2-1/

(vector 1 #t (unit))

⇒ → the pointer to vector

```
movq   free_ptr(%rip), %rdx
addq   $32, free_ptr(%rip)
leaq   type3(%rip), %rax      ] installs ptr to type
movq   %rax, %rdx(0)
movq   $1, %rdx(8)
movq   $1, %rdx(16)
movq   $1, %rdx(24)
```

type3:

```
.quad 3 - Vector
.quad 3 - length
.quad 1 - 564
.quad 2 - Bool
.quad 0 - Unit
```

select (global str) = str(%rip)

(unit) = $1

(allocate num) =
```
movq   fglobal free_ptr) dst;
addq   $(8 * 1+num), free_ptr;
movq   dst, %rax;
leaq   type, %r11;
movq   %r11, %rax(0)
```

tmp 1 = rax

tmp 2 = r11

(vector-ref vecptr num) =
```
movq   (sel vecptr), %rax;
movq   %rax(8*(1+num)), dst
```

(collect num) =
```
movq   ROOT-STACK-REG, %rdi (1st arg)
movq   $num, %rsi (2nd argument)
callq  _collect
```

(vector-set! vp amt nv) =
```
movq   (sel vp), %rax;
movq   (sel nv), %r11;
movq   %r11, %rax(8*(1+num))
```

12-2/ live
conflicts
assign
patch
main

live/ leaq s,d
=> handled like movq

conflicts/ leaq like movq
update callq

assign/
old: vars either go to a register OR the stack = %rsp (-8 x (var#))

old: $\forall v \in L_k$. $\forall r \in$ CALLER-SAVED (add (r,v) to G)

new: $\forall v \in L_k$. where $\Gamma(v) =$ Vector(...) $\forall r \in$ CALLEE-SAVED (add (r,v) to G)

now: if it's not a reg...
is it a vector? ($\Gamma(v) =$ Vector(...))
yes => ROOT-STACK-REG (-8 x (rs-var#)) -> r15
no => STACK-REG (-8 x (var#)) -> rsp

main/ BEGIN:
old: pushed all the used callee-saved-regs
movq %rsp, %rbp
subq $local-count x 8 (maybe +8 if odd), %rsp (allocs space)
jmp BODY

new: subq $object-local-count x 8, % ROOT-STACK-REG
callq _initialize;
movq _rootstack_start, % ROOT-STACK-REG

END:
old: movq %rax, %rdi
callq _print_int or bool

new: leaq ans-type, %rdi
movq %rax, %rsi;
callq _print_value

12-3

runtime.c

import

```
extern int64_t  ty_unit;
                ty_bool;
                ty_s64;
                ty_vector;
```

export

```
int64_t read_int();
int64_t print_value (
        int64_t * ty,
        int64_t  val );
```

```
int print_value ( * ty , va) {
  if ( ty[0] == ty_unit) {
        printf("unit"); }
  else if ( ty[0] == ty_bool ) {
        printf ("%c", val ? 't' : 'f'); }
  else . . .
  else if (ty[0] == ty_vector) {
        . . .
    for (int i=0 ; i < ty[1] ; i++) {
      print_value ( ty[2+i] , val[1+i]);
    . . .
```

□ char debug_p;
int64_t * free_ptr;
int64_t * from_space_end;
int64_t * root_stack_ptr;

void initialize();
void collect (
int64_t** curr_root_stack_ptr,
int64_t alloc_request);

```
initialize() {
    fromspace_begin = malloc ( heap_size);
    fromspace_end = fromspace_begin + heap_size;
    free_ptr = fromspace_begin;
    root_stack_begin = malloc
    root_stack_ptr = root_stack_end;
```

mmap

mprotect

vs malloc