Design Note

Source code for this simulator is on github, you can access it by clicking the following link: https://github.com/ya-gao/ComputerSystemSimulator

We program this simulator in Java using JDK 12. There are six packages in our project, which are cs, Memory, Registers, Instruction, Utilities and program. Following are the detailed descriptions of each package.

CS: The simulator class in this package implemented the user interface. Implementation of simple pipelining is included. For more details on how to use the simulator and what happens when you click on the button, please refer to the Section 2, User Guide.

Memory: There are two classes in this package, called Memory and Cache. Memory is represented by an array of size 2048. The data in instances of memory array are represented in the format of decimal. Cache is comprised of several 16-bit long cache lines, and cache line is implemented using linked list. We implement an add function to add elements into cache.

Registers: A class called Registers is implemented in this package. The constructor of this class initialize all the registers with random values, which are in the format of string. The register number of general purpose registers and index registers are decimal numbers. For each type of register, functions set<RegisterName> and get<RegisterName> are implemented to read and write data from and into the designated registers. The PC value is started from 06 in our simulator.

Instruction: In this package, we separated instructions into different classes based on the type of instruction. There are seven classes in this package, called Instruction, LoadStore, Logic, Arithmetic, Transfer, ShiftRotate and IO_Operation, and each class handles the execution for a specific type of instruction.

Utilities: There are three classes implemented in this package, called DataTypeConvert, Decode and EffectiveAddress. DataTypeConvert deals with issues about data, like convert between different data types and adding padding zeros. Decode acts as a media between cs and Instruction packages to decode the input instructions. EffectiveAddress is used to calculate the effective address.

program: Program package includes program1.class and program2.class. In program1.class, program1.txt will be read in and sent into our simulator. In program2.class, program2.txt will be read in and sent into our simulator.

The basic workflow of our project is CS.Simulator -> Utilities.Decode -> Instruction.XX -> CS.Simulator. First initialize the front panel, registers and memory, and take in some instructions. Then pass the instruction into Utilities.Decode to decode the instruction. The decoded instruction will be passed into Instruction.XX according to the type of instruction. In this process, Registers and Memory packages will be used when needed (eg. retrieving or updating content). After doing some changes to memory and registers, update the associated values to the front panel.