Project: Microprocessor (2019-20)

Cours *Systèmes numériques : de l'algorithme aux circuits*

Hadrien Barral <Hadrien.Barral@normalesup.org>
Georges-Axel Jaloyan <Georges-Axel.Jaloyan@ens.fr>

This project aims at creating from scratch a microprocessor able to run a digital clock as presented in the lectures. To achieve this, you will have to create a circuit simulator, that simulates any (correct) digital synchronous circuit. You will then design a fully functional microprocessor, and simulate it in the simulator you wrote. Finally, you will have to program this microprocessor so that it runs the program of your digital clock.

To put it in another way, you will have to create (in this order):

- A simulator able to run any synchronous digital circuit.
- An architecture of a microprocessor (with its instruction set architecture)
- The program of the digital clock, written using the instruction set architecture of your microprocessor.

1 The project

The *netlist* simulator

The simulator is tasked with the simulation of any digital synchronous circuit. For this, the simulator is provided on the standard input (or any other file) with the description of the circuit, in the form of a *netlist* (a language to describe circuits), as well as the inputs of this circuit at each step of the simulation. In particular, the simulator should reject any invalid circuit, defined as a circuit that has a loop in its description. Additionally, the simulator should provide all features of a standard netlist, including memory primitives such as ROM (*Read-Only Memory*) and RAM (*Random-Access Memory*).

Formally, the simulator takes as input:

- the circuit description in the form of a netlist
- the number n of cycles (or steps) to simulate
- the values of every input on each of the n cycles

it should output:

- the values of every output on each of the n cycles

For the sake of simplicity, the netlist given as an input to your simulator will be in a toy language, represented as an unordered list of logic gates, as well as various hardware net operations. This netlist can be generated by a compiler called MiniJazz, from a higher-level

description of your circuit in a domain-specific language called MiniJazz, available on github https://github.com/inria-parkas/minijazz/. Many off-the-shelf *Hardware-Description Languages* are available, but may not be adequate for this project as the produced netlists are much more complex to simulate.[1]

The microprocessor

You shall design and implement a microprocessor able to run the digital clock as presented in the lectures. You have total freedom of choice on the Instruction Set Architecture (ISA), that could either be a general purpose ISA able to simulate any program to a very specialized ISA targeted at handling digital clocks (e.g. instructions that count modulo 12, 24 or 60). For general purpose ISAs, you may or may not build upon existing ISAs.

To start with, you are required to specify the general architecture of your microprocessor as well as its ISA. On top of the usual specifications for any microprocessor (memory, register size and number, ...), you will have to anticipate the issues raised by inputs (hardware clocks) and outputs (clock display).

You may design a MIPS or RISC-V compatible chip (or another exotic architecture). Do ask one of the TAs for advices about which subset of the chosen architecture to implement (trying to design a fully compliant multi-core RISC-V chip is **not** a good idea.).[2]

Later, you shall simulate your processor in your simulator. You are required to design an assembly language and implement its corresponding assembler (~~as well as a C cross-compiler~~) targeting your architecture. This assembly language will help you writing the program of your digital clock.

Basic use-case: digital clock

Your digital clock shall handle seconds, minutes, hours, days, months and years. Furthermore, your program shall have two behaviors:

- A real time mode where the time is initialized using your host system's clock, and an external clock provides a periodic ticking signal in order for your digital clock to work in real-time.
- A fast-forward mode, where your program runs as fast as the processor allows it to run, without taking into account the clocking signal (a prize may be awarded to the fastest team).

Advanced use-cases

If you want to do some extra work, you may go further by adding some architectural optimizations to your processor (*e.g.* caches, pipelines, ...), or target other applications, like cryptographic instructions (*e.g.* carryless multiplication or the bitslice procedure of AES block cipher).

[1] Obviously, risky bets may pay off, especially if you design your own HDL.

[2] MIPS ISA summary: https://www-licence.ufr-info-p6.jussieu.fr/lmd/licence/2010/ue/LI221-2010oct/td_tp/memento2010.doc.pdf and RISC-V ISA: https://content.riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf

2 Practical details

Organization

Each of you will first implement the netlist simulator, and is expected to submit it. Then, you will be merged in groups of 3 or 4 to design and build the microprocessor, the assembler and the digital clock.

You are free to chose the tools and the languages you wish for every part of this project, inasmuch as your project can be run on a standard Linux computer. We provide you the skeleton of a netlist parser in OCaml, that could be rewritten in any other language, if you wish so.

Several lectures will focus on this project, and its different parts. You are invited to share your design and technical choices either by sending an email, or by discussing IRL (just send an email beforehand to check whether we are at the ENS).

Milestones and deadlines

Every document in this project shall be typed on a computer. The sources of your project shall be packed in a tarball, with a `README.md` file explaining how to compile and execute your programs (like arguments, ...). You may add whatever you deem necessary in this README file. You can also add a Makefile, it will only make us happier. More generally, good engineering practices will be rewarded (this includes automated build, continuous integration, version control, software testing, nice logo/project title and website).[3] Reports and archives should be send by mail to `hadrien.barral@normalesup.org` and `georges-axel.jaloyan@ens.fr` **within** the deadline. All deadlines are AoE.

November 13th Submission of your netlist simulator (individual)
: You shall provide a netlist simulator able to run every circuit provided in the lab session 1, as well as a short summary explaining the behavior of your simulator and the difficulties you encountered.

December 16th Small report on the design of your microprocessor (group)
: This report shall contain the processor architecture and its instruction set architecture specification. Note: we will discuss you report on the 17/12 lab session. You will be allowed adjustments in light of the discussion.

January 20th Final report (group)
: This report shall contain the same elements provided in the two previous milestones (potentially updated if changes occurred), in addition to whatever is required to make your digital clock work. In particular, you should add:

  - The final version of your simulator
  - Your microprocessor
  - An assembler program targeting your processor
  - The assembly code of your digital clock
  - A script allowing to build everything and executing your clock in both real-time and fast-forward modes.

January 21th Project defense (group)
: You shall showcase your project comprehensively, including both real-time and fast-forward modes of your digital clock. Further details will be provided later on.

[3] You may add in the README.md a link to your project on github/gitlab/...