

```
# Activation record for main
```

```
# 0. a
```

```
# 1. b
```

```
# 2. c
```

```
# 3. total
```

```
# r6 frame pointer
```

```
# r7 stack pointer
```

```
main: lli r6 0x00
```

```
lui r6 0x80
```

```
addi r7 r6 4 # 4 comes from what is in the activation rec
```

```
# a = 2;
```

```
lli r1 2 # a = r1
```

```
sw r1 r6 0 # using offset 0 we set a = 0
```

```
# b = 3;
```

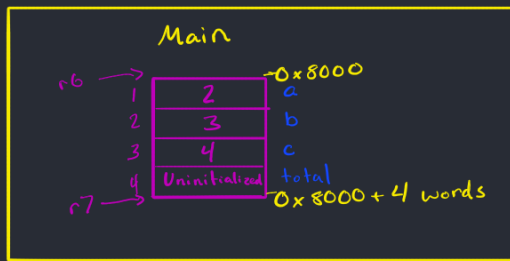
```
lli r1 3
```

```
sw r1 r6 1
```

```
# c = 4
```

```
lli r1 4
```

```
sw r1 r6 2
```



```
# call foo(a, b, c)
```

```
lw r1 r6 0 # r1: a
```

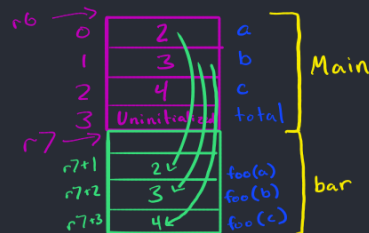
```
sw r1 r7 1 # a --> param 1
```

```
lw r1 r6 1 # r1: b
```

```
sw r1 r7 2 # b --> param 2
```

```
lw r1 r6 2 # r1: c
```

```
sw r1 r7 3 # c --> param 3
```



```
lli r1 &foo
```

```
lui r1 &foo
```

```
jalr r1 r5
```

Jump to &foo and link register r5 as return address

```
# total = < return value (r4) >
```

```
sw r4 r6 3
```

After we return from foo

```
# call bar(b, c)
```

```
lw r1 r6 1 # r1: b
```

```
sw r1 r7 1 # b --> param 1
```

```
lw r1 r6 2 # r1: c
```

```
sw r1 r7 2 # c --> param 2
```

```
lli r1 &bar
```

```
lui r1 &bar
```

```
jalr r1 r5
```

```
# total = total + < return value (r4) >
```

```
lw r1 r6 3
```

```
add r1 r1 r4
```

```
sw r1 r6 3
```

```
# cout << total
```

```
lw r1 r6 3 # Line not needed in this case
```

```
out r1
```

```
# end of main
```

```
.halt
```

Same Idea for the rest. Just overwriting past original stack address (r7)

```
# Activation record for bar
# 0. previous frame pointer
# 1. parameter a
# 2. parameter b
# 3. c
```

```
bar: sw r6 r7 0 # store previous frame ptr
addi r6 r7 0 # move fram ptr to stack ptr
addi r7 r7 4 # move stack ptr to end
```

```
# c = a + b;
lw r1 r6 1 # r1: a
lw r2 r6 2 # r2: b
add r3 r1 r2 # r3: a + b
sw r3 r6 3 # r3: c = a+b
```

```
# return c + 2;
lw r4 r6 3
addi r4 r4 2 # < return val (r4) > = c + 2
```

```
addi r7 r6 0 # move stack ptr to frame ptr
lw r6 r7 0 # restore previous frame ptr
jalr r5 r0 # return
```

```
# Activation record for foo
# 0. previous frame pointer
# 1. parameter x
# 2. parameter y
# 3. parameter z
# 4. sum1
# 5. sum2
# 6. return address
```

```
foo: sw r6 r7 0 # store previous fram ptr
addi r6 r7 0 # move fram ptr to stack ptr
addi r7 r7 7 # move stack ptr to end
sw r5 r6 6 # store return address
```

```
# Call bar(x, y)
lw r1 r6 1
sw r1 r7 1 # x --> param 1
lw r1 r6 2
sw r1 r7 2 # y --> param 2
```

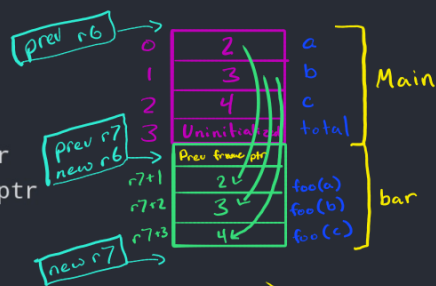
```
lli r1 &bar
lui r1 &bar
jalr r1 r5
```

```
# sum1 = < return val (r4) >
sw r4 r6 4
```

```
# Call bar(y, z)
lw r1 r6 2
sw r1 r7 1 # y --> param 1
lw r1 r6 3
sw r1 r7 2 # z --> param 2
```

```
lli r1 &bar
lui r1 &bar
jalr r1 r5
```

```
# sum 2 = <return val (r4) >
sw r4 r6 5
```



Function Operations

Return r4

Reverse stack and frame ptr before returning



```
# return sum1 + sum2;
lw r1 r6 4
lw r2 r6 5
add r4 r1 r2    # < return value (r4) > = sum1 + sum2

lw r5 r6 6      # restore return address
addi r7 r6 0     # move stack ptr to frame ptr
lw r6 r7 0       # restore previous frame ptr
jalr r5 r0       # return
```