ARM SimSim

Abhishek Gupta, 2016004
Yajur Ahuja, 2016121
Brihi Joshi, 2016142

October 30, 2017

Abstract

ARM or the Advanced RISC Machines and typically require fewer transistors than those with a complex instruction set computing (CISC) architecture (such as the x86 processors found in most personal computers), which improves cost, power consumption, and heat dissipation.

ARM instructions are difficult to decode manually and so the ARM Simulator that we would be defining will cater to Fetch, Decode, Execute, Memory and Writeback the instructions. The code for the same would be written in **Python**

Methodology

The methodology that we would be following for implementing the ARM Simulator has been divided into the various steps an instruction execution has to follow. They are subdivided as follows:

- Fetch - The Fetch instruction reads from the instruction memory and updates the instruction register

  Steps:

  1. Select the file from which instructions have to be read.
  2. Place file pointer at the beginning of file.
  3. Read each instruction along with its address.

- Decode - The Decode instruction reads the instruction register, reads operand1, operand2 from the register file, decides the operation to be performed in execute stage

  Steps:

  1. Convert the hex instruction fetched in last step to binary.
  2. Identify the opcode and the operation it stands for.
  3. Identify and read the registers involved.
  4. Identify and store the immediate values involved.

- Execute - The Execute instruction executes the ALU operation based on ALUop

  Steps:

  1. Perform the operation identified in decode step.
  2. Get the result of performing the said operation.
  3. Store the result in cache(temporary variable).

- Mem - The Mem instruction performs the memory function

  Steps:

  1. If the operation is load/store access the memory address identified and load or store as necessary. Store this value in cache.
  2. If the operation is not load/store skip this step

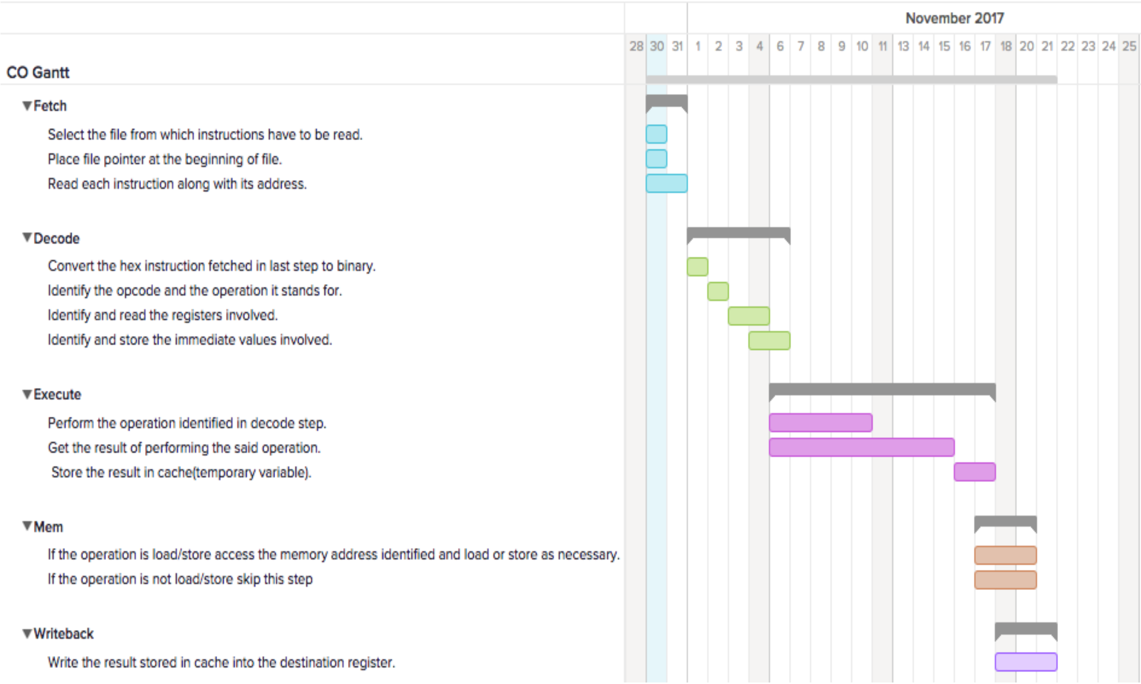
- Writeback - The Writeback instruction writes the results back to register file

  Steps:

  1. Write the result stored in cache into the destination register.

Project Plan

Gantt timeline



CO Gantt

▼Fetch

Select the file from which instructions have to be read.

Place file pointer at the beginning of file.

Read each instruction along with its address.

▼Decode

Convert the hex instruction fetched in last step to binary.

Identify the opcode and the operation it stands for.

Identify and read the registers involved.

Identify and store the immediate values involved.

▼Execute

Perform the operation identified in decode step.

Get the result of performing the said operation.

Store the result in cache(temporary variable).

▼Mem

If the operation is load/store access the memory address identified and load or store as necessary.

If the operation is not load/store skip this step

▼Writeback

Write the result stored in cache into the destination register.

Figure 1: Gantt timeline for our project

Constraints

The constraints in our project are as follows:

- The ARM Simulator will be designed to implement only a subset of the instructions that are present.
- ’Step into’ functionality would not be present.