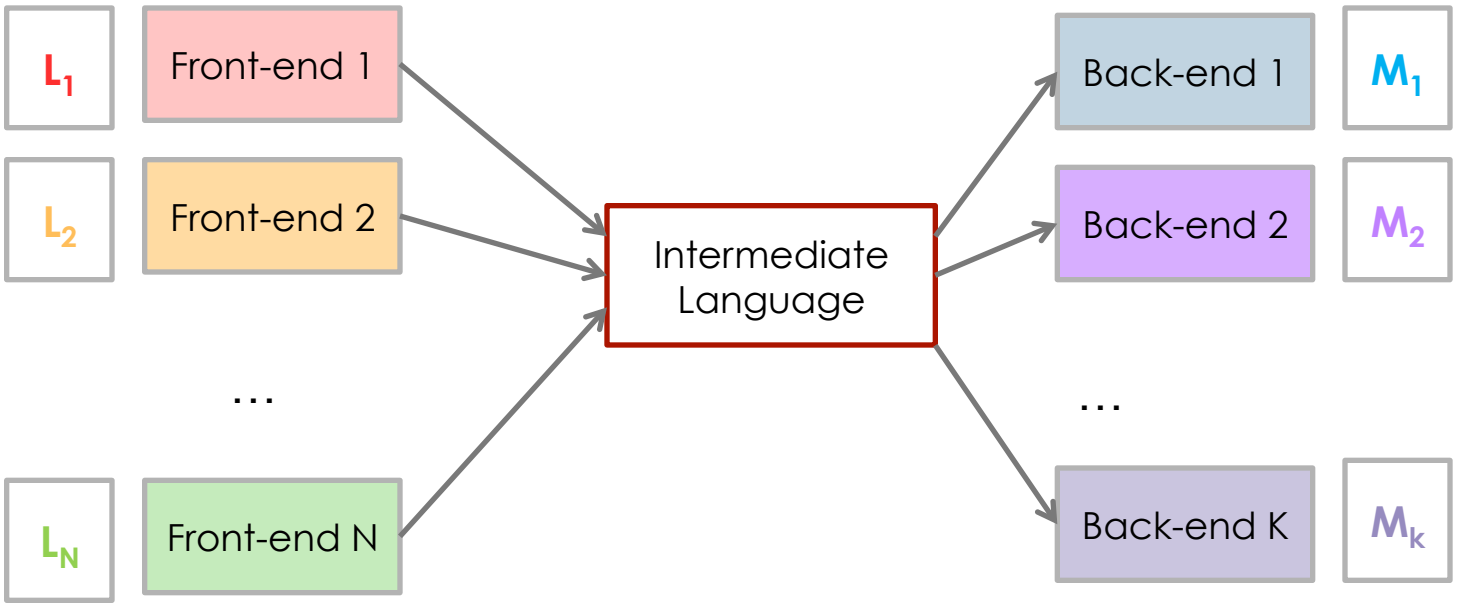
Translating to the target code

Lecture 13

Formal Languages and Compilers 2011

Nataliia Bielova

Front-end and back-end



The target code: C

- Using C like assembler:
  - using one "stack":

    ```
    memory_t stack[10240]
    ```

  - elements of the stack

    ```
    typedef union {int i; float f;} memory_t
    ```

  - stack pointer and base pointer
  - parameters, variables, …, in the stack (name -> offset)
  - "hardware" registers: `memory_t reg[1024]`
- Exploit some functionality of C:
  - commands of `return`
  - using union for data `int` and `float`
  - `printf`, that we all like so much
  - … and many other features!

Example of translation: assignment

crème CAraMeL

```
x := (3 + 7) * 11;
write(x);
```

IC

```
ADD     Val INT: 3  Val INT: 7      reg[2].i
MUL     reg[2].i    Val INT: 11     reg[1].i
CPY     reg[1].i    NULL            offset 0
OUT     offset 0    NULL            NULL
```

C

```
reg [2] .i      = 3 + 7;
reg [1] .i      = reg[2].i * 11;
stack [0].i     = reg[1].i;
printf("%d\n", stack[0].i);
```

How is it implemented?

- Translation "one-by-one" from the intermediate code

```
ADD     Val INT: 3  Val INT: 7      reg[2].i
```

```
reg [2] .i      = 3 + 7;
```

- Code of the compiler:

  http://disi.unitn.it/~bielova/flc/exercises/13-Compiler.zip

- File: target.ml