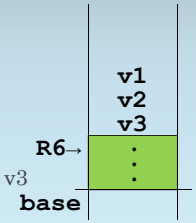
Finish by Restoring R1 and Returning

```
SUM_OF_3
ST R1,SAVE_R1    ; save R1
LDR R0,R6,#0     ; R0 ← v1
LDR R1,R6,#1     ; R1 ← v2
ADD R0,R0,R1     ; R0 ← v1 + v2
LDR R1,R6,#2     ; R1 ← v3
ADD R0,R0,R1     ; R0 ← v1 + v2 + v3
ADD R6,R6,#3     ; pop all three
LD R1,SAVE_R1    ; restore R1
RET
SAVE_R1 .BLKW #1
```



Restore R1 and return.

Breaking the Abstraction Can Be Done Safely

To use SUM_OF_3,
- push three values, call SUM_OF_3, and use the result in R0.
- Or allocate three locations with one ADD, write in three values, then call …

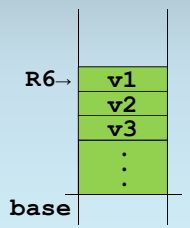
We can **safely use**
- **any data on the stack**
- **if we know that it's there**.

Can We Generalize SUM_OF_3 to SUM_OF_N?

The picture to the right shows
- an **array of three integers**
- on top of the stack.

What if we want to generalize?

Can we write a subroutine
- that **adds a variable number of non-negative numbers**
- **from an array on top of the stack**?



Can We Generalize SUM_OF_3 to SUM_OF_N?

Can we write a subroutine that adds N non-negative numbers from the top of the stack?

Yes!

But **the subroutine must know the value of N**.