Project Report

In this project I have done a programme to simulate MIPS instructions.

How to use the program:

Type “g++ A2Asm.h A2Asm.cpp Assn2FinalCode.cpp -o assembler”; Type “./assembler”. After that, it will output "Please enter the testfile name. (e.g. a.asm)" Then the input file name (e.g. testfile.asm) could be inputted. If you typed a wrong name, it will output "Invalid name.". Then you can input the file name again. In the file, the labels and instructions shouldn’t be connected; otherwise, the disassembling process will lead to errors. (e.g. “main:addi” will lead to error but “main: addi” won’t)

After that the programme will start simulation.

Code Logic:

Fundamental Framework

The A2Asm.h and A2Asm.cpp mainly comes from Project 1, with some modification: for example, “syscall” is included; meanwhile, it could recognize if there is “.data” and “.text”.

According to test files, I am required to separate simulation into 2 stages: storing stage and execution stage. In the storing stage, there are 6 types of data required to be specified, including codes. In this program, they are divided into 3 kinds of structure: strings, integers, and codes. If the first word of the test file is “.data”, the strings and integers will be stored, while the data storing process will skip everything before a “.text”. Strings are stored by function “ascii”, while integers are stored by function “integer”.

After given data are stored, the program will start to disassemble codes, and start to store them in the memory. One instruction will be divided into 4 parts and stored as 4 characters.

After completing storing stage, execution stage will begin.

Storing Stage

During the storing stage, the $sp, $gp, $fp will be initialized. After opening given file, the first word will be examined if it is “.data”. If so, the program will scan every line until there is a “.text”. Data in those lines will be stored. After eliminating comments and labels, if the line is blank, this line will be skipped. Otherwise, the word symbolizing data structure will be read. The rest of the line will be stored in a buffer string “data”. The data structure will be judged and corresponding action will be performed.

String type: The program will align its memory first. The string will be purified to eliminate “ and ”. Then the program will store the string character by character until ‘\0’. Then if the data structure is “.asciiz” the program will align its memory again.

Integer type: The program will align its memory first. The integers will be purified to eliminate ‘,’. Then the program will store the integer by a number of characters. Then if the data structure is “word” or “half” the program will align its memory again.

After meeting “.text”, the given instruction will be disassembled and stored. If there is a “.data”, everything before “.text” will be ignored, thus securing the right value of labels. Every instruction will become a 32-bit long code, which will be stored as 4 characters. Then PC will be initialized. Then it will go to the execution stage.

Execution Stage

As I need to obey the machine cycle, I separated the execution stage into 5 parts: Instruction Fetch, Instruction Decode, Execution, Memory Read and Memory Write.

Instruction Fetch

The Instruction Fetch part will be done by function “IF”. It will fetch 4 characters instructed by PC and combine them together. Then PC will be added by 4. If the fetched instruction is "00000000000000000000000000000000", the program will be stopped.

Instruction Decode

The Instruction Decode part will be done by function “initialize”. It will store specific areas of the code, including op, rd, rs, rt, shamt, funct, im.

Execution

The Execution part is the main component of main function. It will examine different parts of codes and do corresponding actions. If the instruction expect interactions with memory, relevant data will be recorded.

Memory Read

The Memory Read part will be done by function “MEM”. It will decode relevant data and take measures.

Memory Write

The Memory Read part will be done by function “WB”. It will decode relevant data and take measures.

Execution Part

Most of instructions can be directly executed. But for add, sub, mul, div and some other instructions, there should be some functions’ assistance.

For function “add”, it will judge whether the result may overflow. If not unsigned, overflow will lead to an exception.

For function “hlm”, the hilo[0] will be the more significant 32 bits, while hilo[1] will be the less ones; for “hld”, the hilo[0] will be the quotient, while hilo[1] will be the remainder.

For syscall, $v0 will be judged and corresponding measures will be taken. If $v0=17, the $zero will be 1, indicating returning with values.

Memory Read

For some instructions, the program will firstly examine whether the latent data should be aligned. If not, it will lead to an exception.

The positive and negative value of amount indicates the direction of reading: positive means read towards the bigger side. The program will read data character by character and save the holistic integer to the register.

Memory Write

If amount=5(sc), it will operate specially at first; then the program will read data character by character and save the holistic integer to the register.