asc码。

在调用子程序之前先把N和P压入栈中。把EAX,EBX置0。

调用子程序，把BP赋值为SP，然后通过访问BP+12的堆栈段单元，把100放入EAX寄存器中，再访问BP+8的堆栈单元，把6放入EBX寄存器中，然后执行操作，把转换的数字的asc码存入NUM数组中。

程序截图如下：



这里我们发现最终的转换结果是244（6进制），满足要求，右下角的堆栈段0128地址存储的64（即十进制的100），0124的地址存储6（转换的进制）。满足设计要求。

源程序如下：

.386

DATA SEGMENT USE16

N DD 100

P DD 6

NUM DW 20 DUP(0)

DATA ENDS

STACK SEGMENT USE16 STACK

DB 300 DUP(0)

STACK ENDS

CODE SEGMENT USE16

ASSUME DS:DATA,SS:STACK,CS:CODE,ES:DATA

START:

MOV AX,DATA

MOV DS,AX

MOV ES,AX

MOV EAX,N

PUSH EAX

MOV EBX,P

PUSH EBX

MOV EAX,0

MOV EBX,0

MOV SI,OFFSET NUM

CALL RADIX

MOV AH,4CH

INT 21H

RADIX PROC

PUSH CX

PUSH EDX

XOR CX,CX

MOV BP,SP

MOV EAX,[BP+12]

MOV EBX,[BP+8]

LOP1:

XOR EDX,EDX

DIV EBX

PUSH DX

INC CX

OR EAX,EAX

JNZ LOP1

LOP2:

POP AX

CMP AL,10

JB L1

ADD AL,7

L1:

ADD AL,30H

MOV [SI],AL

INC SI

LOOP LOP2

POP EDX

POP CX

RET

RADIX ENDP

CODE ENDS

END START