## Assignment 5

Due: Dec 15th, 11:59 pm
Weight: 10% of total class grade

## Overview

In this assignment, students will add an Input/Output device that writes to/from memory. Sample programs will demonstrate the problems of maintaining memory & cache coherency.

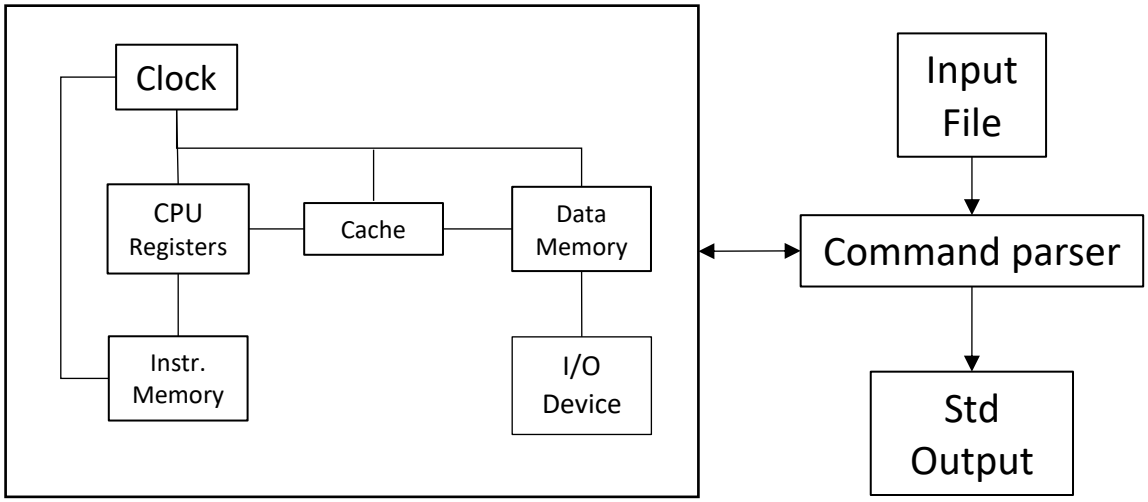
## Program Execution

cs3421_emul <data_file>

## Input Details

The cs3421_emul program reads an arbitrary number of lines, one line at a time from the data file. Each line will begin with a device identifier. The format of the rest of the line will be device dependent. For this assignment, possible devices are “clock”, “cpu”, “memory”, “imemory”, “cache”, and “iodev”.

## Devices




### Clock

The clock device is unchanged from Assignment 4.

### Memory

The memory device is unchanged from Assignment 4.

### Instruction Memory

The instruction memory device is unchanged from Assignment 4.

### CPU

The CPU device is unchanged from Assignment 4. Note that in Assignment 4, an “addi” instruction used an immediate value of 254. This was incorrect. In this assignment, a value of “-1” is being used, which is stored as 0xFF in an 8 bit two’s complement system.

### Cache

The cache device is unchanged from Assignment 4.

## Input/Output Device

The Input/Output (I/O) device has a single 8 bit register, and can read/write directly to Data Memory. The device gets loaded with a "schedule" of when it reads or writes to memory. The contents of the register can be dumped. Only read operations modify the I/O device register. Writes go directly to memory.

### *Reset*

The "reset" command causes the I/O Device to be reset, and sets the 8 bit register to zero.

### *load <filename>*

The "load" command takes the name of a file that contains a list of "I/O schedule events" (no more than 100). An I/O schedule event is as follows:

<clock tick> <read|write> <address> [value]

The "clock tick" is a base 10 number that indicates on what clock tick the operation is to begin execution. Ticks will always be increasing in value. The "read" or "write" indicate the type of operation to perform on Data Memory. The "address" is a hex value that indicates the address in memory on which to operate. The "value" is hex number included only with the "write" operation, and indicates the data to be written to memory. For example:

6 read 0x10
16 read 0xA0
23 write 0xEE 0x27
30 read 0xEE

Reads and writes to memory will take the usual 5 clock ticks to complete. The programmer may assume sufficient time between I/O schedule events is given for each to complete. The programmer may also assume that only a single device is ever accessing Data Memory at a time. Note the CPU may be access cache while the I/O device accesses memory.

### *dump*

The "dump" command prints the contents of the register in hex, followed by a blank line. With a register contents of 0x23, the command "iodev dump" would produce the following:

```
IO Device: 0x23
```

## Language

The developer may use either Java or C/C++.

## Submission Details

Submission will be via Canvas. Create a directory matching your username, and place all files you plan to submit with that directory. To submit your assignment, create a zip file of that directory (<your_username>.zip), and submit the zip file via Canvas. You should verify that the zip file has the correct directory structure, meaning when extracted without any special options, it WILL create a directory corresponding to the username of the student.

The zip file for the assignment (<your_username>.zip) should contain the following files:

<your_username> - directory corresponding to your username which holds all files
    Author.txt – a text file with a single line containing your first & last name

Language.txt – a single line containing the word “C” or “JAVA” (all upper case, no quotes) indicating the language used for the assignment
Readme.txt – Any comments you’d like to share regarding the submission
Makefile – an optional file that will compile your code into the cs3421_emul executable
cs3421_emul source & header files necessary for your project

**Note, if you have known issues with your program, DOCUMENT THAT in the Readme.txt file.** Any use of non-standard libraries or packages should also be documented, as well as how to retrieve & install them.

## Grading

Programs will be primarily graded on the correctness of the output, **and ability to follow submission guidelines**. Given the large number of submissions, the evaluation of the output will be automated. This means it is **CRITICAL** to follow the sample output above. Sample data & output will also be provided to test your programs. The assignment grade may also be based on style, comments, etc. Programs that crash, fail to produce the correct output, or don’t compile/run may lose up to 100% of the points, depending upon severity of the error. Some effort may be made during grading to correct errors, but students should not depend upon this. For grading, programs will be compiled and run on Ubuntu Linux 18.04 LTS.