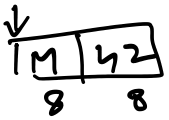



```
(vector-ref (vector-ref (vector (vector 42)) 0) 0)
```

```
(vector-ref
  (vector-ref
    (let ([vecinit7976
      (let ([vecinit7972 42])
        (let ([collectret7974
          (if (< (+ (global-value free_ptr) 16)
            (global-value fromspace_end))
            (void)
            (collect 16)
           )])
        (let ([alloc7971 (allocate 1 (Vector Integer))])
          (let ([initret7973 (vector-set! alloc7971 0 vecinit7972)])
          alloc7971)))])
    (let ([collectret7978
      (if (< (+ (global-value free_ptr) 16)
          (global-value fromspace_end))
        (void)
        (collect 16)
       )])
      (let ([alloc7975 (allocate 1 (Vector (Vector Integer)))])
        (let ([initret7977 (vector-set! alloc7975 0 vecinit7976)])
        alloc7975))))
  0)
0)
```

expose allocation

(define v (Int ..))

1) enough space? yes

2) no → GC

steps

1) check if enough space
   1) allocate & return the addr
2) call GC.
   1)

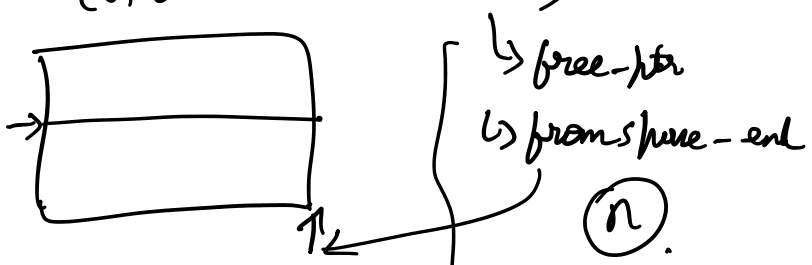
(collect int) → GC

(collect 16) → 16 bytes

(Allocate int type)

(Allocate 8 ~~boot~~)

(Global Value name)



if (free_ptr + n) >= end

(define v ( ))

n

collect
allocate
global value

(let ( )

(+ 4 (+ 2 1))

~~(+ 3)~~

(+ 4 3)

Int

Exp_control

( )

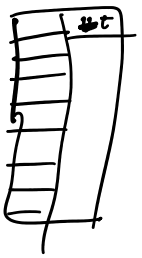
```
block35:
  movq free_ptr(%rip), alloc9024
  addq $16, free_ptr(%rip)
  movq alloc9024, %r11
  movq $131, 0(%r11)
  movq alloc9024, %r11
  movq vecinit9025, 8(%r11)
  movq $0, initret9026
  movq alloc9024, %r11
  movq 8(%r11), tmp9034
  movq tmp9034, %r11
  movq 8(%r11), %rax
  jmp conclusion
block36:
  movq $0, collectret9027
  jmp block35
block38:
  movq free_ptr(%rip), alloc9020
  addq $16, free_ptr(%rip)
  movq alloc9020, %r11
  movq $3, 0(%r11)
  movq alloc9020, %r11
  movq vecinit9021, 8(%r11)
  movq $0, initret9022
  movq alloc9020, vecinit9025
  movq free_ptr(%rip), tmp9031
  movq tmp9031, tmp9032
  addq $16, tmp9032
  movq fromspace_end(%rip), tmp9033
  cmpq tmp9033, tmp9032
  jl block36
  jmp block37
block37:
  movq %r15, %rdi
  movq $16, %rsi
  callq 'collect
  jmp block35
block39:
  movq $0, collectret9023
  jmp block38

start:
  movq $42, vecinit9021
  movq free_ptr(%rip), tmp9028
  movq tmp9028, tmp9029
  addq $16, tmp9029
  movq fromspace_end(%rip), tmp9030
  cmpq tmp9030, tmp9029
  jl block39
  jmp block40
block40:
  movq %r15, %rdi
  movq $16, %rsi
  callq 'collect
  jmp block38
```

uncover-locals

local:

env, info



select inst

lhs = (vector-set! v n arg)

movq vec, (%r11)

movq arg, 8(n+1)(%r11)

movq arg, 8(n+1)(%r11)

movq $0, lhs

movq arg, 8(n+1)(%r11)

← rax ?

movq arg, 8(n+1)(%rax)

-16(%rbp)

movq -16(%rbp) 8(n+1)(%r11)

↓ PI

movq -16(%rbp) (%rax)

movq (%rax) 8(n+1)(%r11)

lhs = (vector-ref v n)

SI

movq v, %r11

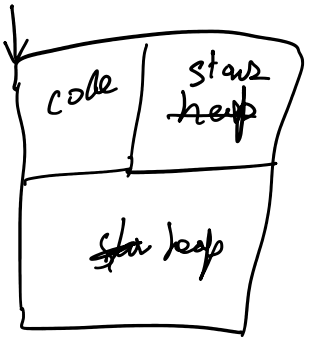
movq 8(n+1)(%r11) lhs

allocate → return address

1) current free_ptr, return value

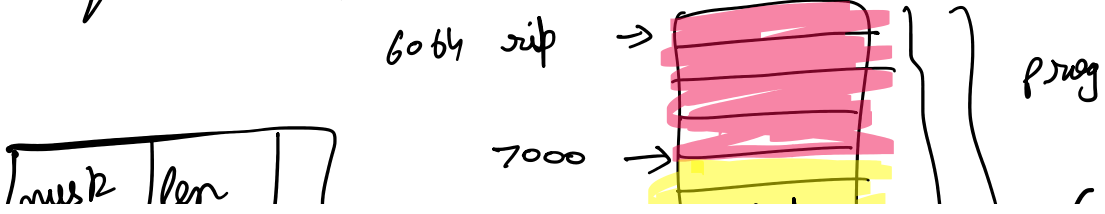
2) update free_ptr, (len+1)

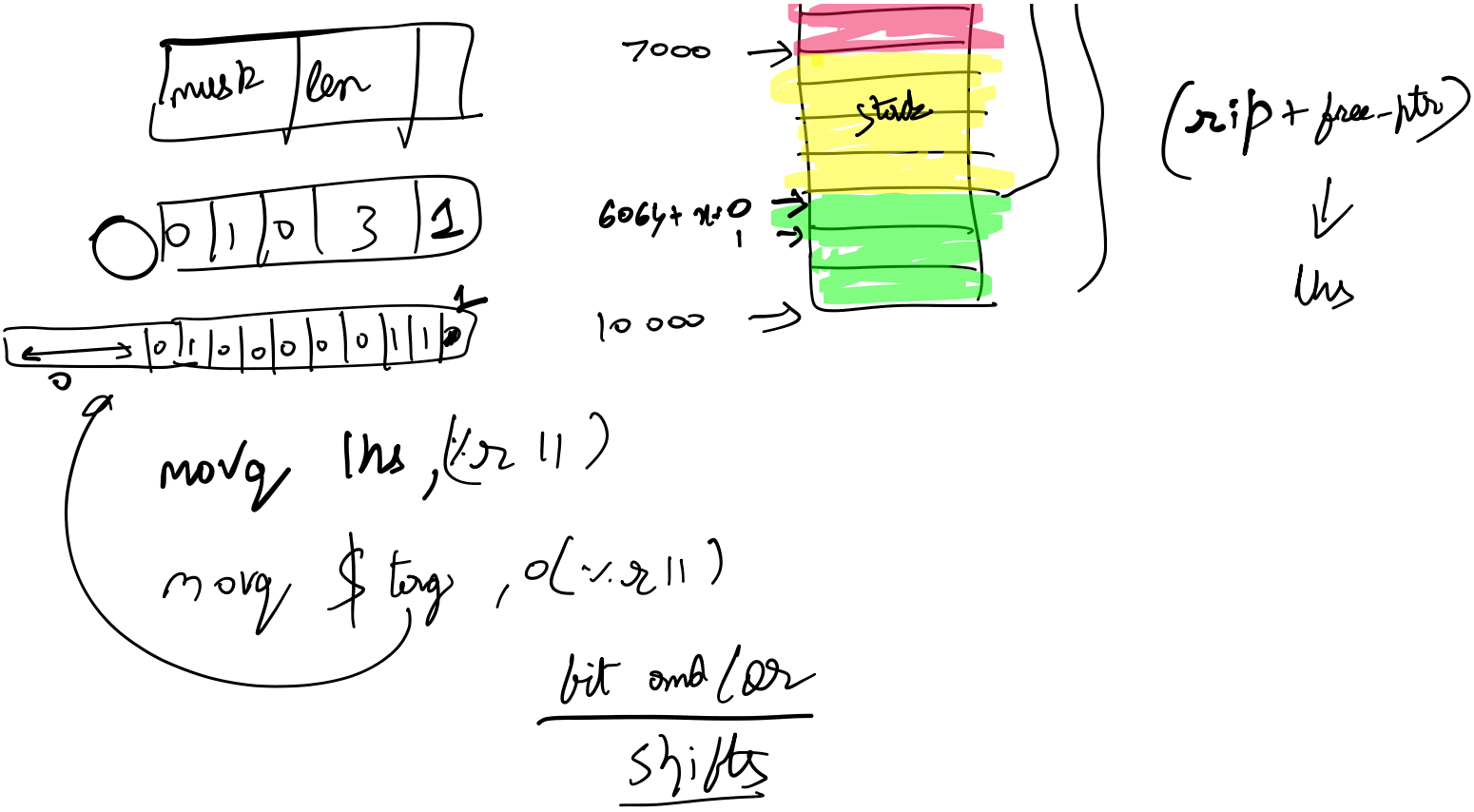
3) update / hopulate meta data



→ movq free_ptr(%rip), lhs

addq 8(len+1), free_ptr(%rip) S2





bit and/or

shifts