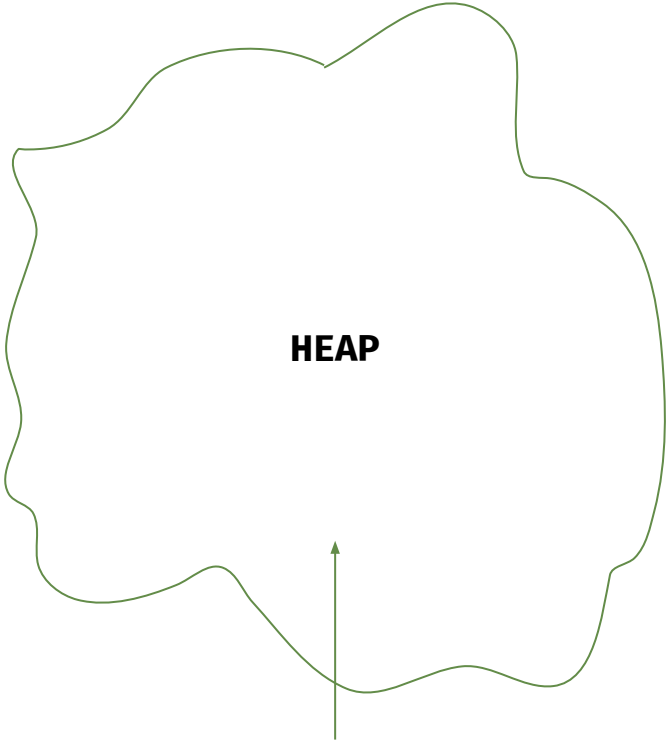
0X BARE C8 METAL

CMPSC 0XC8 ALLEGHENY COLLEGE 0X7E7

Stack and Heap

STACK
0xdef45ea5
0x134a48fd
0x6785efae

Lawful good

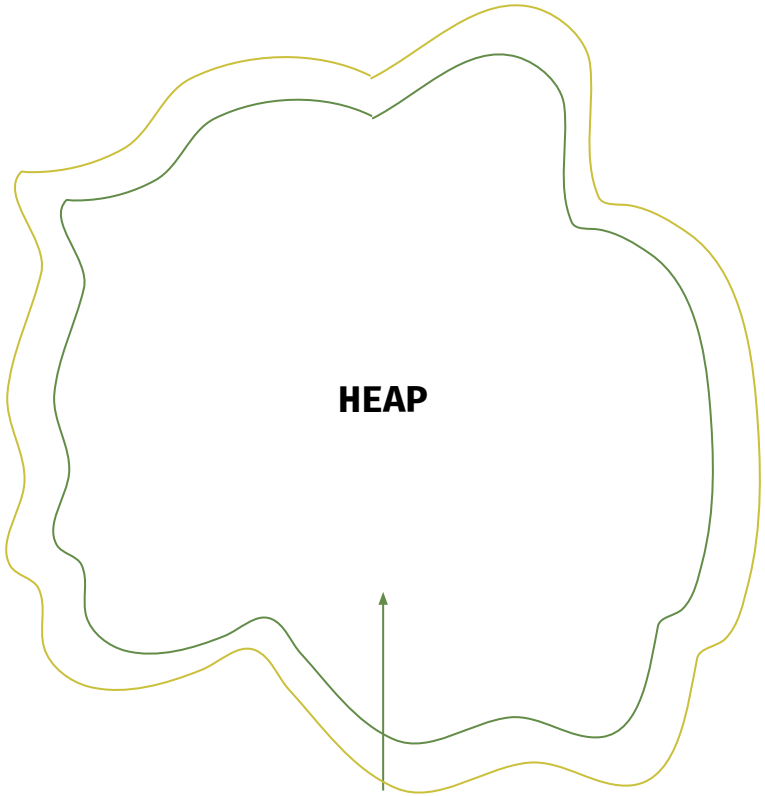


Chaotic neutral

Stack and Heap

STACK
0xdef45ea5
0x134a48fd
0x6785efae

Lawful good



Chaotic neutral

Brief introduction to the “stack”

There exists another quasi-temporary spot for memory which can get us out of a tricky jam: the *stack*.

LIFO
Last In, First Out

VALUE
0xdef45ea5
0x134a48fd
0x6785efae

“grows down”

Brief introduction to the “stack”

PUSH
Places a value in the stack *in order*

REGISTERS	
R0	0xdef45ea5
R1	0x134a48fd
R2	0x6785efae

PUSH {R0-R2}

STACK
0xdef45ea5
0x134a48fd
0x6785efae

Brief introduction to the “stack”

POP
Extracts values from
the stack *in order*

REGISTERS	
R0	0x6785efae
R1	
R2	

POP {R0}

STACK
0xdef45ea5
0x134a48fd
0x6785efae

Brief introduction to the “stack”

POP
Extracts values from
the stack *in order*

REGISTERS	
R0	0x6785efae
R1	0x134a48fd
R2	0xdef45ea5

POP {R1,R2}

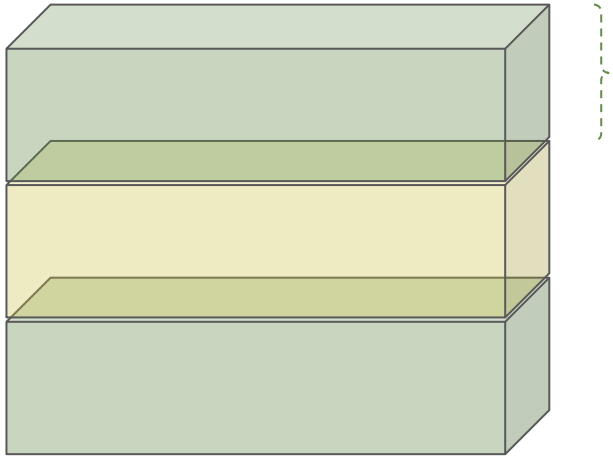
STACK
0xdef45ea5
0x134a48fd
0x6785efae

Brief introduction to the “stack”

PUSH $\{R_M, R_D, \ldots\}$ POP $\{R_M, R_D, \ldots\}$

PUSH $\{R_M - R_D\}$ POP $\{R_M - R_D\}$

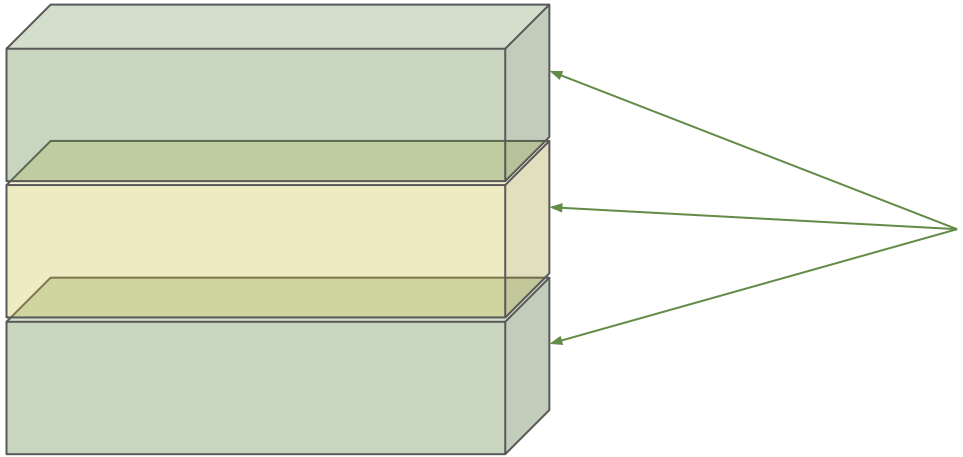
Getting framed



Individual stack “frame”

Contains all PUSH’d, POP’d values for a given subroutine

Getting framed

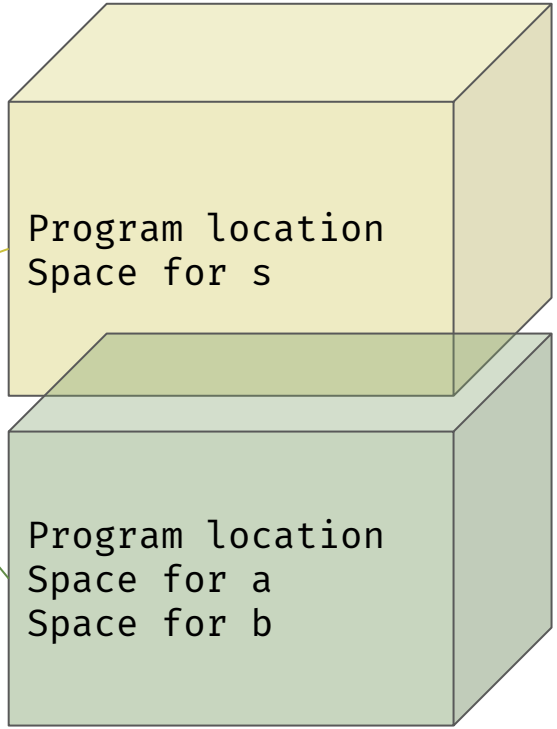


Implies: each subroutine can have its own stack “frame”

Getting framed

```
def add(a: int, b: int) -> int:
    c = a + b
    return c


s = add(2, 3)
print(s)
```

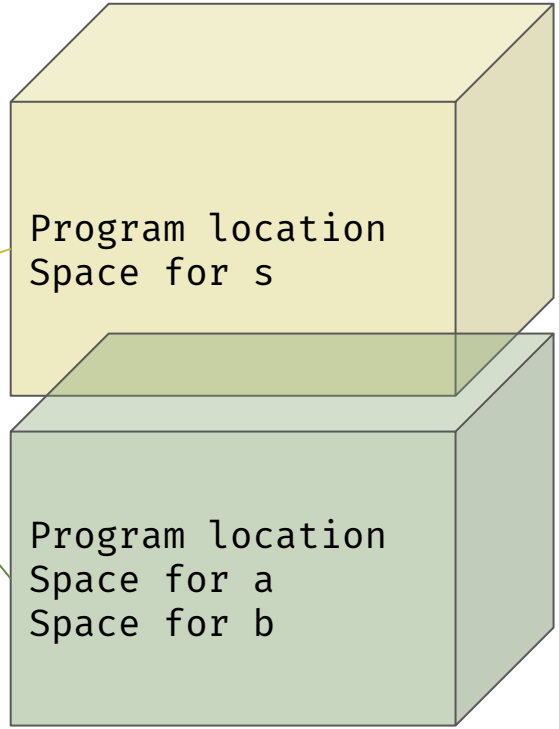


Getting framed

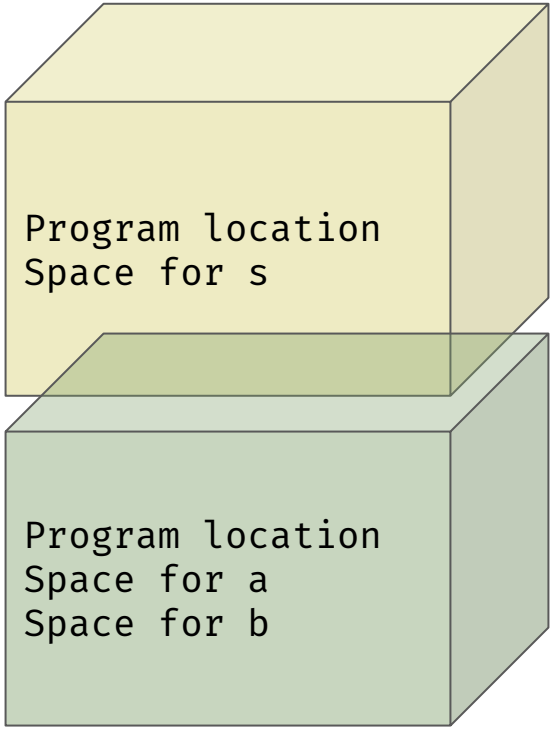
```
def add(a: int, b: int) -> int:
    c = a + b
    return c

s = add(2, 3)
print(s)
```

But we can't do this, can we?



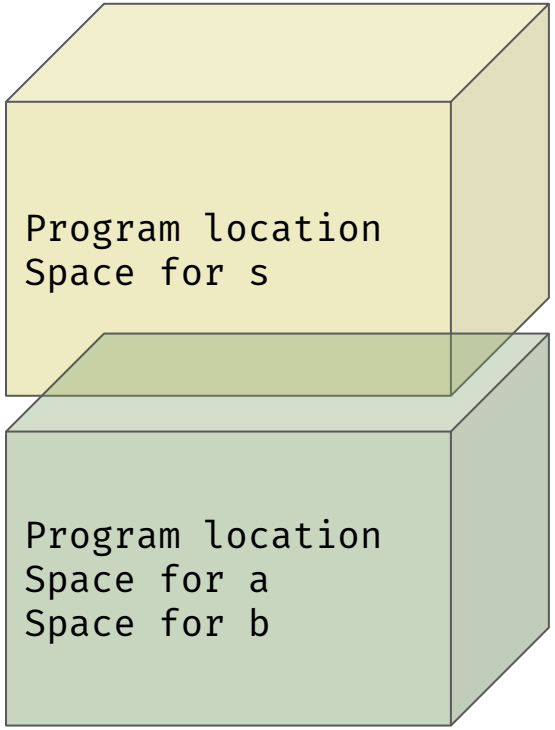
Getting framed

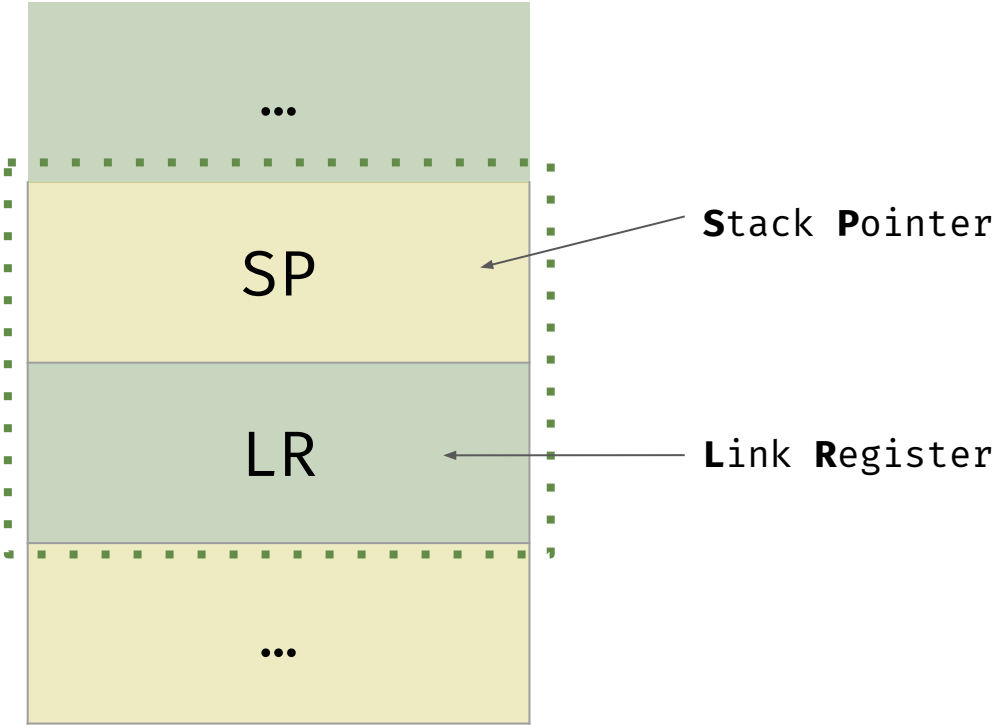


Back to here?

How do we get from here…

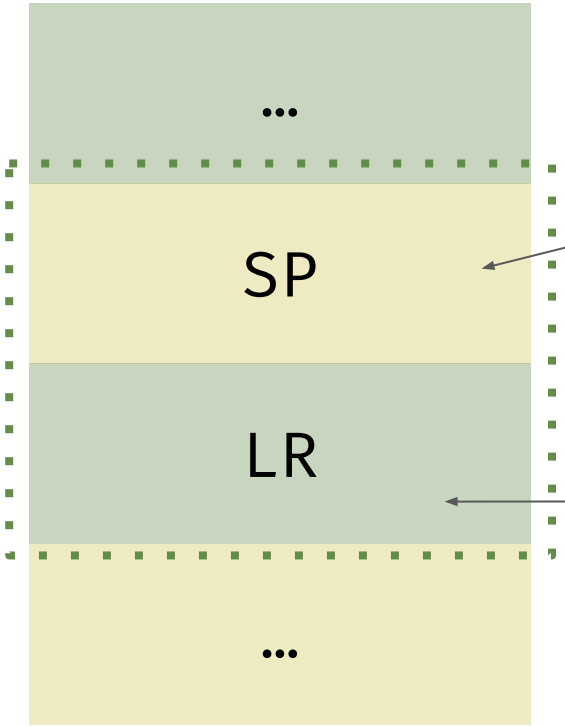
Getting framed





Getting framed

Seems like the CARDIAC had something like this…



Stack **P**ointer

Link **R**egister: stores memory location of the previous point in memory that we branched from.

Going out on a limb…

BL	**B**ranch and **L**ink	Jump to a label; store location in **LR**
BX	**B**ranch and e**X**ecute	Jump back to a memory location and continue to execute

Going out on a limb…

BL LABEL

BX R_D

Has to be a memory location stored in a register…

But can’t POP {LR}

Getting shifty

Logical **S**hift **R**ight

LSR R_D, R_X, $\#_{BITS}$

LSR R5, R5, #24

0xff0000000~~x0~~00000ff

LSL R_D, R_X, $\#_{BITS}$

LSL R5, R5, #24

0x000000ff → 0xff000000

Logical **S**hift **L**eft