dOvs - Codegeneration

Group 9
Miran Hasanagić - 20084902
Jakob Graugaard Laursen - 20093220
Steven Astrup Sørensen - 201206081

November 18, 2015

1 Instruction selection

How did you implement the missing elements in x86gen. Describe how the file works as a whole and explain a few interesting cases.

2 The support modules

Write how you implemented the missing elements in x86frame..

3 Problems encountered & experience gained

If you had any problems in the work process, write it here with what you gained.

4 5 tiger programs

5 interesting tiger programs that you find test your codegeneration in a good way.

4.1 test01.tig

```
/*
Array test for equal of equal things and unequal for unequal things
*/
let
        type A = array of int
        var b := A[2] of 0
        var c := b
        var d := A[1] of 3
in
        b = c & c <> d
end
```

4.2 test02.tig

```
/*
Test for loops,
that we update control variable correctly,
```

```
and that we do not loop to much,
and that we do no enter loops where low > high
*/
let
        var a := 0
in
        for i := 1 to 0 do a := 99; for i := 1 to 3 do a := a + i; a
end
```

4.3 test03.tig

```
/*
Checks if we put the arguments in correct order
*/
let
        function sub(a:int,b:int):int = a - b
in
        sub(3,2)
end
```

4.4 test04.tig

```
let
type rectype = {name: string, address: string, id: int, age: int}
type arrtype = array of rectype
var arr := arrtype [5] of
                  rectype{name="aname", address="somewhere", id=0, age=0}
in
    arr[3].name := "kati";
    arr[1].age := 23;
    arr[1].age
end
```

4.5 test05.tig

```
/* test for recursive functions
 * current version has an infinite loop for negative cases
*/
let
        even(i:int):int = if i = 0 then 1
                                                else if i = 1 then 0
                                                else odd(i-1)
        odd(i:int):int = even(i-1)
in
        even(4) & 0 = odd(4)
end
```