Assignment 1: Build and Extend a CPU

Implement the provided architecture for the QuAC ISA

A Digital Circuit

On this page

In this assessment you are going to complete a number of Digital files to a required spec, much like what you have already been doing in the labs.

> The first part of this assignment builds upon the following labs:
>
> - Lab 1: Introduction
> - Lab 2: ALU
> - Lab 3: Registers
> - Lab 4: Manua
> - Lab 5: Auto C
> - Lab 6: Condit

If you have not completed the tasks in the above labs or do not understand the content, we *strongly* recommend that you first complete the labs and then start the assignment.

Outline

- **Deadline:** 7 April 2024, 11:59 pm
- **Assessment template:** link
- **Specification:** keep reading 🙂

- **Weighting:** 20%
- **Marked out of:** _ / 100

Time Until Deadline (7 April 2024, 11:59 pm): 9:11:41:46

Rules and Policies

- this is an individual assessment task, ensure you fork your repo as **private**

- you may re-use designs / files from your labs
  - if you do, please cite them in the `statement-of-originality.md`
  - it is **your** responsibility to ensure any additional files are included in the repo and pushed to gitlab correctly
  - your main solution still needs to be in the provided files
  - all submitted files should abide by the style guidelines
- late submission is **not permitted** without an extension
- it is **your** responsibility to ensure that your changes have been submitted to gitlab, what is present **on gitlab** at the time of the deadline will be considered your submission

Allowed Components

You may use any component digital provides you, with the following exceptions:

- VHDL / Verilog / Anything outside of Digital
- Some extensions prohibit the use of certain components, check the extension page for details

Submissions that use any of the above will have marks deducted accordingly.

Resources

> If this is your first time reading the document then skip this for now and open links

> as necessary

This assignment has a few external pages that are linked throughout the document, for quick reference, here they are in one spot:

- QuAC ISA Definit
- QuAC Individual I ons
- Part 2 Extension
- QuAC Assembler
- Design Documen
- Digital Style Guide

Background

You have been building the CPU from the ground up during the first six labs. The first part of this assignment will require you to submit the CPU that conforms to the QuAC ISA.

The second part is more open-ended and will require you to extend the base ISA in an interesting way.

Specification

Part 1 (30 marks)

For the first part of the assignment, your task is to deliver a CPU in Digital that implements the full QuAC ISA specification.

> If you have already completed the first six labs that result in a CPU for the QuAC ISA, then congratulations! You have already completed the first part of the assignment... *sort of*. You still need to update the files in the assignment repo and ensure they are passing the tests there. We aren't going to refer to your lab repos, but it is fine for you to copy your components over.

> Complete `src/basic-cpu.dig` to the full QuAC ISA specification and ensure it is passing all provided tests.

> You `src/basic-cpu.dig` should contain **only** what is needed for Part 1. Your extension and further modifications will be completed in `src/extended-cpu.dig`

> . Students who further extend the `src/basic-cpu.dig` file will have marks removed accordingly.

Part 2 (70 marks)

Outline

The QuAC ISA is quite [illegible]bilities compared to established ISAs, such as ARM and RISC-V. We cannot [illegible]ting assembly programs to solve real-world problems. The primary [illegible]AC ISA include:

- Few logical and arithmetic operations
- Limited set of general-purpose registers
- ALU instructions cannot directly operate on immediate values
- Inability to conditionally execute on anything other than the zero flag
- Lack of a stack and dedicated `push` / `pop` instructions (this point relates to functions)

Your objective in this part of the assignment is to extend the QuAC ISA in a *meaningful* manner. To this end, you should note that the QuAC ISA deliberately has large gaps in the specification:

- Eight of the 16 possible operation codes (opcodes) are undefined. New instructions can thus use these opcodes and even define a new encoding format other than R-Mode or I-Mode.
- Bits `[3]` and `[7]` in the R-Mode encoding format are only defined if both are zero (allowing each R-Mode instruction to have three alternate meanings based on the values of bits `[3]` and `[7]`.)
- One of the register codes is unused, allowing for another special purpose register (check here to see which register codes are used).
- Bits `[4:15]` in the flag register are unused.
- The outcome/behaviour of writing to the flag register is undefined. This means that you could define writing to the flag register in the usual fashion (if you for some reason wanted to set/clear the flag bits directly) or you could add a write-only register that shares the same register code as `FL` (reading register code `101` reads from `FL`, writing to register code `101` writes to the new register.)

Your extension can broadly be whatever you like, but keep in mind that sophistication of the extension is part of the marking criteria.

Here is a list of potential extensions that you might like to consider, though if you have your own ideas, ask on Ed or run them by your tutor first to check for suitability.

> Your CPU with ISA extensions ***must*** be backwards compatible with the base QuAC ISA.
> This means that any changes you make must:
>
> - involve new instructions using currently undefined opcodes or
> - use undefined bits in existing instructions (or both)
> - if it interacts with existing instructions it must explicitly be an improvement that does not change the expected outcome from the base ISA

Extension Implementation and Testing

To be able to demonstrate and test the capabilities of your extension, you must make any necessary additions to the `quac.jison` assembler config file to add support for your extensions, such that anyone can use it on your submitted CPU.
Remember that the base ISA must still work, so you cannot change any existing instruction configs.

Once this has been completed, you must write **at least** one of either:

1. An assembly program, `demo.quac`, that demonstrates your extension. For more help on this, check out the assembler guide page.
2. Additional digital tests contained in your `extended-cpu.dig` file that demonstrates your extension.

The quality of your additional test(s) will contribute to your marks for the extended CPU.

Design Report

Your assignment will also require the submission of a **≤1000 words** design report describing the design decisions you made to implement the base ISA and your extension. More specifically, we would like to see:

- A high level overview of your extension
- What modifications you have made to the ISA
  - This section may be omitted if you are implementing an extension that does not require changes to the ISA. Alternatively, you can briefly explain *why* your extension does not require any changes to the ISA.
- The required changes to the microarchitecture
- Analysis of the tradeoffs / limitations with your implementation:
  - What are the key *benefit(s)* of your changes
  - What are some *limitations/tradeoffs* of your changes
  - Impact of your changes on the CPU
- A small explanation of the example program/additional tests you have included.

The `report.md` file in your assignment repository contains section headers for each of the above points.

> You will need to successfully implement a complex extension backed up by an excellent design report to get a mark in the HD range. See more here.

Your design report sho ribe your extension and its behaviour and encoding in a format s A description.

> Note that we say ≤ 1000 words. If you feel you have addressed all of what has been asked of you in less words, please don't feel the need to hit 1000 words. You will end up writing a worse report by reducing the conciseness of it.

> Complete `src/extended-cpu.dig` to the full QuAC ISA specification and then extend it based on the information provided above.
> Provide **at least** one assembly program `demo.quac` that demonstrates your extension (further files should be added and named accordingly, eg: `demo-stack.quac`).
> Finally, write a design report in `report.md` detailing your extension.

Deliverables

Outline

To successfully complete this assignment, the following files must be submitted:

1. `src/basic-cpu.dig` containing your CPU that implements the full QuAC ISA, but no extensions beyond that
2. `src/extended-cpu.dig` containing your CPU that implements the full QuAC ISA **and** has been extended beyond the base capabilities
3. `quac.json` containing the default quac instructions and any extra instructions, etc. that you have added
4. `demo.quac` containing the QuAC assembly program that demonstrates your extension
    - *optional* `demo-XXXXX.quac` containing further assembly programs that demonstrate other aspects of your extension
5. `report.md` containing your **≤1000 word** design report

- *optional* `assets/*` containing any images you want to add in your report

Additional Files / Sub-circuits

We provide the files `src/basic-cpu.dig` and `src/extended-cpu.dig` as templates that contain only the dual-port memory module. You should build the CPU(s) in these files. For any other additional sub-circuits (`alu.dig`, `reg_file.dig`, `control_unit.dig`) you create them in the `src/` directory. If your part-1 or part-2 use differing versions of sub-circuits, then please name them appropriately.

Marking Crit

Your assessment will be evaluated on the following criteria:

1. Correctness of your implementation of the base ISA (30/100)
2. Sophistication and Correctness of your extension of the base ISA (40/100)
3. Explanation and Reasoning of your extension of the base ISA (30/100)

Item 1 will be evaluated primarily through the results of your Digital tests and visual inspection.

Item 2 will be evaluated primarily through visual inspection of the Digital files and the assembly program demonstrations.

Item 3 will be evaluated through your `report.md` and the quality of the writing within.

Items 1 and 2 will *also* be evaluated by visually inspecting your submissions for neatness, efficiency and logical layout. For more information on how to lay things out well in Digital, check the style guide.

We will take the following aspects into account during the marking of the assignment:

- Functional correctness
  - Your CPU must be able to run assembly programs written in the base QuAC ISA correctly.
  - You must demonstrate the correctness of your extended CPU with a carefully written assembly program that uses the new extension. We will verify.
- Succinctness, clarity and neatness of CPU design and organization
  - No spaghetti wiring
  - Wise use of diagonal wires (seldom)
  - Avoid easily preventable wire crossings
  - Avoid running wires under other components
  - Take care that the data flow in typical combinational circuits is left-to-right
  - Appropriate use of comments to explain circuit sections
  - Sensible use of abstraction
    - Your design should be easy to understand, this includes the packaging of abstractions

    - For more info, check the style guide
- Modularity
    - How easy would it be to modify your CPU and/or add other extensions?
- Avoiding wasteful duplication of hardware
    - An ALU that is not using the same full adder for addition and subtraction is wasteful
    - Adding an i ion should not use another adder
    - If an instruc esised from existing instruction(s) (namely pseudo-instruction) hardware for the instruction is wasteful. The one exception is ware for the new instruction results in a more efficient ex ng existing instructions, i.e., the software-only approach.
- Gate minimisation
    - Your design does not need to be optimal, but large sprawling, easily minimizable circuits will be penalised.
- Testing
    - Modifications to the control unit and ALU should have corresponding additional tests added
    - New instructions should have some example code demonstrating how they work

Submission

Submission is through GitLab, the most recently pushed commit of your fork of the assessment template before the deadline is taken to be your assessment submission.

Getting Started

1. read this assessment page **completely**
2. fork and clone the assessment template
    - ensure you fork your project as private
    - do **NOT** change the name or path of the repo, or it may get missed by our software

3. work on each part, testing, committing and pushing as you go
4. make a mistake or get stuck, then ask a good question on the course forum.

Completion Checklist

- you have submitted the files listed above
- you have checked that you haven't used any of the components listed above
- you have run your files local test(s) and they pass successfully
- you have laid out all of your circuits using good practice
- you have saved, committed and pushed your Digital files to gitlab
- you have filled out, committed, and pushed your `statement-of-originality.md`
- you have filled out, committed, and pushed your `report.md`
- you have checked the report pdf artifact on gitlab to ensure it is correct
- you have checked the gitlab ci tests and they are passing

Report pdf Artifact

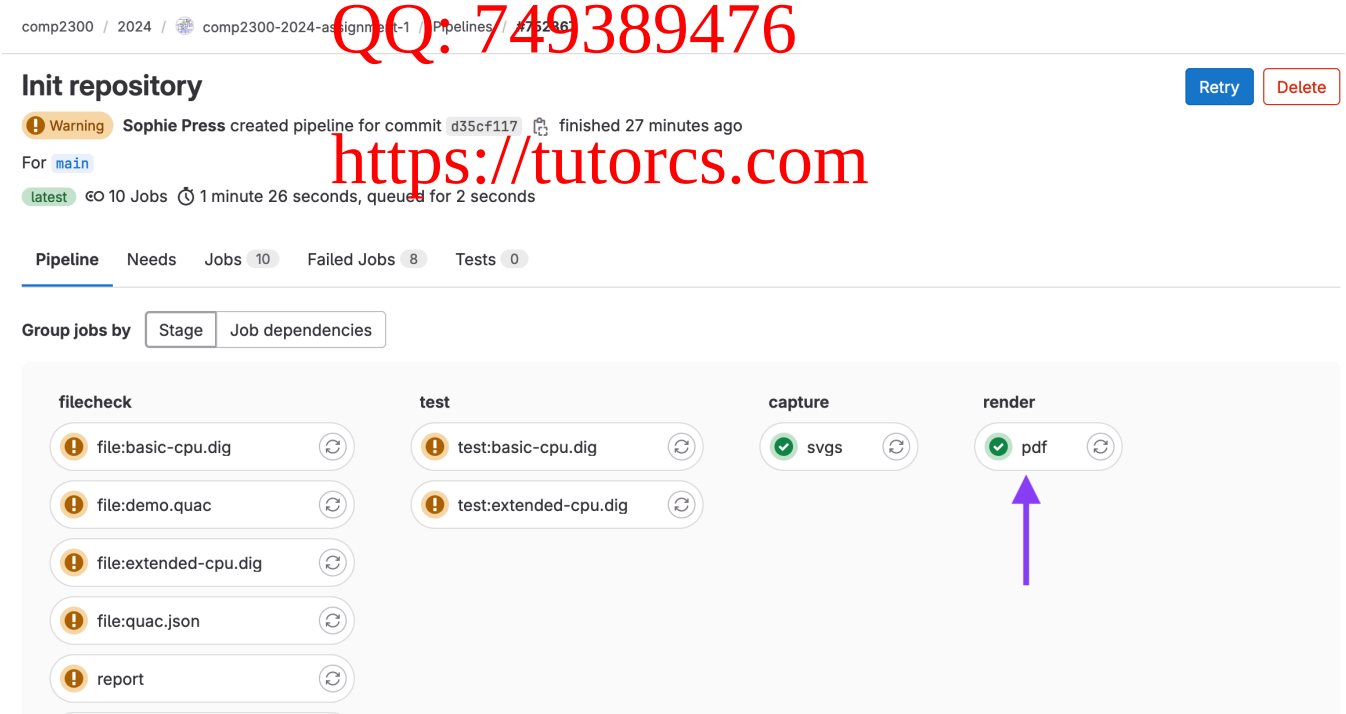
Your repo will be packaged into a report pdf for marking purposes. As such it is important that you see what the result of the `pdf` job is and make sure the output makes sense.

It will:

- take your name and uid from the `statement-of-originality.md`
- take images from your `basic-cpu.dig` and `extended-cpu.dig` files
- take test results of files
- take your report from `report.md`
- take references from the `statement-of-originality.md`

- take images from all *other* digital files
- combine all of them into a single pdf

To view the pdf, first click the cloud icon on your most recent commit (as above), then click on the `pdf` job.



Then, you'll be taken to the job page, where you should see a "Job Artifacts" section, click on the `Browse` button.

This will allow you to preview your pdf. Which should look something like this.

There is more general info about gitlab ci here.

FAQ

My circuits don't work, can I email you for help?

Sorry, you won't get help over email or Teams. We provide a course forum which is the **only** way we are able to help.

Forum posts related to your assessment submission **must** be “private” (as for any individual assessment task).

It’s [5 minutes, 60 minutes, 12 hours] before the deadline and my CI Jobs aren’t finishing!

Unfortunately on the d ment is due, when many students are pushing updates at once, the C p up. You may not see your CI jobs finish before the deadline. You will just heck that your files have been submitted correctly and that you are passi

The best way to avoid t early and finish early 😇

If there’s any issues with your git repository *after the deadline*. Please let us know (after the deadline) through a private forum post and we may be something we can do.

How do I know my assessment has been submitted?

If:

1. the files in *your fork* of the assessment are correct (i.e., the files you intend to submit) when checking on the **gitlab website**
2. the time is before the deadline

then your assessment has been submitted (well done!).

Please don’t ask us to “check”, we would be just doing exactly the same thing as the above steps which you can do yourself.

I’ve done X, is that good enough to get Y marks?

We don’t give out marks before assignments are completed, but it’s fair to wonder “How hard should I work on this?”

The extension page’s grade bands are a guide as to what levels of sophistication we expect in your extension for each grade level, but note that it is only a guide. Just attempting the extension is not enough. If you did not follow the specification correctly (e.g. for I/O extension your I/O components can’t be interacted with via instructions), it does not fully work, or your design report is very weak, you may drop a grade band. A solid submission is still required to achieve the target grade.

Note that submissions including more than one extension will be handled case-by-case. It could be that your second extension is of high enough quality to bring you up a grade band compared to just having the first, or it could not.

Sorry that we can’t be more specific than that.

Acknowledgement of Country

The Australian National University acknowledges, celebrates and pays our respects to the Ngunnawal and Ngambri people of the Canberra region and to all First Nations Australians on whose traditional lands we meet and work, and whose cultures are among the oldest continuing cultures in human history.