Computer Architecture and Mobile Processor

Project #2

Single-cycle MIPS

Nadia Calista Candra The

Dankook University

Spring 2021

Freedays left: 5

디나디아차리스타찬드라
32185144

1. Project Introduction

Building a MIPS simulator that has simple ISA, and optimized for concurrent parallel execution for performance. MIPS ISA defines 31 integer instructions and are categorized into I, R, J types and MIPS ISA defines 32 general purpose registers. Building a CPU simulator that takes MIPS executable binary as an input and emulates the operation in single-cycle MIPS.

2. Motivation

The MIPS simulator mimics the operation of the CPU, so it is good to understand the operational behaviour inside the CPU. Other than that, getting used to the Linux environment which is more manual and by making the project through assam server.

3. Concepts Used

File operations:

Files are opened using fopen and closed using fclose. Contents are read using the fread function.

Struct statement:

Allowing to define types of variables that can hold several data items in the same kind and allows it to combine the data items of different kinds. Using arrow operator to access the elements in the struct.

Define directive:

Using define to define macros within the source code. It will allow the constant value to be declared throughout the code

Bzero function:

It will erase the data in the memory starting from the location that is pointed.

4. Program Structure

Program runs in a loop in main function

+----------

| initialize()

| loadprog()

| fetch()

| decode()

| execute()

+---

In each stage the function will print out the result on the screen (except for initialize()).

initialize() to initialize the program and setting values of sp and ra.

loadprog() to load the program using fread in a loop

fetch() to fetch the instruction

execute() to execute the program and it will go inside an if else function that will differentiate either it is the operand or funct that is wanted or not and will do the instruction that is needed then print it out.

5. Problems and Solutions

At first I didn't know how to make the mips simulator but after listening to class, doing the datapath hw and searching up on the internet I got a picture on how and what I needed to do. At first I am not able to load the program because I didn't put "rb" when it is on the fread function because it was a binary file however, by seeing references I am able to make it. Not only that, I have problems putting the binary file and opening the binary file in my computer. However, after listening in class and seeing how it is done I am able to do that. And then the test files are given through elearning site and I am able to copy it to the assam server by following the instructions that are made.

Then, making errors are extremely easy here because of the similarity in the Opcode instructions and it did take me a lot of time to be able to input all of the instructions that are needed without having a mistake. I also have problem solving how to update the PC but I solved it by adding 4 after they are done with the execution so it will be added by four for the next instruction.

While compiling, I receive problems on the 'move' instruction because that instruction does not have an opcode and because of that I am very bewildered and after that I found out that it was a pseudo instruction and it will be translated into addu by the assembler. I also had problems where it will only show up until the 2nd instruction and doesn't continue to the third instruction. To solve that problem I print on every function it goes and finds out where my problem was.

6. Build Environment

Compilation: Linux Assam with gcc

To compile:

Gcc -o a mips.c

To run:

디나디아차리스타찬드라
32185144

/.a

7. Screen Capture

simple.bin

```
nadia18@assam:~/mips$ ./a
Fet: [0x00000000]: 27BDFFF8
Dec:     opcode: 9, rs: 1d, rt: 1d, rd: 1f, shamt: 3e, funct: 38, imm fff8, jump imm:3bdfff8
ADDIU    R31 (0xffffffff) = R29 + (-8)
Fet: [0x00000004]: AFBE0004
Dec:     opcode: 2b, rs: 1d, rt: 1e, rd: 0, shamt: 0, funct: 4, imm 4, jump imm:3be0004
SW       M[R0 (0x0) + 0] = R30 (0x0)
Fet: [0x00000008]: 03A0F021
Dec:     opcode: 0, rs: 1d, rt: 0, rd: 1e, shamt: 0, funct: 21, imm f021, jump imm:3a0f021
ADDU     R30 (0xfffff8) = R29 + R0
Fet: [0x0000000C]: 00000000
Dec:     opcode: 0, rs: 0, rt: 0, rd: 0, shamt: 0, funct: 0, imm 0, jump imm:0
SLL      R0 (0x0) = R0 (0x0) << 0
Fet: [0x00000010]: 03C0E821
Dec:     opcode: 0, rs: 1e, rt: 0, rd: 1d, shamt: 0, funct: 21, imm e821, jump imm:3c0e821
ADDU     R29 (0xfffff8) = R30 + R0
Fet: [0x00000014]: 8FBE0004
Dec:     opcode: 23, rs: 1d, rt: 1e, rd: 0, shamt: 0, funct: 4, imm 4, jump imm:3be0004
LW       R30 (0x4602080) = M[R0 + (0)]
Fet: [0x00000018]: 27BD0008
Dec:     opcode: 9, rs: 1d, rt: 1d, rd: 0, shamt: 0, funct: 8, imm 8, jump imm:3bd0008
ADDIU    R0 (0x0) = R29 + (0)
Fet: [0x0000001C]: 03E00008
Dec:     opcode: 0, rs: 1f, rt: 0, rd: 0, shamt: 0, funct: 8, imm 8, jump imm:3e00008
JR       PC = R31 (0xffffffff)
nadia18@assam:~/mips$
```

simple2.bin

```
nadia18@assam:~/mips$ ./a
Fet: [0x00000000]: 27BDFFE8
Dec:     opcode: 9, rs: 1d, rt: 1d, rd: 1f, shamt: 3e, funct: 28, imm ffe8, jump imm:3bdffe8
ADDIU    R31 (0xffffffff) = R29 + (-24)
Fet: [0x00000004]: AFBE0014
Dec:     opcode: 2b, rs: 1d, rt: 1e, rd: 0, shamt: 0, funct: 14, imm 14, jump imm:3be0014
SW       M[R0 (0x0) + 0] = R30 (0x0)
Fet: [0x00000008]: 03A0F021
Dec:     opcode: 0, rs: 1d, rt: 0, rd: 1e, shamt: 0, funct: 21, imm f021, jump imm:3a0f021
ADDU     R30 (0xfffffe8) = R29 + R0
Fet: [0x0000000C]: 24020064
Dec:     opcode: 9, rs: 0, rt: 2, rd: 0, shamt: 2, funct: 24, imm 64, jump imm:20064
ADDIU    R0 (0x0) = R0 + (0)
Fet: [0x00000010]: AFC20008
Dec:     opcode: 2b, rs: 1e, rt: 2, rd: 0, shamt: 0, funct: 8, imm 8, jump imm:3c20008
SW       M[R0 (0x0) + 0] = R2 (0x0)
Fet: [0x00000014]: 8FC20008
Dec:     opcode: 23, rs: 1e, rt: 2, rd: 0, shamt: 0, funct: 8, imm 8, jump imm:3c20008
LW       R2 (0x4602040) = M[R0 + (0)]
Fet: [0x00000018]: 03C0E821
Dec:     opcode: 0, rs: 1e, rt: 0, rd: 1d, shamt: 0, funct: 21, imm e821, jump imm:3c0e821
ADDU     R29 (0xfffffe8) = R30 + R0
Fet: [0x0000001C]: 8FBE0014
Dec:     opcode: 23, rs: 1d, rt: 1e, rd: 0, shamt: 0, funct: 14, imm 14, jump imm:3be0014
LW       R30 (0x4602040) = M[R0 + (0)]
Fet: [0x00000020]: 27BD0018
Dec:     opcode: 9, rs: 1d, rt: 1d, rd: 0, shamt: 0, funct: 18, imm 18, jump imm:3bd0018
ADDIU    R0 (0x0) = R29 + (0)
Fet: [0x00000024]: 03E00008
Dec:     opcode: 0, rs: 1f, rt: 0, rd: 0, shamt: 0, funct: 8, imm 8, jump imm:3e00008
JR       PC = R31 (0xffffffff)
nadia18@assam:~/mips$
```

simple3.bin

```
nadia18@assam:~/mips$ ./a
Fet: [0x00000000]: 27BDFFE8
Dec:     opcode: 9, rs: 1d, rt: 1d, rd: 1f, shamt: 3e, funct: 28, imm ffe8, jump imm:3bdffe8
ADDIU    R31 (0xffffffff) = R29 + (-24)
Fet: [0x00000004]: AFBE0014
Dec:     opcode: 2b, rs: 1d, rt: 1e, rd: 0, shamt: 0, funct: 14, imm 14, jump imm:3be0014
SW       M[R0 (0x0) + 0] = R30 (0x0)
Fet: [0x00000008]: 03A0F021
Dec:     opcode: 0, rs: 1d, rt: 0, rd: 1e, shamt: 0, funct: 21, imm f021, jump imm:3a0f021
ADDU     R30 (0xfffe8) = R29 + R0
Fet: [0x0000000C]: AFC00008
Dec:     opcode: 2b, rs: 1e, rt: 0, rd: 0, shamt: 0, funct: 8, imm 8, jump imm:3c00008
SW       M[R0 (0x0) + 0] = R0 (0x0)
Fet: [0x00000010]: AFC0000C
Dec:     opcode: 2b, rs: 1e, rt: 0, rd: 0, shamt: 0, funct: c, imm c, jump imm:3c0000c
SW       M[R0 (0x0) + 0] = R0 (0x0)
Fet: [0x00000014]: AFC00008
Dec:     opcode: 2b, rs: 1e, rt: 0, rd: 0, shamt: 0, funct: 8, imm 8, jump imm:3c00008
SW       M[R0 (0x0) + 0] = R0 (0x0)
Fet: [0x00000018]: 08000011
Dec:     opcode: 2, rs: 0, rt: 0, rd: 0, shamt: 0, funct: 11, imm 11, jump imm:11
J        PC = Jump Address 0x44
Fet: [0x00000044]: 8FC20008
Dec:     opcode: 23, rs: 1e, rt: 2, rd: 0, shamt: 0, funct: 8, imm 8, jump imm:3c20008
LW       R2 (0x4602040) = M[R0 + (0)]
Fet: [0x00000048]: 00000000
Dec:     opcode: 0, rs: 0, rt: 0, rd: 0, shamt: 0, funct: 0, imm 0, jump imm:0
SLL      R0 (0x0) = R0 (0x0) << 0
Fet: [0x0000004C]: 28420065
Dec:     opcode: a, rs: 2, rt: 2, rd: 0, shamt: 2, funct: 25, imm 65, jump imm:420065
SLTI     R2 (0x0) = ((R2 < (0) ? 1 : 0)
nadia18@assam:~/mips$
```

8. Personal Feelings

I feel okay after finishing this project even though I feel that this project is also hard the same as the last project that I did before because at first I didn't know how to make a MIPS simulator but now I am able to make it using C. And doing it by structuring the program makes it a lot easier to read and to understand because if there is a mistake and it is not structured it will be a lot harder to find the mistakes and I've realized it after making this program in structures. I feel quite proud on how I can execute the program however, I would like to learn how to execute it in a more simple and easier way because with the one I wrote, it is quite hard to read. Not only that I would like to learn more on how to improve my simulator and be able to satisfy all of the additional implementations from the assignment if there are any chances in the future.