Computer Architecture

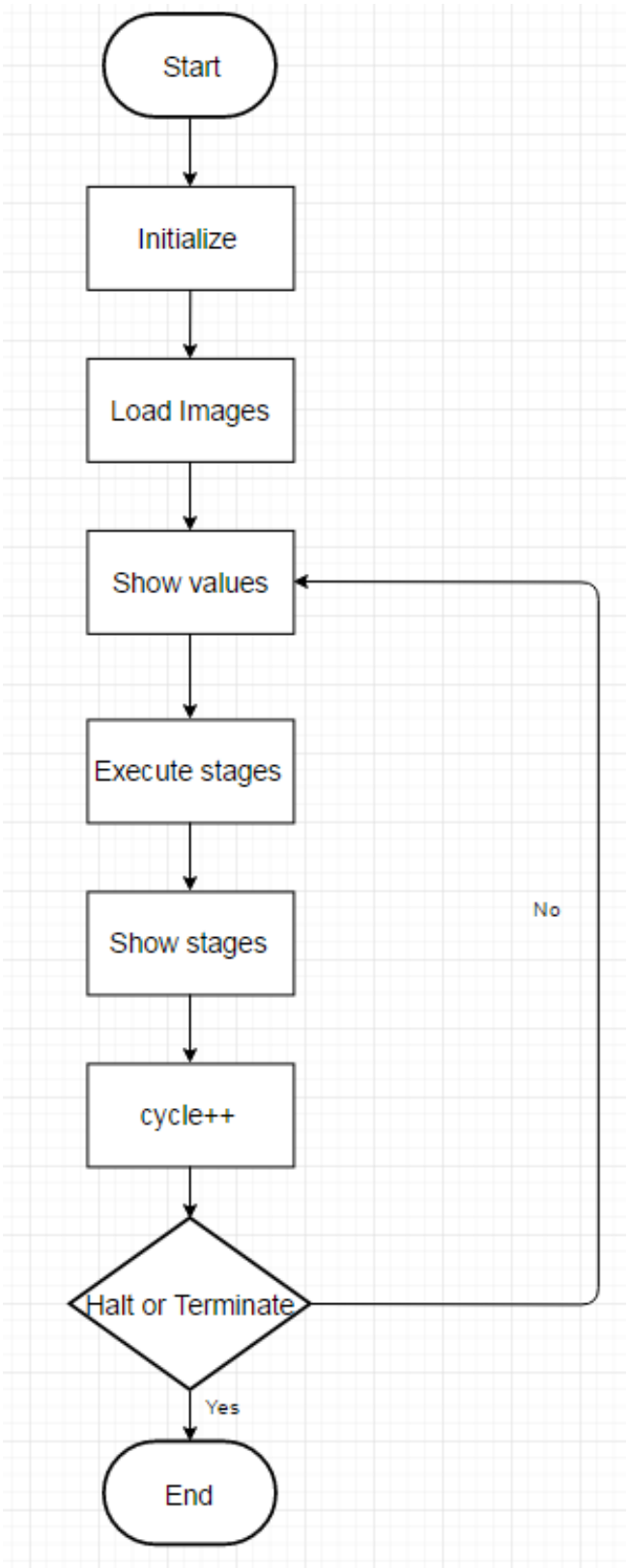
Project 2

Due. 2017/5/6

104062203 陳涵宇

Project Description

1. Flow chart



The picture above is the flow chart about my program. At the beginning, the program will initialize the value of registers and memory arrays to zero, and then read the images in. For the whole pipeline, I implement a class called “Stage”, and I declared a “Stage” called “stage”. stage will also be initialized at the beginning, including setting Boolean variables to false and allocating memory for pointers in a self-defined type called “Instruction”. Then it will show the registers’ values, and starting execute each stages in the order WB, DM, EX, ID and IF.

After executing, the program will print the instruction executed in each stage, then add the cycle to represent that a new cycle is going to start. After doing these, the program would check the status of the pipeline. If an error that would cause the program terminate happened, than the program would be terminated. Besides, if all of the instructions in all five stages, were “NOP”, the program would also be ended. If the two situations mentioned above didn’t happen, then the program would go back to show registers’ values as the flow chart shows.

2. Detailed description-

In memory.cpp:

In error.cpp:

These two files are almost same as the file in project 1.

In regfile.cpp:

I add the function to print out stages in this file.

In simulator.cpp:

This file do the things shown in the flow chart.

In InstructionFetch.cpp:

This file represent stage ”IF”. In this stage, the program will first check if it is needed to be flushed or stalled, and then do the corresponding things. If there is no need to flush or stall, then the program will just get the new instruction from I-memory. At last, the result of Instruction IF will be stored to IFID (which means the register between IF and ID).

In InstructionDecode.cpp:

This file represents the stage “ID”. In this stage, the program will check if it is needed to be flushed. If is needed, the information stored in IFID will be deleted. If not, the program will read the data from IFID and analyze the instruction code, then check whether it is needed and possible to forward from EXDM. After dealing with forwarding, the program will deal with branch-related instruction, such as jr, j, jal. At last, the result will be stored into IDEX.

In Execute.cpp:

This file represents the stage “EX”. At first, the program will check if the previous is stalled. If so, then the data in EX should be “NOP”. If not, the program will load the instruction from IDEX and check about forwarding, including EXDM and DMWB. If needs forwarding, then the program will forward data, or it will simply get data from the corresponding register. After getting data, then the program will start calculating. Errors of number overflow and overwrite HI-LO register will be detected in this stage. The result will be stored to EXMEM.

In MemoryAccess.cpp:

This file represents the stage “DM”. In this stage, the program will load instruction from EXMEM, deal with all kinds of memory-access-related instructions and detect errors of data misaligned and address overflow. Then it will be stored to MEMWB.

In WriteBack.cpp:

This file represents the stage “WB”. The program will load the instruction from MEMWB and write the result calculated in EX stage to the corresponding register. Error of write to $0 is checked here.

Testcase Description

I just try to make hazard and forwarding happen, and the instruction is selected randomly.