C Project Checkpoint

Priyansh Mahajan, Adam Maltby, Hamish McCreanor, Om Singh

June 2022

1 Work Allocation

In an effort to achieve a fair and balanced work distribution we allocated work to members based on the following criteria:

- The (estimated) length of time we thought the task might take.
- The pre-existing workload of each member.
- The areas of strength and weakness for each group member.

Following the "What To Do" section detailed in the specification, *Figure 1* describes the task split for each bullet point.

2 Group Performance

In the aspect of group work we were fortunate in our ability to meet as a group regularly due to convenient living circumstances. This meant collaboration took place almost entirely in person which meant communication was easy and unambiguous. We all feel that we worked effectively as a group and are happy with the allocation of tasks so far.
We believe the ordering of tasks could have been improved somewhat as we divided the work for the execution of instructions prior to the implementation of pipe-lining. This meant we were unable to debug each instruction as we were writing it, instead we had to integrate all of our functionality together and test at that point. In the future we will be more thoughtful as to the sequencing of our work.

3 Emulator Structure

The emulator is controlled in-majority from the emulate.c file that runs the fetch-decode-execute cycle and branches off to separate instruction files each time an instruction is executed.
Regarding re-use of code in the assembler phase, while the two tasks are quite different there are cases where code can be taken from the emulator phase and put to use with little modification. This includes: structs, enumerated types and word manipulation functions.

4 Future Challenges

From previous experience, as stated above, we think the order in which we tackle tasks could prove difficult to manage so that we are able to test effectively. This will be handled by spending more time analysing and discussing which tasks should be attempted in which order. In addition, our plans for the final extension part will require parts that we must source online. This could affect our workflow, so we are thinking ahead and aiming to procure such parts before we need them in order to eliminate bottlenecks in our progress.

Task	Allocated Member(s)	Explanation
Binary loader	Hamish, Om	Hamish and Om started by writing up their own versions of a binary loader separately and then came together to merge their solutions, extracting the best aspects of each solution.
Pipeline	Hamish, Priyansh	Hamish and Priyansh was tasked with implementing the pipeline while the other members implemented various instruction type executions.
Data processing execution	Priyansh	Priyansh was tasked with implementing the execution of data processing instructions.
Multiply	Adam	Adam was tasked with implementing the execution of multiply instructions.
Single-data transfer	Om	Om was tasked with implementing the execution of single-data transfer instructions.
Branch	Adam, Om	Adam and Om were tasked with implementing the execution of branch instructions.
GPIO	Adam, Om	Adam and Om tackled the functionality required for the GPIO pins. This involved a modification to the state struct data type.
Terminate	Hamish	Hamish added the termination functionality for the emulator. This involved writing a pretty print function for the state of the emulator.
Debugging	Adam, Hamish, Om, Priyansh	We all handled the task of debugging and fixing errors that arose in the test cases.

Figure 1: Task allocation table