Конструкција компилатора — Пример LLVM међукода

Милена Вујошевић Јаничић

8. април 2020.

За програм који рачуна минимум два унета броја:

```
#include <stdio.h>

int main()
{
  int a, b, min;
  printf("Unesi dva broja\n");
  scanf("%d%d", &a, &b);
  min = a;
  if(b<min) min = b;
  printf("Najmanji broj je %d\n", min);
  return 0;
}
```

генерише се LLVM кôд приказан на слици 1. Следи кратко објашњење основних елемената LLVM кода који су присутни у овом примеру.

Коментари се у оквиру LLVM кода записују иза ознаке `;` и садрже додатне информације о самим LLVM наредбама, као што су тип променљиве над којом је извршена додела вредности, број коришћења променљиве у наставку кода, или имена блокова претходника датог блока. У оквиру `main` функције постоје четири блока: `entry`, `bb`, `bb1` и `return`. На почетку улазног блока, тј. блока `entry`, алоцирана је меморија за све локалне променљиве функције `main` инструкцијом `alloca`. Локалне променљиве функције садрже префикс `%`. Променљиве које служе за складиштење привремених резултата (да би програм био у форми статички јединствене доделе) уместо именима означавају се бројевима. Кôд садржи инструкције за учитавање и уписивање садржаја у меморију, тј. инструкције `load` и `store`, инструкцију целобројног аритметичког поређења `icmp slt` и инструкцију позива функције `call`. Сваки блок, сем излазног блока `return`, завршава се инструкцијом `br` условног (блок `entry`) или безусловног скока (блокови `bb` и `bb1`) која одређује којим се блоком наставља извршавање програма.

```
; ModuleID = 'minimum.o'
target datalayout = "e-p:32:32:32-i1:8:8-i8:8:8-i16:16:16-i32:32:32-i64:32:64-
                     f32:32:32-f64:32:64-v64:64:64-v128:128:128-a0:0:64-f80:32:32"
target triple = "i386-pc-linux-gnu"

@.str = internal constant [16 x i8] c"Unesi dva broja\00" ; <[16 x i8]*> [#uses=1]
@.str1 = internal constant [5 x i8] c"%d%d\00"    ; <[5 x i8]*> [#uses=1]
@.str2 = internal constant [21 x i8] c"Najmanji broj je %d\0A\00" ; <[21 x i8]*> [#uses=1]

define i32 @main() nounwind {
entry:
  %retval = alloca i32                            ; <i32*> [#uses=2]
  %min = alloca i32                               ; <i32*> [#uses=4]
  %b = alloca i32                                 ; <i32*> [#uses=3]
  %a = alloca i32                                 ; <i32*> [#uses=2]
  %0 = alloca i32                                 ; <i32*> [#uses=2]
  %"alloca point" = bitcast i32 0 to i32          ; <i32> [#uses=0]
  %1 = call i32 @puts(i8* getelementptr ([16 x i8]* @.str, i32 0, i32 0)) nounwind
                                                  ; <i32> [#uses=0]
  %2 = call i32 (i8*, ...)* @scanf(i8* noalias getelementptr ([5 x i8]* @.str1,
                                  i32 0, i32 0), i32* %a, i32* %b) nounwind
                                                  ; <i32> [#uses=0]
  %3 = load i32* %a, align 4                      ; <i32> [#uses=1]
  store i32 %3, i32* %min, align 4
  %4 = load i32* %b, align 4                      ; <i32> [#uses=1]
  %5 = load i32* %min, align 4                    ; <i32> [#uses=1]
  %6 = icmp slt i32 %4, %5                        ; <i1> [#uses=1]
  br i1 %6, label %bb, label %bb1

bb:                                               ; preds = %entry
  %7 = load i32* %b, align 4                      ; <i32> [#uses=1]
  store i32 %7, i32* %min, align 4
  br label %bb1

bb1:                                              ; preds = %bb, %entry
  %8 = load i32* %min, align 4                    ; <i32> [#uses=1]
  %9 = call i32 (i8*, ...)* @printf(i8* noalias getelementptr ([21 x i8]* @.str2,
                                   i32 0, i32 0), i32 %8) nounwind
                                                  ; <i32> [#uses=0]
  store i32 0, i32* %0, align 4
  %10 = load i32* %0, align 4                     ; <i32> [#uses=1]
  store i32 %10, i32* %retval, align 4
  br label %return

return:                                           ; preds = %bb1
  %retval2 = load i32* %retval                    ; <i32> [#uses=1]
  ret i32 %retval2
}

declare i32 @puts(i8*)

declare i32 @scanf(i8* noalias, ...) nounwind

declare i32 @printf(i8* noalias, ...) nounwind
```

Слика 1: LLVM кôд генерисан на основу C програма који рачуна минимум два унета броја.