/\*

\* Supplementary D: Source Code of Opcode Analysis Pass in C language

\*

\* Author: Dr. Ehsan Ali

\* Email: ehssan.aali@gmail.com

\* Last Update: 4th Nov. 2024

\*

\* Used in Article: Innovative Hardware Accelerator Architecture

\* for FPGA-Based General-Purpose RISC Microprocessors

\*

\* URL: https://github.com/ehsan-ali-th/cortex\_m0\_MA/tree/master/Supplementaries

\* \*/

For code:blocks project issue:

$ mkdir build

$ cd build

$ cmake -DCMAKE\_BUILD\_TYPE=Debug -G "CodeBlocks - Unix Makefiles" ../src/

op\_analysis.cpp:

//=========================================================================

// Name : op\_analysis.cpp

// Author :

// Version :

// Copyright : Your copyright notice

// Description : Hello World in C++, Ansi-style

//=========================================================================

#include "tools/tools.h"

#include "tools/OpcodeAnalyzer.h"

#include <iostream>

#include <iterator>

#include <fstream>

using namespace std;

using namespace esi;

int main(int argc, char \*argv[]) {

cout << "Hi !" << endl;

if (argc < 4) {

cout << "Usage: ./op\_analysis input\_file1(opcodes.dat) exidx.dat rodata.dat" << endl;

cout << "input\_file1 (opcodes.dat) = should have .text section content" << endl;

cout << "input\_file2 (exidx.dat) = should have .ARM.exidx section content" << endl;

cout << "input\_file3 (rodata.dat) = should have .rodata section content" << endl;

return -1;

}

string opcodes\_filename = argv[1];

string exidx\_filename = argv[2];

string rodata\_filename = argv[3];

cout << "- Analyzing : " << opcodes\_filename << " - " << exidx\_filename << " - " << rodata\_filename << endl;

// Extract the filename by removing the extension

size\_t lastindex = opcodes\_filename.find\_last\_of(".");

if (lastindex == string::npos) {

cout << "No extension in filename is specified. " <<

"Input filename must have .dat extension" << endl;

return -2;

}

string filename = opcodes\_filename.substr(0, lastindex);

string report\_filename = filename + ".txt";

string vhdl\_filename = "miniature\_accelerator.vhd";

cout << "- VHDL Adaptive module: " << vhdl\_filename << endl;

cout << "- Report file: " << report\_filename << endl;

// Open the files

ifstream opcodes\_file\_in (opcodes\_filename.c\_str(), ios::in);

ifstream exidx\_file\_in (exidx\_filename.c\_str(), ios::in);

ifstream rodata\_file\_in (rodata\_filename.c\_str(), ios::in);

OpcodeAnalyzer opcodeAnalyzer(opcodes\_file\_in, exidx\_file\_in, rodata\_file\_in);

opcodeAnalyzer.list\_functions();

cout << "Number of functions = " << opcodeAnalyzer.count\_functions() << endl;

for (int i = 0; i < opcodeAnalyzer.count\_functions(); i++) {

Function f = opcodeAnalyzer.getFunction(i);

cout << "Function " << f.getName() << " has " << dec << f.count\_MBs() << " Machine Blocks, with " <<

f.getNumOfInstructions() << " instructions" << endl;

f.listMBs();

}

Function func = opcodeAnalyzer.getFunction(0);

opcodeAnalyzer.mineFunction(func);

pair<Instruction, Instruction> mostFreqPair = opcodeAnalyzer.getMostFreqPair();

cout << "Most Frequent pair is (" << mostFreqPair.first.getOpcode() << ", " <<

mostFreqPair.second.getOpcode() << ")" << endl;

vector <uint32\_t> PCSensivityList = opcodeAnalyzer.getPCSensitivityList(0, mostFreqPair);

opcodeAnalyzer.getRC\_PC\_sensivityFile(PCSensivityList);

vector <pair <Instruction, Instruction> > intsr\_pairs = opcodeAnalyzer.genAccelMemVHDL(0, mostFreqPair);

opcodeAnalyzer.getPC\_to\_OP\_File(PCSensivityList, intsr\_pairs);

opcodeAnalyzer.genMainProgramCoeFile();

cout << "End of Program." << endl;

return 0;

}

CMakeLists.txt:

cmake\_minimum\_required(VERSION 3.16.3)

set(CMAKE\_ECLIPSE\_VERSION "4.14.0" CACHE STRING "4.14.0")

# Set the project name

project(op\_analysis)

add\_compile\_options("-Wall" "-Wpedantic" "-Wextra" "-fexceptions" "-Werror" "-O0;-g3;-ggdb")

# Add the executable

add\_executable(op\_analysis op\_analysis.cpp)

target\_link\_libraries(op\_analysis ToolsLib)

add\_subdirectory(tools)