SAARLAND UNIVERSITY
COMPUTER SCIENCE

Lowering and Backend in libFirm

Saarland University

20. Januar 2012

Necessary Preparations for the Backend

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- No unreachable code
- No Bad nodes
- Create vtables
- Lower Sels to address calculations/vtable accesses
- Replace Alloc nodes by calloc()
- Prepare the stack frame of each method
- Select the target

The Print Statement

- Create a global function with the type seen below
- Do not create a graph for it

```
set_entity_visibility(print_entity,
  ir_visibility_external);
```

- Create a call to this function for every PrintStatement
- Link the generated code with a file containing the code below
- Its name depends on the target (later)

```
#include <stdio.h>

void Print(int x)
{
  printf("%d\n", x);
}
```

public static void main(String[] args)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Create a global function for the main method: `int main(void)`
- It always returns 0
- Its name depends on the target (later)

Create vtables

- Create an array type of function pointers with the right length for each class
- Create a global entity of each array type
- Initialise each array with the method pointers

```
ir_graph*          const irg  = get_const_code_irg();
ir_initializer_t* const init =
  create_initializer_compound(n_methods);
for (size_t i = 0; i != n_methods; ++i) {
  union symconst_symbol sym;
  sym.entity_p = method[i];
  ir_node* const symc =
    new_r_SymConst(irg, mode_P, sym, symconst_addr_ent);
  ir_initializer_t* const val =
    create_initializer_const(symc);
  set_initializer_compound_value(init, i, val);
}
set_entity_allocation( entity, allocation_static);
set_entity_visibility( entity, ir_visibility_local);
set_entity_linkage(    entity, IR_LINKAGE_CONSTANT);
set_entity_initializer(entity, init);
```

Sels for Field Accesses

- Calculate the offset for every field in each class

```
set_entity_offset(entity, offset);
```

- Remember to include the pointer to the vtable!
- Replace every Sel for a field access by an Add with the offset of the field

$$addr = Sel_x(p) \rightarrow addr = Add(p, Const_{offset_x})$$

Sels for Virtual Method Calls

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Replace Sels for method calls:

- Load the vtable pointer from the object pointer
- Add the offset of the method in the vtable
- Load the method pointer
- Use this pointer for the Call

$$
\begin{array}{rcl}
 & vtab & = Load(m, p) \\
 & fct & = Load(vtab_m, Add(vtab_{res}, Const_{offset_f})) \\
Call(m, Sel_f(p), \dots) \rightarrow & & Call(fct_m, fct_{res}, \dots)
\end{array}
$$

Replace Alloc Nodes

- Calculate the size of every class type (probably in hand with calculation field offsets)

```
set_type_size_bytes(type, size);
set_type_state(type, layout_fixed);
```

- Create a global function calloc() without graph (like Print() earlier)

```
void* calloc(size_t count, size_t size);
```

- Replace every Alloc by a call to calloc() with count 1 and size of the requested type
- Store the right vtable pointer into the allocated memory

$$
\begin{array}{lcl}
a = Alloc_X(m, 1) & & obj = Call(m, SymConst_{calloc}, 1, Const_{sizeof_X}) \\
 & & s = Store(obj_m, obj_{res}, SymConst_{vtab_X}) \\
p = a_{res} & \rightarrow & p = obj_{res} \\
m = a_m & \rightarrow & m = s_m
\end{array}
$$

Prepare the Stack Frame

- We do not need any entities on the stack, so we set the size to 0:

```
ir_type* const frame = get_irg_frame_type(irg);
set_type_size_bytes(      frame, 0);
set_type_alignment_bytes(frame, 4);
```

Select the Target

- The compiler shall support several target platforms
  - Windows (`--win32`)
  - OS X (`--mac`)
  - Linux (and other Unices) (`--linux`)
- If the target is Windows or OS X then all external names (Print, calloc, main) must to be prepended with an underscore
- You have to pass switches to the backend to select the different targets

Target Switches for the Backend

- Windows:

```
be_parse_arg("ia32-gasmode=mingw");
```

- OS X:

```
be_parse_arg("ia32-gasmode=macho");
be_parse_arg("ia32-stackalign=4");
be_parse_arg("pic");
```

- Linux:

```
be_parse_arg("ia32-gasmode=elf");
```

Call the Backend


```
void be_main(FILE* output, char const* name);
```

- Open a file with the name a.s and pass it to be_main()
- Now you can make a binary from the generated assembler and start it:

```
%cc a.s Print.c
%./a.out
```