Compilation Project

Mohannad ALMASRI Martin Vassor
Grenoble Alpes University
10th December

Project setup

- All announcements will be made by email
- Download the archive archive.zip, and unzip it
- 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 or ocaml
- See README for the organization of the archive

Git setup

- Each student creates an account on github
  - Good habit to use it for all your programming projects
  - Showcase of your work
  - Use something else if you don't like github
- One student creates a project and upload the initial files
- Add all other students as collaborators

Git setup

- Add me as collaborator on github user "LIGMRIM" ," Bromind"
- Project has to be set to private eventually
- Go to education.github.com to request free account upgrade
  - If problem, use bitbucket.com
- Everbody in the group clones the project
- Set up your git locally (see provided doc)
  - Gitcofing
  - Upload public key

Important dates

- Weekly reports
  - Fri. 14/12, 21/12, 11/01 - 19h
- Mid-project submission on 24/12
- Final project submission
  - Thu. 17/01 (Tue. 15/01 for M1 INFO) - 12h00
- Project defense
  - M1 INFO: Wed 16/01 - 09h00 - 17h00
  - M1 MoSIG: Fri. 18/01 - 09h00 - 17h00

Important dates

- Most days, I'll be answering questions in **, either morning or afternoon.
- We may schedule extra lectures if needed
- All missed deadlines will be penalized
- Project will be retrieved from git repository master branch on time

Supervision

- You should be autonomous
  - Set up a github/bitbucket project
  - Follow an ocaml tutorial
  - Find an ARM instruction
  - Write a bash script or compile a java program ...
- However, ask for help if:
  - You are stuck (don't wait for the last day!)
  - Need clarification on provided material
  - Want to make sure you're on the right path

Supervision

- You're expected to work 7-8 hours a day for 4 weeks.
- Use internet with good judgment
  - Stack overflow for all git/programming questions
  - But MinCaml, ASML only used in this project
  - Careful with compilation resources

Communication

- Email
  - Clear and precise questions
  - Try to think first
  - Answer by email, or during presence hour
- Updates, corrections
  - Email to leaders
  - Update on webpage

Group organization

- Leader
  - Global view of the project
  - Communication with supervisor
  - Check deadlines and deliverables
- Difficulties
  - Distribution of tasks (efficiency vs interest, dynamic)
  - Heterogeneity (skills, motivation)
- Daily meeting
- Don't stay in a bubble

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

- Send me an email TODAY with:
  - Team members and their emails (4-6 people)
  - Team name
  - Team « leader » (recommended)
  - Programming language: Java or OCaml
  - Git repository that I can clone
  - I'll confirm every email
  - Without this, you will not be part of the project

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
  - Documentation:
    - “User documentation”: what we need to use your compiler

Grade

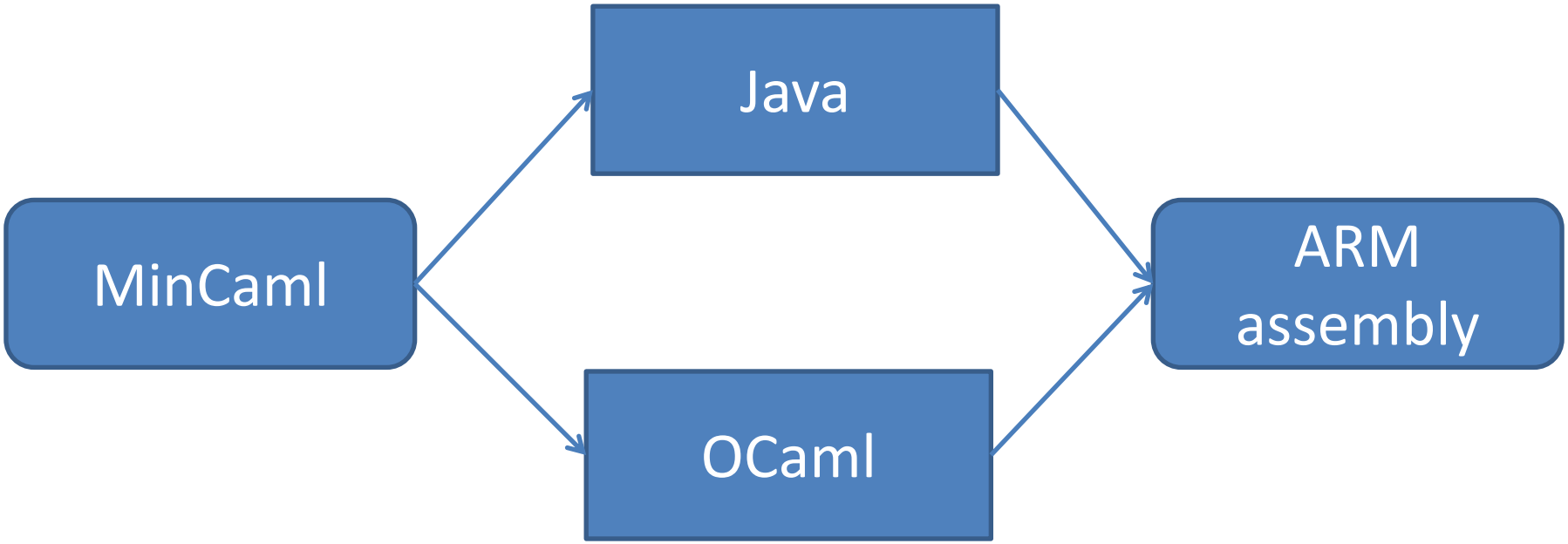
- Grade will take into account:
  - Documentation:
    - “Developer documentation”: what a new developer would need, including documentation of test suite
    - Recommended: javadoc (or ocamldoc)
  - Demo:
    - 20 min. presentation (technical & commercial).
    - 10 min. questions
  - Team management (weekly reports,. . .)

Cheating

- You must write all the code by your hand in, except for code that was provided. You are not allowed to look at anyone else's solution or use code written by someone else.
- But you may discuss your assignments with other students. This is encouraged.
- Plagiarism will be detected (detection tools).
- We will follow you during the project.

The Compiler

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



GCD Example

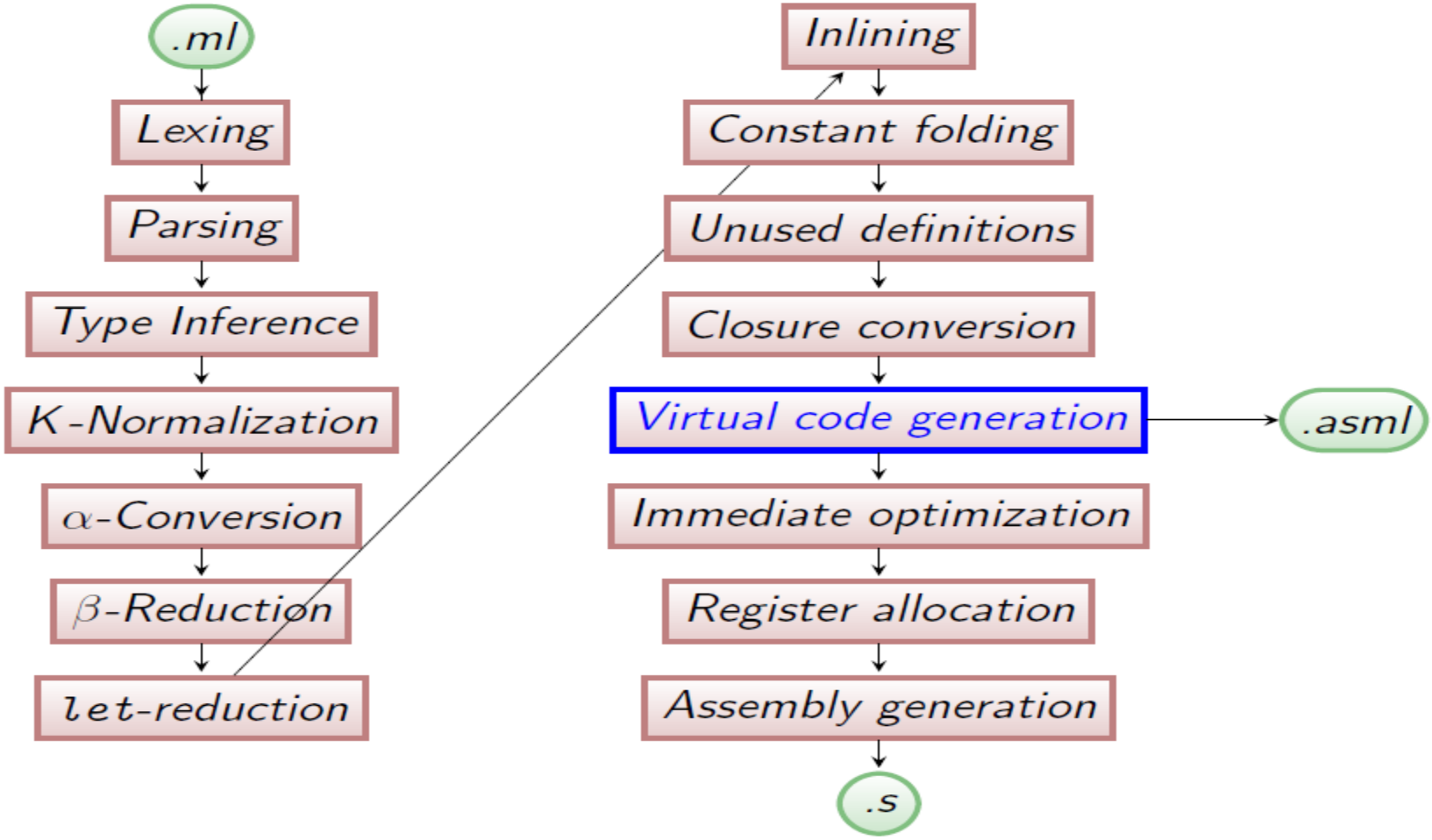
.ml

```
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



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

Tasks and 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 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

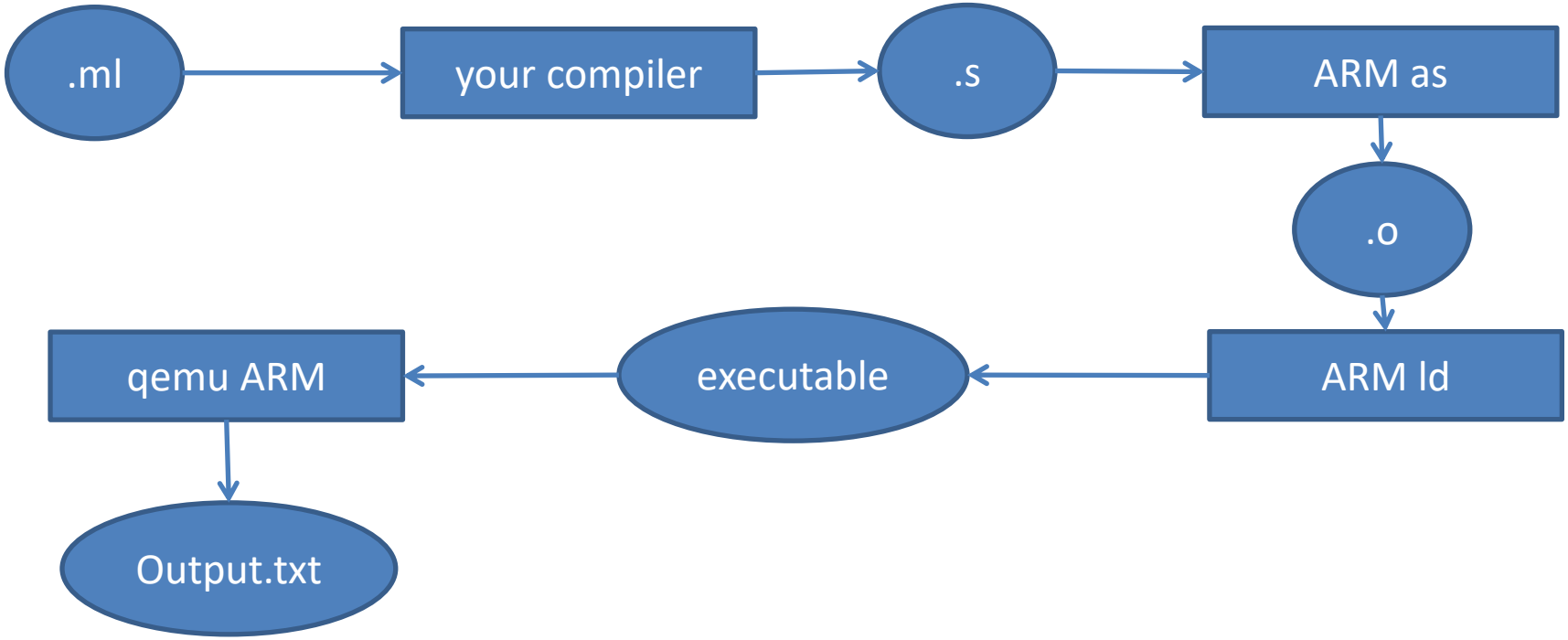
- We provide an interpreter for ASML. You can test the front-end Independent of the back-end!
  - Compare the behavior of the ASML program with the behavior of the source le (using ocaml)
  - But you need to output ASML (easy)
- You can test the back-end independently of the front-end too
  - But you 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
    - -p : parse only
    - -asml : print ASML
    - -my-opt : you can add personal options (optimizations, etc.)

Questions?

Compiler Toolchain



- Comparison:

Toolchain Example

```
$ mincamlc 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.