**ARM SIMULATOR**

In this project, we have created main.c which is running myARMSim.c, with one command line argument when executing.

All the commands are read from the .mem file which is passed as command line argument.

First we load memory with all the instructions from the input file.

In fetch:

We read address and corresponding instruction from the file and store in variables and increase the program counter

We divided the instructions in three types of instructions:

For F=0: Data Processing instruction

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Cond | F | I | Opcode | S | Rn | Rd | Operand 2 |
| No. of bits |  |  |  | 4 | 2 | 1 | 4 | 1 | 4 | 4 | 12 |

For F=1: Data Transfer instruction

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Cond | F | Opcode | Rn | Rd | Offset |
| No. of bits | 4 | 2 | 6 | 4 | 4 | 12 |

For F=2: Branch Instructions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cond | F | Opcode | Offset |
| No. of bits | 4 | 2 | 4 | 24 |

1. Move- puts the value given in the destination register

2. Add- used to give result of addition of the operand 1 and operand 2.

3. Subtract- used to give result of subtraction of the operand 1 and operand 2.

4. Compare- used to compare two values and then update the negative, zero, carry and overflow flags according to the value of operand 1 and operand 2.

5. Load/store- used to load/store a value from/to a register.

6. Exit- Takes us to the end of the program