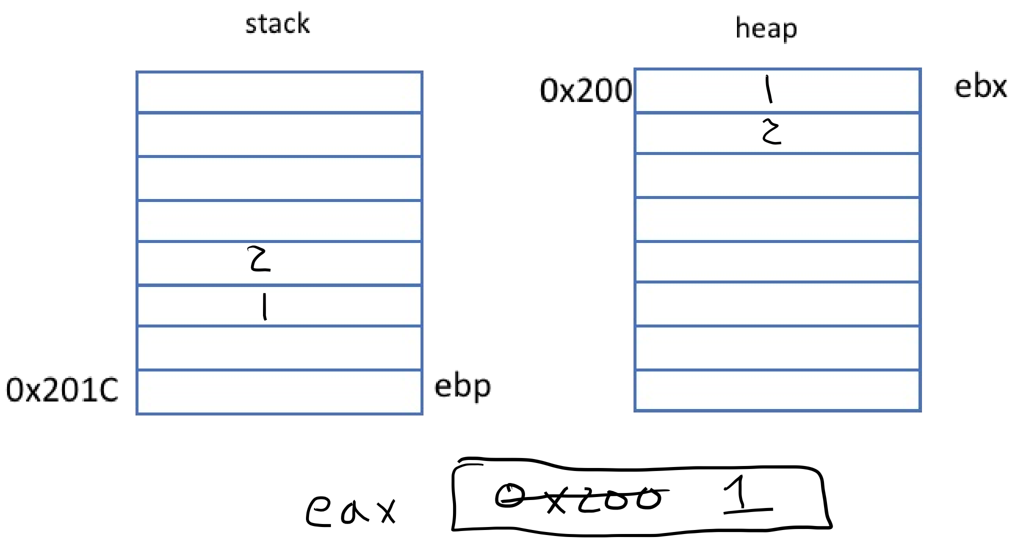
```
type expr =
    . . .
  | EPair of expr * expr
  | EFst of expr
  | ESnd of expr
```

```
(let (x (pair 1 2))
   (fst x))
```

```
(let (x (pair 1 2))
  (fst x))
```

Trace through this part

```
mov eax, 1
mov [ebp - 8], eax
mov eax, 2
mov [ebp - 12], eax
mov eax, [ebp - 8]
mov [ebx], eax
mov eax, [ebp - 12]
mov [ebx + 4], eax
mov eax, ebx
add ebx, 8
mov [ebp - 8], eax
mov eax, [ebp - 8]
mov eax, [eax]
```



ebx → heap pointer (always refers to the next open word on the heap)

```
let rec e_to_is (e : expr) (si : int) (env : tenv) =
...
| EPair(f, s) ->
  let fis = e_to_is f si env in
  let sis = e_to_is s (si + 1) env in
  fis @ [sprintf "mov %s, eax" (stackval si)] @
  sis @ [sprintf "mov %s, eax" (stackval (si + 1))] @
  [
    sprintf "mov eax, %s" (stackval si);
    sprintf "mov [ebx], eax";
    sprintf "mov eax, %s" (stackval (si + 1));
    sprintf "mov [ebx + 4], eax";
    sprintf "mov eax, ebx";
    sprintf "add ebx, 8";
  ]
| EFst(e) -> let cis = e_to_is e si env =
             cis @ ["mov eax, [eax]"]
| ESnd(e) ->
```

fill in code for fst

(fst 1) → Not a tuple

Numbers	0000.....	x1
Bool	1/0 1111....1	10
Pair	xxxxxxxx	00

(extension to tags)

What value is in eax after the bolded part for (pair 1 2)?

A: 0x208

B: 0x200

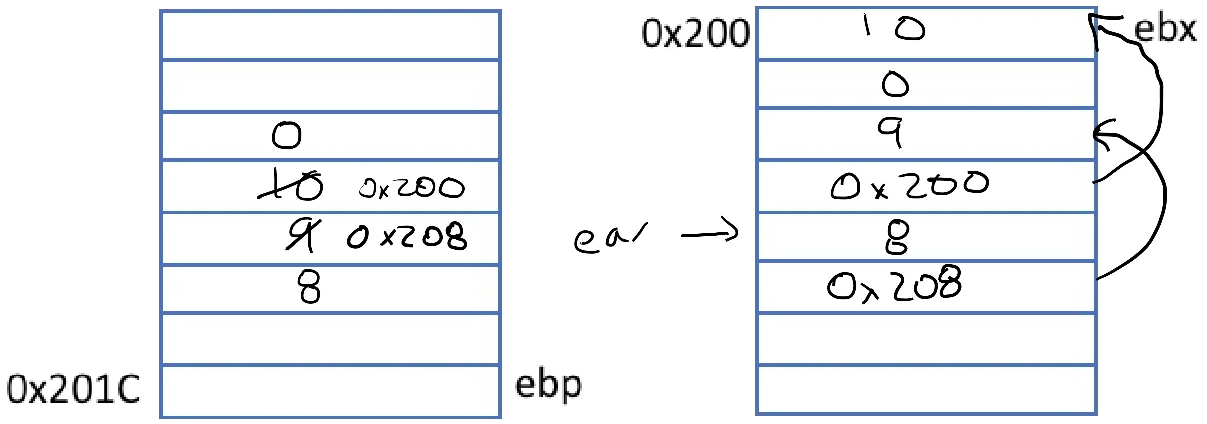
C: 1

D: 2

E: Something else

```
(pair 8
  (pair 9
    (pair 10 false)))
```

```
mov eax, 8
mov [ebp - 8], eax
mov eax, 9
mov [ebp - 12], eax
mov eax, 10
mov [ebp - 16], eax
mov eax, 0
mov [ebp - 20], eax
mov eax, [ebp - 16]
mov [ebx], eax
mov eax, [ebp - 20]
mov [ebx + 4], eax
mov eax, ebx
add ebx, 8
mov [ebp - 16], eax
mov eax, [ebp - 12]
mov [ebx], eax
mov eax, [ebp - 16]
mov [ebx + 4], eax
mov eax, ebx
add ebx, 8
mov [ebp - 12], eax
mov eax, [ebp - 8]
mov [ebx], eax
mov eax, [ebp - 12]
mov [ebx + 4], eax
mov eax, ebx
add ebx, 8
```



What is in eax at the end?

A: 0x208

[✓ B: 0x210]

C: 0x218

D: 0x200

What is in ebx at the end?

A: 0x208

B: 0x210

[✓ C: 0x218]

D: 0x224 (note hexadecimal notation)