Compilation Project

Philippe Bidinger, Florent Bouchez Tichadou

philippe.bidinger@imag.fr
florent.bouchez-tichadou@imag.fr

Université Grenoble Alpes

January 2th, 2017

Contents

Contents

Practical Organization

- Mon. 2 - 13h30 - 17h00 Presentation of the project
- Wed. 4 - 14h00 - 17h00 Git/Caml labs
- Fri. 7, 14, 21 - 18h weekly report (mid-project submission on 14)
- Wed. 25 - 12h00 Final project submission
- Thu. 26 - 09h00 - 13h00 Project defense

We will be there almost every afternoon for answering questions.

Weekly Organization

Team organization

Highly recommended: every day, ≈10 minutes "stand-up meeting" at fixed hour.

For each task:

- - new task: description of task
  - existing task: status of task (work done, work remaining...)
- people assigned on task
- expected time of completion

Assign the "scribe" role: takes log of meeting (must not change during week).

Weekly Organization

Team organization

Highly recommended: every day, ≈10 minutes "stand-up meeting" at fixed hour.

For each task:

- - new task: description of task
  - existing task: status of task (work done, work remaining...)
- people assigned on task
- expected time of completion

Assign the "scribe" role: takes log of meeting (must not change during week).

Weekly report is the list of all five logs of the week.

Group Organization

Subscribe on moodle ASAP, you can then choose a team. Then send us an email with:

- team members (4-5 people)
- team name :-)
- team "leader" (recommended)
- programming language (we recommend OCaml, Java or C)
- project name on the imag forge (+ add us as members)

http://imag-moodle.e.ujf-grenoble.fr/course/view.php?id=234
⇒ watch the forum for info (time slots, lectures...)

Grade

Grade will take into account:

- Respect of requirements
- Code:
  - features
  - writing quality (adhere to coding conventions, write well-documented and modular programs)
  - robustness
  - test suite
- Report:
  - "User documentation": what we need to use your compiler
  - "Developer documentation": what a new developer would need, including documentation of test suite
- Demo:
  - 20 min. presentation (technical & commercial).
  - 10 min. questions
- Team management (weekly reports,...)
- Personal modifier

Grade personal modifier

Each person will anonymously grade him·herself and the others:

- Quality of work
- Quantity of work
- Willingness to help others
- Availability (presence)
- Role in team

Example grades, each on 20:

- <10 = problem
- 10 = limit
- 12 = average
- 14 = good
- 16 = very good
- 18 = excellent

Grade personal modifier example

Example : Teacher's grade for the project: 15/20

	Student 1	Student 2	Student 3	Student 4
Self	15	19	12	14
Pair 1	12	18	9	14
Pair 2	15	16	14	15
Pair 3	16	16	13	13
Student Avrg.	14,5	17,25	12,75	14
Team Avrg.	14,25	14,25	14,25	14,25
Modifier factor	1,0175	1,2105	0,8495	0,9825
Grade	**15,3**	**18,2**	**13,8**	**14,7**

(Note: unless special cases, we limit the modifier to $\pm 20\%$*.)*

Cheating

- You must write all the code you hand in, except for code that was provided. You are not allowed to look at anyone else's solution, but you may discuss your assignments with other students.
- We use plagiarism detection tools
- we will follow you during the project.

Project setup

- Download `archive.tar.gz` from `www-verimag.imag.fr/~bidinger/archive.tar.gz` (or moodle) and untar it (`tar zxvf archive.tar.gz`).
- All the instructions and documentation for the project is accessible from `doc/index.html`.
- Depending on the language you will use for the project, you will start with the code given in java, ocaml or c.
- See README for the organization of the archive.

Git setup

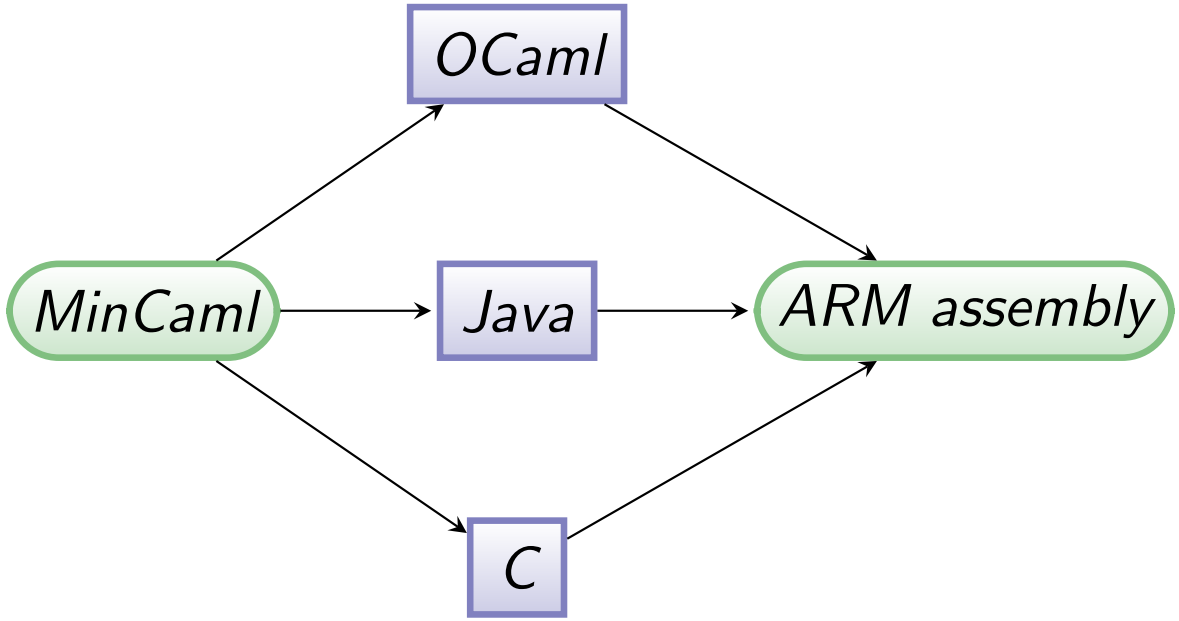
To set-up your git repository on the forge, see the details on moodle.
http://imag-moodle.e.ujf-grenoble.fr/course/view.php?id=234

- Register on forge.imag.fr
- Register a new project, choose `git` as the SCM.
- Add files to your own project
- Then everyone can get a copy

Contents

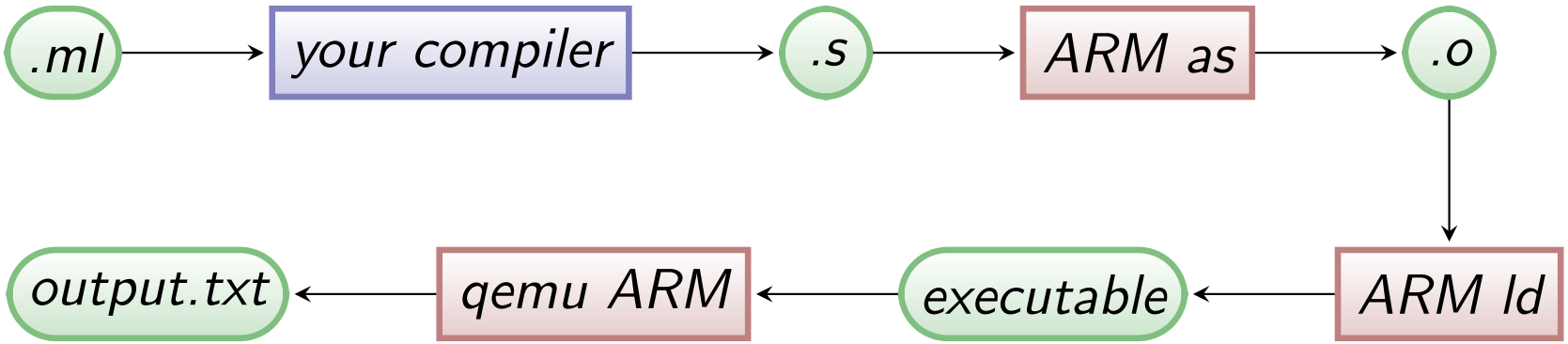
The Compiler

Skeletons given in OCaml, Java, C, with parser and minimal tests.



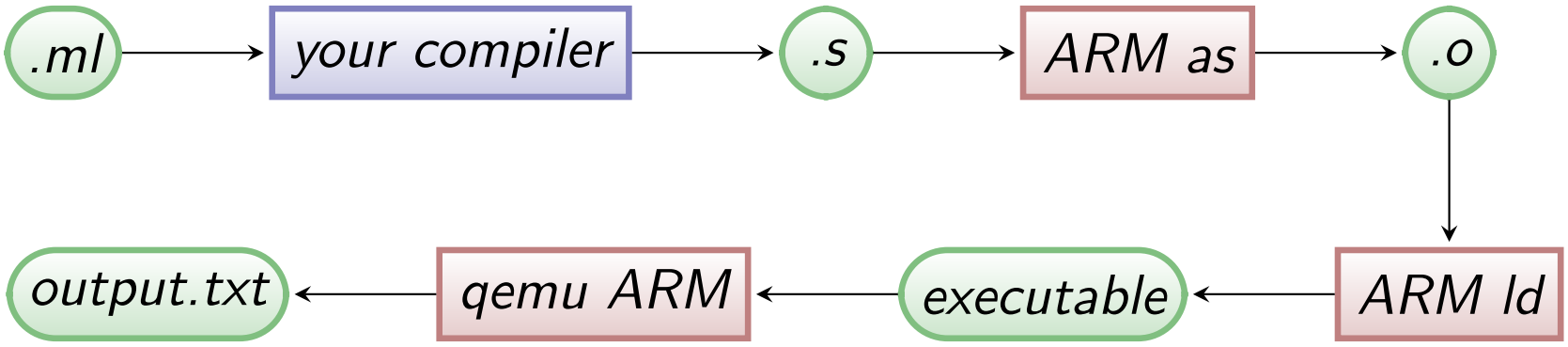
Toolchain

Compiler Toolchain:

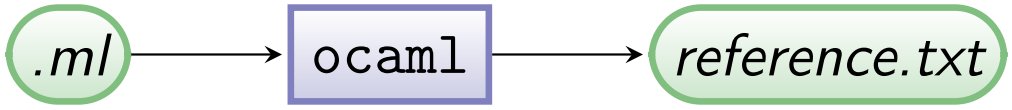


Toolchain

Compiler Toolchain:



Comparison:



Warning: `ocaml` interprets the code (no compilation).

Toolchain Example

```
$ ../min-caml gcd.ml -o gcd.s
$ arm-none-eabi-as -o gcd.o gcd.s libmincaml.S
$ arm-none-eabi-ld -o gcd gcd.o
$ qemu-arm ./gcd
```

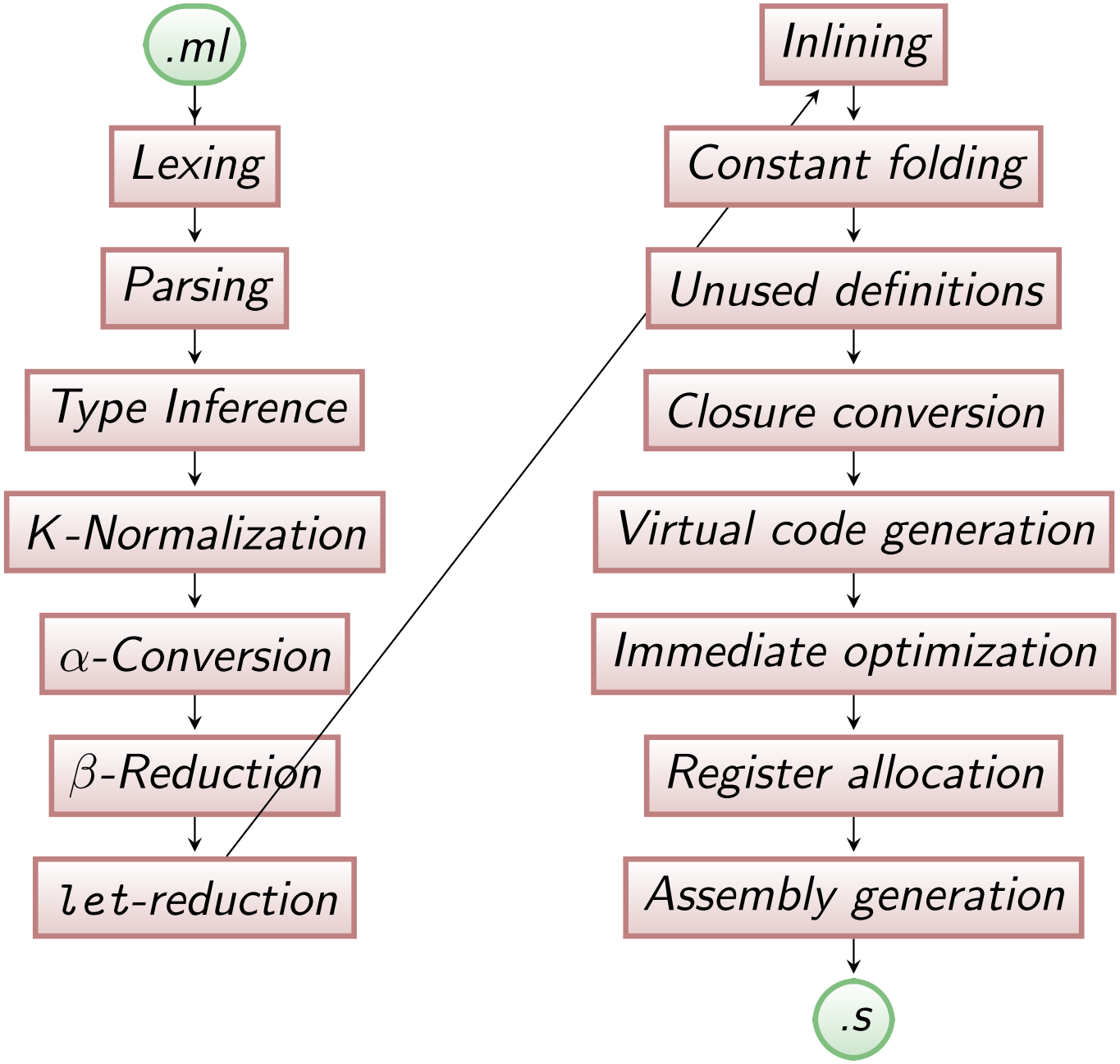
See in the archive for other examples.

GCD Example

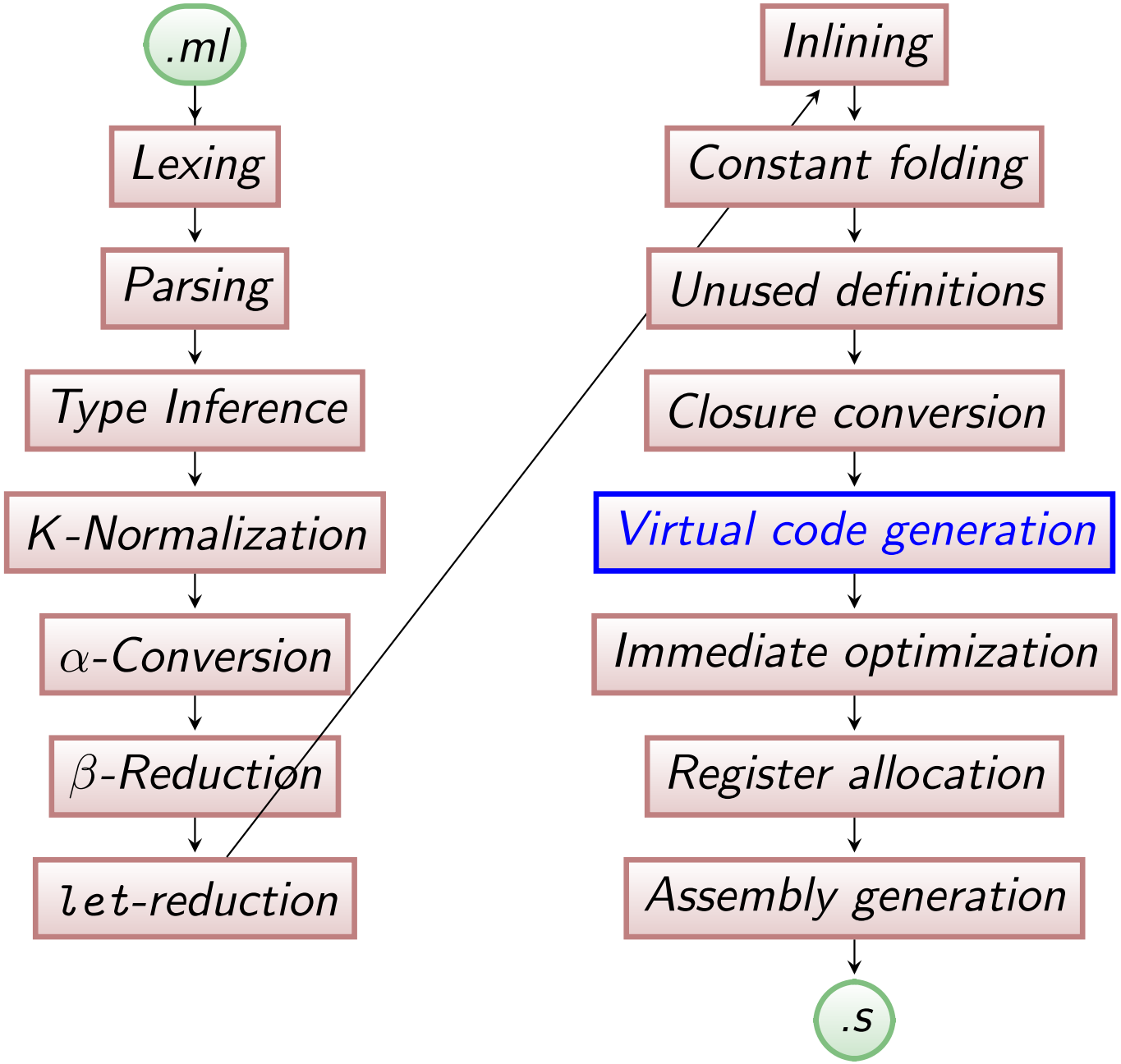
```
let rec gcd m n =
  if m = 0 then n else
  if m <= n then gcd m (n - m) else
  gcd n (m - n)
```

```
        .text
gcd.7:
        cmp   r0, #0
        bne   be_else.17
        mov   r0, r1
        bx    lr
be_else.17:
        cmp   r0, r1
        bgt   ble_else.18
        sub   r1, r1, r0
        b     gcd.7
        nop
ble_else.18:
        sub   r0, r0, r1
        mov   r14, r1
        mov   r1, r0
        mov   r0, r14
        b     gcd.7
        nop
```

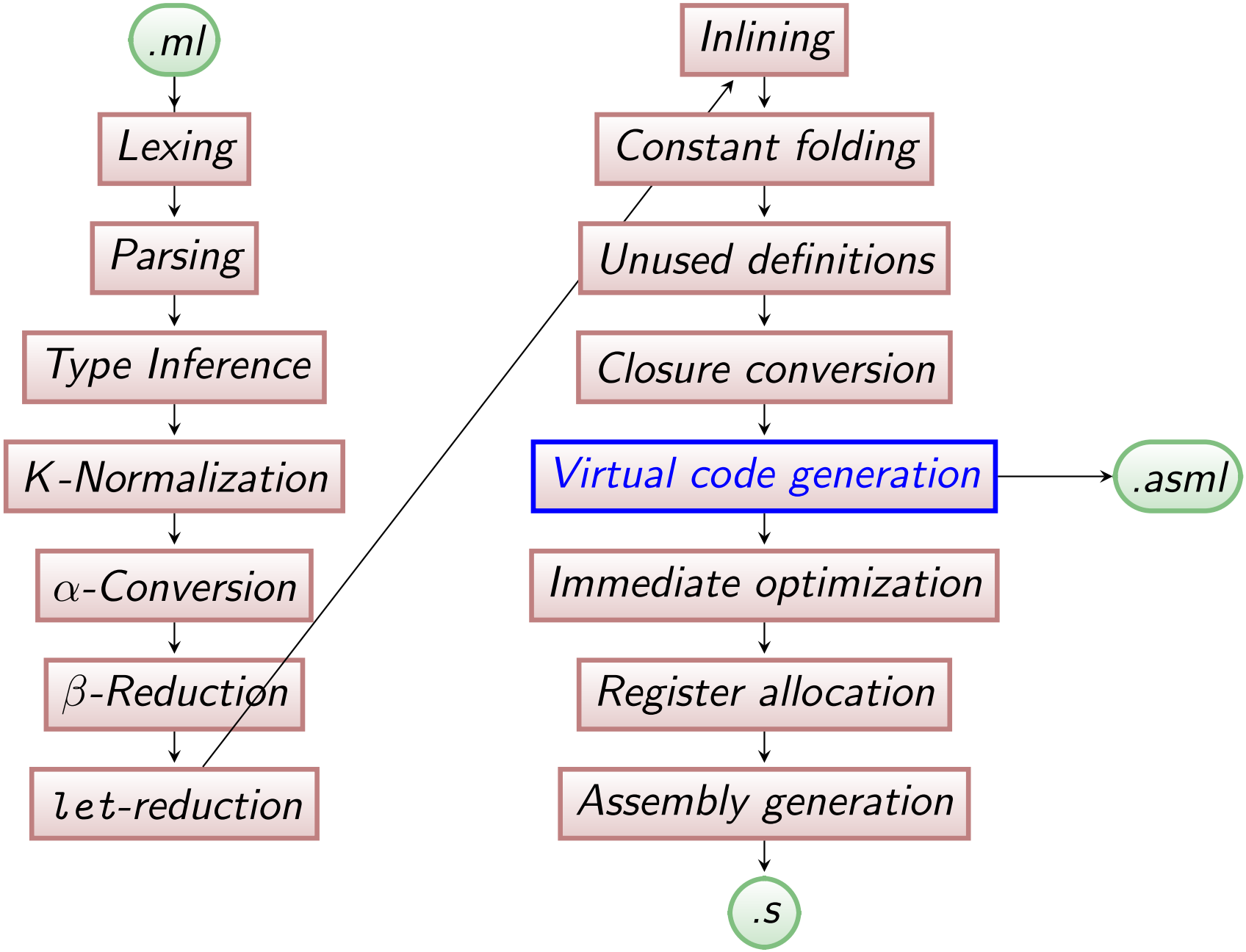
Compiler Passes



Compiler Passes



Compiler Passes



Tasks and Incremental Development

Decide what you want to do!

- Front-end (35%)
- Type inference and type checking (15%)
- Back-end (35%)
- Testing (15%)

Each task can be subdivided furthermore.

- Don't work one pass a at a time, but one feature at a time
- Integrate ASAP

Incremental development

- simple arithmetic expression
- call to external functions
- if-then-else
- functions (let rec)
- tuples and arrays
- closures
- floats

The ASML intermediate representation

ASML is at the interface between front-end and back-end.
It is a *Virtual Machine Code*.

Features

- values stored in temporaries (“registers”)
- functions represented by labels
- no nested definitions
- memory accesses are explicit
- if-then-else constructs
- operators for memory allocations

The ASML intermediate representation

ASML is at the interface between front-end and back-end.
It is a *Virtual Machine Code*.

ASML example

```
let _ =
  let size = 2 in
  let init = 0 in
  let arr = call _min_caml_create_array size init in
  let i0 = 0 in
  let v0 = 1 in
  let tu = mem(arr + i0) <- v0 in
  let i1 = 1 in
  let v1 = mem(arr + i0) in
  mem(arr + i1) <- v1
```

The ASML intermediate representation

- We provide an interpeter for ASML. You can test the front-end independently of the back-end!
  - Compare the behavior of the ASML program with the behavior of the source file (using ocaml)
  - But you need to output ASML (easy)
- you can test the back-end independently of the front-end too
  - But you need to be able to parse ASML (more difficult!). Maybe a simpler representation (JSON?)
- In any case, you will have to write a datatype that encode ASML programs.

The Command Line

We will use our test suite on your compilers.

Return value

- 0 on success
- 1 on error, plus error message on stderr

Required command-line options

-o <file> : output file

-h : display help

-v : display version

-t : only type checking

-asml : print ASML

The Command Line

We will use our test suite on your compilers.

Return value

- 0 on success
- 1 on error, plus error message on stderr

Required command-line options

-o <file>	:	output file
-h	:	display help
-v	:	display version
-t	:	only type checking
-asml	:	print ASML
-my-opt	:	you can add personal options (optimizations, etc.)

Git Advices

We will clone your git repositories on Friday 15th January at 6pm.

We will only consider the **master** branch!

make	must compile your compiler
make test	must launch your test suite

Git Advices

We will clone your git repositories on Friday 15th January at 6pm.

We will only consider the **master** branch!

`make` must compile your compiler

`make test` must launch your test suite

- we advise you to work on development branches
  ```
  $ git checkout -b devel
  $ git checkout -b <login>
  ```
- push on `devel` to share development with others
  ```
  $ git checkout devel
  $ git merge <login>
  $ git push
  $ git checkout <login>
  ```
- push on `master` only when it's safe

Master must always be in a working state!

Compressed TODO List

- Register on moodle (COMPPROJ course)
- Leader sends email at philippe.bidinger@imag.fr and florent.bouchez-tichadou@imag.fr
- Read the documentation, including the min-caml article
- Note the other available information moodle
- Register project for your team on forge.imag.fr
- Watch for news on moodle forum (esp. time&place of meetings)
- Start organization! (stand-up meetings, etc.)