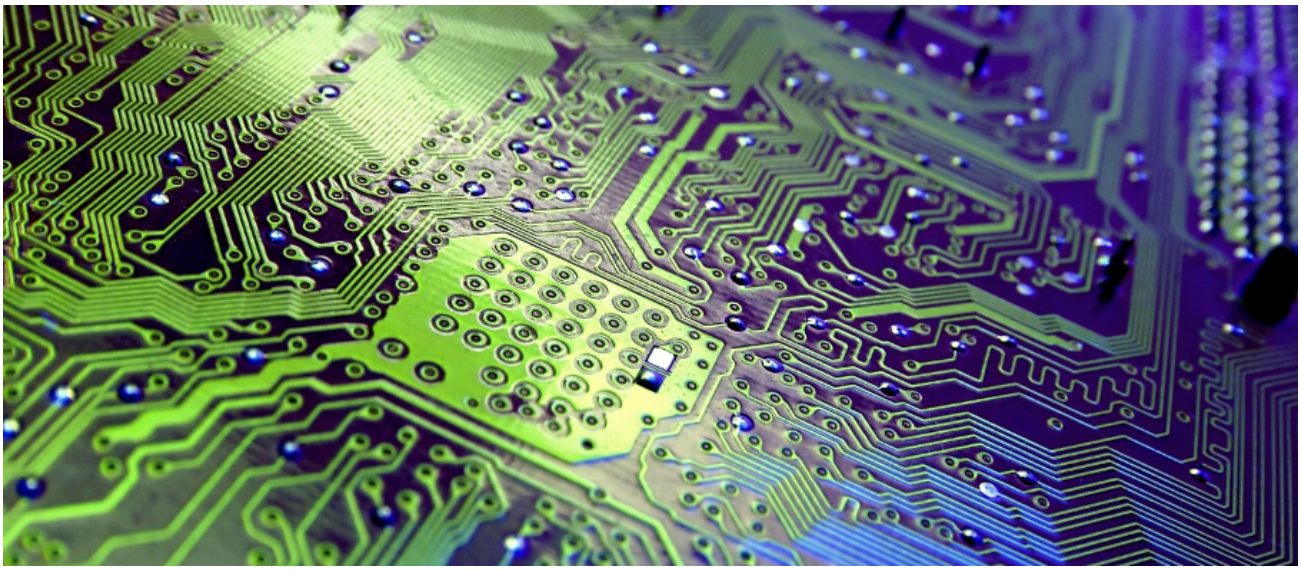
CSIS0801 FINAL YEAR PROJECT

PHASE 1 DELIVERABLE

PROJECT PLAN

TOPIC: COMPUTER SYSTEM SIMULATOR



WONG JING HIN

UID: 3035051060

Project Background

Current Situation

Computer Organisation is a core course for undergraduates of Computer Science Major, which covers the study of operational units of a computer that are involved in Instruction Execution Cycle (e.g. CPU) and their interconnections. There are several kinds of those operational units which further contains a number of components in each of them, for example in a CPU, there are different types of registers as well as ALU inside.

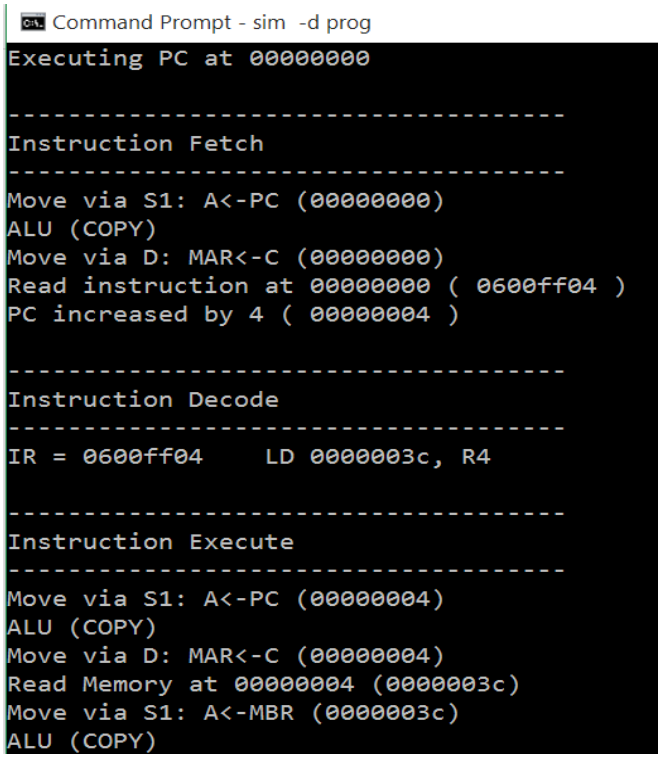
In an instruction execution, there are lots data movements among different components and transformations within components. Moreover, components are connected with each other differently, forming a complicated architecture of a computer system. Therefore, lots of undergraduate students find the flow and relevant concepts difficult to understand just based on verbal descriptions by lecturer and on lecture note or textbooks.

Existing Teaching Aids

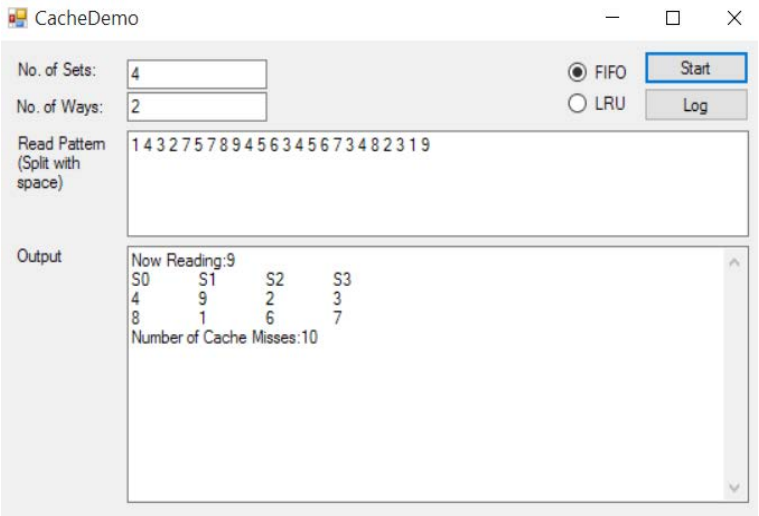
In order to explain the concepts and the data flow among components of computer system, there are 2 mini simulators available as teaching aids (possibly jointly developed by the lecturer and tutor) for the Computer Organisation course.

The first one is a program written in C++ programming language which simulates the data flow in Control Unit (CU) and Arithmetic and Logic Unit (ALU) of CPU. This program has to be run in command-line interpreter with a binary instruction set text file as input, the sequence of data movement and transformation of every instruction will be displayed as pseudocode-like descriptions on the console output.

The screen capture on the right is the sample output of this CPU simulating program:



Another one is a cache memory simulator written in C#. It is a small program simulating mapping and replacement algorithm of cache memory with Graphical User Interface (GUI). The screen capture on the right shows a sample run of this cache memory simulator:



Problems and Proposed Solution

There are several major drawbacks of the above existing teaching aids. Firstly, the two operational units of computer system architecture are simulated separately, it failed to show full and real picture of the operation of an instruction set in the computer system. Also, for the CPU simulating program, all configurations such as memory size, register file size and operation representing code (e.g. 00000101 as MOV operation) are "hardcoded" in the program source code, direct amendments on the program code is needed if we want to perform the simulation in different configurations. In addition, for the first simulating program, although the data movements and transformations are shown in sequence, it is shown in form of "sentences" in command line prompt, not only the full picture of the CPU operation cannot be shown (as only content of affected components are displayed), command line output format is also unattractive to read. Moreover, the output has low readability due to command-line form display, there are chunks of statements displayed on the screen for every instruction execution, which contributes a reducing clearness on the illustration of the flow and relevant concepts, thus students still find it difficult to learn the computer system operations even with these teaching aids.

As a "user" using these teaching aids before, I appreciate the efforts of developing such programs but unfortunately it has to be admitted that these programs are not effective enough in aiding students to learn computer organization (still they are good references). Thus, developing a new integrated computer system simulator which can cover more operational units, more functionalities and with greater flexibility in configurations is undeniably desirable and meaningful.

Project Objective

The primary objective of the project is to develop a simulator for a computer system based on a simple instruction set that simulates the instruction execution process, cache memory and memory hierarchy for teaching purposes, with the following features:

Interconnection of different Operational Units

The data movements and transformations occurring in the processors, cache memory and memory hierarchy (main memory & storage) will be covered in instruction set simulation of the simulator. Users will be able to see the changes of all these operational units in the execution and the interactions among them.

Graphical User Interface (GUI)

One major drawback of the existing teaching aids programs are its command-line type display, thus the proposed simulator of this project will be developed with GUI. Components of the processor (e.g. registers, program counter and stack pointer), cache memory and memory hierarchy will be displayed on the screen in form of graphic. There are 3 benefits of GUI for the simulator, firstly the structure of the computer system and interconnections among components can been shown clearly; also the data movements and transformation in different components and instruction execution can be more effectively illustrated; students will also be able to view the full picture of instruction set execution process easier than before as the whole relevant architecture is displayed in the output screen, they can see which components are affected and which are not affected.

Flexible Configuration

Configuration will be independent of the core program source code. When a user runs the simulator, the simulator will automatically read the external configuration file to set necessary configuration for simulation, after that it will receive user's input of the storage path of the instruction set binary file for the simulation. Users only need to amend the configuration file and then re-execute the simulator program if they want to do the simulation in different configurations.

Further Enhancement

In real execution, some instructions may be able to be processed in parallel in order to shorten the processing time, the simulator can be enhanced to cater this and specify the execution order of instructions in the instruction set.

Project Methodology

Phased methodology will be adopted for this project.

The system architectural design of the simulator will be complicated as it covers not only one operational units in computer system, which contains sub-components and interrelates among themselves differently. At the same time, CPU is the core part of instruction set execution process, the functionalities of the CPU part should be completed before integrating further operational units to the simulator. Therefore, the development of the simulator should be carried out in a unit-by-unit approach, and so phased development methodology will be best-fit for this situation. The main advantage of phased development is that it allows developers to put focus on each feature one by one that scopes are well defined under each phase, hence the development burden can be eased and quality of each feature of the system can be ensured.

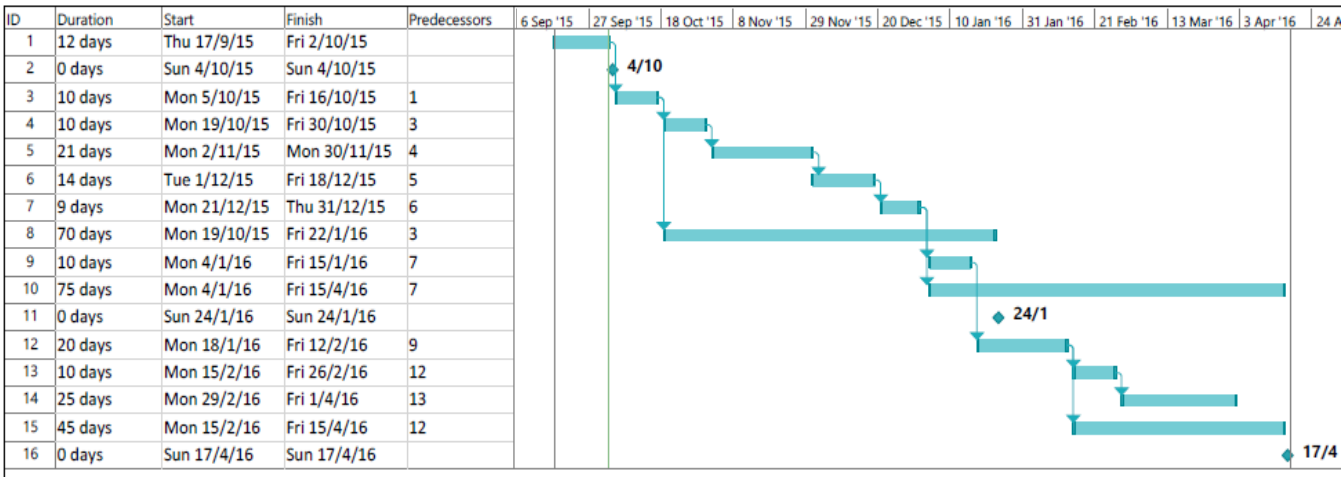
The development cycle of the simulator will be categorized into 3 phases. The first two phases will be the design and development of operational units, the order of design and development will be in the other way round of the accessing order of operation units in instruction set execution (e.g. if accessing order is: cache memory -> CPU, then development order will be: CPU -> cache memory), the latter phase will be extension of the deliverable from previous phase development. The final phase will be further enhancements for the simulator as mentioned in the previous part. Moreover, for each phase, it will be further divided into 2 sub-phases, the development will be in command-line output first, when the output for the 1st sub-phase is correct (i.e. the logic flow implementation of the part is correct), the output format will be transformed into GUI form as the 2nd sub-phase development.

Project Schedule and Milestones

The tentative schedule of the project is shown in the following table:

ID	Task Name	Duration	Starting Date	Predecessors
1	Requirements Gathering & Analysis	2 Weeks	17/9/15	
2	**Milestone: Project Plan Submission**	**N/A**	**4/10/15**	
3	Preliminary Study & Research	2 Weeks	5/10/15	1
4	Phase 1 (CPU) Analysis & Design	2 Weeks	19/10/15	3
5	Phase 1 Coding & Testing I	4 Weeks	2/11/15	4
6	Break for Preparing Final Examinations	3 Weeks	1/12/15	5
7	Phase 1 Coding & Testing II	2 Weeks	21/12/15	6
8	Documentation for Phase 1	14 Weeks	19/10/15	3
9	Phase 2 (Cache Memory) Analysis & Design	2 Weeks	4/1/16	7
10	Documentation for Phase 2	15 Weeks	4/1/16	7
11	**Milestone: Interim Report Submission**	**N/A**	**24/1/16**	
12	Phase 2 Coding & Testing	4 Weeks	18/1/16	9
13	Phase 3 (Enhancements) Analysis & Design	2 Weeks	15/2/16	12
14	Phase 3 Coding & Testing	5 Weeks	29/2/16	13
15	Documentation for Phase 3	9 Weeks	15/2/16	12
16	**Milestone: Final Report Submission**	**N/A**	**17/4/16**	

Project Gantt Chart:



End of Project Plan