Homework 1

（1）

（a）简要注释每条汇编代码；

```
.file "foo.c"        //表示汇编来源的原始文件名
    .text            //代码开始
.globl fact          //定义一个全局符号,fact 作为函数名
    .type    fact,@function    //定义符号 fact 的类型为一个 function
fact:                //局部函数 fact 开始
    pushl    %ebp          //将 ebp 寄存器中长字类型的内容压入栈中
    movl     %esp, %ebp        //将 esp 中的内容传送到 ebp 中
    subl$4, %esp           //esp 指针向下移动 4 个字节，给局部变量留出空间
    cmpl     $0, 8(%ebp)       //将 ebp+8 所对应地址中的数与 0 作比较
    jg   .L2              //若比较结果大于 0 则跳转至 L2
    movl     $1, -4(%ebp)      //将立即数 1 存入 ebp-4 所对应的地址
    jmp .L1               //跳转至 L1
.L2:
    subl$12, %esp          //esp 指针向下移动 12 个字节
    movl     8(%ebp), %eax         //将 ebp+8 对应地址的内容送入 eax 中
    decl%eax          //执行减一指令，eax 中存的数值减一
    pushl    %eax          //将 eax 中的内容入栈
    call fact       //调用 fact 函数
    addl$16, %esp          //让 esp 指向 esp+16 所对应的地址单元
    imull    8(%ebp), %eax         //将 ebp+8 对应地址的内容与 eax 中内容相乘并
存入 eax
    movl     %eax, -4(%ebp)        //将 eax 中的内容送入 ebp-4 对应的地址中
.L1:
    movl     -4(%ebp), %eax        //将 ebp-4 对应地址中的内容送入 eax 中
    leave                 //关闭栈帧
    ret                   //恢复断点，返回主程序
.Lfe1:
    .size    fact,.Lfe1-fact    //fact 函数大小
    .ident   "GCC: (GNU) 3.2.2 20030222 (Red Hat Linux 3.2.2-5)"//编译器标识
```

（b）尝试指出 C 程序与汇编代码间的联系，

```
int fact( int n )
    {
        if (n<= 0) return 1;
        else return ( n*fact(n-1));
    }
```

1、**函数声明**

C 程序中 fact 为函数，对应汇编代码.typefact,@function

2、**函数参数**

fact 函数的参数 n 在汇编中以 8(%ebp)表示，其中汇编代码 cmpl $0, 8(%ebp)对应 C 程序中的判断语句 if (n<= 0)，由此可以看出参数 n 在 ebp+8 这个内存单元中

3、**if 条件语句**

if 语句对应满足 n<=0 以及 n>0 的两种情况

其中 n<=0 时对应汇编代码中的 movl$1，-4(%ebp)以及 jmp .L1，跳转至汇编代码中 L1 代码段，L1 代码段的执行会结束函数的运行并返回 1，对应递归程序中 n<=0 时返回值为 1 的功能；

而 n>0 时，汇编代码对应执行 jg .L2，即跳转至 L2 代码段

4、**递归与阶乘**

对于 C 程序中 n*fact(n-1)，其中参数 n-1 对应 L2 代码段中的 decl %eax，对 eax 中的值执行减一操作，并再次调用 fact 函数实现递归，对应汇编中的 call fact，传入参数为 n-1, 计算阶乘则对应汇编 imull 8(%ebp), %eax

（2）针对以下 C 程序，给出其输出结果。

并简要分析其中带有下划线的语句：**cout << i << endl;**的执行或输出情况。

输出结果：0434207836

结果分析：根据嵌套的多层 switch 语句以及其判断条件分析可知，输出的值分别为 0、4、3、4、20、7、8、36

执行情况：cout << i << endl 语句在代码运行过程中不会被执行，因为此语句在 switch 语句中不对应任何一个 case，并且也不在 default 语句中，因此不会被执行。

（3）针对以下 C/C++程序：

（3.1）用文字简要描述以下 C 变量 p 的类型信息。

int (*(*(*p(int x))())[20])(int *y);

变量 p 是一个函数，该函数有一个整型参数，p 函数返回一个指针，该指针指向一个无参的函数，该无参的函数返回一个指针，该指针指向一个具有 20 个元素的数组，该数组的每一个元素都是一个指针，并且指向一个有整型参数的函数，该函数返回一个整数

（3.2）给出以下程序在你的电脑上运行结果：

```
    void main(){
int iii = 1000;
int & ii = iii;
int *pii = &ii;
int * &pr = pii;
cout << "iii = " << iii << " @: " << &iii << endl;
cout << "ii = " << ii << " @: " << &ii << endl;
cout << "pii points to: " << pii << " with value = " << *pii << " @: " << &pii << endl;
cout << "pr points to: " << pr << " with value = " << *pr << " @: " << &pr << endl;
return ;
}
```

运行结果为：

iii = 1000 @: 0x61fe0c
ii = 1000 @: 0x61fe0c
pii points to: 0x61fe0c with value = 1000 @: 0x61fe00
pr points to: 0x61fe0c with value = 1000 @: 0x61fe00