notes

Pavel Rubinson

February 6, 2015

Contents

1 Stuff to go-over before coding the compiler:

1.1 TODO Understand the structure of the expected stack:

1.1.1 Environment

Looks like a 2d array arranged by major and minor numbers. Expanded when a new closure is evaluated. Variable lookup is then O(1), because we know in compile time the lexical location (i.e. major+minor indexes) of each variable. That is - the variable names are replaced by minor and major numbers, which are indexes in the lexical environment array.

1. **TODO** find out how the actual environment extension is done.

1.1.2 Stack pointer

Points to the top of the stack.

1.1.3 Frame pointer

Points to the base of the current frame in the stack. Used to access stack fields whose position is known relative to the base of the stack, such as function arguments. When a new frame is opened - the old FP needs to be saved on the stack, in order to be retreived when the frame pops.

2 Stuff to remember:

2.1 New C macros

Define macros to comfortably retrieve stack arguments

2.2 C comments

Adding C comments is apparently important (Mayer said that about half of the generated code should be C comments) He recommends generating comments for compiled expressions that say which expression it is supposed to be. For Example:

```
/* (pvar x j) */
MOVE(R0, FPARG(2 + j));
/* End of (pvar x j) */
```

3 Recommended implementation order:

- Void, (), #f, #t
- Seq, or, if, (and)
- lambda-simple, applic

- tc-applic
- lambda-opt, lambda-var

- Consts

- fvar, def

- Lib