**第 5周作业 函数调用**

1. CPU 在执行CALL指令、RET指令时，会分别完成什么操作？
2. 设执行如下语句前， ESP = 0x12345678，指令地址为 0x004A1090  
   004A1090 E8 32 00 00 00 call func ; 子程序直接调用  
   在执行CALL 后， EIP = ？ ESP =？ 栈顶的双字数据是多少？
3. 设 EAX = 0x004A1090 EBX = 0x004A109A ESP=0x12345678

执行如下程序段 PUSH EAX

PUSH EBX

RET

执行 RET 指令后， EIP = ？ ESP = ？

1. **对下面的程序进行调试 （x86, Debug版），观察反汇编窗口、寄存器窗口、内存窗口等，回答问题**

注意：为了看到更紧凑的变量空间分配方式，可将项目属性 C/C++ -> 代码生成-> “基本运行时检测”设置为“默认值”； “安全检查”设置为“禁用安全检查”。每个同学都根据自己的观察回答问题。

int fadd(int a, int b)

{ int temp;

a+=10;

b+=20;

temp = a + b;

return temp;

}

void main( )

{ int x = 0x1234;

int y = -32;

int result = 0;

result = fadd(x, y);

x=30;

}

1. 画出 执行result=fadd(x,y) 之前的堆栈示意图。

要求在堆栈中标明 x、y、result位置，以及具体的地址值；指出 ESP的值、EBP的值；在堆栈中标明 ESP指向的位置。中间不相关的内容可以用省略号。

(2) 给出result=fadd(x,y); 的反汇编截图；

(3) 调用函数时是如何传递参数的，有何规律？参数所占的空间是如何“释放的”？（即参数在的位置）

(4) 画出进入函数fadd，保存 EBP 后，堆栈的示意图（要包括main中局部变量x,y,result等相关内容，在栈单元外，标明地址；中间有些不相关的内存单元可用省略号代替）。写出参数 a、b的地址表达形式，以及具体的地址值。写出fadd中temp的地址表达形式，以及具体的地址值。  
  
 (5) 执行完 result=fadd(x,y)后，ESP、EBP的值各是多少？与执行result=fadd(x,y) 之前的 ESP、EBP是否相同？如何看待函数执行对栈空间的使用（参数以及局部变量的空间分配与释放如何完成的）？

1. 设有如下程序

int fadd(int a, int b)

{ int temp;

temp = a + b;

return temp;

}

int\* fadd\_bug(int a, int b)

{ int temp; // 编译时会有警告

temp = a + b;

return &temp;

}

void main( )

{ int x = 1;

int y = 2;

int result;

int \*p;

p = fadd\_bug(x, y);

result = \*p; // ①

x = fadd(10,20);

result =\*p; // ②

}

同上一题：“基本运行时检测”设置为“默认值”； “安全检查”设置为“禁用安全检查”。

1. 在程序编译时，会给出什么警告信息？
2. 在 执行①处的 result =\*p后， result=? p=? (指p中存放的地址)  
    在 执行②处的 result =\*p后， result=? p=?
3. 为什么返回局部变量的地址是一种危险的做法？