

## Stage 5 Codegen Example

Tuesday, May 7, 2024 2:57 AM

```

.data
// error messages
out_of_bounds_msg: .string "out-of-bounds array access"
invalid_alloc_msg: .string "invalid allocation amount"

.text

.globl main
main:
    pushq %rbp
    movq %rsp, %rbp
    subq $32, %rsp
    movq $0, -8(%rbp)
    movq $0, -16(%rbp)
    movq $0, -24(%rbp)
    jmp main_entry

fn main() -> int {
    let x:&int, y:int, z:&int

    entry:
        x = $alloc 10
        z = $gep x 5
        y = $load z
        y = $arith add y 1
        $store z y
        $ret y
    }

    main_entry:
        movq $10, %r8
        cmpq $0, %r8
        jle .invalid_alloc_length // error if failed
        // call _cflat_alloc(10 + 1)
        movq $1, %rdi
        imulq %r8, %rdi
        incq %rdi
        call _cflat_alloc
        movq $10, %r8
        movq %r8, 0(%rax)
        addq $8, %rax
        movq %rax, -8(%rbp)
        // store 10 in header
        // store (header + <word>) in x

        movq $5, %r8
        cmpq $0, %r8
        jl .out_of_bounds // error if failed
        // check 5 <= 0
        movq -8(%rbp), %r9
        movq -8(%r9), %r10
        cmpq %r10, %r8
        jge .out_of_bounds // error if failed
        // z = address of x[5]
        imulq $8, %r8
        addq %r9, %r8
        movq %r8, -24(%rbp)

        movq -24(%rbp), %r8
        movq 0(%r8), %r9
        movq %r9, -16(%rbp)
        // y = load z

        movq -16(%rbp), %r8
        addq $1, %r8
        movq %r8, -16(%rbp)
        // y = y + 1

        movq -16(%rbp), %r8
        movq -24(%rbp), %r9
        movq %r8, 0(%r9)
        // store y into z

        movq -16(%rbp), %rax
        // return y
        jmp main_epilogue

    main_epilogue:
        movq %rbp, %rsp
        popq %rbp
        ret
}
// error blocks

```

*Handwritten annotations:*

- alloc**: A bracket groups the `$alloc 10` instruction in the C code and the `call _cflat_alloc` instruction in the assembly.
- gep**: A bracket groups the `$gep x 5` instruction in the C code and the `movq -8(%rbp), %r9`, `movq -8(%r9), %r10`, `cmpq %r10, %r8`, `jge .out_of_bounds`, `imulq $8, %r8`, and `addq %r9, %r8` instructions in the assembly.
- load**: A bracket groups the `$load z` instruction in the C code and the `movq -24(%rbp), %r8`, `movq 0(%r8), %r9`, and `movq %r9, -16(%rbp)` instructions in the assembly.
- store**: A bracket groups the `$store z y` instruction in the C code and the `movq -16(%rbp), %r8`, `movq -24(%rbp), %r9`, and `movq %r8, 0(%r9)` instructions in the assembly.
- Diagram**: A memory layout diagram showing a header with the value 10 and a series of slots. Arrows indicate the address of the header and the address of the first slot.

ret

// error blocks

.out\_of\_bounds:

lea out\_of\_bounds\_msg(%rip), %rdi  
call \_cflat\_panic

.invalid\_alloc\_length:

lea invalid\_alloc\_msg(%rip), %rdi  
call \_cflat\_panic