ARM Simulator

Design and implement the function simulator in C/C++/Java for subset of ARM instructions discussed in class.

Project Goal

The objective of this project is to design and build an ARM Simulator. The project should be written in C/C++/Java, which would read the instruction from instruction memory, decode the instruction, read the register, execute the operation, and write back to the register file.

The instruction set supported is same as given in the lecture notes. (MOV, ADD, SUB, LDR, STR, etc). Basic SWI instructions should be implemented: Read, Print, Exit etc (Read should be console input in C/C++/Java)

The execution of instruction continues till it reaches instruction “swi 0x11”. In other words as soon as instruction reads “0xEF000011”, the simulator stops execution.

All the instructions in the given in the input MEM file are executed as per the functional behavior of the instructions. Each instruction must go through the following phases:
- Fetch
- Decode
- Execute
- Memory
- Writeback

The simulator also prints messages for each stage, for example for the third instruction above following messages are printed.

- Fetch prints:
  - “FETCH:Fetch instruction 0xE3A0200A from address 0x0”
- Decode
  - “DECODE: Operation is ADD, first operand R2, Second operand R3, destination register R1”
  - “DECODE: Read registers R2 = 10, R3 = 2”
- Execute
  - “EXECUTE: ADD 10 and 2”
- Memory
  - “MEMORY:No memory operation”
- Write-back
  - “WRITEBACK: write 12 to R1”

Input/Output format

Input consists of a single .MEM file containing the the encoded instruction and the corresponding address at which instruction is supposed to be stored, separated by space. For example, a simple program to add two numbers may look like:

```
0x0 0xE3A0200A
0x4 0xE3A03002
0x8 0xE0821003
0xA 0xEF000011
```

Sample input:

```
0x0 0xE3A0200A
0x4 0xE3A03002
0x8 0xE0821003
0xC 0xEF000011
```

Sample output:

```
Fetch instruction 0xe3a0200a from address 0x0
DECODE: Operation is MOV, First Operand is R0, immediate Second Operand is 10,
Destination Register is R2.
Read Registers: R0 = 0
EXECUTE: MOV 10 in R2
MEMORY: No memory operation
WRITEBACK: write 10 to R2

Fetch instruction 0xe3a03002 from address 0x4
DECODE: Operation is MOV, First Operand is R0, immediate Second Operand is 2,
Destination Register is R3.
Read Registers: R0 = 0
EXECUTE: MOV 2 in R3
MEMORY: No memory operation
WRITEBACK: write 2 to R3

Fetch instruction 0xe0821003 from address 0x8
DECODE: Operation is ADD, First Operand is R2, Second Operand is R3, Destination
Register is R1.
Read Registers: R2 = 10, R3 = 2
EXECUTE: ADD 10 and 2
MEMORY: No memory operation
WRITEBACK: write 12 to R1

Fetch instruction 0xef000011 from address 0xc
MEMORY: No memory operation
EXIT:
```

Simulator Flow:

There are two steps:

1. Memory is loaded with input memory file.
2. Simulator executes instruction one by one.

The simulator exits only when the instruction sequence reads “SWI 0x11”. The implementation of fetch, decode, execute, memory, and write-back function should be implemented as separate functions in C/C++/Java.

FETCH: Reads from the instruction memory and updates the instruction register

DECODE: Reads the instruction register, reads operand1, operand2 from the register file, decides the operation to be performed in execute stage

EXECUTE: Executes the ALU operation based on ALUop

MEM: Performs the memory function

WRITE-BACK: Writes the results back to register file

Extra ideas for interested students(might be criteria for bonus):

- Heap functions(malloc and free) using SWI
- Anything else that you like.