LAB5: Pipeline Stall Detector/Simulator Report

Gaurav Choudekar - CS22BTECH11015

1 Introduction

This report is regarding lab5 conducted on pipeline stall detector.This code takes a file as input and outputs their respective code with nops in 2 cases,one with forwarding and other without forwarding.

2 About Code

The code is written in C.It can be executed using gcc as mentioned in readme file.

3 Input format

A file is given as input,name of file is to be modified accordingly.in each line of file,one instruction is given with spaces and comma,it is assumed that no blank line is given.file name to be modified at 2 places.One before without forwarding case and other before forwarding case.Segmentation fault may be occured for first time when file is used as input,in such case,create a line with enter and then use backspace.Segmentation fault may be debugged with this.

4 Code Flow

4.1 Main Function-beginning

In the main function we are giving a file as input at 2 places as mentioned above.First,without forwarding case is executed.one global variable is used to calculate total number of cycles.this is again adjusted to 2 before starting new calculation for with forwarding case.Buffer string is used to store read line from file using fgets and stored in other string for further use.Last character of each line made to null.Their duplicates are also made so that our original instruction can be printed after processing it.

4.2 Main Function-general idea

This code calculates nops with help of 3 instructions,first one gets removed and next gets added to list each time.Some cases are made according to number of nops in between a,b b,c(assume its their order).

4.3 Calculating NOPS

We only check NOPS between a and c when there is no NOP between a and b as well as b and c,in such a case,if 2 NOPS are calculated between a and c,then its equivalent to 1 NOP as b instruction is there between a and c,otherwise,no NOP is between b and c due to a and c.NOP count due to a and b and b and c due to them itself does not get changed,if any of both of them have non-zero NOPs,then no need to check between a and c for between b and c,count of NOPS between a and b as well as b and c remains unchanged in that case.

4.4 Functions for NOP calculation

There are 2 functions for with forwarding and without forwarding cases,namely processf and process.It takes 2 instructions as input and returns number of NOPS as if they were consecutive.Further,it is used to print NOPS as mentioned above.NOPS are calculated keeping in mind destination and source registers.

4.5 Function for different naming

One seperate function is made to replace name of register in 'x' series if it is given in other valid name,namenly change.one more function is used to get register name in case it is having brackets in its token.

5 Accuracy check

Different input files are made and collected from different people and our answers were cross-checked,necessary corrections were made to ensure good accuracy

6 Run the program

1. Compile the program using gcc coa5.c -o coa5
2. run it by typing ./coa5

7 Test case files

Some of sample test case file are given in this zip file.

8 Results

This code outputs assembly level instruction with added NOPS where required in 2 cases,one where forwarding is considered and other forwarding is not considered.Number of cycles is also printed calculating number of NOPS,number of instructions and number of stages.

9 Conclusion

This lab helps us to understand about different stages of standard-5 cycle processor,and helps to understand importance of NOPS and forwarding.