CAP : Lab 5 - MIF08 : Lab 4

November & December 2017

Code Generation

Input: a mu file:

```
var n:int;
n=6;
```

Output: a leia file:

```
1       ; ; ( stat ( assignment n = ( expr (atom 6) ) ;) )
2       .LET r1 6
3       COPY r2 r1
```

Code Generation, first step

- 3-address code generation according to the code generation rules of the course:

```
// e1+e2 code generation rule
  t1 <- GenCodeExpr(e_1)
  t2 <- GenCodeExpr(e_2)
  dr <- newTemp()
  code.add(InstructionADD(dr, t1, t2))
  return dr
```

- **TODO: implement them**:

```
tmpl = self.visit(ctx.expr(0))
tmpr = self.visit(ctx.expr(1))
dr = self._prog.newtmp()
if ctx.myop.type == MuParser.PLUS:
    self._prog.addInstructionADD(dr, tmpl, tmpr)
```

- this is not executable code

Result after first step

The previous step uses instructions of an API like:

```
self._prog.addInstructionADD(dr, tmpl, tmpr)
```

which side effect is to construct a LEIA prog as a list of 3 adresses instructions with temporaries (virtual registers, from the class `VirtualRegister`.

This list can be dumped (with `printCode` in the API) into a .s file:

```
1       ;; (stat (assignment n = (expr (atom 6)) ;))
2       .LET temp_1 6
3       COPY temp_2 temp_1
```

We cannot test!

Code Generation, second step

The allocation process:

- takes as input the preceding result
- modifies the list of instructions with temporaries into list of instructions with physical registers or accesses to memory.
- a trivial allocator is given.

TODO : all in memory allocation (see course)

Code Infrastructure

```
...code/Mu-codegen$ ls
Allocations.py   Main.py    MyMuCodeGen3AVisitor.py
APICodeLEIA.py   Makefile   test_codegen.py
ExpandJump.py    Mu.g4      Instructop,3A.py Operands
```

- The Mu grammar in `Mu.g4`,
- A Makefile, Main as usual.
- Library and unit tests in `test*.py`.
- API for generating LEIA code : `APICodeLEIA`, `Instruction3A.py` `Operands.py`
- `ExpandJump.py` is used to dump the conditional jump instruction into legal LEIA code.
- `Allocations.py`: allocators for LEIA code.
- **TODO : edit and fill `MyMuCodeGen3AVisitor.py` mainly**. You may have other changes to make in other files (Allocation).

LEIA API

In this API (APICodeLEIA, Instruction3A, Operands) :

- A class for a program `LEIAProg`. The program contains a list of instructions, methods to add instructions, to increment temporaries numbers, ...
- Classes for instructions: `Instruction`, `Instru3A`, `Label` ...
- A 3 address instruction contains arguments that can be `Immediate`, `VirtualRegister`, `Register`, or a `Condition` in the special case of the condjump. ...
- the `CondJump` instruction (label,dr1,cond,dr2) has the meaning:

  `if (dr1 cond dr2) jump to label.`
- Ignore code concerning graphs (print dot, add edges) and dataflow (in and out sets), this is for next lab.