The general form of the operand is

$$A(B, I, S)$$

and it stands for address $A + B + I \times S$ where

- A is a 32-bit signed constant
- I is 0 when omitted
- $S \in \{1, 2, 4, 8\}$ (is 1 when omitted)

Example:

```
movq    -8(%rax,%rdi,4), %rbx  # rbx <- mem[-8+rax+4*rdi]
```

Compilation of get

$C(e_1[e_2]) \stackrel{\text{def}}{=}$

$C(e_1)$
```
pushq %rdi
```
$C(e_2)$
```
movq 8(%rdi), %rdi
popq %rsi
movq 16(%rsi, %rdi, 8), %rdi
```

Operation `lea` computes the effective address of the operand

$$A(B, I, S)$$

```
leaq    -8(%rax,%rdi,4), %rbx    # rbx <- -8+rax+4*rdi
```

Note: we can make use of it to perform arithmetic

```
leaq  (%rax,%rax,2), %rbx  # rbx <- 3*%rax
```

Compilation of `for` loop

$C(\texttt{for } x \texttt{ in } e\texttt{: } s) \stackrel{\text{def}}{=}$

```
        C(e)
        movq 8(%rdi), %rcx
        leaq 16(%rdi, %rcx, 8), %rcx
        pushq %rcx
        leaq 16(%rdi), %rcx
        pushq %rcx
L_start:
        movq (%rsp), %rdi
        movq 8(%rsp), %rsi
        cmpq %rdi, %rsi
        je L_end
        movq (%rdi), %rcx
        movq %rcx, ofs(%rbp)
        addq $8, %rdi
        movq %rdi, (%rsp)
        C(s)
        jmp L_start
L_end:
        addq $16, %rsp
```

where `L_start` and `L_end` are new labels, and *ofs* stands for the offset of variable x.

Note: I assume $C(e)$ yields a list in `%rdi`. We can implement an assembly routine to check it.