Compiler Construction: Assignment 5

Fabian Krause

January 24, 2022

Assignment 5: Compiler for $\mathcal{L}_{Tup}$

6/7 passes:

1. (Shrink: $\mathcal{L}_{Tup} \rightsquigarrow \mathcal{L}_{Tup}$)
2. Expose Allocation: $\mathcal{L}_{Tup} \rightsquigarrow \mathcal{L}_{Tup}^{exposed}$
3. Remove Complex Operands: $\mathcal{L}_{Tup}^{exposed} \rightsquigarrow \mathcal{L}_{Alloc}^{mon}$
4. Explicate Control: $\mathcal{L}_{Alloc}^{mon} \rightsquigarrow \mathcal{C}_{Tup}$
5. Select Instructions: $\mathcal{C}_{Tup} \rightsquigarrow x86_{Global}^{Var}$
6. Register Allocation: $x86_{Global}^{Var} \rightsquigarrow x86_{Global}$
7. Prelude and Conclusion: $x86_{Global} \rightsquigarrow x86_{Global}$

1. Shrink: $\mathcal{L}_{tup} \rightsquigarrow \mathcal{L}_{tup}$

Missing in the book & assignment, concerns following cases:

```
(3, 4, x or y)
```

```
(x or y,)[0]
```

2. Expose Allocation: $\mathcal{L}_{Tup} \rightsquigarrow \mathcal{L}_{Tup}^{exposed}$

Translation given in the book, straight-forward implementation.

3. Remove Complex Operands: $\mathcal{L}_{Tup}^{exposed} \rightsquigarrow \mathcal{L}_{Alloc}^{mon}$

Add new cases for:

- `exp`: `Allocate`, `GlobalValue`, `Call(Name('len'), [...])`, `Subscript(e, e', Load())`
- `stm`: `Subscript(e, e', Store())`, `Collect(n)`

4. Explicate Control: $\mathcal{L}^{mon}_{Alloc} \rightsquigarrow \mathcal{C}_{Tup}$

Add new cases for:

- `explicate_stmt`: `Collect(n)`
- `explicate_pred`: `Subscript(e, e', Load())`

5. Select Instructions: $\mathcal{C}_{Tup} \rightsquigarrow x86^{Var}_{Global}$

Interesting cases: `Allocate` and `Call(Name('len'), [...])`
Working with bits:

- $1 \ll i \mathrel{\widehat{=}} 1 \cdot 2^i \mathrel{\widehat{=}} 1\underbrace{0 \ldots 0}_{i}$
- If x and y have 1's at different positions, then $x + y \mathrel{\widehat{=}} x \vee y$

Tag can be calculated as:

$$\texttt{pointer\_mask} = \sum_i (\texttt{isTupleType}(i) \ll i)$$

$$\texttt{tag} = (\texttt{length} \ll 1) + (\texttt{pointer\_mask} \ll 7) + 1$$

For `Call(Name('len'), [...])`, we can shift the tag to the right and then `andq` with $111111 \mathrel{\widehat{=}} 63$:

$$\texttt{len} = (\texttt{tag} \gg 1) \wedge 111111$$

6. Register Allocation: $x86^{Var}_{Global} \rightsquigarrow x86_{Global}$

- `assign_homes`: Different mapping of color to location for tuple-typed variables.
- `build_interference`: Add edges to callee-saved-registers for variables live at `Collect(n)`.
- `new_color`: Exclude registers reserved for tuple operations: `%r11`, `%r15`

7. Prelude and Conclusion: $x86_{Global} \rightsquigarrow x86_{Global}$

- Initialize garbage collection
- Store root-stack pointer in `%r15`
- Zero-out locations on the root-stack
- Allocate root-stack space

Questions?