Script generated by TTT

Title: Petter: Compiler Construction (02.07.2020)

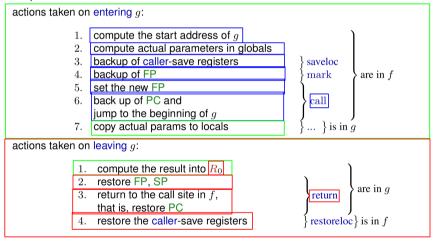
- 57: Caller-Callee Interaction

Date: Fri Jul 03 13:33:20 CEST 2020

Duration: 12:29 min

Pages: 6

Principle of Function Call and Return



Split of Obligations

Definition Let f be the current function that calls a function g. • f is dubbed caller • g is dubbed callee

The code for managing function calls has to be split between caller and callee.

This split cannot be done arbitrarily since some information is only known in that caller or only in the callee.

Observation:

The space requirement for parameters is only know by the caller: Example: printf

39/49

Managing Registers during Function Calls

The two register sets (global and local) are used as follows:

- automatic variables live in *local* registers R_i
- intermediate results also live in *local* registers R_i
- parameters live in *global* registers R_i (with $i \le 0$)
- global variables:

41/49

40/49

Managing Registers during Function Calls

The two register sets (global and local) are used as follows:

- automatic variables live in *local* registers R_i
- intermediate results also live in *local* registers R_i
- parameters live in *global* registers R_i (with i < 0)
- global variables: let's suppose there are none

convention:

41/49

41/49

convention:

41/49

Managing Registers during Function Calls

• automatic variables live in *local* registers R_i • intermediate results also live in *local* registers R_i

• parameters live in *global* registers R_i (with i < 0)

local registers are saved before calling a function

• global variables: let's suppose there are none

ullet the result of a function is stored in R_0

The two register sets (global and local) are used as follows:

• the i th argument of a function is passed in register R_{-i}

Managing Registers during Function Calls

The two register sets (global and local) are used as follows:

- automatic variables live in *local* registers R_i
- intermediate results also live in *local* registers R_i
- parameters live in *global* registers R_i (with $i \leq 0$)
- global variables: let's suppose there are none

convention:

- the i th argument of a function is passed in register R_{-i}
- the result of a function is stored in R_0
- local registers are saved before calling a function

Definition

Let f be a function that calls g. A register R_i is called

- caller-saved if f backs up R_i and q may overwrite it
- callee-saved if f does not back up R_i , and g must restore it before returning