CSE305 Computer Architecture
201911036 Chanjung Kim
Professor Daehoon Kim
Project #3 – Simulating Pipelined Execution

Introduction

This program is an emulator which runs executables generated from the Simple MIPS assembler, which was the last assignment. This emulator does not use any special algorithms or technologies to run the executable faster.

Project Structure

The project contains three directories: `Public`, `Source`, and `Tests`. `Public` contains header files, `Source` contains C++ source files, and `Tests` contains unit tests. Source files in `Tests` are not necessary to build `runfile`.

`Public` and `Source` contains several source and header files, whose roles are as the following:

- `Common.hh` / `Common.cc` implements common functions like hex integer parsing.
- `Components.hh` contains definition of classes which are added to `Emulator` as delegates (Component Pattern).
- `Emulator.hh` / `Emulator.cc` implements `EmulatorBuilder` and `Emulator` (Builder Pattern).
- `File.hh` / `File.cc` contains file I/O functions.
- `Implementations.hh` / `Implementations.cc` contains implementations derived from classes in `Components.hh`
- `Main.cc` contains the entry point, parses the command-line argument, and converts errors into strings.
- `Memory.hh` / `Memory.cc` implements the register file and the memory.
- `NamedEntryMap.hh` / `NamedEntryMap.cc` contains helper classes used to contain state registers' names.

List of State Registers

The following is the list of state registers. You can refer `Implementations.hh` and `Implementations.cc` to see which datapath stage reads/writes each register.

Name	Description
{IF_ID, ID_EX, EX_MEM, MEM_WB, WB}_PC	Contains memory address in which the instruction is located. These are used to print the current pipeline state.
{IF_ID, ID_EX, EX_MEM}_NextPC	Contains PC + 4.
{IF_ID, ID_EX, EX_MEM, MEM_WB, WB}_Instr	Contains the instruction.
{EX_MEM, MEM_WB}_ALUResult	Contains the result computed in the EX stage.
{ID_EX, EX_MEM, MEM_WB}_RegWrite	Contains 1 if the instruction writes a value to the register file. Contains 0 otherwise.
{ID_EX, EX_MEM, MEM_WB}_MemRead	Contains 1 if the instruction reads a value from the memory. Contains 0 otherwise.
{ID_EX, EX_MEM}_MemWrite	Contains 1 if the instruction writes a value to the memory. Contains 0 otherwise.
ID_EX_Reg1Value	Contains value from rs.
{ID_EX, EX_MEM}_Reg2Value	Contains value from rt.
ID_EX_Imm	Contains the immediate value.
ID_EX_Reg1	Contains Instruction[25-21].
{ID_EX, EX_MEM}_Reg2	Contains Instruction[20-16].
ID_EX_Reg3	Contains Instruction[15-11].
{ID_EX, EX_MEM, MEM_WB}_RAWrite	Contains 1 if the instruction writes a value to the RA register. Contains 0 otherwise.
{ID_EX, EX_MEM, MEM_WB}_RAValue	Contains the value which will be written to the RA register.
{EX_MEM, MEM_WB}_DestReg	Contains the index of the register to which the value will be written.
MEM_WB_ReadData	Contains the value read from the memory.

Build Instructions

This emulator is written in C++17. Although GCC 7 supports C++17, it lacks some standard C++17 headers (e.g. `<filesystem>` or `<charconv>`). To properly build the program, please use GCC 9 or higher. To install GCC 9 on Ubuntu 18.04, run:

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo apt update
sudo apt install gcc-9 g++-9
```

This project uses CMake 3.13. To build the program with CMake, run:

```
mkdir build
cd build
cmake -DCMAKE_CXX_COMPILER=g++-9 ..
cmake --build .

./runfile -atp -n 10 -p -d input.o
```

If CMake 3.13 or higher is not available in your system, you can use this command instead:

```
g++-9 -std=c++17 -I./Public          \
      -o runfile                     \
      ./Source/Common.cc             \
      ./Source/Emulator.cc           \
      ./Source/File.cc               \
      ./Source/Implementations.cc \
      ./source/Main.cc               \
      ./Source/Memory.cc             \
      ./Source/NamedEntryMap.cc

./runfile -n 10 -d input.o
```

Unit Tests

To run the unit tests, please download Vcpkg from https://github.com/microsoft/vcpkg and install GTest. To build the unit tests with CMake, run:

```
mkdir build
cd build
cmake \
  -DCMAKE_CXX_COMPILER=g++-9 \
  -DCMAKE_TOOLCHAIN_FILE=/vcpkg/scripts/buildsystems/vcpkg.cmake \
  -DENABLE_SIMPLE_MIPS_EMU_TEST=ON \
  ..
cmake --build .
ctest
```