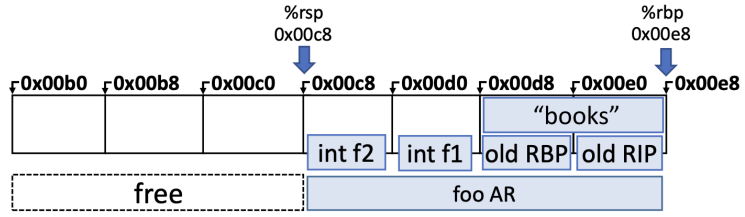
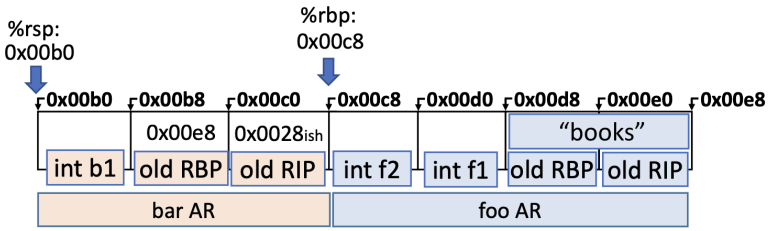
enter fn
getarg 1, [a]
[tmp1] := [c] * [d]
[global] := [a] **MULT64** 2
[tmp0] := 9 LT64 [var]
setret 42
L_fn_end: leave fn

enter v
setarg 1, 42
call fn
getret [k]
L_v_end: leave v






```
void bar(int f1, int f2, int f3, int f4, int f5, int f6, int f7, int f8){
  int b;
  b = f8;
}
void foo(){
  int loc;
  loc = 8;
  bar(1,2,3,4,5,6,7,loc);
}
```

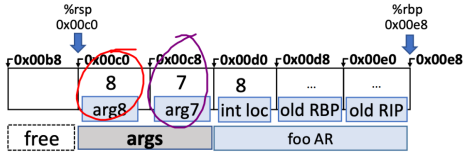
.globl _start
.data
global_var_a: .quad 7
str: .asciiz "hi"
.text
_start:
  movq (global_var_a), %r10
  movq $str, %r9
  movq $60, %rax # choose syscall exit
  movq $4, %rdi # set syscall arg - return code
  syscall

lbl_fn: pushq %rbp
movq %rsp, %rbp
addq $16, %rbp
subq $16, %rsp # **AR%16==0**
lbl_start: nop
movq -32(%rbp) %rax
movq $12, %rbx
cmpq %rbx, %rax **(opposite order)**
setlt %cl # jge LBL_after
andq 0x1, %rcx
movq %rcx, -40(%rsp)
movq -40(%rsp), %rax
cmpq $0, %rax
je LBL_after
movq -32(%rsp), %rax
addq $1, %rax
movq %rax, -32(%rsp)
jmp lbl_start
lbl_after: nop
addq $16, %rsp
popq %rbp
retq

X64 for call to bar

```
movq $1, %rdi
movq $2, %rsi
movq $3, %rdx
movq $4, %rcx
movq $5, %r8
movq  $6, %r9
pushq  $7
movq -24(%rbp), %r12
pushq %r12
callq bar
addq $16, %rsp
```

call

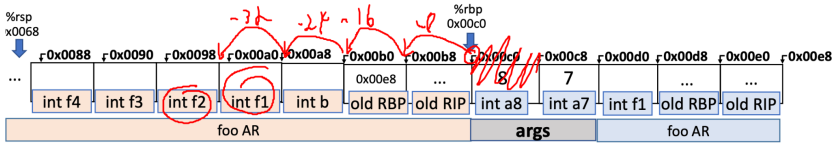


Args 1 – 6

- Were passed in register
- Should be allocated saved/in current AR

```
getarg 1, [f1]      movq %rdi, -32(%rbp)
getarg 2, [f2]      movq %rsi, -40(%rbp)
getarg 3, [f3]      movq %rdx, -48(%rbp)
getarg 4, [f4]      movq %r08, -56(%rbp)
getarg 5, [f5]      movq %r09, -64(%rbp)
getarg 6, [f6]
```

(keeps them from getting clobbered if the callee calls something else)

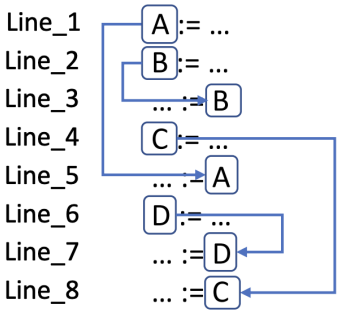


leaq -8(%rbp), %rax # like movq, but address, not value
movq $2, %rax
movq $4, %r11
imulq %r11 # %rdx:%rax = %rax * o1 # 8
idivq o # %rax = %rdx:%rax / o1 # %rdx = %rdx:%rax % o1
.text .data heap free stack. Function arguments: %rdi rsi rdx rcx r08 r09

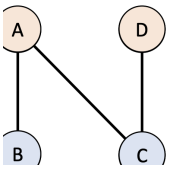
movsx %bl, %rbx

Runtime: platform on which code depends. Platform: soft&hard guarantees
cc -> .s -> as -> .o -> ld -> .exe -> loader
Reference counting (cycles). Mark&sweep (freeze)

Use/Definition Sequence



Interference Graph



Preserved (callee-saved): rbx, rsp, rbp, r12, r13, r14, r15
Volatile (caller-saved): rax, rdi, rsi, rdx, rcx, r8, r9, r10, r11
Peephole: store/load, add;add, jump to next
Constant(copy) propagation (const/not/? down): inline const
Constant folding: 1+2
Flowgraph: Leader (first line, label, after T) /Terminator (last, jump, callq)
Dead code elimination (dataflow fact sets: live/dead/? up): live variable analysis
Static **S**ingle **A**ssignment: $a_3 = \phi(a_1, a_2)$

X dominates Y if all paths to Y must pass though X
Dominance frontier: all the things it dominates
Sound/Complete

$$IN(b) = \bigcup_{p}^{pred(b)} OUT(p) \quad \text{(loops->saturation)}$$

$$OUT(b) = GEN(b) \cup (IN(b) - KILL(b))$$