```
long swap_add(long *xp, long *yp)
{
    long x = *xp;
    long y = *yp;
    *xp = y;
    *yp = x;
    return x + y;
}

long caller()
{
    long arg1 = 534;
    long arg2 = 1057;
    long sum = swap_add(&arg1, &arg2);
    long diff = arg1 - arg2;
    return sum * diff;
}
```

a）swap_add和调用函数的代码

```
    long caller()
1   caller:
2     subq    $16, %rsp          Allocate 16 bytes for stack frame
3     movq    $534, (%rsp)       Store 534 in arg1
4     movq    $1057, 8(%rsp)     Store 1057 in arg2
5     leaq    8(%rsp), %rsi      Compute &arg2 as second argument
6     movq    %rsp, %rdi         Compute &arg1 as first argument
7     call    swap_add           Call swap_add(&arg1, &arg2)
8     movq    (%rsp), %rdx       Get arg1
9     subq    8(%rsp), %rdx      Compute diff = arg1 - arg2
10    imulq   %rdx, %rax         Compute sum * diff
11    addq    $16, %rsp          Deallocate stack frame
12    ret                        Return
```

b）调用函数生成的汇编代码

图 3-31　过程定义和调用的示例。由于会使用地址运算符，所以调用代码必须分配一个栈帧

```
long call_proc()
{
    long  x1 = 1; int  x2 = 2;
    short x3 = 3; char x4 = 4;
    proc(x1, &x1, x2, &x2, x3, &x3, x4, &x4);
    return (x1+x2)*(x3-x4);
}
```

a）swap_add和调用函数的代码

图 3-32　调用在图 3-29 中定义的函数 proc 的代码示例。该代码创建了一个栈帧