练习题

- （1）已知某32位整数X，其值为-101（十进制），则其16进制补码为___，另一32位整数Y的补码为0xFFFFFF6A，则X+Y的16进制补码(32位)为____，X-Y的16进制补码为____。

- （2）在所有由四个“1”和四个“0”组成的8位二进制整数（补码）中，最小的数是（ ）。
  - （1）－128　（2）－122　（3）－121　（4）－64

- （3）给定一个浮点数格式，有k位指数和n位小数，对于下列数，写出E 、M、小数f和值V的公式，并请描述其位表示。
  - A. 数5.0;
  - B. 能够被准确描述的最大奇整数；
  - C. 最小的正规格化数。

（4）

一个函数的原型为

```
int decode2(int x, int y, int z);
```

```
    x at %ebp+8, y at %ebp+12, z at %ebp+16
1       movl    16(%ebp), %edx
2       subl    12(%ebp), %edx
3       movl    %edx, %eax
4       sall    $15, %eax
5       sarl    $15, %eax
6       xorl    8(%ebp), %edx
7       imull   %edx, %eax
```

参数 x、y 和 z 存放在存储器中相对于寄存器 %ebp 中地址偏移量为 8、12 和 16 的地方。代码将返回值存放在寄存器 %eax 中。

写出等价于我们汇编代码的 decode2 的 C 代码。

（5）

一种C函数调用的参数传递与栈恢复模式被称为fastcall，其与我们课堂上学的默认的C函数参数传递与栈恢复方法不同(称为cdecl)。请根据下列的C代码与汇编代码的对应关系，写出fastcall的参数是如何传递的，即哪些参数是通过通用寄存器传递（包括对应关系如何），哪些是通过栈传递、压栈顺序如何，以及传递参数的栈是由被调者还是调用者负责恢复。

```
int __fastcall dummy(int w, int x, int y, int z )
{
    return ((w*2 + (x+y))>>2 - y*23 ) * z;
}

void calldummy(int w, int x, int y, int z)
{
    int res = dummy(w, x, y, z);
}
```

```
_calldummy:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        ……
        call    dummy
        subl    $8, %esp
        leave
        ret
```

```
dummy:
        ……
        movl    8(%ebp), %ebx
        addl    %ebx, %edx
        leal    (%edx,%ecx,2), %eax
        leal    (%ebx,%ebx,2), %edx
        movl    $2, %ecx
        sall    $3, %edx
        subl    %ebx, %edx
        subl    %edx, %ecx
        sarl    %cl, %eax
        movl    12(%ebp), %ecx
        imull   %ecx, %eax
        ……
        ret     $8
……
```

```
_function:
        pushl   %ebp
        movl    %esp, %ebp
        pushl   %edi
        xorl    %edi, %edi
        pushl   %esi
        xorl    %esi, %esi
        pushl   %ebx
        xorl    %ebx, %ebx
L11:
        xorl    %ecx, %ecx
        leal    _N(%edi), %edx
        jmp     L10
L18:
        movl    (%edx), %eax
        incl    %ecx
        addl    $4, %edx
        addl    %eax, %ebx
        cmpl    $9, %ecx
        jg      L17
L10:
        leal    (%esi,%ecx), %eax
        testb   $1, %al
        jne     L18
        movl    (%edx), %eax
        incl    %ecx
        addl    $4, %edx
        subl    %eax, %ebx
        cmpl    $9, %ecx
        jle     L10
```

```
L17:
        incl    %esi
        addl    $40, %edi
        cmpl    $9, %esi
        jle     L11
        movl    %ebx, %eax

        popl    %ebx
        popl    %esi
        popl    %edi
        popl    %ebp

        ret

.comm   _N, 400   # 400
```

```
int N[10][10];

//初始化数组省略!
int function(int n)
{
        int res = 0;
        int i ,j;
        for( i = 0; i < 10; i++)
        {
                //????
        }
        return res;
}
```

（6）

（7）C函数返回struct类型是如何实现的？

```
typedef struct{
int age; int bye; int coo; int ddd; int eee;
} TEST_Struct;

int i = 2;

TEST_Struct __cdecl return_struct(int n )
{
        TEST_Struct local_struct;
        local_struct.age = n;
        local_struct.bye = n;
        local_struct.coo = 2*n;
        local_struct.ddd = n;
        local_struct.eee = n;

        i = local_struct.eee + local_struct.age *2 ;

        return local_struct;
}

int function1()
{
        TEST_Struct main_struct =
                return_struct(i);

        return 0;
}
```


?

（8）C函数是如何传入struct类型参数的？

```
typedef struct{
int age; int bye; int coo; int ddd; int eee;
} TEST_Struct;

int i = 2;
int input_struct(TEST_Struct in_struct)
{
        return in_struct.eee + in_struct.age*2 ;
}
int function2()
{
        TEST_Struct main_struct;
        main_struct.age = i;
        main_struct.bye = i;
        main_struct.coo = 2*i;
        main_struct.ddd = i;
        main_struct.eee = I;
        return input_struct(main_struct);
}
```

?

（9）过程调用以及返回的顺序在一般情况下都是“过程返回的顺序恰好与调用顺序相反”，但是我们可以利用汇编以及对运行栈的理解来编写汇编过程打破这一惯例。有如下汇编代码，其中GET过程唯一的输入参数是一个用于存储当前处理器以及栈信息的内存块地址（假设该内存块的空间足够大）。而SET过程则用于恢复被GET过程所保存的处理器及栈信息，其唯一的输入参数也是该内存块地址。

问题：

（1）SET过程的返回地址是什么，其返回值是多少？

（2）代码段中的（A）指令执行后，eax中存放的是什么？(B) 指令执行后，ecx中存放的是什么？(C) 指令的作用是什么?（E）指令的作用是什么？并将（D）指令补充完整。

```
GET:
movl 4(%esp), %eax   #(A)
…
movl %edi, 20(%eax)
movl %esi, 24(%eax)
movl %ebp, 28(%eax)
movl %ebx, 36(%eax)
movl %edx, 40(%eax)
movl %ecx, 44(%eax)
movl $1, 48(%eax)
movl (%esp), %ecx   #(B)
movl %ecx, 60(%eax)
leal 4(%esp), %ecx   #(C)
movl %ecx, 72(%eax)
movl 44(%eax), %ecx
movl $0, %eax

ret
```

```
SET:
movl 4(%esp), %eax

…
movl 20(%eax), %edi
movl 24(%eax), %esi
movl 28(%eax), %ebp
movl 36(%eax), %ebx
movl 40(%eax), %edx
movl 44(%eax), %ecx
movl ____(%eax), %esp   #(D)
pushl 60(%eax)  #(E)
movl 48(%eax), %eax


ret
```