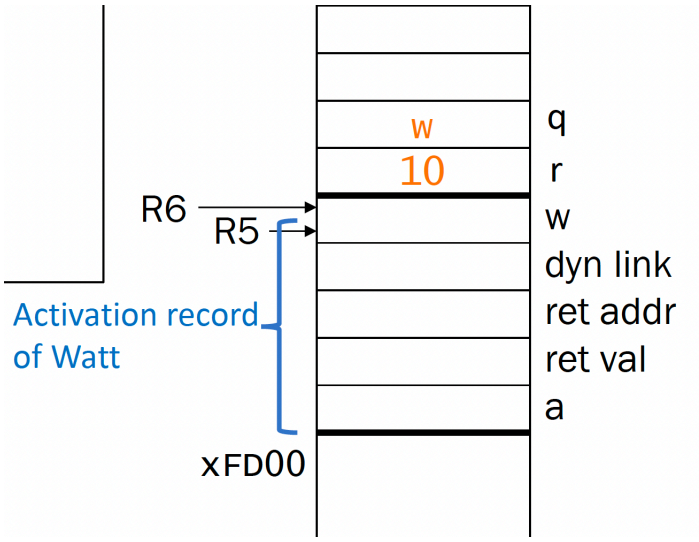
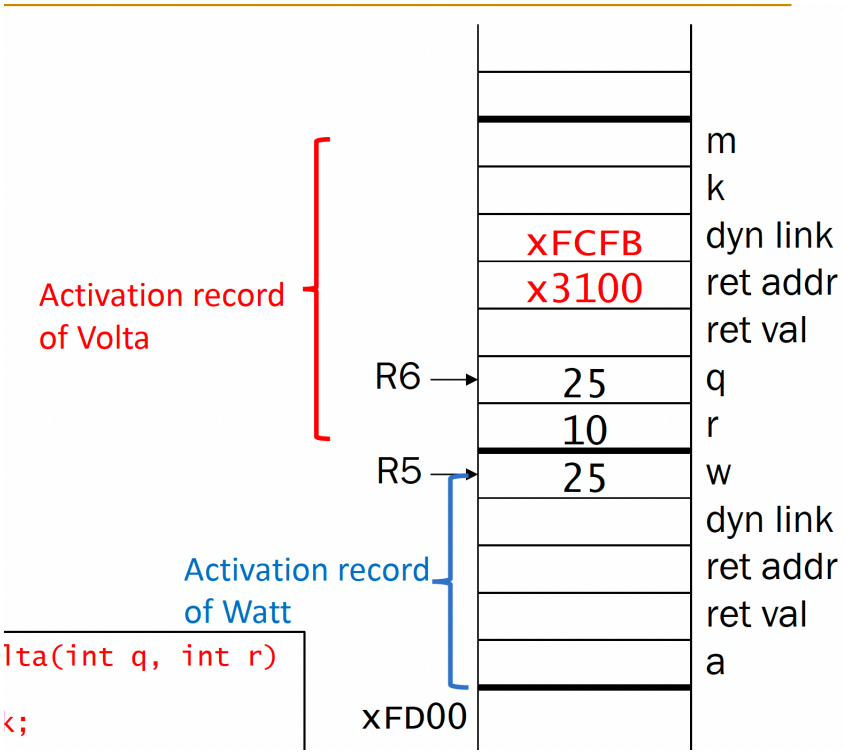
```
int Volta(int q, int r){
    int k;
    int m;
    ...
    return k;
}
int Watt(int a){
    int w;
    ...
    w = Volta(w,10);
    ...
    return w;
}
```




```
;此处Watt本身也是一个callee，所以它也有对应的return value，return address，dynamic link
; 下面的函数对应于caller setup(push callee's arguments onto stack)和pass control to callee
Watt
...
; Push second arg (10)
AND R0, R0, #0
ADD R0, R0, #10 ; R0 = 10
ADD R6, R6, #-1
STR R0, R6, #0 ; 把R0存入R6的值代表的地址中
;  Push first arg (w)
LDR R0, R5, #0 ; R0 = w，因为R5在callee中总是在其内部的第一个局部变量的位置，即R5的值是该位置
ADD R6, R6, #-1
STR R0, R6, #0 ; 把R0存入R6的值代表的地址中，此时栈中图像如上所示

JSR Volta ; 跳转到Volta
```



; 下面的函数对应于Callee setup(push bookkeeping info and local variables onto stack)和execute funciton Volta

ADD R6, R6, #-1; 给return value腾出位置

ADD R6, R6, #-1; 现在R6应该位于return address的位置

STR R7, R6, #0

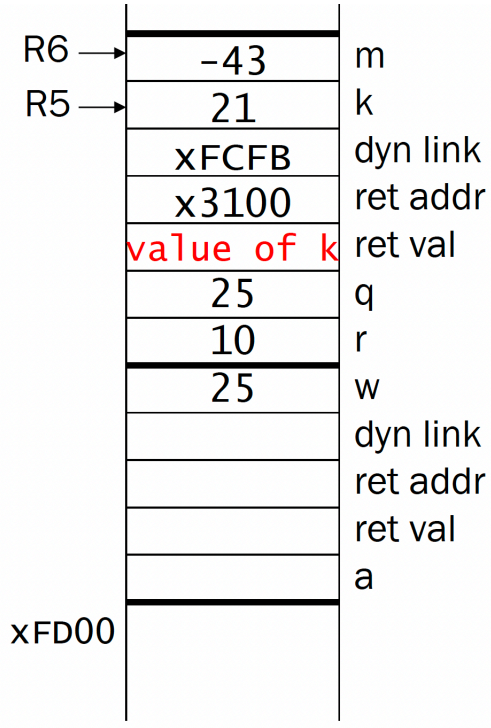
ADD R6, R6, #-1; 现在R6应该位于dynamic link(caller's frame ptr)的位置

STR R5, R6, #0; 把上一个R5中的值存到这里，此时这个位置中的值是caller中第一个局部变量的地址，即w(假定为25)

; 此时栈图如上

ADD R5, R6, #-1; 现在R5就是callee中第一个局部变量的地址

ADD R6, R6, #-2; 现在R6在callee中第二个局部变量的地址，作为栈顶



; 下面的函数对应于Callee tear-down (update return value, pop local variables, caller's frame pointer and return address from stack)以及return to caller
; 假如我已经通过一些操作把形参k和m的值赋好了，如上图所示
LDR R0, R5, #0 ; 就是把Volta的第一个局部变量的值(k)赋值到R0中
STR R0, R5, #3 ; 由于R5指向第一个局部变量的地址，所以它上面三个就是return value，这一步就是把函数得到的数值存入return value的地址中，其可以直接用R5+3的偏移量代替
ADD R6, R5, #1 ; 把局部变量都弹出栈，此时栈顶在R5+1，也就是存放dynamic link的地址，而dynamic link的值就是caller function的第一个局部变量的地址
LDR R5, R6, #0 ; 把dynamic link的值还给R5
ADD R6, R6, #1 ; 此时R6在存放return address的地址
LDR R7, R6, #0 ; 把return address的值还给R7
ADD R6, R6, #1 ; 此时R6在return value的地址
RET ; 返回caller函数

; 下面的函数对应于caller tear-down(pop callee's return value and arguments from stack)
LDR R0, R6, #0 ; 在上面的代码中，R6就是return value的地址，因此用LDR把return value放到R0中
STR R0, R5, #0 ; 这里面的偏移量不一定是#0，看你要修改的变量是第几个局部变量
ADD R6, R6, #1 ; 此时R6对应的是callee最后一个形参(最左边的形参)的地址
ADD R6, R6, #2 ; 由于形参一共俩，所以我先+1再+2之后，栈中就没有callee的参数了，也就没有callee的任何痕迹了。