COMPUTER ORGANIZATION

PROJECT REPORT

Names of the team members:

Kaustav Vats - 2016048
Meghna Gupta - 2016056
Shravika Mittal - 2016093

25.10.2017

PROJECT TOPIC

Project - 1: ARM Simulator

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

METHODOLOGY

1. Go through the instruction set architecture of ARM Assembly from the book - Computer Organisation and Design - ARM Edition by David A. Patterson and John L. Hennessy.
2. Write a basic program for reading a file in C
3. Making separate FETCH, DECODE, EXECUTE, MEMORY and WRITE-BACK functions and testing them for the instruction set taught in class
4. Attempt for bonus by expanding our ARM Simulator for more instructions.

PROJECT PLAN

ABSTRACT

We need to replicate the functionality of an ARM Simulator which would read a memory file that contains the various instruction formats and memory addresses for a ARM Assembly code. After this, the ARM Simulator fetches the instruction from instruction memory, decode the instruction, execute the specified operation, reads the Data Memory if necessary and then writes back to the register file.

The ARM Simulator prints the messages about all the 5 stages in an execution process for the current instruction. The stages are:

1) Fetch
2) Decode
3) Execute
4) Memory
5) Writeback

This process goes on till the execution comes to an end, i.e. witnesses a swi 0x11 command.

So, basically our ARM Simulator is providing us with the entire flow of the program.

STEPS INVOLVED

1. Initial Database Setup
    a. Declare a global array for all the R0-R15 registers initialized to 0 which would be later updated as per the instructions.
    b. Declare a global memory array initialised to 0 for fetching the instructions at a particular location in the instruction memory.

    c. Declare a array for all the ARM commands that are being supported by our ARM Simulator and make another helper array that stores the corresponding opcodes of the commands.
2. Implementing Stage Specific Functions
    a. Make a function for each stage involved in the execution process - Fetch, Decode, Execute, Memory Read, Write-Back
    b. **Fetch** - R[15] is the PC counter in ARM Assembly. We would add the base address of our instruction memory array to this in order to get the particular instruction at that location. PC counter is also incremented by 4.
    c. **Decode** - This function extracts the opcode, operator and the respective operands Rn and Operand 2. There would be multiple cases for the type of instruction on the basis of their opcodes. Exit condition would also be handled in this function.
    d. **Execute** - This function performs the operation according to the extracted opcode and then updates the program counter accordingly.
    e. **Memory** - Since we know that all instructions exception logical and arithmetic involves memory, therefore this operation is executed only for these instructions also on the basis of the extracted opcodes.
    f. **Writeback** - This would print whatever changes that occur to the register file because of the instruction execution depending on the opcodes and also updates our global register array..
3. Input File Handling
    a. Read the input text file line by line and split about space, to obtain the address and the instruction.
    b. The global Memory array would be populated with the instructions at indexes defined using their addresses.
4. Calling all the functions defined

COMMANDS SUPPORTED

S NO.	TYPE OF INSTRUCTION	NAMES
1.	ARITHMETIC	ADD, SUB, MUL
2.	DATA TRANSFER	LDR, STR, LDRH, STRH, LDRB, STRB, SWP, MOV
3.	LOGICAL	AND, ORR, MVN, LSL, LSR, CMP
4.	CONDITIONAL BRANCH	B (NE, NQ, LT, LE, GT, GE, AL)
5.	UNCONDITIONAL BRANCH	B, BL

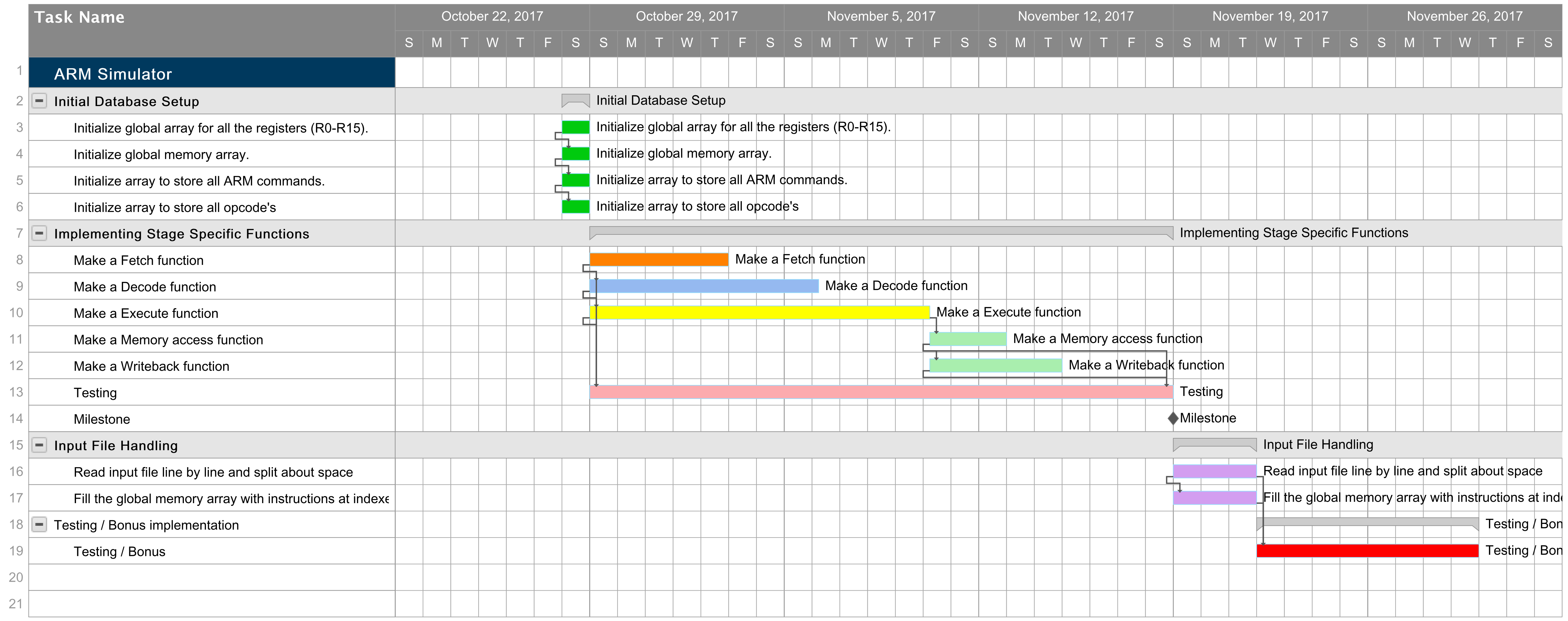
PROJECT TIMELINE

**Gantt Diagram on the last page.*

REFERENCES

1) ARMSim User Guide - http://armsim.cs.uvic.ca/AttachedFiles/ARMSim_UserGuide4Plus.pdf
2) Computer Organization and Design ARM Edition: The Hardware and Software Interface - David A. Patterson, John L. Hennessey
3) ARM Instruction Set resource - http://simplemachines.it/doc/arm_inst.pdf

Arm Simulator Project



	Task Name
1	ARM Simulator
2	Initial Database Setup
3	Initialize global array for all the registers (R0-R15).
4	Initialize global memory array.
5	Initialize array to store all ARM commands.
6	Initialize array to store all opcode's
7	Implementing Stage Specific Functions
8	Make a Fetch function
9	Make a Decode function
10	Make a Execute function
11	Make a Memory access function
12	Make a Writeback function
13	Testing
14	Milestone
15	Input File Handling
16	Read input file line by line and split about space
17	Fill the global memory array with instructions at indexe
18	Testing / Bonus implementation
19	Testing / Bonus
20	
21	