CMSC 430: Introduction to Compilers

Loot: lambda

Loot

- Loot adds lambda expressions.

```
(λ (x0 ...) e0)
```

- Like

```
(define (f0 x00 ...) e0)
```

  Lambda expressions create procedures, but
  - There is no function name in the λ-expression; it is an **anonymous** function.
  - It can appear anywhere as a subexpression in a program, whereas definitions were restricted to be at the top-level.

Implement lambda

- ast.rkt
- parse.rkt
- Interp.rkt

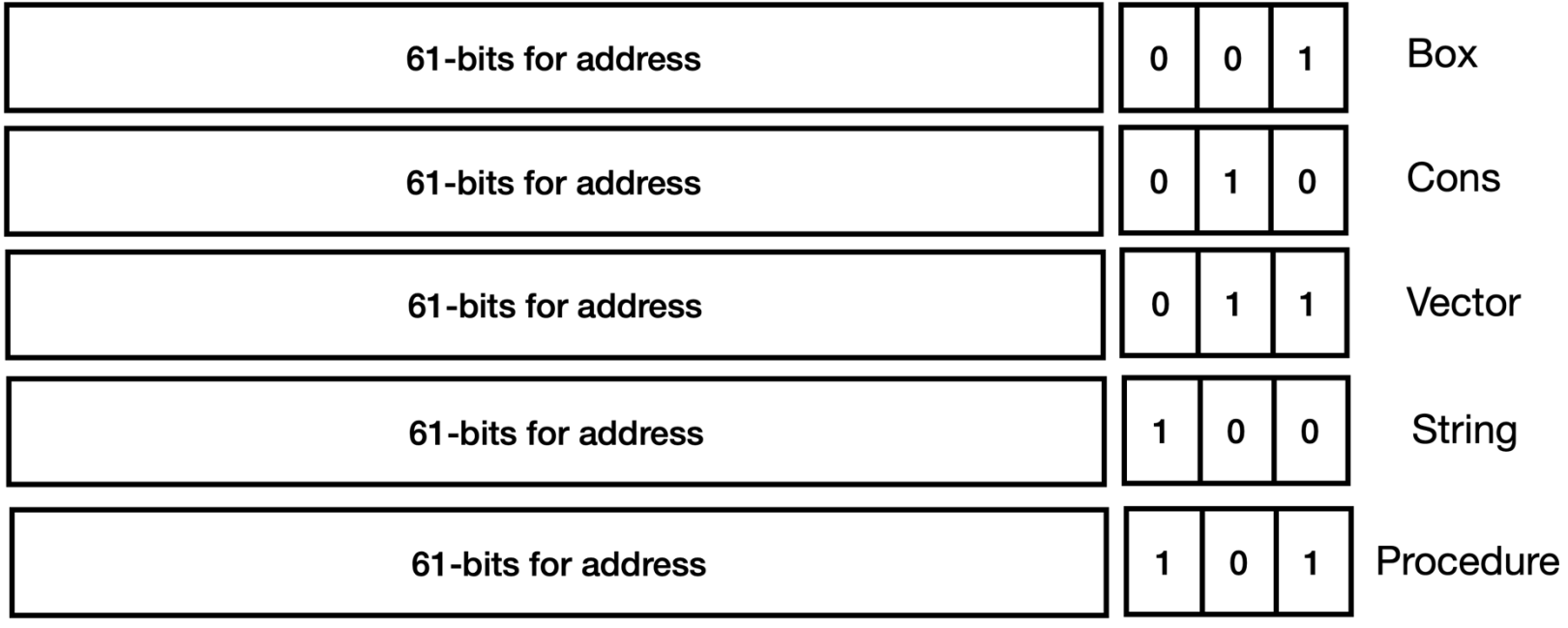
How to interp a lambda?

```
[(App e es)
  (let ((f (interp-e e r ds))
    (vs (interp-e* es r ds)))
    (if (procedure? f)
        (apply f vs)
        (raise 'err)))]

[(Lam f xs e)
  (λ vs
   ; check arity matches
  (if (= (length xs) (length vs))
   (interp-e e (append (zip xs vs) r) ds)
  (raise 'err)))]
```

Defunctionalization

- **Convert any program with lambdas into one without**
  - What does λ do for you?
    - packs up the parameters, body, and environment at point that λ-expression is evaluated
    - unpacks, binding arguments to parameters in environment and running the body when the function is applied
- We could do this with a data structure instead
  - Closure

Loot: Encoding procedure pointer



Compiling Lambdas

```
((let ((z 10)) (λ (x y) z)) 1 2)
```

text

1. expression
2. Lambda definitions
3. Top level definitions

…

heap

Closures
1. Pointer to lambda definition
2. Env: all the free variables

stack

Arguments
Free variables (copy from closure)

```
(compile-defines ds)
(compile-lambda-defines
   (lambdas p))
```

```
(compile-e e …
```

```
apply
```

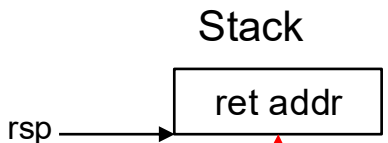
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Xor 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)

(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```

```
(Label 'label99)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 (Mem '(+ rax 3)))
(Push 'r9)
(Mov 'rax (Mem 'rsp 0))
(Add 'rsp 32)
(Ret)
```

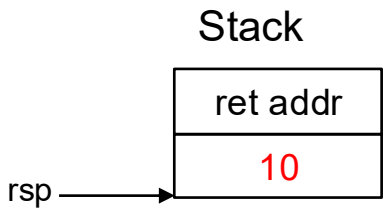
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)

(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



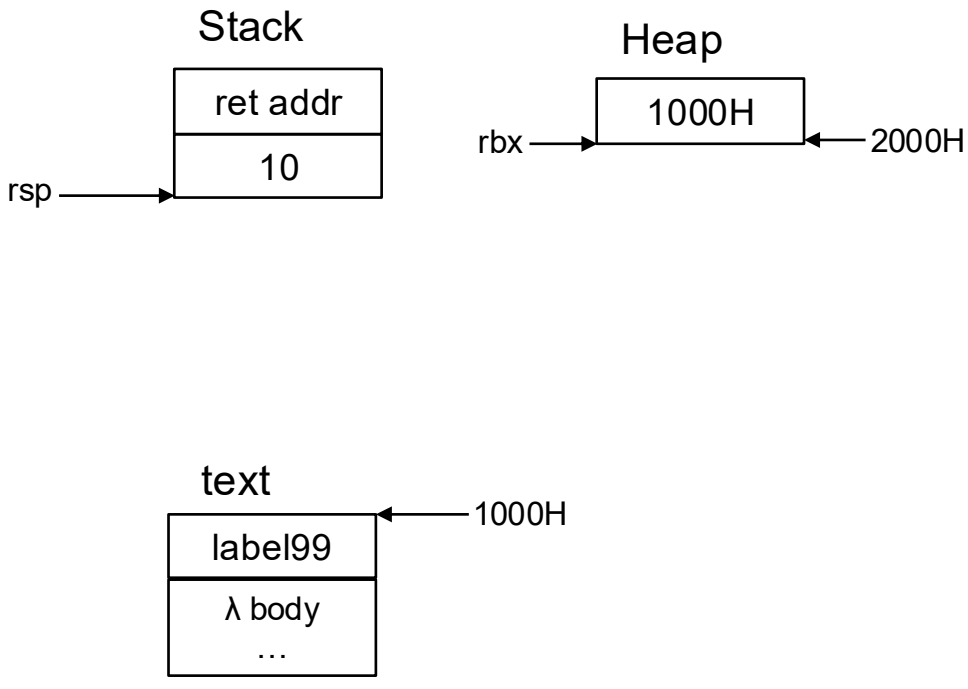
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)

(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



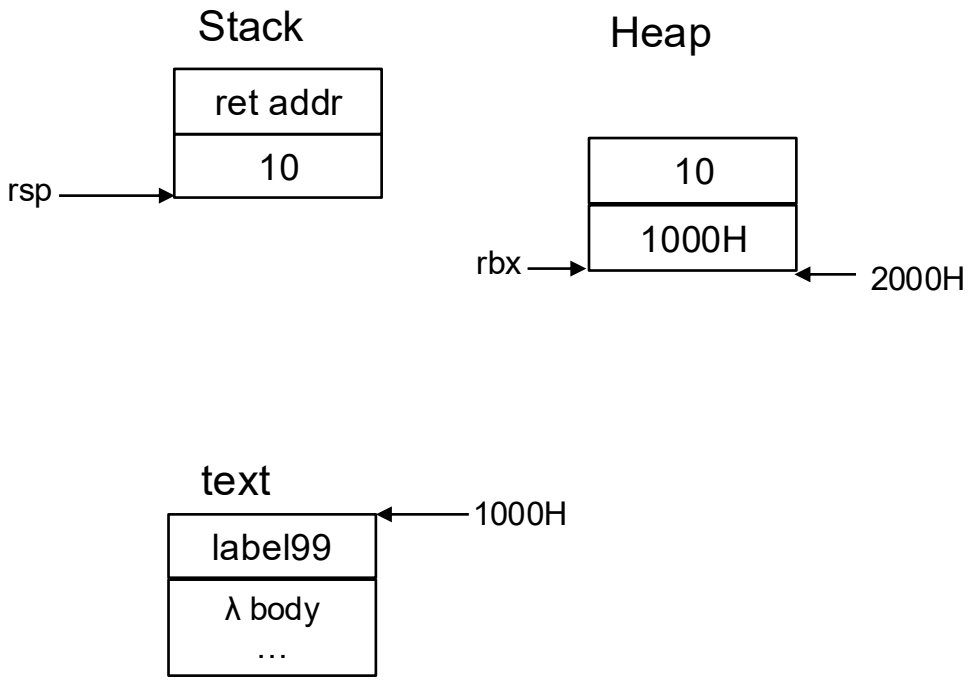
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)

(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



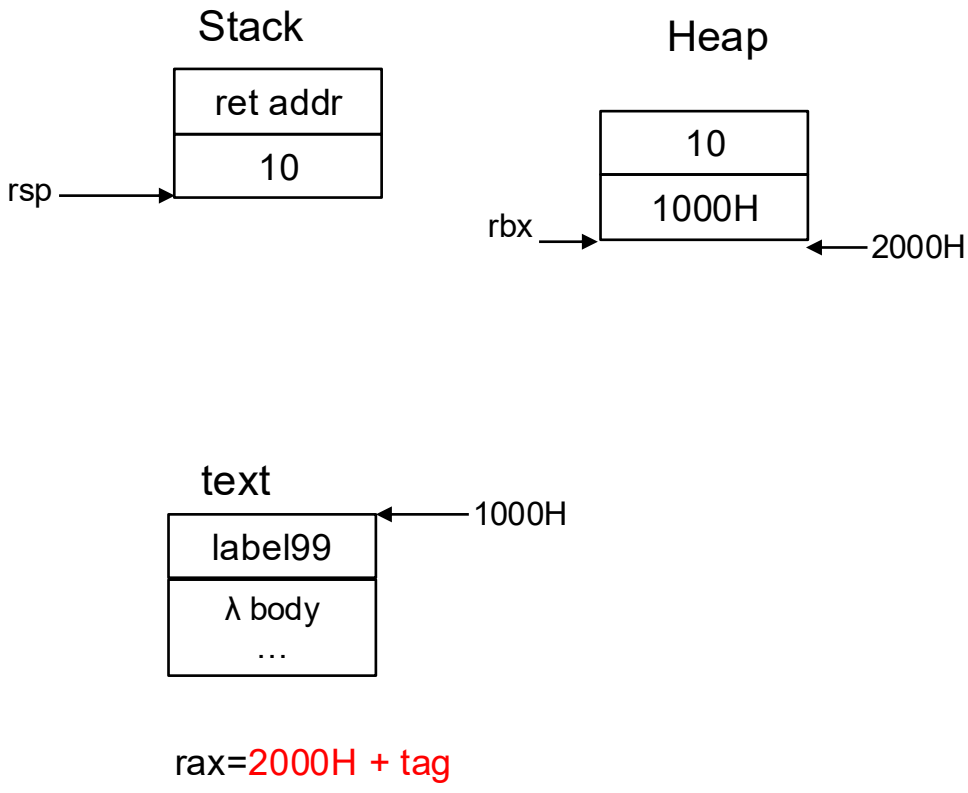
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax 5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



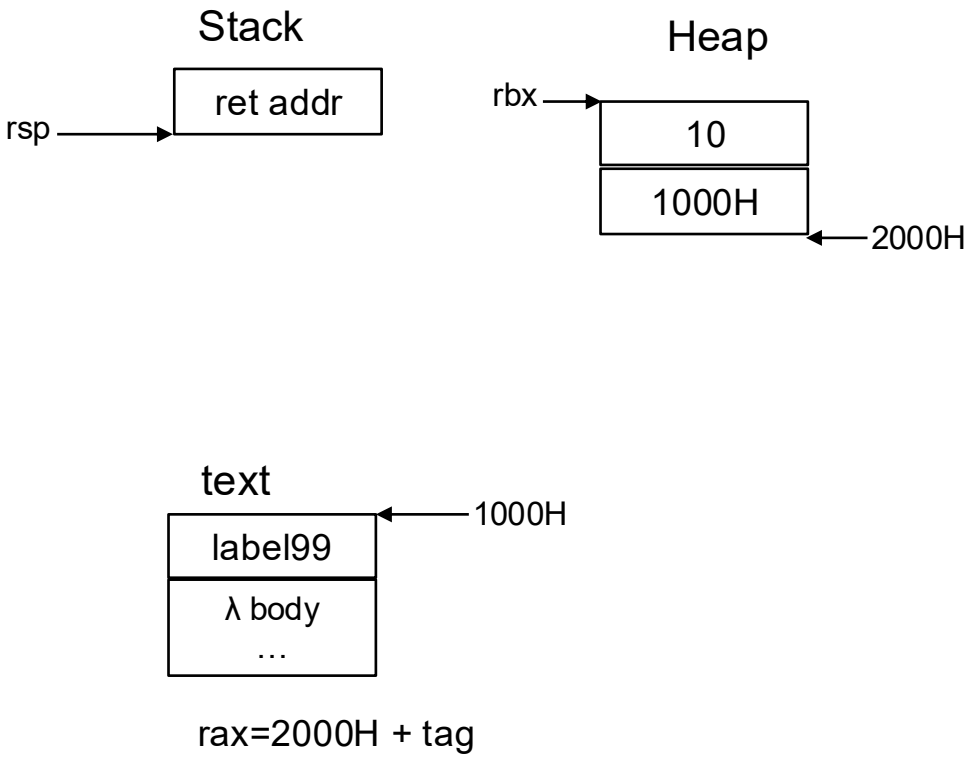
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



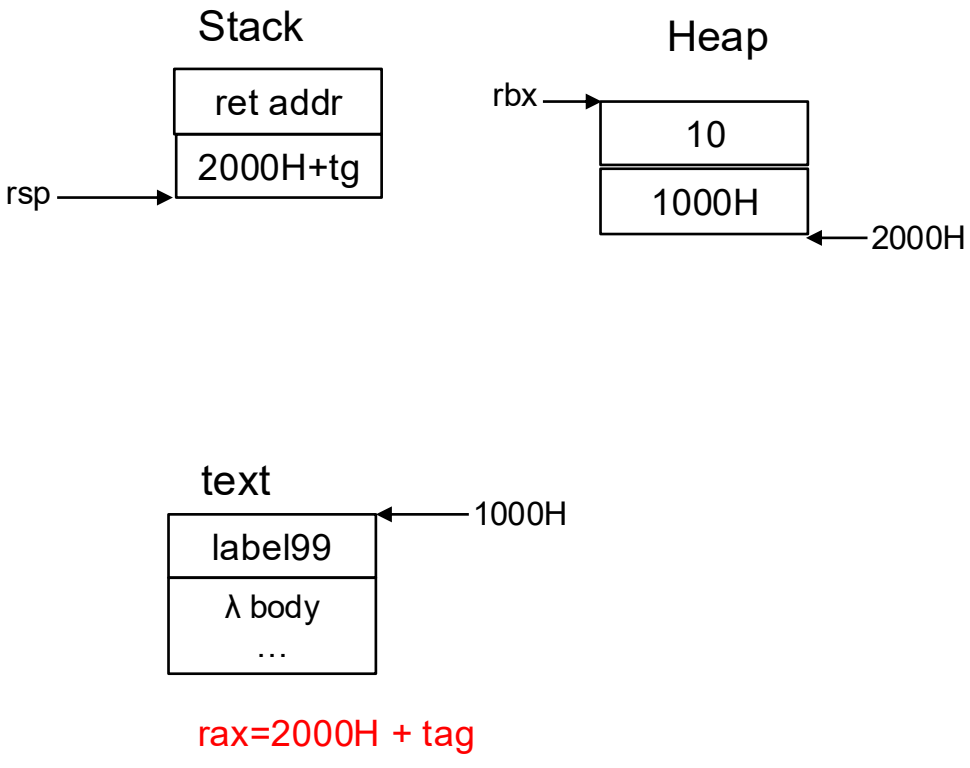
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



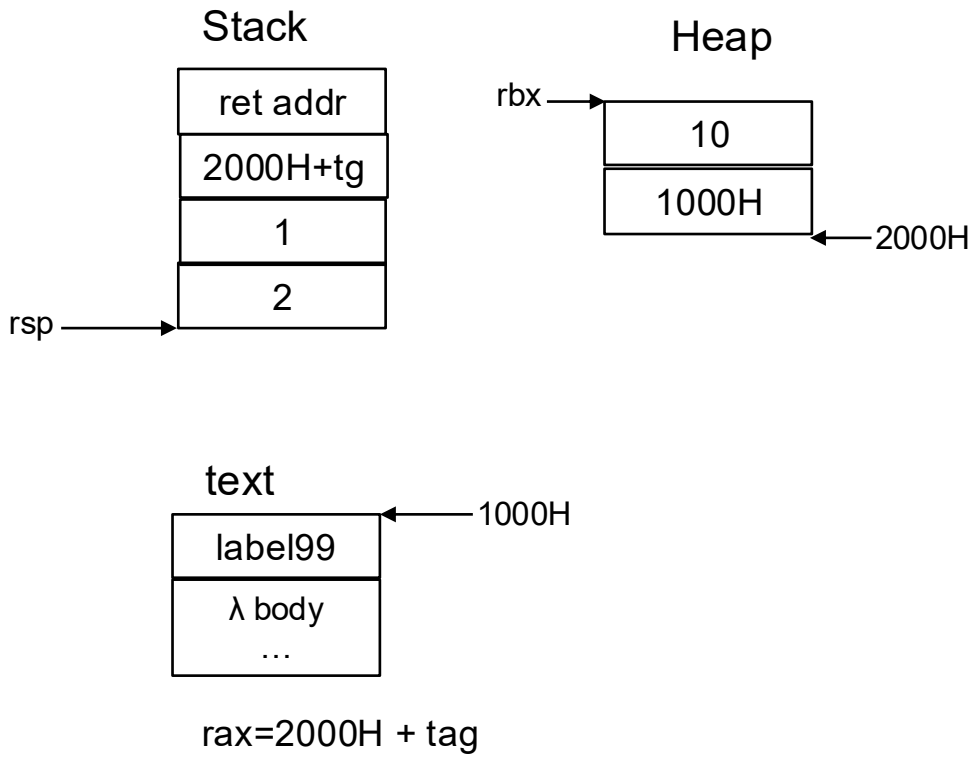
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



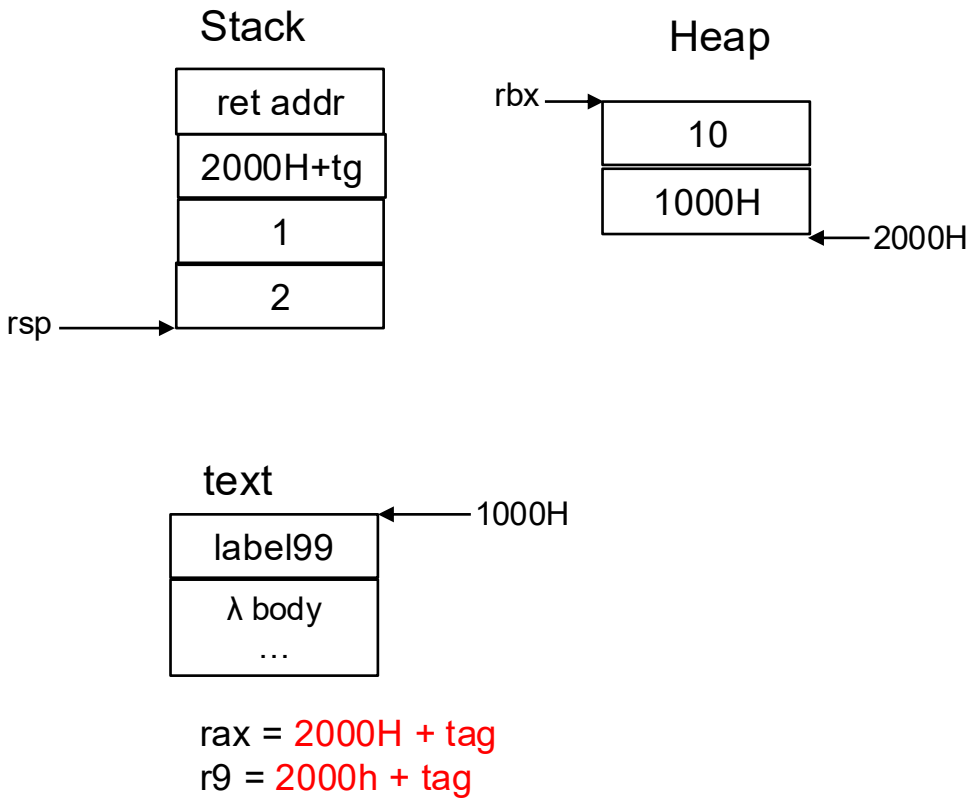
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



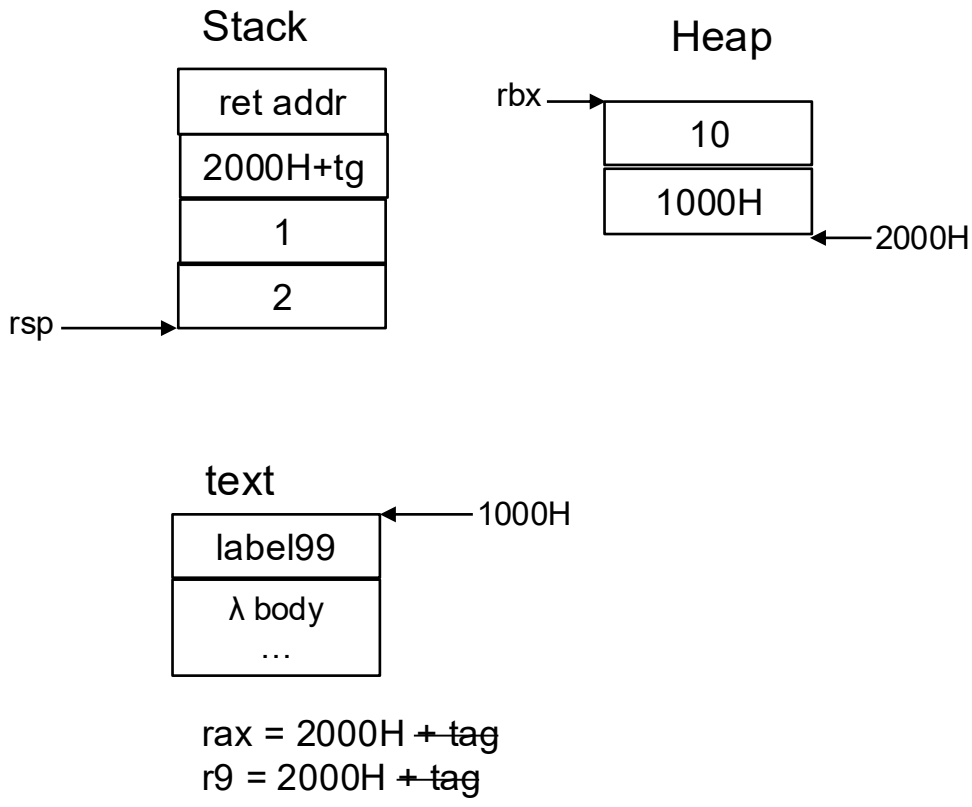
((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



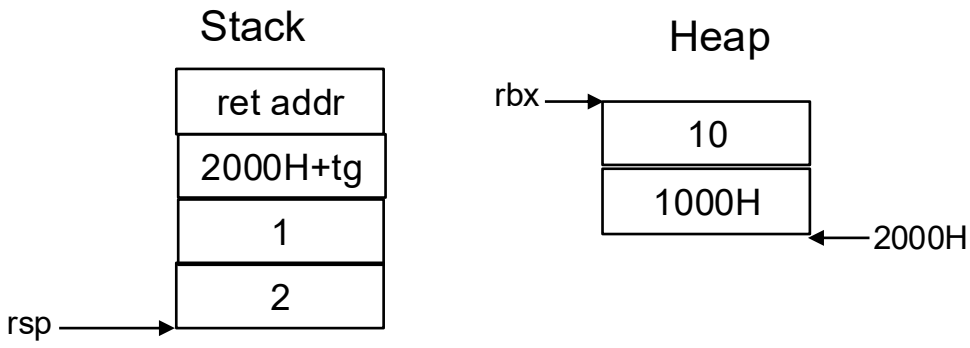
((let ((z 10)) (λ (x y) z)) 1 2))

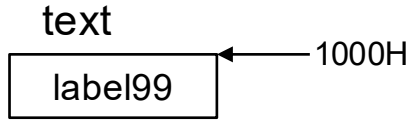
```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)

(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



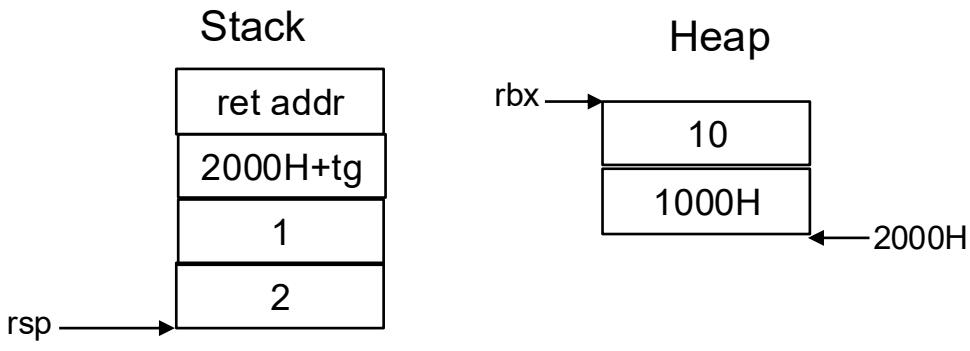


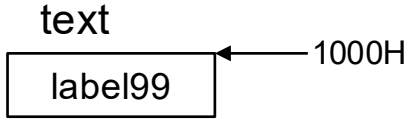
rax = 1000H

```
(Label 'label99)
 (Mov 'rax (Mem 'rsp 16))
 (Mov 'r9 (Mem rax 3))
 (Push 'r9)
 (Mov 'rax (Mem 'rsp 0))
 (Add 'rsp 32)
 (Ret)
```

((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



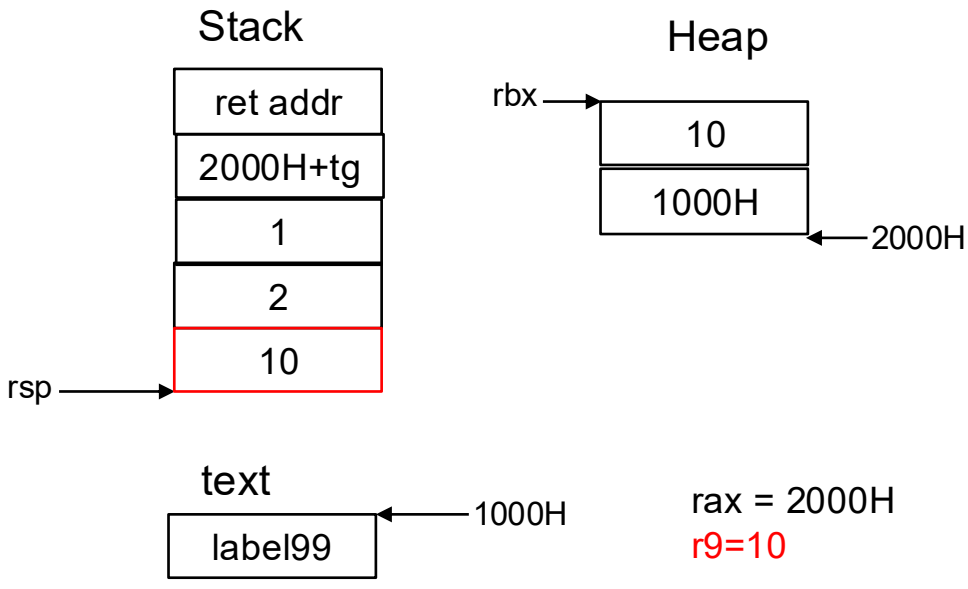


rax = 2000H

```
(Label 'label99)
 (Mov 'rax (Mem 'rsp 16))
 (Mov 'r9 (Mem rax 3))
 (Push 'r9)
 (Mov 'rax (Mem 'rsp 0))
 (Add 'rsp 32)
 (Ret)
```

((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```

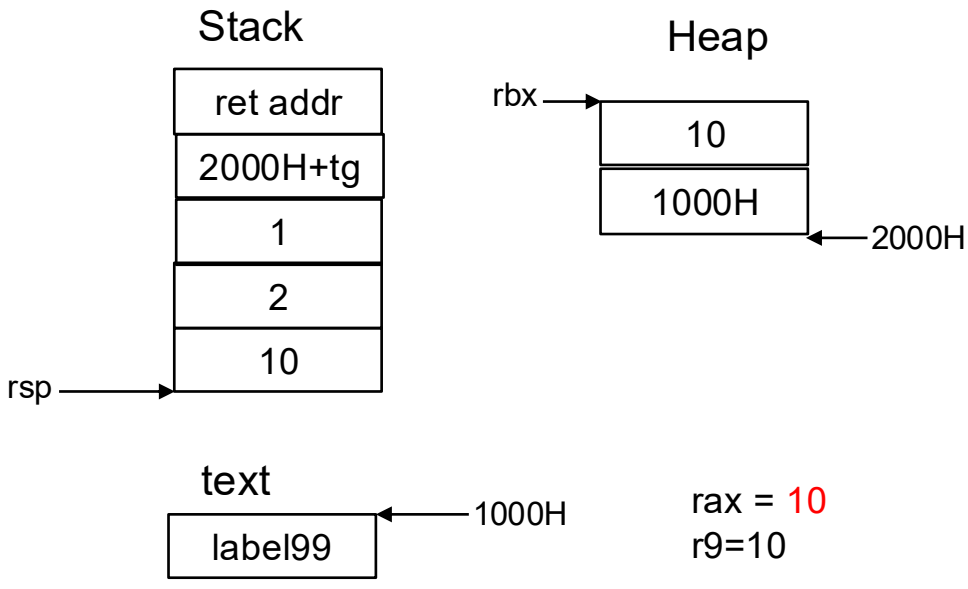


rax = 2000H
r9=10

```
(Label 'label99)
 (Mov 'rax (Mem 'rsp 16))
 (Mov 'r9 (Mem rax 3))
 (Push 'r9)
 (Mov 'rax (Mem 'rsp 0))
 (Add 'rsp 32)
 (Ret)
```

((let ((z 10)) (λ (x y) z)) 1 2))

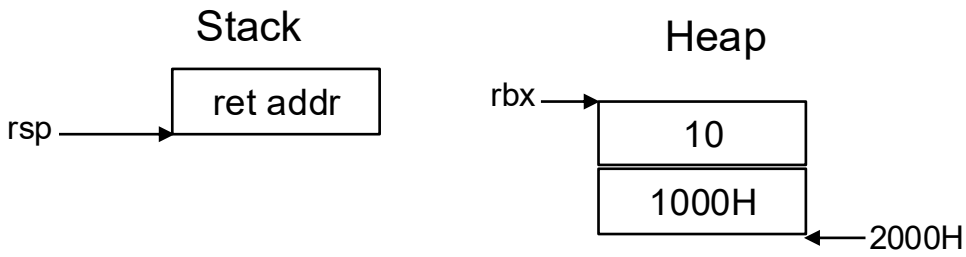
```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```

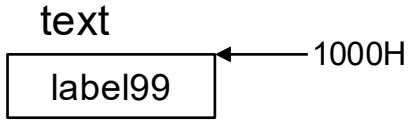



```
(Label 'label99)
 (Mov 'rax (Mem 'rsp 16))
 (Mov 'r9 (Mem rax 3))
 (Push 'r9)
 (Mov 'rax (Mem 'rsp 0))
 (Add 'rsp 32)
 (Ret)
```

((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)
(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



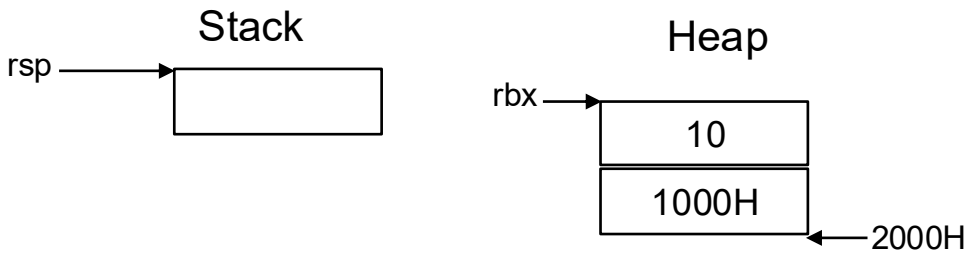


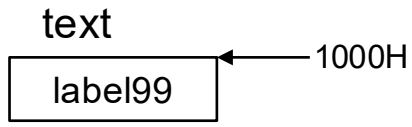
rax = 10
r9=10

```
(Label 'label99)
 (Mov 'rax (Mem 'rsp 16))
 (Mov 'r9 (Mem rax 3))
 (Push 'r9)
 (Mov 'rax (Mem 'rsp 0))
 (Add 'rsp 32)
 (Ret)
```

((let ((z 10)) (λ (x y) z)) 1 2))

```
(Lea 'rax 'ret236579)
(Push 'rax)
(Mov 'rax 160)
(Push 'rax)
(Lea 'rax 'label99)
(Mov (Mem 'rbx 0) 'rax)
(Mov 'r8 (Mem 'rsp 0))
(Mov (Mem 'rbx 8) 'r8)
(Mov 'rax 'rbx)
(Or 'rax 5)
(Add 'rbx 16)
(Add 'rsp 8)
(Push 'rax)
(Mov 'rax 16)
(Push 'rax)
(Mov 'rax 32)
(Push 'rax)
(Mov 'rax (Mem 'rsp 16))
(Mov 'r9 'rax)
(And 'r9 7)
(Cmp 'r9 5)
(Jne 'err)

(Mov 'rax (Mem 'rax -5))
(Jmp 'rax)
(Label 'ret236579)
(Ret)
```



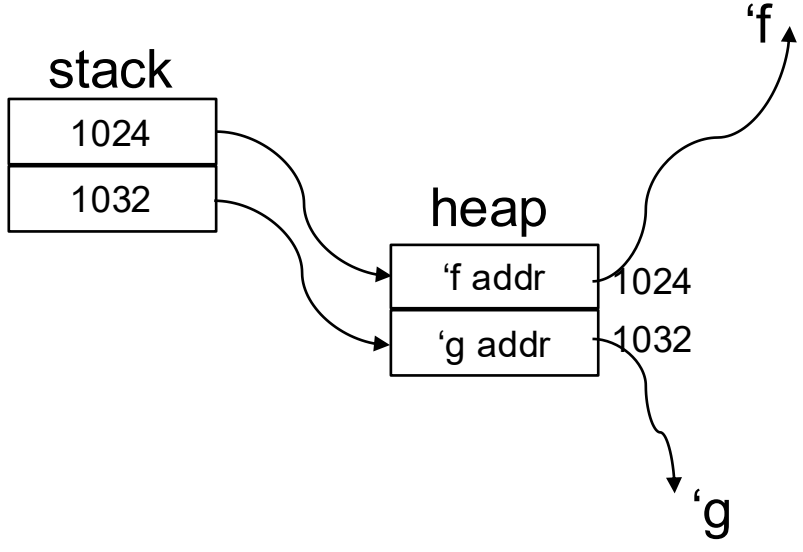


rax = 10
r9=10

```
(Label 'label99)
 (Mov 'rax (Mem 'rsp 16))
 (Mov 'r9 (Mem rax 3))
 (Push 'r9)
 (Mov 'rax (Mem 'rsp 0))
 (Add 'rsp 32)

(Ret)
```

alloc-defines

```
'(define (f x) 10)
'(define (g x y) x)

(alloc-defines ds 0)
```




```
(Lea 'rax 'f)
 (Mov (Mem rbx 0) 'rax)
 (Mov 'rax 'rbx)
 (Add 'rax 0)
 (Xor 'rax 5)
 (Push 'rax)
 (Lea 'rax 'g)
 (Mov (Mem rbx 8) 'rax)
 (Mov 'rax 'rbx)
 (Add 'rax 8)
 (Xor 'rax 5)
 (Push 'rax))
```

define-ids

```
'(define (f x) 10)
'(define (g x y) x)

(define-ids ds)

'(f g)
```

init-defines

```
(define (f x) (+ z y))
'(define (g x y) (+ a b))
 (init-defines ds '(z a y b c) 8)])
```

```
(Mov 'r8 (Mem rsp 0)))
(Mov (Mem rbx 8) 'r8)
(Mov 'r8 (Mem rsp 16))
(Mov (Mem rbx 16) 'r8)
(Mov 'r8 (Mem rsp 8))
(Mov (Mem rbx 32) 'r8)
(Mov 'r8 (Mem rsp 24))
(Mov (Mem rbx 40) 'r8))
```