Bird: Function Naming

```
def rax(n):

end
rax(5)
```

```
fn_rax:
```

driver.c

```
    int64t bird_main()     asm("bird_main");
```

NotEagle

```
<expr> ::= ....
         | (<expr>, <expr>)
         | fst <expr>
         | snd <expr>
```

Bird

4

true

(1,2)

Machine

0x8

0xFFF....FF

0xNNN···NN[nn01]

↪ 0xNNN····NN[nn00] pointer to heap

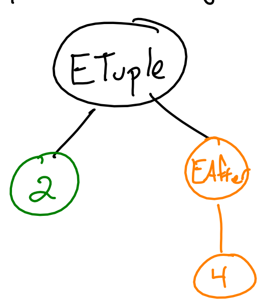
- Start of NotEagle program, use malloc to allocate a slab of memory
- Pass ptr to bird_main
- bird_main will use ptr as heap
- Use a "global variable" heap_cursor to track next free byte of heap memory

---

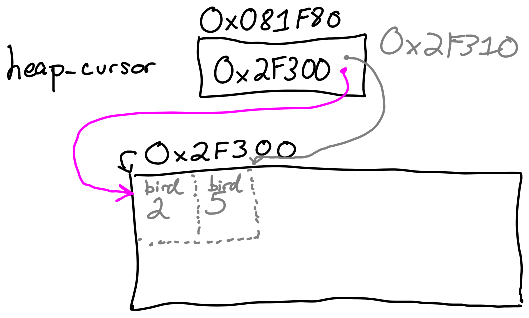
int64_t bird_main(int64_t* heap)

NotEagle Program:

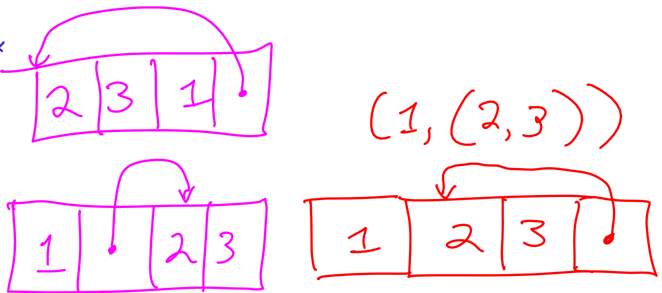
(2, after(4))



```
section .data
  align 8
  heap_cursor:
  dq 0
section .text
  extern ....
  extern ....
  bird_main:
    push rbp
    mov rbp, rsp               } stack setup
    sub rsp, #
    mov [heap_cursor], rdi     } Bird setup
    mov rax, 4                 } compute & store left
    mov [rbp-8], rax
    mov rax, 8
    add rax, 2                 } compute & store right
    mov [rbp-16], rax
    mov rax, [heap_cursor]
    mov r10, rax
    add r10, 16   ; r10 = 0x2F310   } advance heap-cursor
    mov [heap_cursor], r10
    mov r11, [rbp-8]           } Copy 1st value into heap
    mov [rax], r11
    mov r11, [rbp-16]          } copy 2nd
    mov [rax+8], r11
    or rax, 1                  } set up rax
```



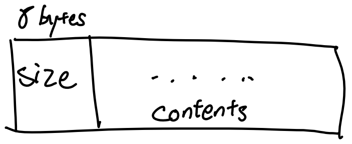
(1, (2, 3))



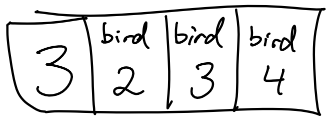
Eagle

```
<expr> ::= ....
        | (<expr>, ....)
        | <expr>[<expr>]
```

Heap Representation



(2, 3, 4)


3
bird
2
bird
3
bird
4