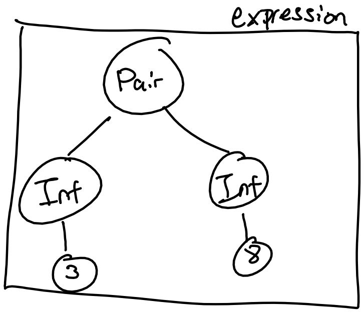
Pairs

Syntax :

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

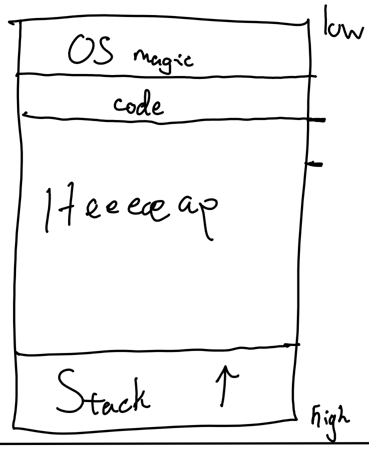
Semantics : (3,8)



representation

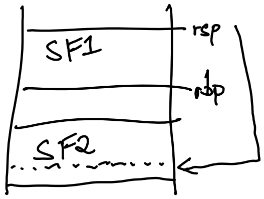
pointer to heap

NOT TO SCALE



true

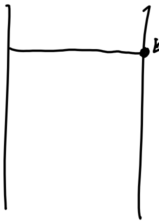
0xFFFF .... FFFF



Eagle Binary Representation

Machine	Bird
0x ZZZ .... ZZ[zzz0]	0xZZZ ... ZZZ[zzz]
0xFFFF .... FFFF	true
0x7FF .... FFF	false
0x ZZZ .... ZZ[z001]	ptr to 0xZZZ .... ZZ[z000]

Heap Mem



0xFE8 } point to

0x00 ... 0FE9
||
0x00 ... 0FE[1001] } store this

means

ptr 0x00 ... 0FE[1000]
||
ptr 0x00 ... 0FE8

Managing Heap Memory

Initialization: call malloc, ask for 1,000,000 words

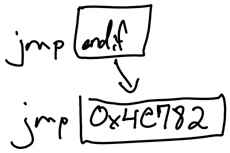
pass ptr to bird_main

in bird_main, store ptr as heap

```
section .text
bird_main:
    ⋮
section .data
    heap_cursor: dq 0
```

Not Doing

1. Garbage collection
2. User does not ask for or free memory
3. Not expanding heap

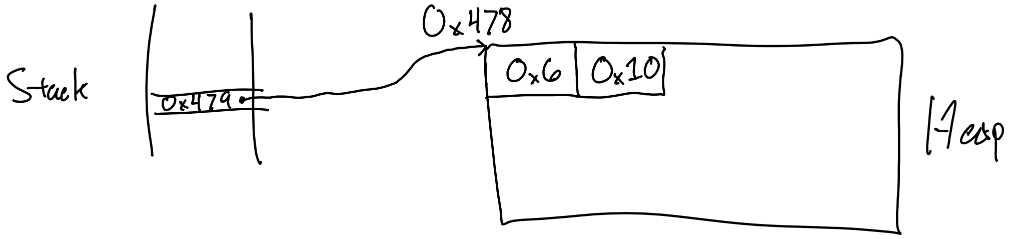


```
mov r11, 0x800
mov [heap_cursor], r11

mov [heap_cursor], rdi
```

heap cursor always points to the next free byte of heap memory

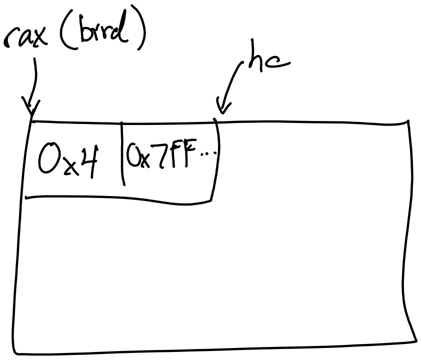
value=(3,8)



Emu Heap Layout (pairs only)

How to compile (1+1, false)?

1. Evaluate subexpressions (store in temp vars)
2. Store temp vars in heap
```
mov r10, [heap_cursor]
mov [r10], temp1
mov [r10+8], temp2
```
3. Store ptr to pair (in Bird form)
```
mov rax, r10
add rax, 1
```
4. Move heap cursor
```
add r10, 16
mov [heap_cursor], r10
```



Think about:

how to compile "fst x"?