Final Project Report

Team Members:

- Abhimanyu Singh Chaudhary - 2016003
- Hanit Banga - 2016040
- Vaibhav Goel - 2016111

Project Topic: ARM Simulator (Project 1)

Methodology/Project Plan:

The main task is decoding the hex instruction that is provided as the input. For that we have to study the structure of the binary instruction and figure out which parts correspond to what part of an ARM instruction.

Along with the binary structure, we have to know the various flags set by each instruction. In ARM, there are 4 flags and every instruction can set some flags based on the result of the computation. Further, each instruction can be conditionally executed based on the current values of the 4 flags.

After studying the structure, the next task is to compile a list of functions that we have to implement in this project, and figure out how they look in binary.

Once the list is compiled and we know the structure of binary code, we start implementing the function simulator in C++.

In the code, these are the main functions:

- Fetch: Fetches the instruction from main memory and updates the Program Counter.
- Decode: This is the most important functionality. It decodes the hex code to figure out what exactly we need to do, and sets global variables to tell the later functions what they need to do. This is mostly done by if-else blocks on the binary of the instruction.
- Execute: Depending upon the decoding of the hex code, this function simulates the ALU task that is to be performed
- Mem: Does all the interaction with data memory.
- Write back: Writes the final result, back into destination register after the execution stage.

Implementation:

The registers are an array of integers, the memory is an array of characters and the condition flags are boolean variables.

There are 5 main functions that are called for every line.

1. fetch() : fetch reads the input file and reads the instructions line by line and updates program counter.

2. decode() : decode first checks if the instruction is an swi instruction. If it is not an swi instruction, the condition flags are then extracted. Then we determine what kind of instruction it is. We use multiple if blocks to find the instruction type and give correct value to opcode and ALUop in case of arithmetic operations.

3. execute() : Carries out the respective arithmetic operation decided by the opcode and stores then in a variable. It also updates the condition flags.

4. mem() : If the instruction is LDR or STR then this function reads value from memory or writes value to memory, according to the opcode extracted in decode function.

5. write_back() : This is the final function, it writes the variable that has the answer computed in execute function to the destination register.

There are helper functions like:

1. read_word : reads one word from memory.

2. write_word : writes one word to memory.

3. get_bits : get bits in a range from an integer variable

Bonus:

Made a working GUI that shows register values at each step and a step_into feature.

Timeline:

15th -17th October: Read about the binary structure of instructions in ARM, and which bits correspond to what.

18th -19th October: Compiled a list of all arm functionalities covered in class and their respective binary/hex codes.

21st -22nd October: Started and finished implementing the ARM simulator in C++.

23rd October: Tested the simulator on various samples, including the two listed in the design document.

24th -29th October: Update the README and design document, provide extensive documentation of the code.