ARM SIMULATOR

In the program we have created, we start our program by running main.c which has myARMSim.c and myARMSim.h already included and takes 2 command line arguments. If it is less than 2 then our program exits after printing "Incorrect no. of arguments". If the arguments are 2 then our program first initializes all registers to 0 and also memory is also reset.

All the commands are read from a file whose extension should be .mem and the commands are encoded in the hexadecimal format.

In fetch, we are reading a word from program memory and saving the content to an array. Also we increase the program counter after that.

**For F=0**

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

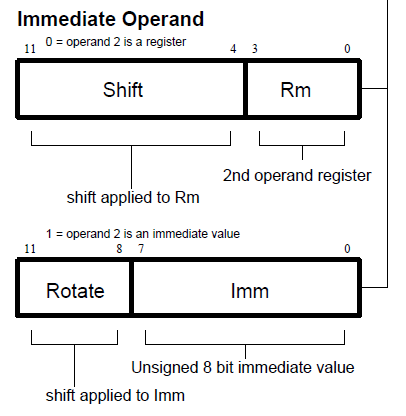
**For F=1**

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

**For F=2**

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

After reading the content from mem file the program need to understand what is that command all about. This is done in the decode part. In our code we have done "bitwise &" and then shifted it right with commands so that we are able to extract the desired information from the command. For each command we read we have extracted its condition, format, immediate, opcode, set, operand 1, destination register, operand 2 and offset. In operand 2 too there were two cases, i.e. with immediate and without it.

[](http://emucode.blogspot.com/2010/09/decoding-arm-instruction-set.html)

Based on the opcode value we perform the desired operations in the execute stage. Following operations can be used in the arm simulator:

1. Move (opcode 0x1101) - puts the value given in the destination register

2. Add (opcode 0x100) (program checks if there is immediate or not and operates accordingly) - used to give result of addition of the operand 1 and operand 2.

3. Subtract (opcode 0x0010) (program checks if there is immediate or not and operates accordingly) - used to give result of subtraction of the operand 1 and operand 2.

4. Compare (opcode 0x1010) - 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 (opcode 0x1100) (further if set is 1 its load and if set is 0 its store) - used to load/store a value from/to a register.

6. Exit (opcode 0x11) - Takes us to the end of the program

There is a memory stage in which we store the data back in the memory.

Also there is a write back stage in which we write the data to the register.

We have a function called “read\_word” which takes memory address as input and takes data from that address.

Also there is a function called “write\_word” which takes the data and address in memory as input and puts the data at that address.