CSIS0801 Final Year Project
First Presentation

Project Title: Computer System Simulator

Supervisor: Dr. K.P. Chan

Presented By: Wong Jing Hin, Kent

UID: 3035051060

Agenda

- Background
- Objective
- Features
- Progress & Demonstration
- 2nd Semester Schedule

Project Background

- Computer Organisation
  - CS undergraduates Core Course
  - Study of operational units of Computer involved in Instruction Execution Cycle & their interconnections

Project Background

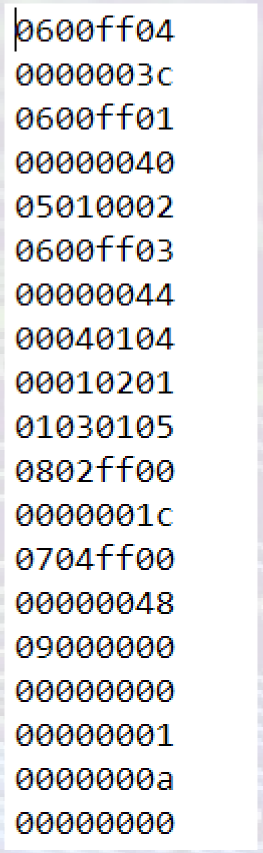
- In Instruction Execution:
  - Lots of data movements & transformation
  - Components connected with each other differently
    - Forming complicated architecture
- Student find flow & concepts difficult to understand

Existing Teaching Aids

- 2 Mini Simulators
  - Instruction Execution Simulator
  - Cache Memory Simulator
- Act as a based reference for this project

Instruction Execution Simulator

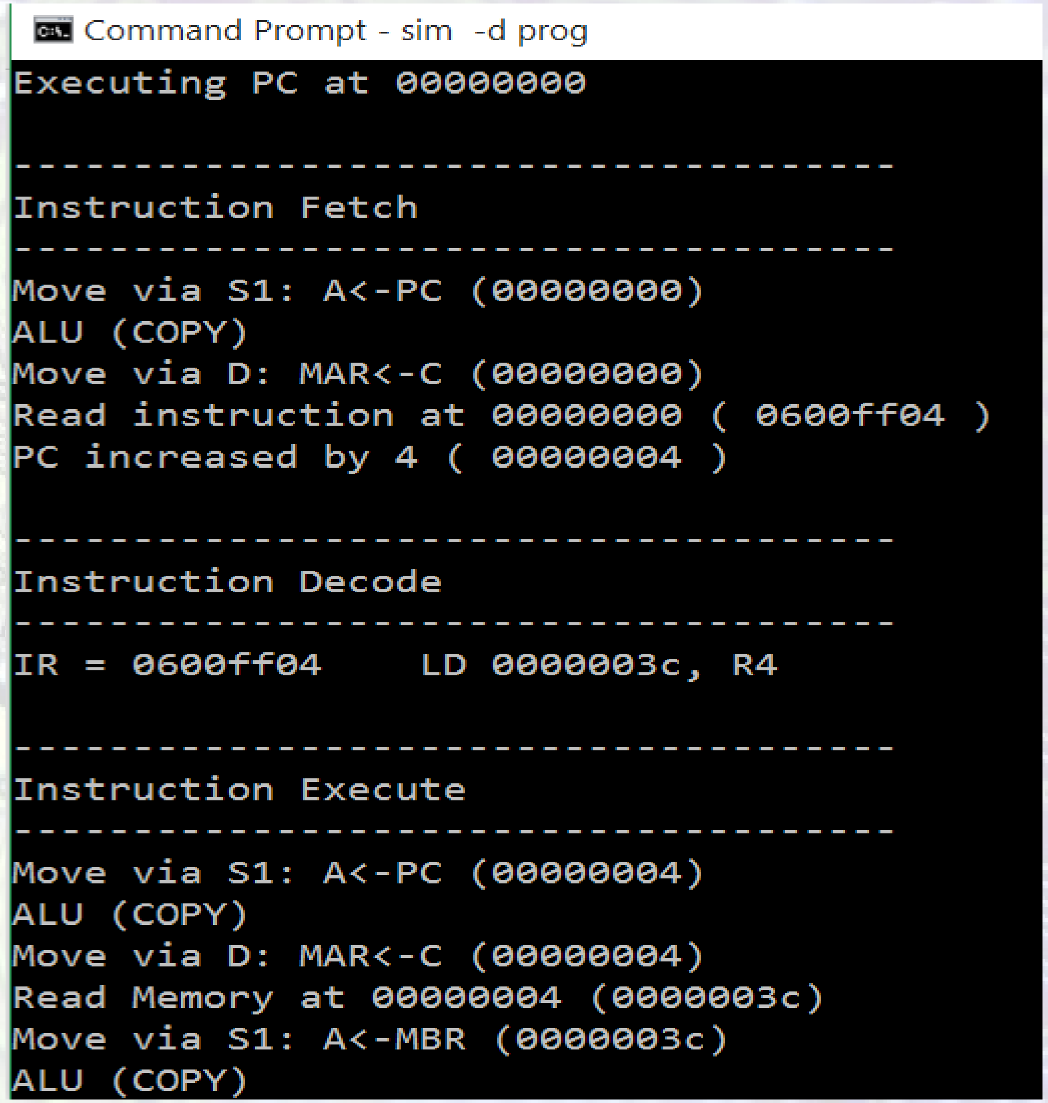
- Written in C++
- Simulates data flow in CU & ALU
- Run in command-line interpreter, e.g. Command Prompt in Windows & Terminal in Linux/Mac OS
- Input
  - Text file containing instruction set in binary form



```
0600ff04
0000003c
0600ff01
00000040
05010002
0600ff03
00000044
00040104
00010201
01030105
0802ff00
0000001c
0704ff00
00000048
09000000
00000000
00000001
0000000a
00000000
```

Instruction Execution Simulator

- Output
  - On console
  - Sequence of data movement & transformation of every instruction
  - Displayed as Pseudocode-like descriptions

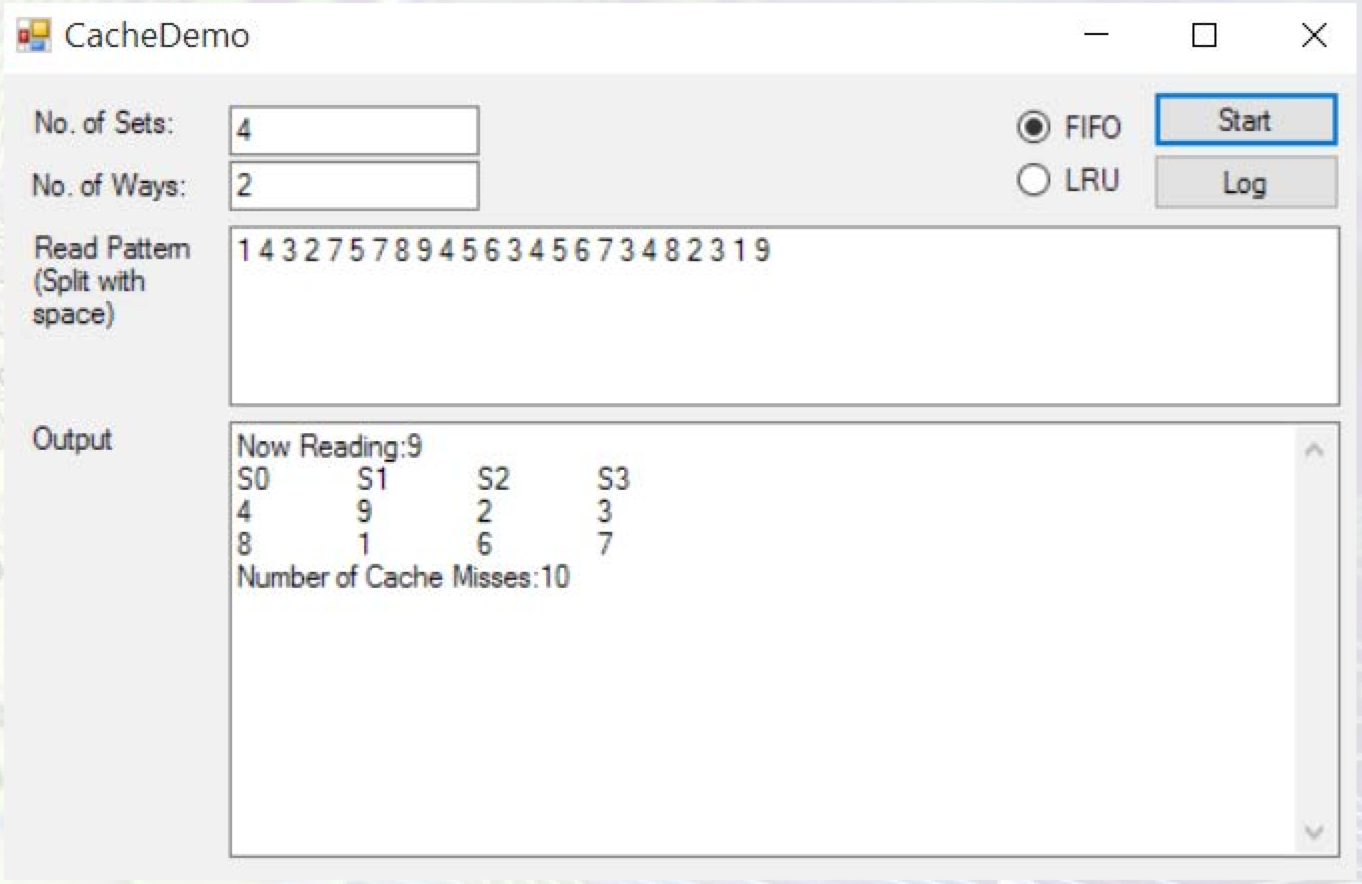


Command Prompt - sim -d prog

```
Executing PC at 00000000

-------------------------------------
Instruction Fetch
-------------------------------------
Move via S1: A<-PC (00000000)
ALU (COPY)
Move via D: MAR<-C (00000000)
Read instruction at 00000000 ( 0600ff04 )
PC increased by 4 ( 00000004 )

-------------------------------------
Instruction Decode
-------------------------------------
IR = 0600ff04    LD 0000003c, R4

-------------------------------------
Instruction Execute
-------------------------------------
Move via S1: A<-PC (00000004)
ALU (COPY)
Move via D: MAR<-C (00000004)
Read Memory at 00000004 (0000003c)
Move via S1: A<-MBR (0000003c)
ALU (COPY)
```

Cache Memory Simulator

- Written in C#
- Simulate Mapping and Replacement Algorithm
- With GUI



No. of Sets: 4

No. of Ways: 2

[✓ FIFO] LRU

Read Pattern (Split with space)

```
1 4 3 2 7 5 7 8 9 4 5 6 3 4 5 6 7 3 4 8 2 3 1 9
```

Output

```
Now Reading:9
S0        S1        S2        S3
4         9         2         3
8         1         6         7
Number of Cache Misses:10
```

Existing teaching Aids

- Problems:
  - 2 operational units of computer system architecture (i.e. CPU & Memory) simulated separately
  - Configuration are hardcoded
  - Command line output format is unattractive and with low readability
- There is need for an integrated computer system simulator which can cover more operational units, more functionalities and with greater flexibility in configurations.

Project Objective

- Primary Objective:
  - 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

Project Features

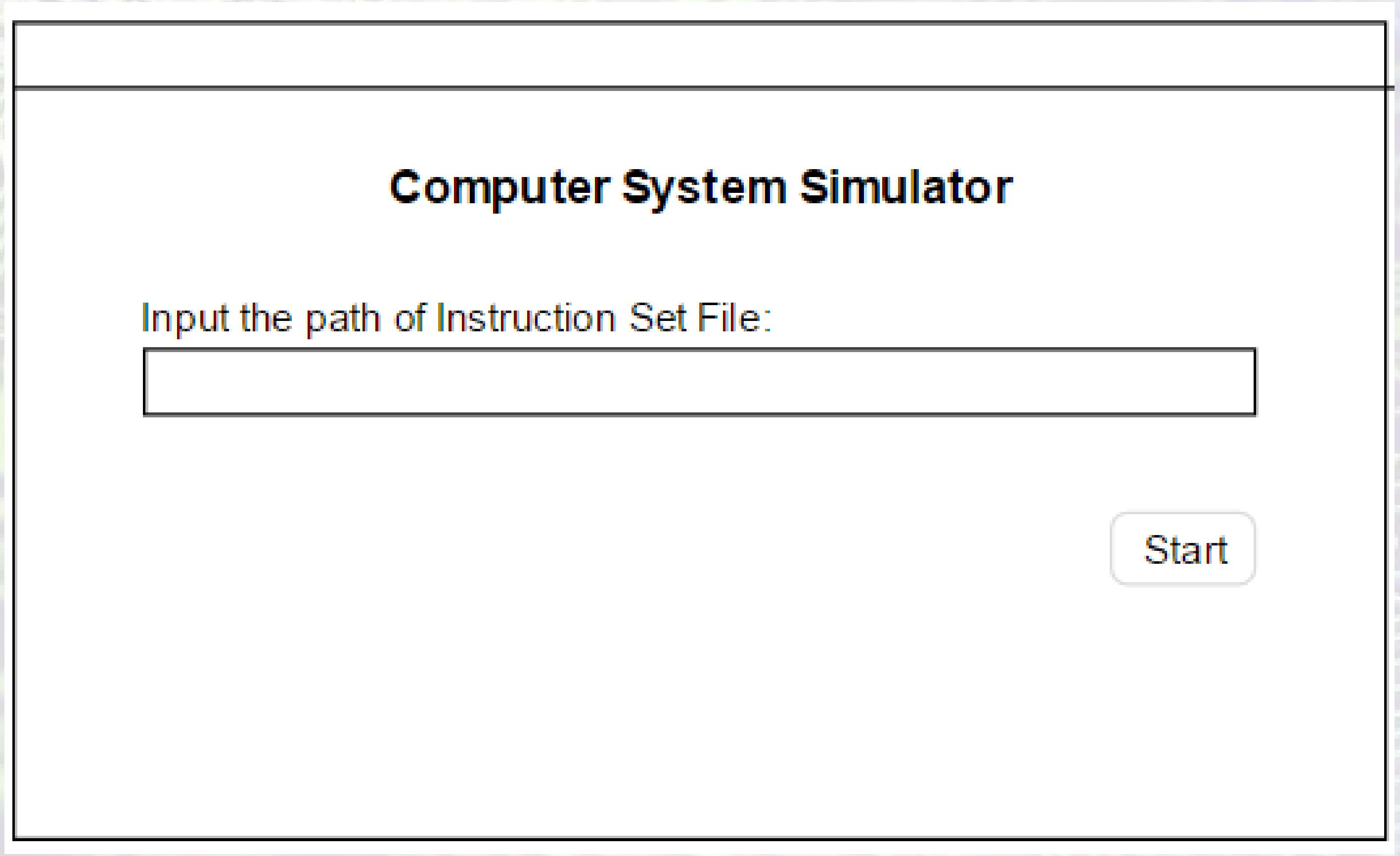
- Interconnection of different Operational Units
  - Cover Data movements & transformation in one single simulator
    - Processor (CPU)
    - Memory Hierarchy (Cache Memory, Main Memory)

Project Features

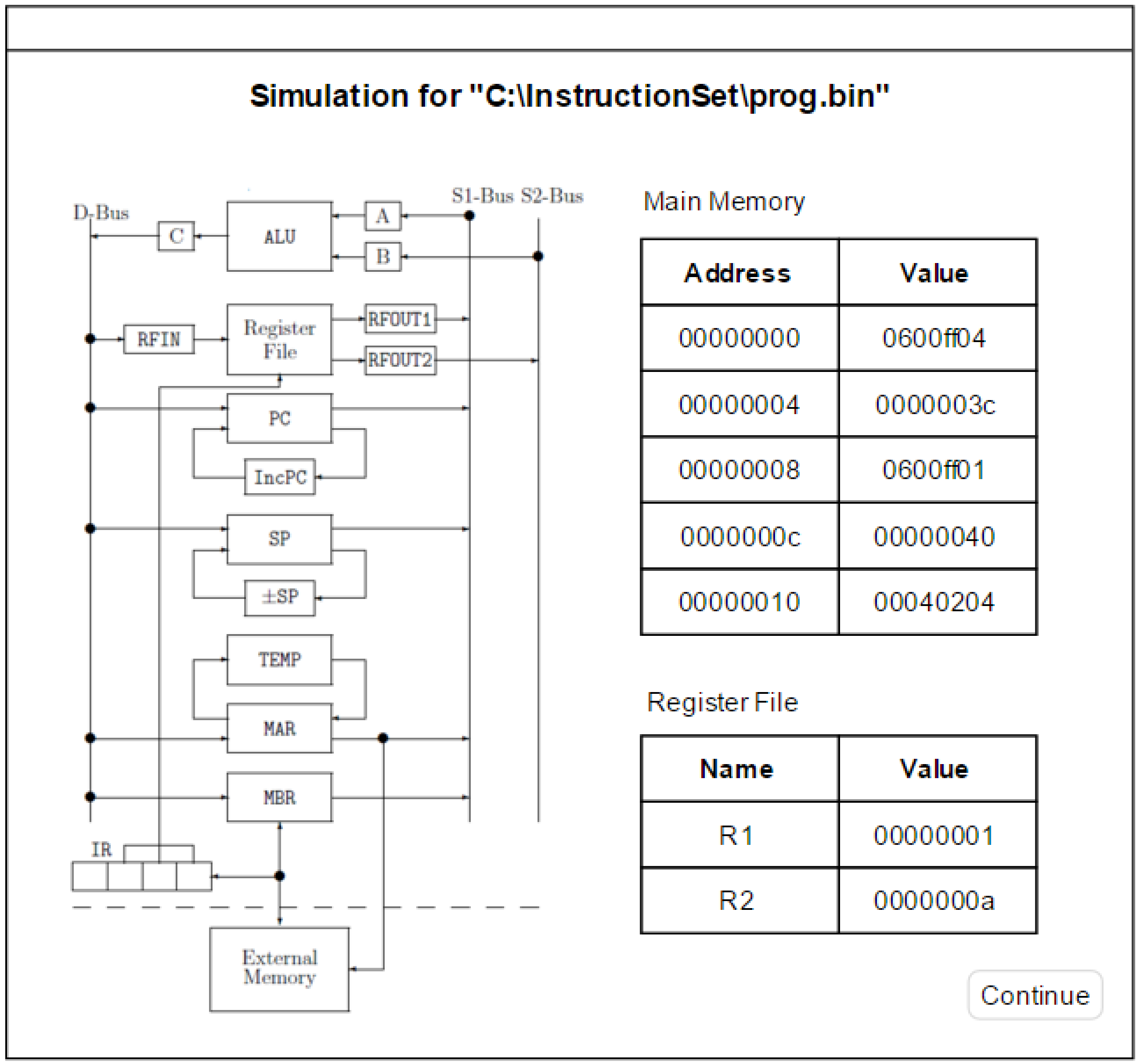
- Graphical User Interface (GUI)
  - Components of processor (e.g. registers, stack pointer, ALU) displayed in form of graphic
  - Data movement & Transformation illustrated using graphics
    - e.g. highlighting updated components
  - Benefits:
    - Structure can been shown clearly
    - Able to view full picture of instruction set execution process

Sample GUI Design

- Index Plane of the Simulator Program


Computer System Simulator
Input the path of Instruction Set File:
Start

Sample GUI Design



Simulation for "C:\InstructionSet\prog.bin"

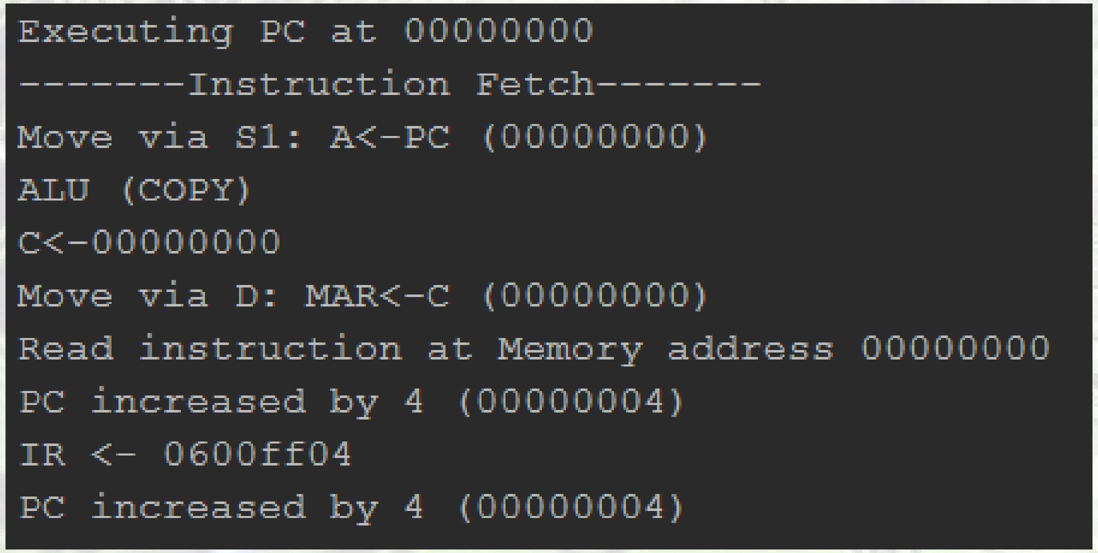
Main Memory

Address	Value
00000000	0600ff04
00000004	0000003c
00000008	0600ff01
0000000c	00000040
00000010	00040204

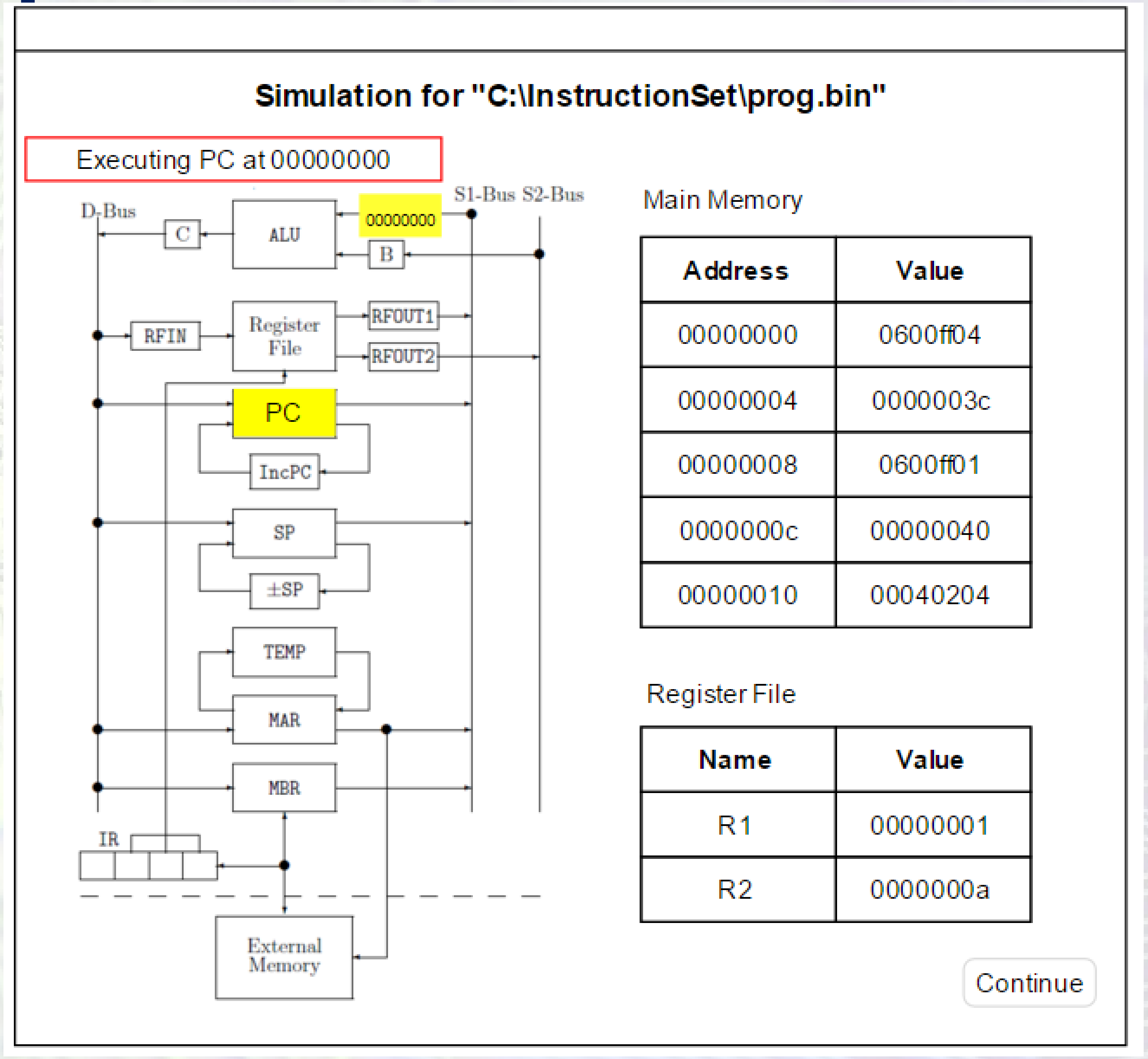
Register File

Name	Value
R1	00000001
R2	0000000a

Example Output: CMD vs GUI

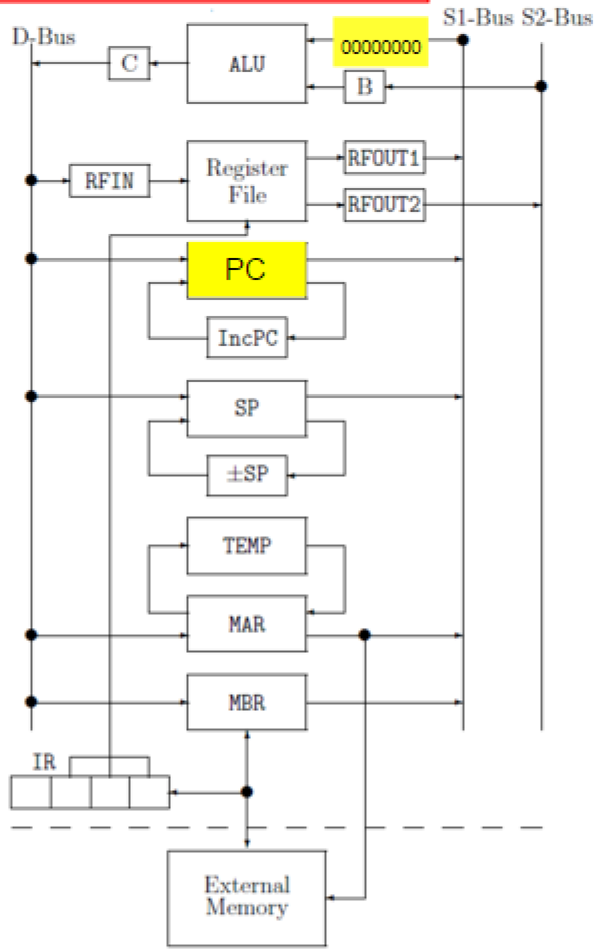


```
Executing PC at 00000000
-------Instruction Fetch-------
Move via S1: A<-PC (00000000)
ALU (COPY)
C<-00000000
Move via D: MAR<-C (00000000)
Read instruction at Memory address 00000000
PC increased by 4 (00000004)
IR <- 0600ff04
PC increased by 4 (00000004)
```



Simulation for "C:\InstructionSet\prog.bin"

Executing PC at 00000000



Main Memory

Address	Value
00000000	0600ff04
00000004	0000003c
00000008	0600ff01
0000000c	00000040
00000010	00040204

Register File

Name	Value
R1	00000001
R2	0000000a

Project Features

- Flexible Configuration
  - Configuration will be independent of core program source code
  - Read external configuration file when simulator runs

exampleConfig.properties

```
#Computer System Component

#CPU Config
cpu.numOfCPU = 1
cpu.numOfCore = 1
register.size = 32
#(Size in MB)

#Main Memory (RAM) Config (size in MB)
memory.size = 1024

#Operation Code Translation
#in binary form
ADD = 00000000
SUB = 00000001
MOV = 00000101
LOAD = 00000110
STORE = 00000111
```

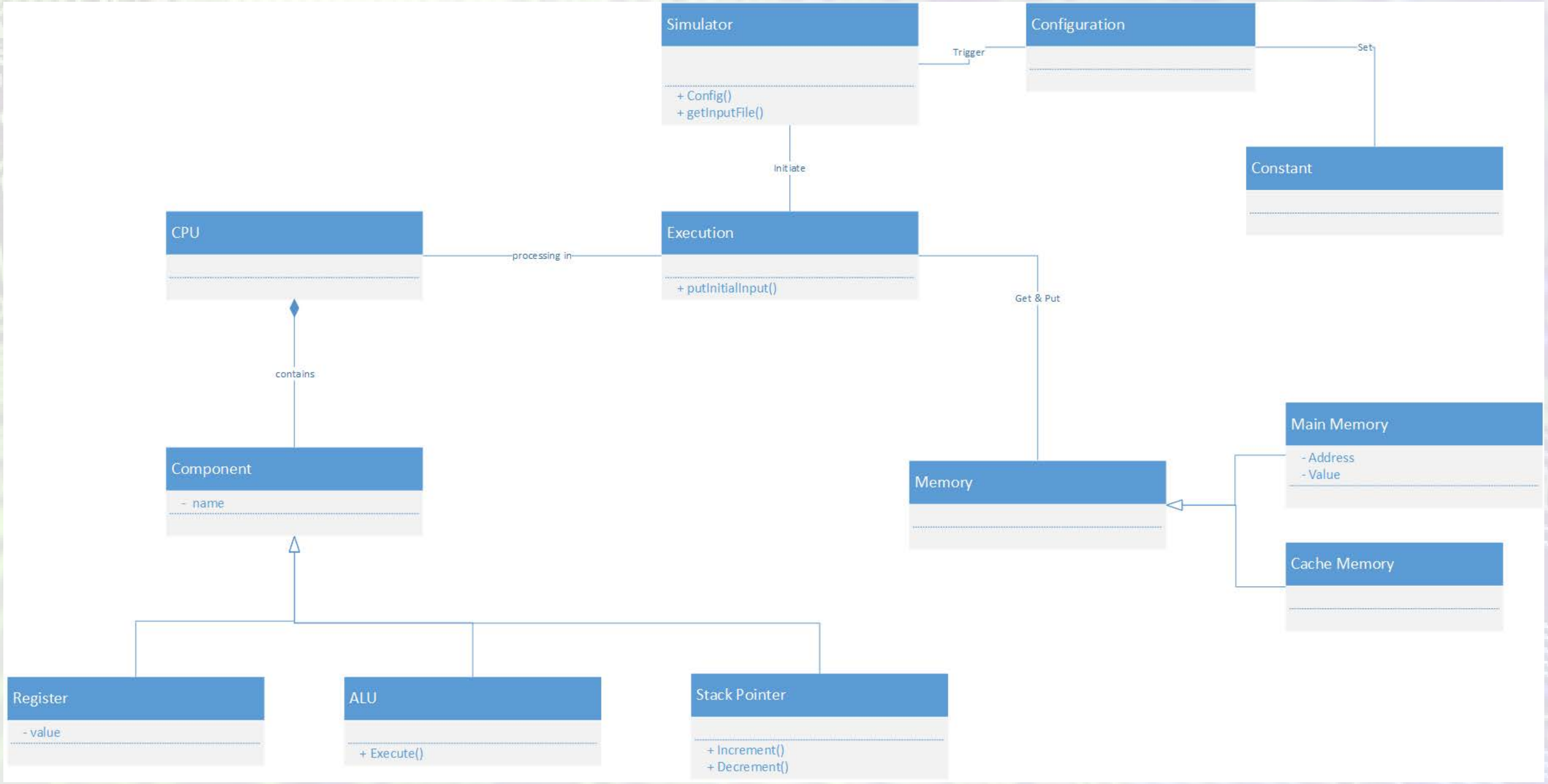
Project Features

- Further Enhancement
  - To cater other addressing mode (e.g. displacement)
  - Specify execution order of instructions
    - Some instructions may be able to be processed in parallel

Program Features

- Programming language used: Java
  - Object-oriented programming language
  - Cross-platform
  - Lots of existing library that useful for implementing project features, e.g. configuration

Class Diagram



Progress

October

- Completion of Project Plan
- Preliminary Study & Research
- Program Class Design

November

- Commencement of Phase 1 Coding (CPU)

Late December – Early January

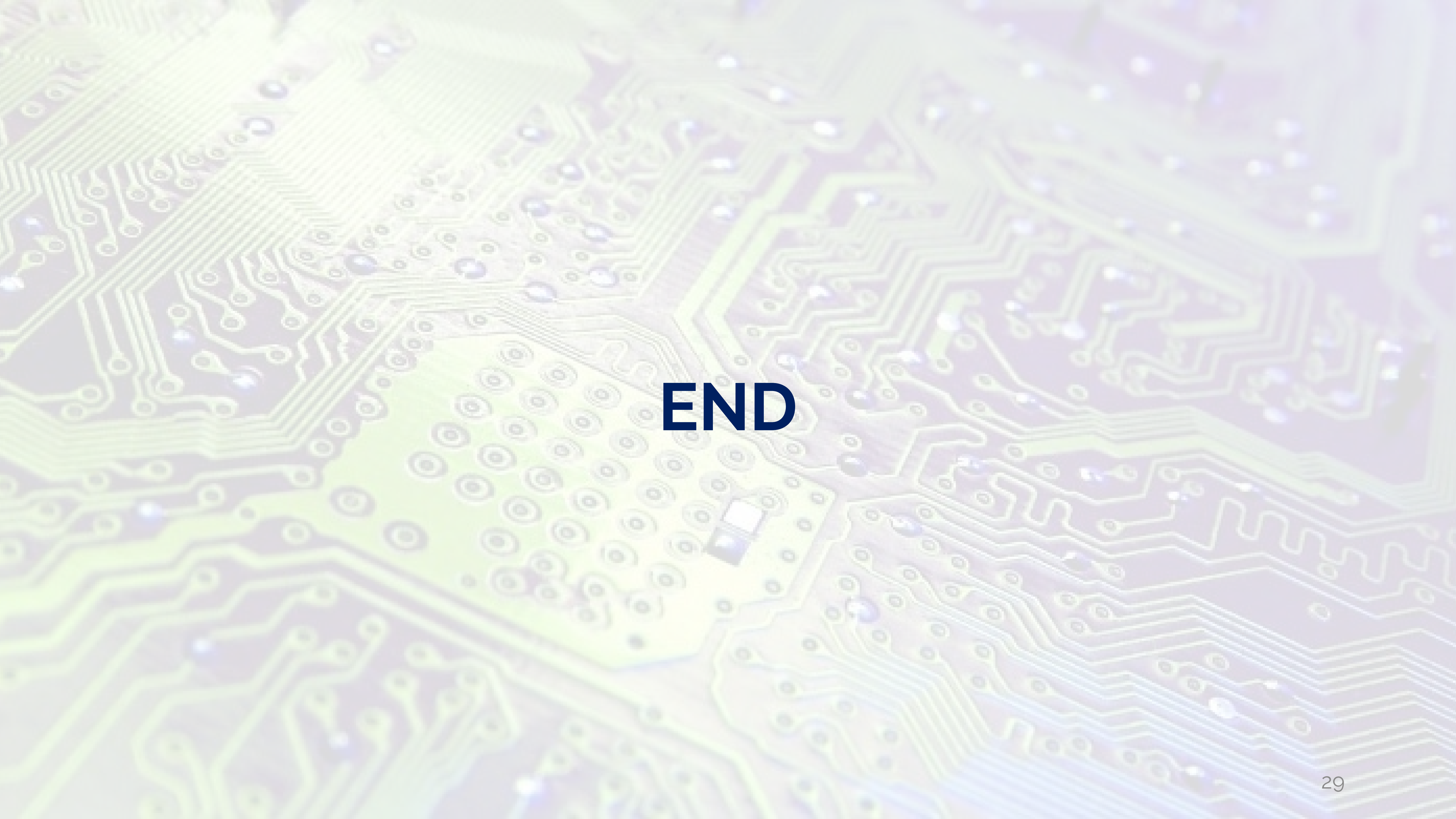
- Completion of Phase 1
- Commencement of Phase 2 (Cache Memory)

2nd Semester Schedule

Feb
- Completion of Phase 2 (Cache Memory)
- Documentation for Phase 2
- Commencement of Phase 3 (GUI)

March
- Completion of Phase 3
- Commencement of Phase 4 (Enhancement)
- Documentation for Phase 3&4

April
- Completion of Phase 4
- Integrated Testing & Debugging
- Completion of Documentation



END