Compiler Principle

Prof. Dongming LU

Apr. 15th, 2024

Content

7 Translation to Intermediate Code

7.3 Declarations

- **The call to transDec: side-effect the frame data structure.**
  - ✓**Variable declaration within a function , additional space will be reserved in the *frame*.**

  - ✓**Function declaration, a new "fragment" of Tree code will be kept for the function's body.**

Variable Definition

- TransDec function: update the value environment and return an augmented type environment

- The initialization of a variable translates into a Tree expression
  - ✓Be put just before the body of the function.

- The translator is applied to function and type declarations
  - ✓The result will be a "no-op“ expression such as Ex(CONST(0)).

```
function f( a:int ,b:int, c:int) =
   {  print_int (a+c);
         let var j:= a+b
            var a:= “hello”
         in print(a);  print_int(j)
         end;
        print_int(b)
    }
```

Function Definition

- **A function is translated into a *prologue*, a *body*, and an *epilogue*.**

```
function f( a:int ,b:int, c:int) =
   {  print_int (a+c);

        let var j:= a+b
              var a:= "hello"
           in print(a);  print_int(j)
        end;

       print_int(b)
    }
```

Function Definition

- A prologue contains
  - (1) pseudo-instructions to mark the beginning of a function;
  - (2) a label definition for the function name;
  - (3) an instruction to adjust the stack pointer
  - (4) instructions to save "escaping" arguments into the frame, and to move nonescaping arguments into fresh temporary registers;
  - (5) store instructions to save any callee-save registers - including the return address register

- Item (1, 3): depend on exact knowledge of the frame size
  - ✓ These instructions should be generated very late , in frame function called procEntryExit3();
  - ✓ Item(2) are also handled at that time.

Function Definition

- **A prologue contains**
  - **(1) pseudo-instructions to mark the beginning of a function;**
  - **(2) a label definition for the function name;**
  - **(3) an instruction to adjust the stack pointer**
  - **(4) instructions to save "escaping" arguments into the frame, and to move nonescaping arguments into fresh temporary registers;**
  - **(5) store instructions to save any callee-save registers - including the return address register**

- **Item (4, 5): part of the view shift, done by a function in the frame module:**

```
/*frame.h*/
…
T_stm F_procEntryExit1(F_frame frame, T_stm, stm)
```

Function Definition

- **The epilogue contains**
  - **(7) an instruction to move the return value (result of the function) to the register;**
  - **(8) load instructions to restore the callee-save registers;**
  - **(9) an instruction to reset the stack pointer;**
  - **(10) a *return* instruction;**
  - **(11) pseudo-instructions, as needed, to announce the end of a function.**

- **Item (9, 11): depend on exact knowledge of the frame size**
  - ✓ **These instructions should be generated very late , in frame function called procEntryExit3();**
  - ✓ **Item(10) are also handled at that time.**

Function Definition

- **The epilogue contains**
  - **(7) an instruction to move the return value (result of the function) to the register;**
  - **(8) load instructions to restore the callee-save registers;**
  - **(9) an instruction to reset the stack pointer;**
  - **(10) a *return* instruction;**
  - **(11) pseudo-instructions, as needed, to announce the end of a function.**

- **For item (7), the translate phase should generate:**

  MOVE(RV, body)

  /*frame.h*/
  Temp_temp F_RV(void)

Function Definition

- **The epilogue contains**
  - **(7) an instruction to move the return value (result of the function) to the register;**
  - **(8) load instructions to restore the callee-save registers;**
  - **(9) an instruction to reset the stack pointer;**
  - **(10) a *return* instruction;**
  - **(11) pseudo-instructions, as needed, to announce the end of a function.**

- **Item (8): part of the view shift, done by a function in the frame module:**

```
/*frame.h*/
...
T_stm F_procEntryExit1(F_frame frame, T_stm,
stm)
```

FRAGMENTS

- A descriptor for the function:
  - ✓ Frame : The frame descriptor containing machine-specific information about local variables and parameters;
  - ✓ Body: The result returned from procEntryExit1.

- Call this pair a fragment to be translated to assembly language.

FRAGMENTS

```
/*frame.h*/
Typedef struct F_frag_ *F_frag;
Struct F_frag_ {enum { F_stringFrag, F_procFrag} kind;
                    union{
                        struct { Temp_label Label;
                                       String str;} string;
                        struct {T_stm body; F_frame frame;}
proc;
                        } u;
                     };

F_frag F_StringFrag(Temp_label label, string str);
F_frag F_ProcFrag(T_stm body, F_frame frame)
```

FRAGMENTS

```
Typedef struct F_fraglist_ *F_fraglist;
Struct F_fragList_ {F_frag head; F_fragList tail;};
F_fragList F_FragList(F_frag head, F_fragList tail);

/*translate.h*/
…
Void Tr_procEntryExit(Tr_level level, Tr_exp body,
                                   Tr_accessList formals);
F_fragList Tr_getResult(void)
```

The end of Chapter 7(3)