ARM11 - Emulator report

Seiya Aoki, Isabel Li, Jimin Park, Alex Liu

31st May 2019

1 Short Introduction to Project

Our group project was to write an emulator in C to simulate the execution of an ARM binary file. We undertook independent learning to utilise many of the features of the C language, such as bit masking operations.

2 Explanation of our Implementation

We represented our memory as an array of 8bit unsigned integers with size 65536 (which is equivalent to a 64KB memory), and our registers as an array of 32bit unsigned integers with size 17 and placed them onto the heap. We chose to place them onto the heap rather than the stack so that they could be used globally and since they occupied more space.

In our three stage pipeline we had a fetch, decode and an execute stage. However, we decided to load the contents of the binary file into an internal memory array to simplify the fetch process.

- Fetch would read contents of the memory, and fetch the first four bytes to return it as a 32bit integer. We assumed that the emulator would be only run on Little Endian Machines, and therefore we did not have to do any explicit reordering. However if it were run on Big Endian Machines we would have to implement a reordering of the four bytes to obtain the correct interpretation of the instructions.

- Our decoder takes an instruction represented as an unsigned 32bit integer and extracts specific bits using bit-masking in order to create a struct to represent the instruction. We used one struct to represent all the four instruction types with the fields: Instruction type, Cond, OpCode, Offset , registers and flags. Instruction type, Cond and OpCode were represented as Enums. Offset and Registers were represented as uint32t and all flags were represented as bools. These structures we implemented can be reused in the assembler stage.

- The executor would then take the decoded instruction as a struct, then work out what instruction type it was. Execution took place by calling one of the four execution functions specific to each instruction type.

Currently our file structure and code style is sub-optimal at the moment, but we plan to refactor and restructure our code in the near future.

3 Working as a group

On the first day we collectively decided to work through the tutorial questions in order to get a good understanding of C and to familiarise ourselves with the syntax. If any of us encountered problems, we would help each other debug.

We first discussed the project as a group to come up with ideas on how to optimally implement the emulator. This was useful as by considering each other's ideas we could individually deepen our own understanding and come up with better ideas together.

We split up the work based on our individual strengths. We collaborated on trickier functions and worked through easier ones alone. We aimed to merge frequently on Gitlab so that any conflicts did not accumulate. Once we finished writing all the functions and ran the testsuite, we found we had many bugs and segmentation faults. We then worked together to debug each test we failed until we passed all of them (except for tests gpio and loop01). Currently we have a timeout error for loop01, but testing locally yields the correct answer albeit taking slightly longer than the other functions. Therefore we will need to optimise our code so that it runs on slower machines.

We're good friends outside of the project and we had previously been a team at this year's IChack. We already had some confidence in our compatibility as a result.

We work well together as we're not afraid to ask each other for help or opinions, which we think is essential for any team who wants to efficiently work together.

4 Future

In the future, I think we need to become more formal when using Git, like giving meaningful and concise commit messages and Git branch names. Otherwise, We think that the current way our group is working is quite efficient. Currently we have made a lot of progress with the Assembler, so many of our concerns about Assembler have been mitigated. Members of our group haven't used a Raspberry Pi, so we may face some challenges in part iii of the project. We're slightly concerned about managing time between working on the extension and revising for the C exam, and also what we will do for the extension.