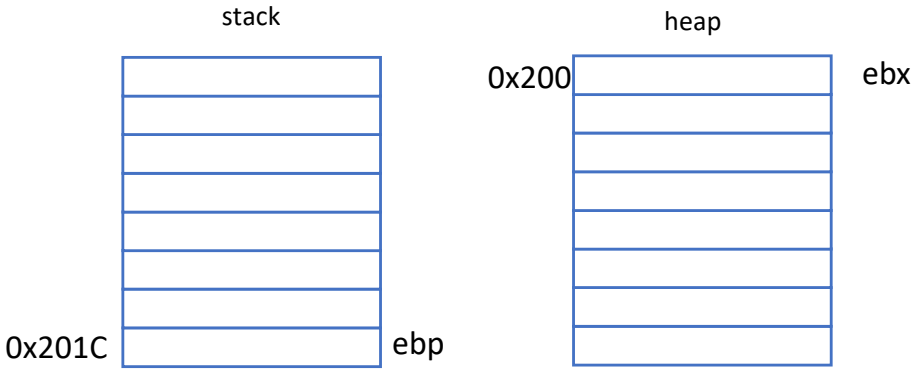
```
let rec e_to_is (e : expr) (si : int) (env : tenv) =
...
| EPair(f, s) ->
  let fis = e_to_is f si env in
  let sis = e_to_is s (si + 1) env in
  fis @ [sprintf "mov %s, eax" (stackval si)] @
  sis @ [sprintf "mov %s, eax" (stackval (si + 1))] @
  [
    sprintf "mov eax, %s" (stackval si);
    sprintf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


| ESnd(e) ->
```

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


int main(int argc, char** argv) {
  int input = 0;

  int* MEMORY = calloc(10000, sizeof(int));

  if(argc > 1) { input = atoi(argv[1]); }
  int result = our_code_starts_here(input, MEMORY);
  printf("%p %d\n", (int*)result, result);
  fflush(stdout);
  return 0;
}
```

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

```
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]
```



```
(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
```