ESE 345: Computer Architecture

Pipelined SIMD Multimedia Unit Design

Part 2: Pipelining and Processor Control

Kyle Han 113799110

Summer Wang 113961753

Table of Contents

About

This project aims to develop an HDL model of a four-stage multimedia unit and its modules. The unit is designed to execute a custom instruction set and includes an instruction buffer, a register file, an ALU, and a forwarding unit to handle data hazards. The goals include creating synthesizable VHDL code, developing an assembler in C to convert assembly files to machine code, and generating simulation results with waveforms and results files. This report outlines the design procedure for each module, including the assembler, instruction buffer, register file, forwarding unit, ALU, and write-back unit. Testbenches have been implemented to validate the functionality of each entity. The pipelined multimedia SIMD unit is presented through a block diagram, and simulation results are discussed. The project's additional feature is the ability to output the unit's status at each clock cycle to a results file.

Goals

1. We would like to develop the HDL model of a four-stage multimedia unit and its modules. This unit must execute the custom instruction set provided. The model must include an instruction buffer that holds a maximum of 64 25-bit instructions, a register file that simulates 32 128-bit registers, an ALU, and a forwarding unit to handle data hazards.
2. During each clock cycle,the unit must output its current state at every stage to a file to be inspected.
3. An assembler must be written in any language of choice that converts an assembly file to machine code to run through the HDL model.
4. As an extra goal, we wanted the VHDL written to be synthesizable. Most of the code written is synthesizable, with a few exceptions. This is a reach goal that is not within the scope of the assigned project. However, we felt that it would be a nice touch.

Design Procedure

Assembler

The Assembler was written in C. Its intention is to take a text file containing the custom assembly language and convert it to the 25-bit machine code that will be used by the instruction buffer.

To design the assembler, we used a tokenizer to separate the assembly language into its subparts. After tokenizing, the opcode is fitted together based on searching the machine code for specific patterns. For example, load immediate instructions are denoted by bit 24 being set to 0. R4 instructions are denoted by “10” in bits [24:23]. By searching for patterns, we can determine the machine code to be outputted.

Custom functions, such as comments (Denoted as //), register references (Referred to as r12 for register 12), and disregarding empty lines were added to ease development. The final instruction count will also be printed to the console.

Instruction Buffer

The instruction buffer was designed to take a string and clock as an input. The string is used to dictate the file containing the machine code, while the clock increments the program counter. On the first clock cycle, the file is opened and read in its entirety, copying its contents into an array of 25-bit std_logic_vectors. Program Counter (PC) is then set as 0. Every following rising edge, PC is incremented by 1. Instruction out is controlled using a dataflow model.

The code for the **instruction buffer** can be found in the appendix.

Register File

The register file is an unclocked entity that writes to an array of vectors. It is used as memory, or RAM in our instance. There are 32 registers, each 128-bits long. These registers can be read from using a 5-bit address input to the register file, whereby the value of the register will be output through a port. The registers can be written to using the same addressing and value. However, a writeEnable signal must be set to '1' to write to a register successfully. When set to '0', nothing will be written to the registers.

The code for the **register file** can be found in the appendix.

Forwarding Unit

The forwarding unit exists to ensure that any data hazards that may appear as a pipelined processor is handled. These hazards exist when there are two instructions in a row that may affect each other. For ease of reference, the first instruction will be called **inst1**. The second instruction is called **inst2**. The following conditions result in a hazard that the forwarding unit handles.

1. A data hazard exists anytime **inst1** is a load immediate and **inst2** is a load immediate of the same Rd.
2. A data hazard exists anytime **inst1**'s rd is used as an input to **inst2** (Either Rs1, Rs2, or Rs3).
    a. However, this fails when **inst1** is a load immediate and **inst2** is any other function.
    b. This also fails when **inst1** is a nop and **inst2** is any other function.

The code for the **forwarding unit** can be found in the appendix.

ALU

The ALU was designed in part 1 of this project. To see that report, please refer to the previous report.

The code for the **ALU** can be found in the previous report.

Write Back Unit

The write back unit is a simple unit that doesn't do much. Asynchronously, it sets the Write Enable signal to a 0 when the machine code for **nop** is found. Every other instruction writes something and therefore must have the signal set to 1. The data and address are then forwarded to the register file to be written back to the register file.

The code for the **write back unit** can be found in the appendix.

Stage Registers

The stage registers are a clocked entity that form our varying stages. Every other entity, besides the instruction buffer, is unclocked, and therefore operates combinationally. That is to say, their outputs can change anytime any input is changed. With the stage registers, however, they can only output on a positive clock edge. On a positive clock edge, they will forward the data at its inputs to the outputs. These outputs will not change until the next rising clock.

The stage registers are used 3 times in this structural architecture. They are named after the stages they represent.

ID/IF = Instruction Decode/Instruction Fetch
IF/EX = Instruction Fetch/Execute
EX/WB = Execute/Write Back

The ID/IF register is slightly special in the following two ways.

1. It functions to slice the machine code into the formatted parts. For example, rs3 will get the bits[19:15] of the machine code. The same is done for rs2, rs1, and rd for the following sets of 5 bits.

2. The ID/IF register also adjusts rs1 based on the opcode. Due to the special nature of the instruction set, rd needs to receive the data before loading the immediate into the register. This is done through rs1. In this entity, if the opcode is a load immediate function, it will send the address of rd through rs1.

The code for **all 3 stage registers** can be found in the appendix.

Test Benches

Testbenches have been written to test functionality of all the entities besides the stage registers. Through the testbenches, we have confirmed that all entities function independently as expected, eliminating one possibility of errors when designing the structural unit. The testbenches have signals that connect to every port on the entities, allowing us to control the inputs and monitor the outputs to manually verify our design.

Pipelined Multimedia SIMD Unit

The unit was designed using the block diagram shown in Figure 1. This diagram was first developed to guide development of the multimedia unit and shows each entity's inputs and outputs. The data paths and control signals are also shown, along with the corresponding bit length. The entity is written in a structural modeling style, simply connecting the various entities together through signals. The block diagram in the appendix shows the result of the **Code2Graphics** wizard within Aldec Active HDL.

The entity also has an additional feature of writing the output to a file. This was required to meet goal 2. The process runs at every clock cycle and prints the status of each of the entities in the unit. This unit was chosen to host the results.txt file since all the signals have already been declared and used to host/move data between entities.

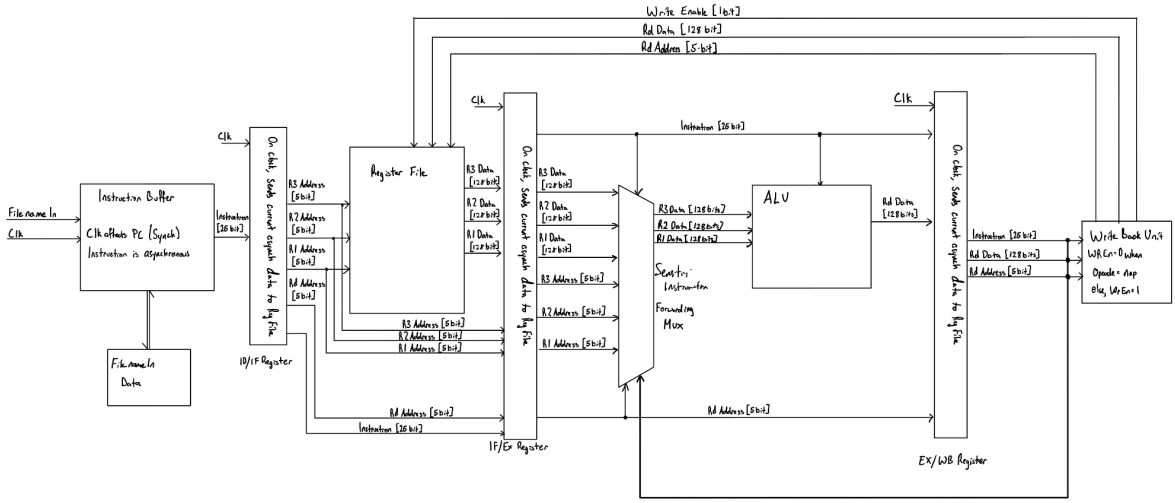
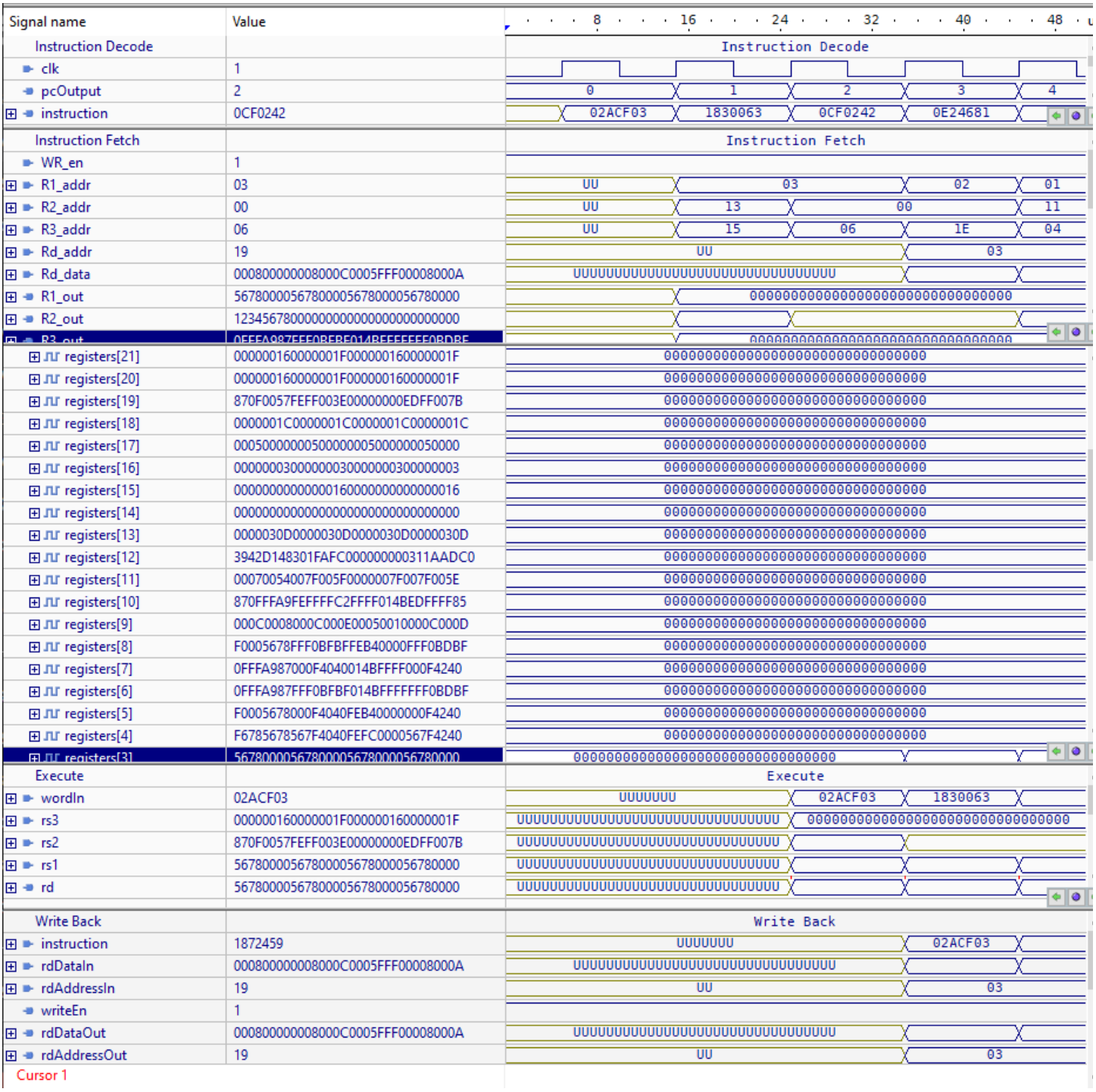


Figure 1: A diagram of the pipelined SIMD unit's entities and their connections.

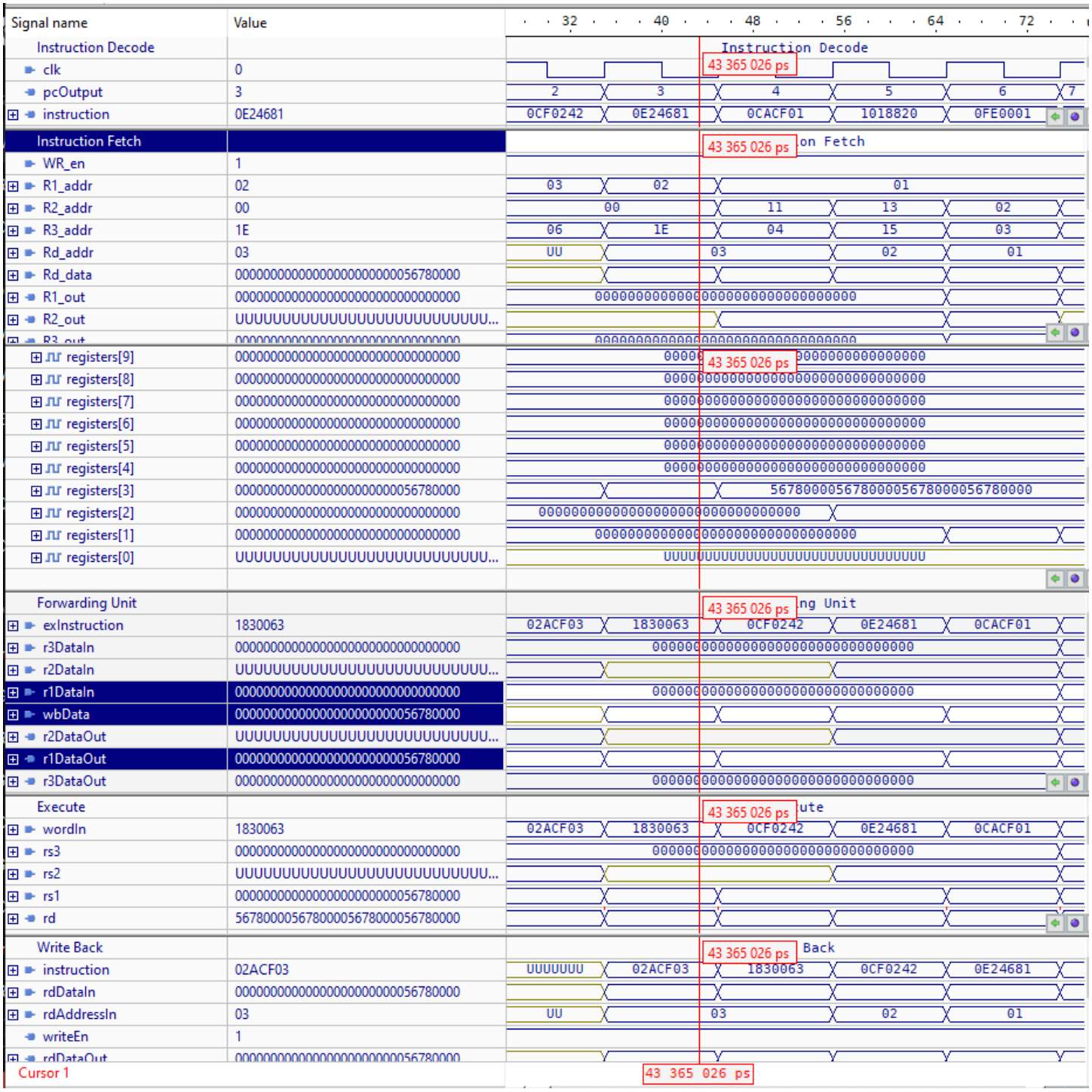
Simulation Results

Waveform Screenshot: Instruction Propagation



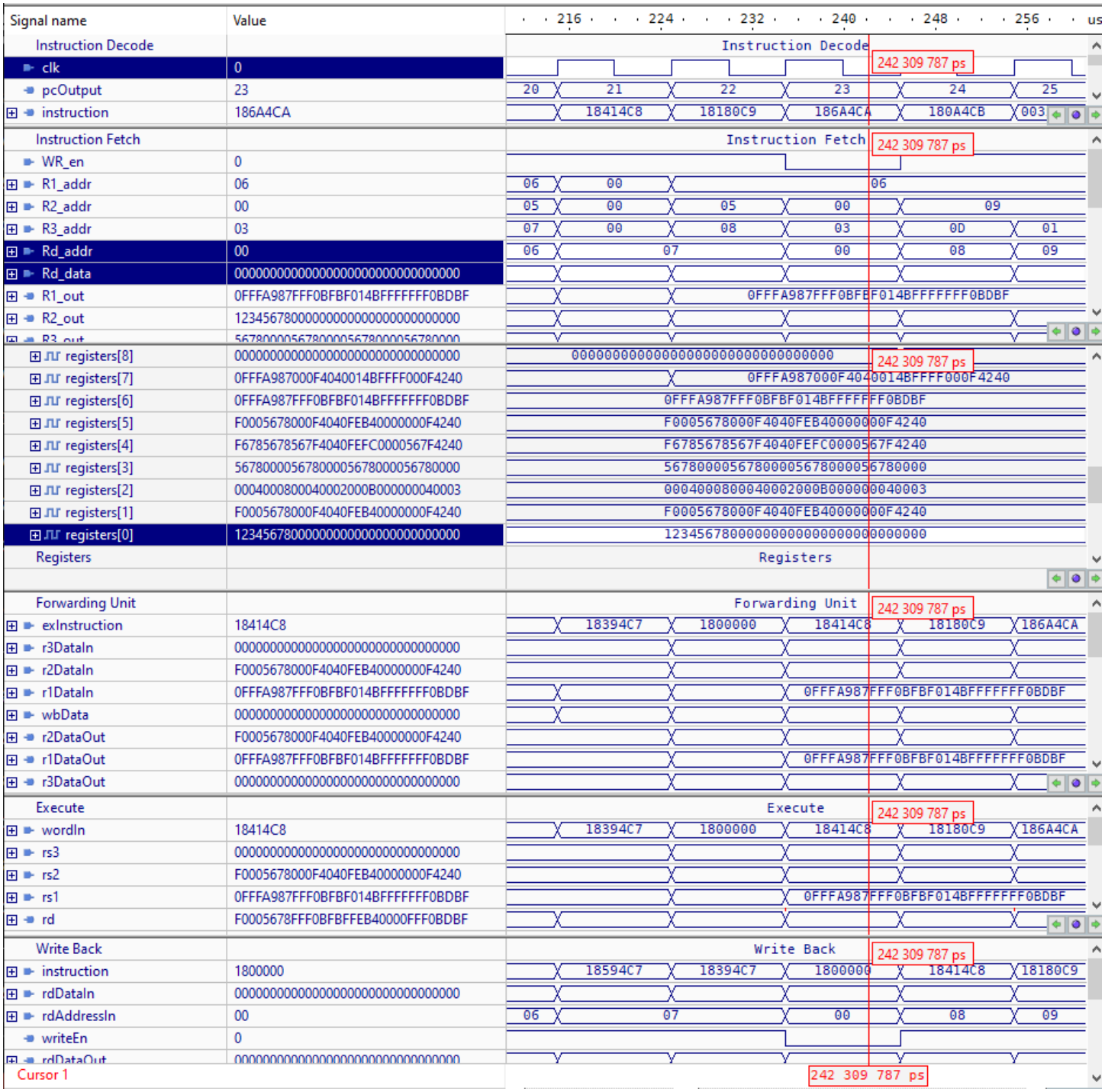
Through the image shown, we can see the propagation of the instruction. In the instruction decode, the first rising edge allows the first opcode through. Each of the following stages have a one clock delay until it receives the first instruction.

Waveform Screenshot: Functioning Forwarding Unit



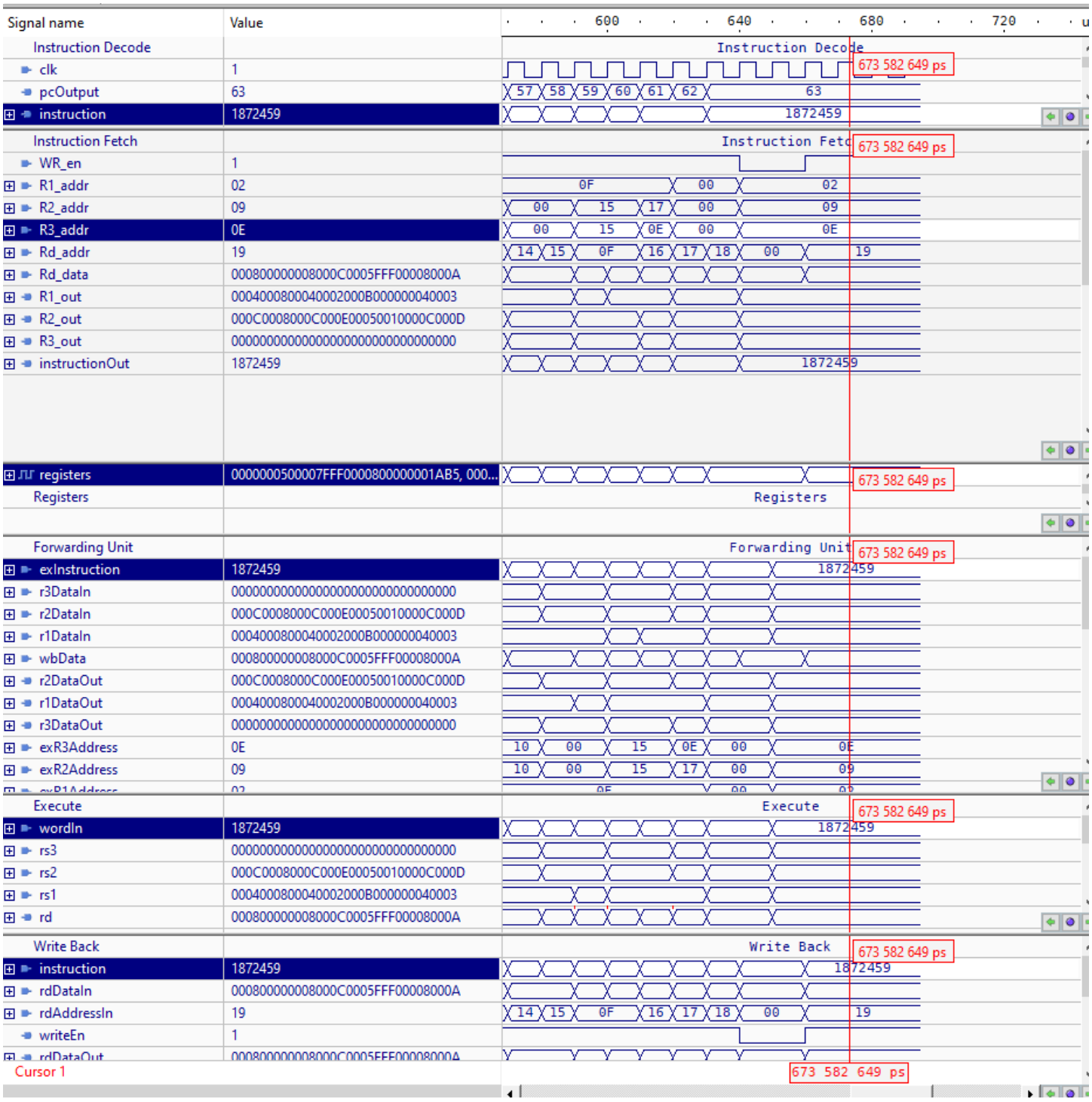
At the current cursor position, we see that a data hazard exists. The highlighted signals shows that r1DataIn does not equal r1DataOut. What we can observe is the data from wbData (data from the writeback unit) being forwarded through to r1DataOut.

Waveform Screenshot: Nop Write Back



The writeback unit is currently dealing with a nop (Opcode 0x0180_0000). Thus, we see the same signal get reflected back to the Instruction fetch stage (That is, the register file). We see through the highlighted signals that the Rd, the address is set to register 0. We also see that the data is set to 0. However, register[0] never gets updated with the value, proving our writeEnable line successfully controls the register file.

Waveform Screenshot: Ending Instruction Propagation



This image shows the propagation nature of the pipelined SIMD unit. At the end of the instructions, PC stays stuck at 64, allowing the same instruction to propagate through to the writeback unit.

Instruction Input

Every function that the ALU is capable of performing has been used to generate the machine code. The assembly instructions appended to this document show the expected inputs and outputs through commented code. The expected outputs have been compiled and is shown below for the reader's convenience. A comparison to the output of the SIMD unit is also shown.

Expected Results

At the end of program execution, contents in the register file are outputed with register names shown below. The left image shows expected results and the right image shows actual outputs. Produced output matches the expected results.

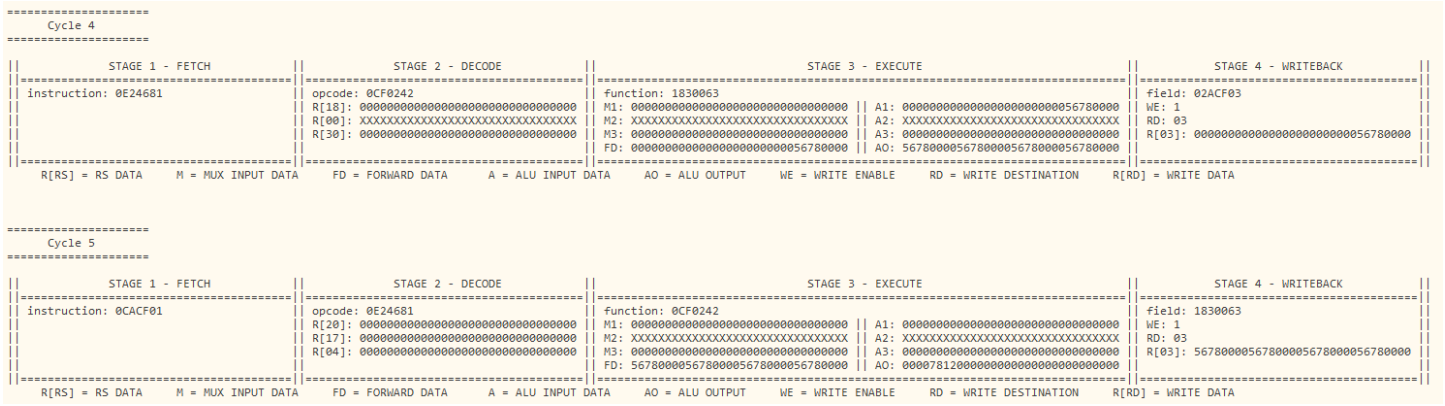
```
R31 0000000500007FFF0000800000001AB5
R30 0000000700000005000080000009263
R29 0000000C00007FFF000080000000AD18
R28 6543210100000007FFFFFFF000C0000
R27 6543210100000007FFFFFFF000F4240
R26 CA86420200000000FFFFFFFE001B4240
R25 0000000000000000000000000008000A
R24 0000000000001CE00000000000001B8
R23 0000000000001CE00000000000001CE
R22 0000000000003AB00000000000003AB
R21 00000016000001F00000016000001F
R20 00000016000001F00000016000001F
R19 870F0057FEFF003E00000000EDFF007B
R18 0000001C0000001C0000001C0000001C
R17 0005000000050000000500000050000
R16 0000000300000003000000300000003
R15 0000000000000016000000000000016
R14 0000000000000000000000000000000
R13 0000030D0000030D0000030D0000030D
R12 3942D148301FAFC000000000311AADC0
R11 00070054007F005F0000007F007F005E
R10 870FFFA9FEFFFFC2FFFF014BEDFFFF85
R9  000C0008000C000E00050010000C000D
R8  F0005678FFF0BFBFFEB40000FFF0BDBF
R7  0FFFA987000F4040014BFFFF000F4240
R6  0FFFA987FFF0BFBF014BFFFFFFF0BDBF
R5  F0005678000F4040FEB40000000F4240
R4  F6785678567F4040FEFC0000567F4240
R3  5678000056780000567800005678000
R2  0004000800040002000B000000040003
R1  F0005678000F4040FEB40000000F4240
R0  12345678000000000000000000000000
```

```
R0  12345678000000000000000000000000
R1  F0005678000F4040FEB40000000F4240
R2  0004000800040002000B000000040003
R3  5678000056780000567800005678000
R4  F6785678567F4040FEFC0000567F4240
R5  F0005678000F4040FEB40000000F4240
R6  0FFFA987FFF0BFBF014BFFFFFFF0BDBF
R7  0FFFA987000F4040014BFFFF000F4240
R8  F0005678FFF0BFBFFEB40000FFF0BDBF
R9  000C0008000C000E00050010000C000D
R10 870FFFA9FEFFFFC2FFFF014BEDFFFF85
R11 00070054007F005F0000007F007F005E
R12 3942D148301FAFC000000000311AADC0
R13 0000030D0000030D0000030D0000030D
R14 0000000000000000000000000000000
R15 0000000000000016000000000000016
R16 0000000300000003000000300000003
R17 0005000000050000000500000050000
R18 0000001C0000001C0000001C0000001C
R19 870F0057FEFF003E00000000EDFF007B
R20 00000016000001F00000016000001F
R21 00000016000001F00000016000001F
R22 0000000000003AB00000000000003AB
R23 0000000000001CE00000000000001CE
R24 0000000000001CE00000000000001B8
R25 0000000000000000000000000008000A
R26 CA86420200000000FFFFFFFE001B4240
R27 6543210100000007FFFFFFF000F4240
R28 6543210100000007FFFFFFF000C0000
R29 0000000C00007FFF000080000000AD18
R30 0000000700000005000080000009263
R31 0000000500007FFF0000800000001AB5
```

Results File

The results file is shown below. Information regarding each stage is shown in a table format for easier reading. For stage 1 instruction fetch stage, instruction in hexadecimal value is displayed. For stage 2 instruction decode stage, opcode, register 1 through register 3 are displayed. For stage 3 execute stage, opcode is also displayed in hexadecimal value, plus M1 - M3, as 3 registers going into IF/EX register, r1, r2, r3 inputs into the ALU as A1 - A3. FD is the data being forwarded. For stage 4 write back stage, opcode is shown in addition to write_en value, rd register, and rd data value.

Currently the cycle 4 shown have 1830063 which is instruction `bcw r3, r3` in stage 3 and `li r3, 1, 0x5678` in stage 4. R3 is to be written at stage 4 while to be used at stage 3. At stage 3, r3 has not been written back to register file so r3 does not have the newest value. There is a data hazard. The forwarding unit detects the hazard and forward data back so the next instruction that needs r3 can get the most updated data. In Cycle 4, A1 (r3) is being forwarded, also shown at stage 4 r[03] with its data. FD is the same as A1 which can also show r3 is bring forwarded. Ao has the output of the ALU which is the broadbasted output. At the next instruction forwarding happens again, so FD gets the Ao value, and r[03] shows the same value as well.



Conclusion

Through this project, we learned about pipelining as well as data forwarding, testbench design, reinforced knowledge of computer architecture though actively writing and debugging behavioral and structural designs using VHDL. Extensive tests have been done to ensure data hazards are handled properly by the data forwarding unit. We gained a lot of familiarity with VHDL via this project. As a result, Pipelined SIMD multimedia unit was successfully built. It was a great learning experience designing, building, and testing this project with the goal of writing synthesizable code in mind.

Appendix

Assembler.c

```c
/*
================================================================================
  File:           345assembler.c
  Description:    Assembles a custom instruction set into machine code for a pipelined SIMD unit

  Author:         Kyle Han and Summer Wang
  Company:        Stony Brook University - ESE 345 Computer Architecture
  Email:          your.email@example.com
  Date:           December 3, 2023

  Version:        1.0

  License:        This code is released under the MIT License.

  Notes:          This code is meant for terminal use. When compiled to an executable, it can be run standalone, but will not provide any feedback

  Usage:

  Compilation:    Compile using standard gcc.

  Dependencies:

================================================================================
*/



#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>



void removeChar(char *str, char c) {
    //If the string is null, do nothing.
    //Otherwise, continue on
    if (str != NULL) {
        int i, j;
        int len = strlen(str);
        for (i = j = 0; i < len; i++) {
            if (str[i] != c) {
                str[j++] = str[i];
            }
        }
        str[j] = '\0';
    }
}

void slice(const char* str, char* result, size_t start, size_t end) {
    strncpy(result, str + start, end - start);
}
```

```
54  //This function will convert a long to a character array of a binary
    equivalent
55  char* long_to_binary(unsigned long k)
56
57  {
58          static char c[65];
59          c[0] = '\0';
60
61          unsigned long val;
62          for (val = 1UL << (sizeof(unsigned long)*8-1); val > 0; val >>= 1)
    {
63              strcat(c, ((k & val) == val) ? "1" : "0");
64          }
65          return c;
66      }
67
68  //This function will take a base m character string and return a n bit
    binary value
69  //Ex: Input 7 --> 00111
70  //Ex: Input 31 --> 11111
71  //Ex: Input FFFF --> 1111111111111111
72  //Ex: Input 0000 --> 0000000000000000
73  char* char2Bin(char* charInput, int n, int m) {
74      char* pEnd;
75
76      //Technically I malloc here but never free.. I don't know where to,
    since it'll return the address and immedeately get used.
77      char* binaryValue = malloc (sizeof(char)*n);
78
79      //Convert the given string to a long
80      long int li1 = strtol(charInput, &pEnd, m);
81
82      //A little checksum - if the long li1 is greater than the bits that
    2**n can hold, throw a warning in the console.
83      if (li1 > pow(2, n) - 1) {
84          printf("Error: %li cannot fit within a %d-bit binary", li1, n);
85          printf("\nExiting with error");
86          exit(1);
87      }
88
89      //Convert the long to a binary encoded string
90      pEnd = long_to_binary(li1);
91
92      //Limit to n bits
93      //For some reason, when I copy-paste strlen(pEnd)-5 directly into the
    slice parameter, I get a segmentation fault
94      int tmp = strlen(pEnd)-n;
95      slice(pEnd, binaryValue, tmp, strlen(pEnd));
96      binaryValue[n] = '\0';
97
98      return binaryValue;
99  }
100
101 int main() {
102     //Open a inputFile called "input.txt" in read only mode
103     FILE* inputFile = fopen("assembly.txt", "r");
104
```

```
105     //Open an outputFile called "output.txt" in write only mode
106     FILE* outputFile = fopen("machineCode.txt", "w");
107     char line[100];
108     int lineIndex = 0;
109 
110     //Go line by line (Or until we hit 64 lines, our limit for the
    instruction buffer)
111     while (fgets(line, sizeof(line), inputFile) && lineIndex < 64) {
112         //There can be up to 5 arguements per line, each limited to 7
    characters long + 1 null terminating character
113         char* args[5] = { '\0' };
114 
115         //The OPCode to be printed to the outputfile. Is 25 characters
    long + 1 null terminating character
116         char opcodeOut[25+1] = { '\0' };
117         int spaceIndex = 0;
118         int currentArg = 0;
119 
120         printf("\n\n%s", line);
121 
122         //Check for comments, denoted by a "//" as the first 2 characters
123         //Also check for just an empty line.
124         if (line[0] != '/' && line[1] != '/' && line[0] != '\n'){
125                 //Remove the commas and the rs (which stand for registers)
126             removeChar(line, ',');
127             removeChar(line, '\n');
128 
129             //First, make sure that everything is lowercase
130             for (int i = 0; line[i]; i++) {
131                 line[i] = tolower(line[i]);
132             }
133 /*
134             //Tokenize Version 1
135             //Going char by char in the line, we will also parse out the
    arguements to put in the 2d array args
136             for (int i = 0; line[i]; i++) {
137                 //First, make sure that everything is lowercase
138                 line[i] = tolower(line[i]);
139 
140                 //If we detect a space or a newline char, we are at the
    end of the word
141                 //spaceIndex identifies the last space, while i represents
    the current space.
142                 //Between these is the arguement to be parsed. Copy that
    over to the args array
143                 if (line[i] == ' ' || line[i] == '\n') {
144                     //Copy the string to the args array, starting from the
    spaceIndex character and for i-spaceIndex characters
145                     strncpy(args[currentArg], line+spaceIndex,
    i-spaceIndex);
146                     //strncpy does not add a null terminator. We must do
    that (Even though the whole array is already initialized to \0, this is
    just good practice)
147                     args[currentArg][i-spaceIndex] = '\0';
148                     spaceIndex = i+1;
149                     currentArg++;
150                 }
```

```
151             }
152             */
153 
154 
155             //Using built in tokenizer
156             int j = 0;
157             char* token;
158             char delimiter[] = " ";
159             token = strtok(line, delimiter);
160             args[j] = token;
161             j++;
162             while (token) {
163                 token = strtok(NULL, delimiter);
164                 args[j] = token;
165                 j++;
166             }
167 
168 
169             //Arguments have been seperated by spaces
170             //Remove the r (Which stands for registers) from every
    arguement beyond arg[0]
171             //removeChar(args[1], 'r');
172 
173 
174             for (int i = 1; i < 5; i++) {
175                 removeChar(args[i], 'r');
176             }
177             //There's probably a better way to do this. I can think of
    one: A hashmap and a switch statement.
178             //However, since this is a simple instructionset with minimal
    instructions, I have elected to use a if-else ladder
179             //If this was a more complicated assembler, the smarter way
    would be a hashmap
180             if(strcmp(args[0], "li") == 0) {
181                 //For some reason, without this printf statement, the
    program segfaults. THIS IS IMPORTANT
182                 strcat(opcodeOut, "0");
183                 strcat(opcodeOut, char2Bin(args[2], 3, 10));
184                 strcat(opcodeOut, char2Bin(args[3], 16, 16));
185                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
186             } else if (strcmp(args[0], "simal") == 0){
187                 //Do Something
188                 strcat(opcodeOut, "10");
189                 strcat(opcodeOut, "000");
190                 strcat(opcodeOut, char2Bin(args[4], 5, 10));
191                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
192                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
193                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
194             } else if (strcmp(args[0], "simah") == 0){
195                 //Do Something
196                 strcat(opcodeOut, "10");
197                 strcat(opcodeOut, "001");
198                 strcat(opcodeOut, char2Bin(args[4], 5, 10));
199                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
200                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
201                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
202             } else if (strcmp(args[0], "simsl") == 0){
```

```
203                 //Do Something
204                 strcat(opcodeOut, "10");
205                 strcat(opcodeOut, "010");
206                 strcat(opcodeOut, char2Bin(args[4], 5, 10));
207                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
208                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
209                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
210             } else if (strcmp(args[0], "simsh") == 0){
211                 //Do Something
212                 strcat(opcodeOut, "10");
213                 strcat(opcodeOut, "011");
214                 strcat(opcodeOut, char2Bin(args[4], 5, 10));
215                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
216                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
217                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
218             } else if (strcmp(args[0], "slmal") == 0){
219                 //Do Something
220                 strcat(opcodeOut, "10");
221                 strcat(opcodeOut, "100");
222                 strcat(opcodeOut, char2Bin(args[4], 5, 10));
223                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
224                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
225                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
226             } else if (strcmp(args[0], "slmah") == 0){
227                 //Do Something
228                 strcat(opcodeOut, "10");
229                 strcat(opcodeOut, "101");
230                 strcat(opcodeOut, char2Bin(args[4], 5, 10));
231                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
232                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
233                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
234             } else if (strcmp(args[0], "slmsl") == 0){
235                 //Do Something
236                 strcat(opcodeOut, "10");
237                 strcat(opcodeOut, "110");
238                 strcat(opcodeOut, char2Bin(args[4], 5, 10));
239                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
240                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
241                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
242             } else if (strcmp(args[0], "slmsh") == 0){
243                 //Do Something
244                 strcat(opcodeOut, "10");
245                 strcat(opcodeOut, "111");
246                 strcat(opcodeOut, char2Bin(args[4], 5, 10));
247                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
248                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
249                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
250             } else if (strcmp(args[0], "nop") == 0){
251                 strcat(opcodeOut, "11");
252                 strcat(opcodeOut, "00000000");
253                 strcat(opcodeOut, "00000");
254                 strcat(opcodeOut, "00000");
255                 strcat(opcodeOut, "00000");
256             } else if (strcmp(args[0], "shrhi") == 0){
257                 strcat(opcodeOut, "11");
258                 strcat(opcodeOut, "00000001");
259                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
```

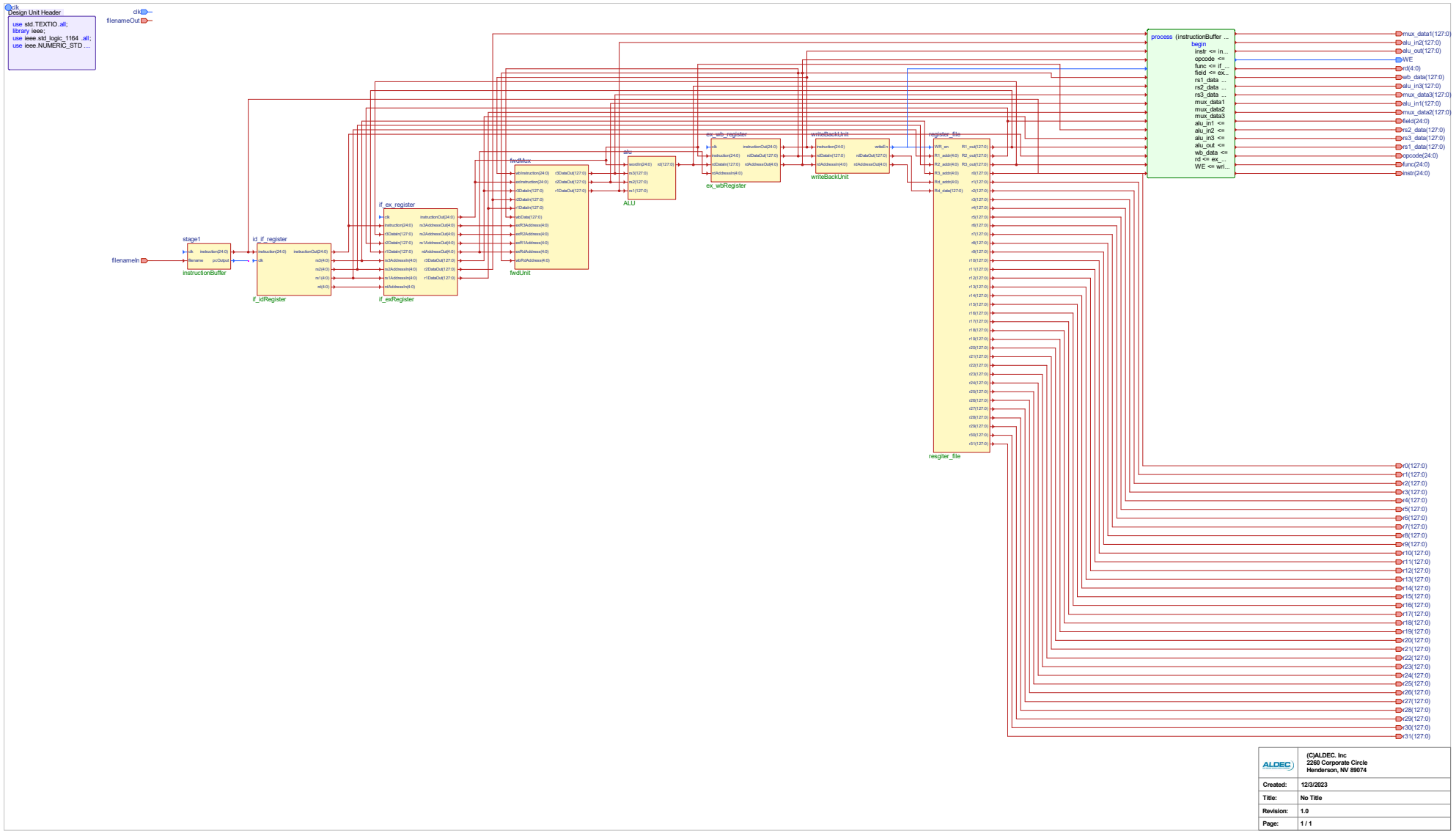
```
260                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
261                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
262             } else if (strcmp(args[0], "au") == 0){
263                 strcat(opcodeOut, "11");
264                 strcat(opcodeOut, "00000010");
265                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
266                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
267                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
268             } else if (strcmp(args[0], "cnt1h") == 0){
269                 //Do Something
270                 strcat(opcodeOut, "11");
271                 strcat(opcodeOut, "00000011");
272 
273     //cnt1h doesn't have 3 registers - only 2. Therefore, replace rs2 with
    all 0s
274                 strcat(opcodeOut, "00000");
275                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
276                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
277             } else if (strcmp(args[0], "ahs") == 0){
278                 //Do Something
279                 strcat(opcodeOut, "11");
280                 strcat(opcodeOut, "00000100");
281 
282                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
283                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
284                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
285             } else if (strcmp(args[0], "or") == 0){
286                 //Do Something
287                 strcat(opcodeOut, "11");
288                 strcat(opcodeOut, "00000101");
289 
290                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
291                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
292                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
293             } else if (strcmp(args[0], "bcw") == 0){
294                 //Do Something
295                 strcat(opcodeOut, "11");
296                 strcat(opcodeOut, "00000110");
297 
298                 //bcw doesn't have 3 registers - only 2. Therefore,
    replace rs2 with all 0s
299                 strcat(opcodeOut, "00000");
300                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
301                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
302             } else if (strcmp(args[0], "maxws") == 0){
303                 //Do Something
304                 strcat(opcodeOut, "11");
305                 strcat(opcodeOut, "00000111");
306 
307                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
308                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
309                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
310             } else if (strcmp(args[0], "minws") == 0){
311                 //Do Something
312                 strcat(opcodeOut, "11");
313                 strcat(opcodeOut, "00001000");
314 
```

```
315                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
316                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
317                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
318             } else if (strcmp(args[0], "mlhu") == 0){
319                 //Do Something
320                 strcat(opcodeOut, "11");
321                 strcat(opcodeOut, "00001001");
322 
323                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
324                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
325                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
326             } else if (strcmp(args[0], "mlhss") == 0){
327                 //Do Something
328                 strcat(opcodeOut, "11");
329                 strcat(opcodeOut, "00001010");
330 
331                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
332                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
333                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
334             } else if (strcmp(args[0], "and") == 0){
335                 //Do Something
336                 strcat(opcodeOut, "11");
337                 strcat(opcodeOut, "00001011");
338 
339                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
340                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
341                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
342             } else if (strcmp(args[0], "invb") == 0){
343                 //Do Something
344                 strcat(opcodeOut, "11");
345                 strcat(opcodeOut, "00001100");
346 
347                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
348                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
349                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
350             } else if (strcmp(args[0], "rotw") == 0){
351                 //Do Something
352                 strcat(opcodeOut, "11");
353                 strcat(opcodeOut, "00001101");
354 
355                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
356                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
357                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
358             } else if (strcmp(args[0], "sfwu") == 0){
359                 //Do Something
360                 strcat(opcodeOut, "11");
361                 strcat(opcodeOut, "00001110");
362 
363                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
364                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
365                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
366             } else if (strcmp(args[0], "sfhs") == 0){
367                 //Do Something
368                 strcat(opcodeOut, "11");
369                 strcat(opcodeOut, "00001111");
370 
371                 strcat(opcodeOut, char2Bin(args[3], 5, 10));
```

```
372                 strcat(opcodeOut, char2Bin(args[2], 5, 10));
373                 strcat(opcodeOut, char2Bin(args[1], 5, 10));
374             } else {
375                 printf("\nInstruction not found: %s", args[0]);
376                 exit(1);
377 
378             }
379 
380             printf("Opcode: %s", opcodeOut);
381             fprintf(outputFile, opcodeOut);
382             fprintf(outputFile, "\n");
383             lineIndex++;
384         }
385 
386 
387 
388     }
389     printf("\nThere are a total of %d instructions in this assembly file."
    , lineIndex);
390 
391     //Once we're done, fill the rest of the machine code with nop
    instructions
392     while (lineIndex < 64) {
393         fprintf(outputFile, "11000000000000000000000000\n");
394         lineIndex++;
395     }
396     fclose(inputFile);
397     fclose(outputFile);
398     return 0;
399 }
```

Code2Graphics Block Diagram

Project ALU


clk
Design Unit Header
use std.TEXTIO.all;
library ieee;
use ieee.std_logic_1164 .all;
use ieee.NUMERIC_STD ....
clk
filenameOut
stage1
filenameIn
instructionBuffer
id_if_register
if_idRegister
if_ex_register
if_exRegister
fwdMux
fwdUnit
alu
ALU
ex_wb_register
ex_wbRegister
writeBackUnit
writeBackUnit
register_file
resgiter_file
process (instructionBuffer ...
begin
instr <= in...
opcode <=
func <= if_...
field <= ex...
rs1_data ...
rs2_data ...
rs3_data ...
mux_data1
mux_data2
mux_data3
alu_in1 <=
alu_in2 <=
alu_in3 <=
alu_out <=
wb_data <=
rd <= ex_...
WE <= wri...
mux_data1(127:0)
alu_in2(127:0)
alu_out(127:0)
WE
rd(4:0)
wb_data(127:0)
alu_in3(127:0)
mux_data3(127:0)
alu_in1(127:0)
mux_data2(127:0)
field(24:0)
rs2_data(127:0)
rs3_data(127:0)
rs1_data(127:0)
opcode(24:0)
func(24:0)
instr(24:0)
r0(127:0)
r1(127:0)
r2(127:0)
r3(127:0)
r4(127:0)
r5(127:0)
r6(127:0)
r7(127:0)
r8(127:0)
r9(127:0)
r10(127:0)
r11(127:0)
r12(127:0)
r13(127:0)
r14(127:0)
r15(127:0)
r16(127:0)
r17(127:0)
r18(127:0)
r19(127:0)
r20(127:0)
r21(127:0)
r22(127:0)
r23(127:0)
r24(127:0)
r25(127:0)
r26(127:0)
r27(127:0)
r28(127:0)
r29(127:0)
r30(127:0)
r31(127:0)
ALDEC
(C)ALDEC. Inc
2260 Corporate Circle
Henderson, NV 89074
Created: 12/3/2023
Title: No Title
Revision: 1.0
Page: 1 / 1

InstructionBuffer.vhd

```
1   ----------------------------------------------------------------------------
    -------
2   -- Company: Stony Brook University - ESE 345 Computer Architecture
3   -- Engineers: Kyle Han and Summer Wang
4   --
5   -- Create Date: 11/29/2023 08:24:40 PM
6   -- Design Name:
7   -- Module Name: Instruction Buffer - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------
    -------
20
21  library IEEE;
22  library std;
23  use IEEE.STD_LOGIC_1164.ALL;
24  use std.textio.all;
25  use IEEE.NUMERIC_STD.ALL;
26  use work.all;
27
28  entity instructionBuffer is
29      Port(
30      --Inputs
31      clk : in std_logic;
32      filename: in string;
33
34      --Outputs
35      instruction: out std_logic_vector(24 downto 0);
36      pcOutput: out integer);
37  end instructionBuffer;
38
39
40  architecture Behavioral of instructionBuffer is
41      signal PC: integer := 0;
42
43
44      type instArray is array (0 to 63) of std_logic_vector(24 downto 0);
45      signal instBuffer: instArray;
46  begin
47      process(clk)
48          file inputFile : text;
49          variable lineContents : line;
50          variable i: integer := 0;
51          variable readFile: integer := 0;
52          variable tempInst: std_logic_vector(24 downto 0);
53      begin
54          --On each rising edge
55          if(rising_edge(clk)) then
```

```
56              --If the file hasn't been read into memory, read the file.
    Then, set readFile to 1 so that we won't enter.
57              if(readFile = 0) then
58                  file_open(inputFile, filename, READ_MODE);
59                  while not endfile(inputFile) loop
60                      readline(inputFile, lineContents);        -- Reads text
    line in file, stores line into line_contents
61                      read(lineContents, tempInst);  -- Reads line_contents
    and stores it into a temporary variable
62                      instBuffer(i) <= tempInst;     -- Takes the information
    from the temp variable and inserts it into the instBuffer(i), which is an
    entry in the 64 25-bit instruction set
63                      i := i + 1;
64                  end loop;
65                  file_close(inputFile);
66                  readFile := 1;
67                  PC <= 0;
68              --If the program counter has not reached the end, increment.
69              elsif(PC < 63) then
70                  PC <= PC + 1;
71              else
72                  PC <= PC;
73              end if;
74          end if;
75      end process;
76      -- At any time, the output should be whatever the instruction is
77      instruction <= instBuffer(PC);
78      pcOutput <= PC;
79
80  end Behavioral;
```

InstructionBuffer_tb.vhd

```
1   ----------------------------------------------------------------------------
    -------
2   -- Company: Stony Brook University - ESE 345 Computer Architecture
3   -- Engineers: Kyle Han and Summer Wang
4   --
5   -- Create Date: 11/29/2023 9:57:21 PM
6   -- Design Name:
7   -- Module Name: insructionBuffer_TB - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------
    -------
20
21  library IEEE;
22  library std;
23  use IEEE.STD_LOGIC_1164.ALL;
24  use std.textio.all;
25  use IEEE.NUMERIC_STD.ALL;
26  use work.all;
27  use std.env.finish;
28
29  entity instructionBuffer_tb is
30
31  end instructionBuffer_tb;
32
33  architecture behavioral of instructionBuffer_tb is
34      signal clk: std_logic := '0';
35
36      constant BL_STR :string :="machineCode.txt";
37      signal filename: string(1 to BL_STR'length) := BL_STR ;
38      --signal filename: string(1 to 16) := "machineCode.txt"; --With this
    singular line of code, vivado didn't want to simulate. The constant above
    is a workaround
39      signal instruction: std_logic_vector(24 downto 0);
40
41      constant period : time := 1 us;
42  begin
43      UUT: entity instructionBuffer
44          port map(clk => clk,
45          filename => filename,
46          instruction => instruction);
47          testing: process
48          begin
49              for i in 0 to 128 loop
50                  wait for period/2;
51                  clk <= not clk;
52              end loop;
53              std.env.finish;
```

```
54          end process;
55  end behavioral;
```

IF_IDRegister.vhd

```
1   ----------------------------------------------------------------------------
    -------
2   -- Company: Stony Brook University - ESE 345 Computer Architecture
3   -- Engineers: Kyle Han and Summer Wang
4   --
5   -- Create Date: 11/30/2023 2:33:21 PM
6   -- Design Name:
7   -- Module Name: IF/ID Register - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------
    -------
20
21
22  library IEEE;
23  library std;
24  use IEEE.STD_LOGIC_1164.ALL;
25  use std.textio.all;
26  use IEEE.NUMERIC_STD.ALL;
27  use work.all;
28
29  entity if_idRegister is
30      Port(
31      --Inputs
32      instruction: in std_logic_vector(24 downto 0);
33      clk: in std_logic;
34
35      --Outputs
36      instructionOut: out std_logic_vector(24 downto 0);
37      rs3: out std_logic_vector(4 downto 0);
38      rs2: out std_logic_vector(4 downto 0);
39      rs1: out std_logic_vector(4 downto 0);
40      rd: out std_logic_vector(4 downto 0));
41  end if_idRegister;
42
43
44  architecture behavioral of if_idRegister is
45
46  begin
47      process(clk)
48      begin
49          if(rising_edge(clk)) then
50              --On a rising clock edge, we want to see the asynchronous data
    sent out. So whatever is in instruction is split
51              instructionOut <= instruction;
52              rs3 <= instruction(19 downto 15);
53              rs2 <= instruction(14 downto 10);
54              --Rs1 = Rs1 unless we have a li instruciton. Then, rs1 gets rd.
```

```
55              rs1 <= instruction(9 downto 5) when (instruction(24) = '1')
    else
56                      instruction(4 downto 0);
57              rd <= instruction(4 downto 0);
58          end if;
59      end process;
60
61  end behavioral;
62
```

RegisterFile.vhd

```
1    library IEEE;
2    use IEEE.STD_LOGIC_1164.ALL;
3    USE ieee.numeric_std.ALL;
4
5
6    entity resgiter_file is
7    port(
8        WR_en: in std_logic;    -- Write enable
9        R1_addr: in std_logic_vector(4 downto 0);  --   R1 Address input
10       R2_addr: in std_logic_vector(4 downto 0);    -- R2 Address input
11       R3_addr: in std_logic_vector(4 downto 0);    -- R3 Address input
12       Rd_addr: in std_logic_vector(4 downto 0);  -- Rd Address input
13       Rd_data: in std_logic_vector(127 downto 0); -- Rd data in from Write
     Back
14   --  CLK: in std_logic; -- clock input for RAM
15       R1_out: out std_logic_vector(127 downto 0); -- R2 register output
16       R2_out: out std_logic_vector(127 downto 0); -- R2 register output
17       R3_out: out std_logic_vector(127 downto 0); -- Data output of RAM
18       -- below are all registers to write to file
19       r0: out std_logic_vector(127 downto 0);
20       r1: out std_logic_vector(127 downto 0);
21       r2: out std_logic_vector(127 downto 0);
22       r3: out std_logic_vector(127 downto 0);
23       r4: out std_logic_vector(127 downto 0);
24       r5: out std_logic_vector(127 downto 0);
25       r6: out std_logic_vector(127 downto 0);
26       r7: out std_logic_vector(127 downto 0);
27       r8: out std_logic_vector(127 downto 0);
28       r9: out std_logic_vector(127 downto 0);
29       r10: out std_logic_vector(127 downto 0);
30       r11: out std_logic_vector(127 downto 0);
31       r12: out std_logic_vector(127 downto 0);
32       r13: out std_logic_vector(127 downto 0);
33       r14: out std_logic_vector(127 downto 0);
34       r15: out std_logic_vector(127 downto 0);
35       r16: out std_logic_vector(127 downto 0);
36       r17: out std_logic_vector(127 downto 0);
37       r18: out std_logic_vector(127 downto 0);
38       r19: out std_logic_vector(127 downto 0);
39       r20: out std_logic_vector(127 downto 0);
40       r21: out std_logic_vector(127 downto 0);
41       r22: out std_logic_vector(127 downto 0);
42       r23: out std_logic_vector(127 downto 0);
43       r24: out std_logic_vector(127 downto 0);
44       r25: out std_logic_vector(127 downto 0);
45       r26: out std_logic_vector(127 downto 0);
46       r27: out std_logic_vector(127 downto 0);
47       r28: out std_logic_vector(127 downto 0);
48       r29: out std_logic_vector(127 downto 0);
49       r30: out std_logic_vector(127 downto 0);
50       r31: out std_logic_vector(127 downto 0)
51       );
52   end resgiter_file;
53
54   architecture comb of resgiter_file is
55   -- define the new type for the 32*128 register file
56   type registers_array is array (31 downto 0) of std_logic_vector (127
     downto 0);
```

```
57   -- initial values in the resgiter file
58   signal registers: registers_array :=(           -- initializes every bit
     of each register to be 0
59       others => (others => '0') );
60
61   begin
62       process(all)
63      --  draft for instruction names: instructionBufferOut,
     id_if_instructionOut, if_ex_Instruction, ex_wb_instruction
64       begin
65       if(WR_en='1') then -- when write enable = 1,
66           -- write back data into Rd in register file at the provided Rd
     address
67               registers(to_integer(unsigned(Rd_addr))) <= Rd_Data;
68               -- The index of the registers array needs to be integer so
69               -- converts addr from std_logic_vector -> Unsigned -> Interger
     using numeric_std library
70
71
72
73
74           end if;
75           --for i in 0 to 31 loop
76               --  ri <= registers(i);
77               -- end loop;
78           R1_out <= registers(to_integer(unsigned(R1_addr)));
79           R2_out <= registers(to_integer(unsigned(R2_addr)));
80           R3_out <= registers(to_integer(unsigned(R3_addr)));
81
82       end process;
83       -- Data to be read out
84
85
86       r0 <= registers(0);
87               r1 <= registers(1);
88               r2 <= registers(2);
89               r3 <= registers(3);
90               r4 <= registers(4);
91               r5 <= registers(5);
92               r6 <= registers(6);
93               r7 <= registers(7);
94               r8 <= registers(8);
95               r9 <= registers(9);
96               r10 <= registers(10);
97               r11 <= registers(11);
98               r12 <= registers(12);
99               r13 <= registers(13);
100              r14 <= registers(14);
101              r15 <= registers(15);
102              r16 <= registers(16);
103              r17 <= registers(17);
104              r18 <= registers(18);
105              r19 <= registers(19);
106              r20 <= registers(20);
107              r21 <= registers(21);
108              r22 <= registers(22);
109              r23 <= registers(23);
```

```
110             r24 <= registers(24);
111             r25 <= registers(25);
112             r26 <= registers(26);
113             r27 <= registers(27);
114             r28 <= registers(28);
115             r29 <= registers(29);
116             r30 <= registers(30);
117             r31 <= registers(31);
118
119  end comb;
```

RegisterFile_tb.vhd

```
1    LIBRARY ieee;
2    use ieee.std_logic_1164.ALL;
3    use ieee.std_logic_unsigned.ALL;
4    use work.all;
5    use std.env.finish;
6
7    -- VHDL testbench for register file
8    ENTITY tb_Reg_File IS
9    END tb_Reg_File;
10
11   ARCHITECTURE Behavioral OF tb_Reg_File IS
12
13       -- Component Declaration for register file in VHDL
14
15       COMPONENT resgiter_file
16       PORT(
17           WR_en: in std_logic;    -- Write enable
18           R1_addr: in std_logic_vector(4 downto 0);  --   R1 Address input
19           R2_addr: in std_logic_vector(4 downto 0);    -- R2 Address input
20           R3_addr: in std_logic_vector(4 downto 0);    -- R3 Address input
21           Rd_addr: in std_logic_vector(4 downto 0);  -- Rd Address input
22           Rd_data: in std_logic_vector(127 downto 0); -- Rd data in from
     Write Back
23       --  CLK: in std_logic; -- clock input for RAM
24           R1_out: out std_logic_vector(127 downto 0); -- R2 register output
25           R2_out: out std_logic_vector(127 downto 0); -- R2 register output
26           R3_out: out std_logic_vector(127 downto 0) -- Data output of RAM
27           );
28       END COMPONENT;
29
30
31       --Inputs
32       signal WR_en: std_logic := '0'; -- Write enable
33       signal R1_addr: std_logic_vector(4 downto 0) := (others => '0');  --
     R1 Address input
34       signal R2_addr: std_logic_vector(4 downto 0);    -- R2 Address input
35       signal R3_addr: std_logic_vector(4 downto 0);    -- R3 Address input
36       signal Rd_addr: std_logic_vector(4 downto 0);  -- Rd Address input
37       signal Rd_data: std_logic_vector(127 downto 0); -- Rd data in from
     Write Back
38
39       --Outputs
40       signal R1_out: std_logic_vector(127 downto 0);
41       signal R2_out: std_logic_vector(127 downto 0);
42       signal R3_out: std_logic_vector(127 downto 0);
43
44   begin
45       -- Instantiate the single-port RAM in VHDL
46       uut: entity resgiter_file
47           port map(
48             WR_en => WR_en,
49             R1_addr => R1_addr,
50             R2_addr => R2_addr,
51             R3_addr => R3_addr,
52             Rd_addr => Rd_addr,
53             Rd_data => Rd_data,
54             R1_out => R1_out,
```

```
55              R2_out => R2_out,
56              R3_out => R3_out
57          );
58
59          test: process
60          begin
61              WR_en <= '1';
62              R1_addr <= "00001";
63              R2_addr <= "00010";
64              R3_addr <= "00011";
65              Rd_addr <= "00001";  -- supposed to write back to $r1
66              Rd_data <= (others => '1');
67  --Rd_data <= (7 => '1', 5 downto 1 => '1', 6 => B_BIT, others => '0');
68              wait for 20 ns;
69
70              WR_en <= '1';
71              R1_addr <= "00001";
72              R2_addr <= "00010";
73              R3_addr <= "00011";
74              Rd_addr <= "00010";  -- supposed to write back to $r2
75              Rd_data <= (7 => '1', 4 downto 0 => '1', others => '0');
76              wait for 20 ns;
77
78              -- when WR_en is 0, verify that nothing gets written in
79              WR_en <= '0';
80              R1_addr <= "00001";
81              R2_addr <= "00000";
82              R3_addr <= "00011";
83              Rd_addr <= "00000";  -- supposed to write back to $r3
84              Rd_data <= (7 => '1', 4 downto 0 => '1', others => '0');
85              wait for 20 ns;
86
87              WR_en <= '1';
88              R1_addr <= "00001";
89              R2_addr <= "00010";
90              R3_addr <= "00011";
91              Rd_addr <= "00100";  -- supposed to write back to $r4
92              Rd_data <= (7 => '1', 4 downto 0 => '1', others => '0');
93              wait for 20 ns;
94
95              WR_en <= '1';
96              R1_addr <= "00100";  -- $r4
97              R2_addr <= "00010";  -- $r2
98              R3_addr <= "00011";  -- $r3
99              Rd_addr <= "00100";  -- supposed to write back to $r4
100             Rd_data <= (6 => '1', 3 downto 0 => '1', others => '0');
101             wait for 20 ns;
102
103             WR_en <= '1';
104             R1_addr <= "00001";
105             R2_addr <= "00010";
106             R3_addr <= "00011";
107             Rd_addr <= "00101";  -- supposed to write back to $r5
108             Rd_data <= (7 => '1', 4 downto 0 => '1', others => '0');
109             wait for 20 ns;
110
111             WR_en <= '1';
```

```
112             R1_addr <= "00001";
113             R2_addr <= "00010";
114             R3_addr <= "00011";
115             Rd_addr <= "00110";  -- supposed to write back to $r6
116             Rd_data <= (8 => '1', 4 downto 0 => '1', others => '0');
117             wait for 20 ns;
118 
119             WR_en <= '1';
120             R1_addr <= "00100";
121             R2_addr <= "00110";  -- supposed to spit out what's written in
122             R3_addr <= "00101";
123             Rd_addr <= "00110";  -- supposed to write back to $r6
124             Rd_data <= (7 => '1', 4 downto 0 => '1', others => '0');
125             wait for 20 ns;
126 
127             WR_en <= '1';
128             R1_addr <= "00100";
129             R2_addr <= "00110";  -- supposed to spit out what's written in
130             R3_addr <= "10101";
131             Rd_addr <= "10101";  -- supposed to write back to $r6
132             Rd_data <= (100 => '1', 4 downto 0 => '1', others => '0');
133             wait for 20 ns;
134 
135         finish;
136         end process;
137 end Behavioral;
```

IF_EXRegister.vhd

```
1   ----------------------------------------------------------------------------
    -------
2   -- Company: Stony Brook University - ESE 345 Computer Architecture
3   -- Engineers: Kyle Han and Summer Wang
4   --
5   -- Create Date: 11/30/2023 2:33:21 PM
6   -- Design Name:
7   -- Module Name: IF/EX Register - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------
    -------
20
21
22  library IEEE;
23  library std;
24  use IEEE.STD_LOGIC_1164.ALL;
25  use std.textio.all;
26  use IEEE.NUMERIC_STD.ALL;
27  use work.all;
28
29  entity if_exRegister is
30      Port(
31      --Inputs
32      clk: in std_logic;
33
34      instruction: in std_logic_vector(24 downto 0);
35
36      r3DataIn: in std_logic_vector(127 downto 0);
37      r2DataIn: in std_logic_vector(127 downto 0);
38      r1DataIn: in std_logic_vector(127 downto 0);
39
40      rs3AddressIn: in std_logic_vector(4 downto 0);
41      rs2AddressIn: in std_logic_vector(4 downto 0);
42      rs1AddressIn: in std_logic_vector(4 downto 0);
43      rdAddressIn: in std_logic_vector(4 downto 0);
44
45
46      --Outputs
47      instructionOut: out std_logic_vector(24 downto 0);
48
49      rs3AddressOut: out std_logic_vector(4 downto 0);
50      rs2AddressOut: out std_logic_vector(4 downto 0);
51      rs1AddressOut: out std_logic_vector(4 downto 0);
52      rdAddressOut: out std_logic_vector(4 downto 0);
53
54      r3DataOut: out std_logic_vector(127 downto 0);
55      r2DataOut: out std_logic_vector(127 downto 0);
```

```
56      r1DataOut: out std_logic_vector(127 downto 0)
57
58      );
59  end if_exRegister;
60
61
62  architecture behavioral of if_exRegister is
63
64  begin
65      process(clk)
66      begin
67          if(rising_edge(clk)) then
68              --On a rising clock edge, we want to see the asynchronous data
    sent out. So whatever is in instruction is split
69              instructionOut <= instruction;
70
71              rs3AddressOut <= rs3AddressIn;
72              rs2AddressOut <= rs2AddressIn;
73              rs1AddressOut <= rs1AddressIn;
74              rdAddressOut <= rdAddressIn;
75
76              r3DataOut <= r3DataIn;
77              r2DataOut <= r2DataIn;
78              r1DataOut <= r1DataIn;
79          end if;
80      end process;
81
82  end behavioral;
83
```

forwardingUnit.vhd

```
1    ----------------------------------------------------------------------------
     --------
2    -- Company: Stony Brook University - ESE 345 Computer Architecture
3    -- Engineers: Kyle Han and Summer Wang
4    --
5    -- Create Date: 11/30/2023 11:24:21 PM
6    -- Design Name:
7    -- Module Name: Forwarding Unit - Behavioral
8    -- Project Name:
9    -- Target Devices:
10   -- Tool Versions:
11   -- Description:
12   --
13   -- Dependencies:
14   --
15   -- Revision:
16   -- Revision 0.01 - File Created
17   -- Additional Comments:
18   --
19   ----------------------------------------------------------------------------
     --------
20
21   library IEEE;
22   library std;
23   use IEEE.STD_LOGIC_1164.ALL;
24   use std.textio.all;
25   use IEEE.NUMERIC_STD.ALL;
26
27   entity fwdUnit is
28       Port(
29       --Inputs
30       --The instruction currently being written back
31       wbInstruction: in std_logic_vector(24 downto 0);
32       --The instruction in the execution stage
33       exInstruction: in std_logic_vector(24 downto 0);
34
35       r3DataIn: in std_logic_vector(127 downto 0);
36       r2DataIn: in std_logic_vector(127 downto 0);
37       r1DataIn: in std_logic_vector(127 downto 0);
38
39       --Data from the output of the wb register file
40       wbData: in std_logic_vector(127 downto 0);
41
42       --Addresses of R3-R1 of the execute stage
43       exR3Address: in std_logic_vector(4 downto 0);
44       exR2Address: in std_logic_vector(4 downto 0);
45       exR1Address: in std_logic_vector(4 downto 0);
46       exRdAddress: in std_logic_vector(4 downto 0);
47
48       --Address of Rd of the execute stage
49       wbRdAddress: in std_logic_vector(4 downto 0);
50
51
52       --Outputs
53       r3DataOut: out std_logic_vector(127 downto 0);
54       r2DataOut: out std_logic_vector(127 downto 0);
55       r1DataOut: out std_logic_vector(127 downto 0)
```

```
56      );
57  end fwdUnit;
58
59  architecture behavioral of fwdUnit is
60  begin
61      --For some reason, sensitivity list doesn't work with all
62      --Originally, I used wbInstruction and exInstruction as a part of the
    sensitivity list.
63      --Howev
64      forward: process(r3DataIn, r2DataIn, r1DataIn, wbData, exR3Address,
    exR2Address, exR1Address, exRdAddress, wbRdAddress)
65      begin
66          --By default, route the data right through.
67          r3DataOut <= r3DataIn;
68          r2DataOut <= r2DataIn;
69          r1DataOut <= r1DataIn;
70
71          --If the execute instruction is a nop, just forward the data
    through
72          --There is no wbRd, so there's nothing to check.
73          if (wbInstruction(24 downto 23) = "11" and wbInstruction(18 downto
    15) = "0000") then
74              --Do nothing, just let the data forward through
75
76          --If the ex opcode is a load imm, we'll compare wbRd to garbage
    addresses.
77          --No data gets forwarded
78          elsif (exInstruction(24) = '0') then
79          --However, if wb instruction is load imm, and ex is also li with
    same rd, we have a hazard.
80              if (exRdAddress = wbRdAddress) then
81                  r1DataOut <= wbData;
82              end if;
83          else
84              --Handling any instruction into r3/r4
85              --If wb rd = ex rs1, data hazard
86              if (wbRdAddress = exR1Address) then
87                  r1DataOut <= wbData;
88              end if;
89
90              --If wb rd = ex rs2, data hazard
91              if (wbRdAddress = exR2Address) then
92                  r2DataOut <= wbData;
93              end if;
94
95              --If wb rd = ex rs3, data hazard
96              if (wbRdAddress = exR3Address) then
97                  r3DataOut <= wbData;
98              end if;
99
100         end if;
101     end process;
102
103 end behavioral;
```

forwardingUnit_tb.vhd

```
1   ----------------------------------------------------------------------------
    -------
2   -- Company: Stony Brook University - ESE 345 Computer Architecture
3   -- Engineers: Kyle Han and Summer Wang
4   --
5   -- Create Date: 12/2/2023 2:47:21 AM
6   -- Design Name:
7   -- Module Name: forwardingUnit_TB - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------
    -------
20
21  library IEEE;
22  library std;
23  use IEEE.STD_LOGIC_1164.ALL;
24  use std.textio.all;
25  use IEEE.NUMERIC_STD.ALL;
26  use work.all;
27  use std.env.finish;
28
29  entity forwardingUnit_tb is
30
31  end forwardingUnit_tb;
32
33  architecture behavioral of forwardingUnit_tb is
34      signal ex_wb_instruction: std_logic_vector(24 downto 0);
35      signal ex_wb_rdData: std_logic_vector(127 downto 0);
36      signal ex_wb_rdAddress: std_logic_vector(4 downto 0);
37
38      signal if_ex_Instruction: std_logic_vector(24 downto 0);
39
40      signal if_ex_rs3Address, if_ex_rs2Address, if_ex_rs1Address,
    if_ex_rdAddress: std_logic_vector(4 downto 0);
41      signal if_ex_rs3Data, if_ex_rs2Data, if_ex_rs1Data: std_logic_vector(
    127 downto 0);
42
43      signal fwdR3Data: std_logic_vector(127 downto 0);
44      signal fwdR2Data: std_logic_vector(127 downto 0);
45      signal fwdR1Data: std_logic_vector(127 downto 0);
46  begin
47      UUT: entity fwdUnit
48          port map(wbInstruction => ex_wb_instruction,
49          exInstruction => if_ex_Instruction,
50
51          r3DataIn => if_ex_rs3Data,
52          r2DataIn => if_ex_rs2Data,
53          r1DataIn => if_ex_rs1Data,
```

```
54
55          wbData => ex_wb_rdData,
56
57          exR3Address => if_ex_rs3Address,
58          exR2Address => if_ex_rs2Address,
59          exR1Address => if_ex_rs1Address,
60          exRdAddress => if_ex_rdAddress,
61
62          wbRdAddress => ex_wb_rdAddress,
63
64          --Outputs from the mux
65          r3DataOut => fwdR3Data,
66          r2DataOut => fwdR2Data,
67          r1DataOut => fwdR1Data
68          );
69          testing: process
70          begin
71
72              --1st instruction (in Wb unit) is nop
73              --2nd instruction (in ex unit) is au
74              ex_wb_instruction <= "11000000000000000000000000";
75              if_ex_Instruction <= "11000000100000100000000010";
76
77              if_ex_rs3Data <= X"10000000000000000000000000000000";
78              if_ex_rs2Data <= X"F0000000000F404000000000123F4240";
79              if_ex_rs1Data <= X"F0000000000F404000000000123F4240";
80              wait for 10ns;
81
82
83              std.env.finish;
84          end process;
85  end behavioral;
```

alu.vhd

This code can be found in the previous report.

alu_tb.vhd

This code can be found in the previous report.

EX_WBRegister.vhd

```
1   ----------------------------------------------------------------------------
    -------
2   -- Company: Stony Brook University - ESE 345 Computer Architecture
3   -- Engineers: Kyle Han and Summer Wang
4   --
5   -- Create Date: 11/30/2023 2:33:21 PM
6   -- Design Name:
7   -- Module Name: EX/WB Register - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------
    -------
20
21
22  library IEEE;
23  library std;
24  use IEEE.STD_LOGIC_1164.ALL;
25  use std.textio.all;
26  use IEEE.NUMERIC_STD.ALL;
27  use work.all;
28
29  entity ex_wbRegister is
30      Port(
31      --Inputs
32      clk: in std_logic;
33
34      instruction: in std_logic_vector(24 downto 0);
35
36      rdDataIn: in std_logic_vector(127 downto 0);
37
38      rdAddressIn: in std_logic_vector(4 downto 0);
39
40
41      --Outputs
42      instructionOut: out std_logic_vector(24 downto 0);
43
44      rdDataOut: out std_logic_vector(127 downto 0);
45
46      rdAddressOut: out std_logic_vector(4 downto 0)
47      );
48  end ex_wbRegister;
49
50
51  architecture behavioral of ex_wbRegister is
52
53  begin
54      process(clk)
55      begin
```

```
56          if(rising_edge(clk)) then
57              --On a rising clock edge, we want to see the asynchronous data
    sent out. So whatever is in instruction is split
58              instructionOut <= instruction;
59
60              rdAddressOut <= rdAddressIn;
61
62              rdDataOut <= rdDataIn;
63          end if;
64      end process;
65
66  end behavioral;
67
```

writeBackUnit.vhd

```
1   ----------------------------------------------------------------------------
    -------
2   -- Company: Stony Brook University - ESE 345 Computer Architecture
3   -- Engineers: Kyle Han and Summer Wang
4   --
5   -- Create Date: 11/30/2023 2:33:21 PM
6   -- Design Name:
7   -- Module Name: Write Back Unit - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------
    -------
20
21  library IEEE;
22  library std;
23  use IEEE.STD_LOGIC_1164.ALL;
24  use std.textio.all;
25  use IEEE.NUMERIC_STD.ALL;
26  use work.all;
27
28  entity writeBackUnit is
29      Port(
30      --Inputs
31      instruction: in std_logic_vector(24 downto 0);
32      rdDataIn: in std_logic_vector(127 downto 0);
33      rdAddressIn: in std_logic_vector(4 downto 0);
34
35      --Outputs
36      writeEn: out std_logic;
37      rdDataOut: out std_logic_vector(127 downto 0);
38      rdAddressOut: out std_logic_vector(4 downto 0)
39      );
40  end writeBackUnit;
41
42
43  architecture behavioral of writeBackUnit is
44  begin
45      rdDataOut <= rdDataIn;
46      rdAddressOut <= rdAddressIn;
47
48      --Write enable is 0 when the instruction is NOP
49      writeEn <= '0' when (instruction(24 downto 23) = "11" and instruction(
    18 downto 15) = "0000") else '1';
50  end behavioral;
51
```

writeBackUnit_tb.vhd

```
1   ----------------------------------------------------------------------------
    -------
2   -- Company: Stony Brook University - ESE 345 Computer Architecture
3   -- Engineers: Kyle Han and Summer Wang
4   --
5   -- Create Date: 11/30/2023 9:57:21 PM
6   -- Design Name:
7   -- Module Name: insructionBuffer_TB - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------
    -------
20
21  library IEEE;
22  library std;
23  use IEEE.STD_LOGIC_1164.ALL;
24  use std.textio.all;
25  use IEEE.NUMERIC_STD.ALL;
26  use work.all;
27  use std.env.finish;
28
29  entity writeBackUnit_tb is
30
31  end writeBackUnit_tb;
32
33  architecture behavioral of writeBackUnit_tb is
34      signal instruction: std_logic_vector(24 downto 0);
35
36      signal rdData: std_logic_vector(127 downto 0);
37      signal rdAddress: std_logic_vector(4 downto 0);
38  begin
39      UUT: entity writeBackUnit
40          port map(instruction => instruction,
41          rdDataIn => rdData,
42          rdAddressIn => rdAddress);
43          testing: process
44          begin
45
46              --WriteEn should be 0
47              instruction <= "1100000000000000000000000";
48              rdAddress <= std_logic_vector(to_unsigned(31, 5));
49              rdData <= (100 downto 80 => '1', others => '0');
50              wait for 10ns;
51
52              --WriteEn should be 1 for the rest of these
53              instruction <= "1000000000000000000000000";
54              rdAddress <= std_logic_vector(to_unsigned(5, 5));
55              rdData <= (127 downto 80 => '1', others => '0');
```

```
56              wait for 10ns;
57
58              instruction <= "11000011000000000000000000";
59              rdAddress <= std_logic_vector(to_unsigned(12, 5));
60              rdData <= (60 downto 20 => '1', others => '0');
61              wait for 10ns;
62
63              --WriteEn should be 0
64              instruction <= "11111100000000000000000000";
65              rdAddress <= std_logic_vector(to_unsigned(31, 5));
66              rdData <= (100 downto 80 => '1', others => '0');
67              wait for 10ns;
68
69              --WriteEn should be 1
70              instruction <= "11111100010000000000000000";
71              rdAddress <= std_logic_vector(to_unsigned(31, 5));
72              rdData <= (100 downto 80 => '1', others => '0');
73              wait for 10ns;
74              std.env.finish;
75          end process;
76  end behavioral;
```

resultFile.txt

```
                                                    **********************************
                                                    *                                *
                                                    *          RESULT FILE           *
                                                    *                                *
                                                    **********************************


====================
      Cycle 0
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: XXXXXXX                   || opcode: XXXXXXX                         || function: XXXXXXX                                                              || field: XXXXXXX                        ||
||                                        || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || M1: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || A1: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || WE: U                           ||
||                                        || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || M2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || A2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || RD: 00                            ||
||                                        || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || M3: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || A3: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ||
||                                        ||                                         || FD: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || AO: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 1
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 02ACF03                   || opcode: XXXXXXX                         || function: XXXXXXX                                                              || field: XXXXXXX                        ||
||                                        || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || M1: 00000000000000000000000000000000 || A1: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || WE: 1                           ||
||                                        || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || M2: 00000000000000000000000000000000 || A2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || RD: 00                            ||
||                                        || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || M3: 00000000000000000000000000000000 || A3: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ||
||                                        ||                                         || FD: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || AO: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 2
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 1830063                   || opcode: 02ACF03                         || function: XXXXXXX                                                              || field: XXXXXXX                        ||
||                                        || R[24]: 00000000000000000000000000000000 || M1: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || A1: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || WE: 1                           ||
||                                        || R[19]: 00000000000000000000000000000000 || M2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || A2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || RD: 00                            ||
||                                        || R[21]: 00000000000000000000000000000000 || M3: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || A3: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ||
||                                        ||                                         || FD: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || AO: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 3
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 0CF0242                   || opcode: 1830063                         || function: 02ACF03                                                              || field: XXXXXXX                        ||
||                                        || R[03]: 00000000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000000000000 || WE: 1                           ||
||                                        || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || M2: 00000000000000000000000000000000 || A2: 00000000000000000000000000000000 || RD: 00                            ||
||                                        || R[06]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX ||
||                                        ||                                         || FD: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || AO: 00000000000000000000000056780000 ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 4
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 0E24681                   || opcode: 0CF0242                         || function: 1830063                                                              || field: 02ACF03                        ||
||                                        || R[18]: 00000000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000056780000 || WE: 1                           ||
||                                        || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || M2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || A2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || RD: 03                            ||
||                                        || R[30]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[03]: 00000000000000000000000056780000 ||
||                                        ||                                         || FD: 00000000000000000000000056780000 || AO: 56780000567800005678000056780000 ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 5
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 0CACF01                   || opcode: 0E24681                         || function: 0CF0242                                                              || field: 1830063                        ||
||                                        || R[20]: 00000000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000000000000 || WE: 1                           ||
||                                        || R[17]: 00000000000000000000000000000000 || M2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || A2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || RD: 03                            ||
||                                        || R[04]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[03]: 56780000567800005678000056780000 ||
||                                        ||                                         || FD: 56780000567800005678000056780000 || AO: 00007812000000000000000000000000 ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA
```

```
====================
      Cycle 6
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 1018820                   || opcode: 0CACF01                         || function: 0E24681                                                              || field: 0CF0242                        ||
||                                        || R[24]: 00000000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000000000000 || WE: 1                               ||
||                                        || R[19]: 00000000000000000000000000000000 || M2: 00000000000000000000000000000000 || A2: 00000000000000000000000000000000 || RD: 02                               ||
||                                        || R[21]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[02]: 00007812000000000000000000000000 ||
||                                        ||                                         || FD: 00007812000000000000000000000000 || AO: 12340000000000000000000000000000 ||                                       ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 7
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 0FE0001                   || opcode: 1018820                         || function: 0CACF01                                                              || field: 0E24681                        ||
||                                        || R[01]: 12340000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: 12340000000000000000000000000000 || WE: 1                               ||
||                                        || R[02]: 00007812000000000000000000000000 || M2: 00000000000000000000000000000000 || A2: 00000000000000000000000000000000 || RD: 01                               ||
||                                        || R[03]: 56780000567800005678000056780000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[01]: 12340000000000000000000000000000 ||
||                                        ||                                         || FD: 12340000000000000000000000000000 || AO: 12345678000000000000000000000000 ||                                       ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 8
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 0CACF01                   || opcode: 0FE0001                         || function: 1018820                                                              || field: 0CACF01                        ||
||                                        || R[00]: 12345678000000000000000000000000 || M1: 12340000000000000000000000000000 || A1: 12345678000000000000000000000000 || WE: 1                               ||
||                                        || R[00]: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || M2: 00007812000000000000000000000000 || A2: 00007812000000000000000000000000 || RD: 01                               ||
||                                        || R[28]: 00000000000000000000000000000000 || M3: 56780000567800005678000056780000 || A3: 56780000567800005678000056780000 || R[01]: 12345678000000000000000000000000 ||
||                                        ||                                         || FD: 12345678000000000000000000000000 || AO: 12345678000000000000000000000000 ||                                       ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 9
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 0A001E1                   || opcode: 0CACF01                         || function: 0FE0001                                                              || field: 1018820                        ||
||                                        || R[24]: 12345678000000000000000000000000 || M1: 12345678000000000000000000000000 || A1: 12345678000000000000000000000000 || WE: 1                               ||
||                                        || R[19]: 00000000000000000000000000000000 || M2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || A2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX || RD: 00                               ||
||                                        || R[21]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[00]: 12345678000000000000000000000000 ||
||                                        ||                                         || FD: 12345678000000000000000000000000 || AO: F0005678000000000000000000000000 ||                                       ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 10
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 0880801                   || opcode: 0A001E1                         || function: 0CACF01                                                              || field: 0FE0001                        ||
||                                        || R[15]: F0005678000000000000000000000000 || M1: 12345678000000000000000000000000 || A1: F0005678000000000000000000000000 || WE: 1                               ||
||                                        || R[00]: 12345678000000000000000000000000 || M2: 00000000000000000000000000000000 || A2: 00000000000000000000000000000000 || RD: 01                               ||
||                                        || R[00]: 12345678000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[01]: F0005678000000000000000000000000 ||
||                                        ||                                         || FD: F0005678000000000000000000000000 || AO: F0005678000000000000000000000000 ||                                       ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 11
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 07FD681                   || opcode: 0880801                         || function: 0A001E1                                                              || field: 0CACF01                        ||
||                                        || R[00]: F0005678000000000000000000000000 || M1: F0005678000000000000000000000000 || A1: F0005678000000000000000000000000 || WE: 1                               ||
||                                        || R[02]: 00007812000000000000000000000000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 01                               ||
||                                        || R[16]: 00000000000000000000000000000000 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[01]: F0005678000000000000000000000000 ||
||                                        ||                                         || FD: F0005678000000000000000000000000 || AO: F0005678000F00000000000000000000 ||                                       ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 12
====================
```

```
||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||===================================================================================||=======================================||
|| instruction: 0400001                  || opcode: 07FD681                        || function: 0880801                                                                 || field: 0A001E1                       ||
||                                       || R[20]: F0005678000F0000000000000000000 || M1: F0005678000000000000000000000000 || A1: F0005678000F0000000000000000000 || WE: 1                            ||
||                                       || R[21]: 0000000000000000000000000000000 || M2: 0000781200000000000000000000000 || A2: 0000781200000000000000000000000 || RD: 01                            ||
||                                       || R[31]: 0000000000000000000000000000000 || M3: 0000000000000000000000000000000 || A3: 0000000000000000000000000000000 || R[01]: F0005678000F0000000000000000000 ||
||                                       ||                                        || FD: F0005678000F0000000000000000000 || AO: F0005678000F4040000000000000000 ||                                       ||
||=======================================||========================================||===================================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 13
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||===================================================================================||=======================================||
|| instruction: 02001E1                  || opcode: 0400001                        || function: 07FD681                                                                 || field: 0880801                       ||
||                                       || R[00]: F0005678000F4040000000000000000 || M1: F0005678000F0000000000000000000 || A1: F0005678000F4040000000000000000 || WE: 1                            ||
||                                       || R[00]: 1234567800000000000000000000000 || M2: 0000000000000000000000000000000 || A2: 0000000000000000000000000000000 || RD: 01                            ||
||                                       || R[00]: 1234567800000000000000000000000 || M3: 0000000000000000000000000000000 || A3: 0000000000000000000000000000000 || R[01]: F0005678000F4040000000000000000 ||
||                                       ||                                        || FD: F0005678000F4040000000000000000 || AO: F0005678000F4040FEB4000000000000 ||                                       ||
||=======================================||========================================||===================================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 14
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||===================================================================================||=======================================||
|| instruction: 0084801                  || opcode: 02001E1                        || function: 0400001                                                                 || field: 07FD681                       ||
||                                       || R[15]: F0005678000F4040FEB4000000000000 || M1: F0005678000F4040000000000000000 || A1: F0005678000F4040FEB4000000000000 || WE: 1                            ||
||                                       || R[00]: 1234567800000000000000000000000 || M2: 1234567800000000000000000000000 || A2: 1234567800000000000000000000000 || RD: 01                            ||
||                                       || R[00]: 1234567800000000000000000000000 || M3: 1234567800000000000000000000000 || A3: 1234567800000000000000000000000 || R[01]: F0005678000F4040FEB4000000000000 ||
||                                       ||                                        || FD: F0005678000F4040FEB4000000000000 || AO: F0005678000F4040FEB4000000000000 ||                                       ||
||=======================================||========================================||===================================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 15
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||===================================================================================||=======================================||
|| instruction: 1818022                  || opcode: 0084801                        || function: 02001E1                                                                 || field: 0400001                       ||
||                                       || R[00]: F0005678000F4040FEB4000000000000 || M1: F0005678000F4040FEB4000000000000 || A1: F0005678000F4040FEB4000000000000 || WE: 1                            ||
||                                       || R[18]: 0000000000000000000000000000000 || M2: 1234567800000000000000000000000 || A2: 1234567800000000000000000000000 || RD: 01                            ||
||                                       || R[16]: 0000000000000000000000000000000 || M3: 1234567800000000000000000000000 || A3: 1234567800000000000000000000000 || R[01]: F0005678000F4040FEB4000000000000 ||
||                                       ||                                        || FD: F0005678000F4040FEB4000000000000 || AO: F0005678000F4040FEB40000000F0000 ||                                       ||
||=======================================||========================================||===================================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 16
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||===================================================================================||=======================================||
|| instruction: 1828C24                  || opcode: 1818022                        || function: 0084801                                                                 || field: 02001E1                       ||
||                                       || R[01]: F0005678000F4040FEB40000000F0000 || M1: F0005678000F4040FEB4000000000000 || A1: F0005678000F4040FEB40000000F0000 || WE: 1                            ||
||                                       || R[00]: 1234567800000000000000000000000 || M2: 0000000000000000000000000000000 || A2: 0000000000000000000000000000000 || RD: 01                            ||
||                                       || R[03]: 5678000056780000567800005678000 || M3: 0000000000000000000000000000000 || A3: 0000000000000000000000000000000 || R[01]: F0005678000F4040FEB40000000F0000 ||
||                                       ||                                        || FD: F0005678000F4040FEB40000000F0000 || AO: F0005678000F4040FEB40000000F4240 ||                                       ||
||=======================================||========================================||===================================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 17
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||===================================================================================||=======================================||
|| instruction: 1858485                  || opcode: 1828C24                        || function: 1818022                                                                 || field: 0084801                       ||
||                                       || R[01]: F0005678000F4040FEB40000000F4240 || M1: F0005678000F4040FEB40000000F0000 || A1: F0005678000F4040FEB40000000F4240 || WE: 1                            ||
||                                       || R[03]: 5678000056780000567800005678000 || M2: 1234567800000000000000000000000 || A2: 1234567800000000000000000000000 || RD: 01                            ||
||                                       || R[05]: 0000000000000000000000000000000 || M3: 5678000056780000567800005678000 || A3: 5678000056780000567800005678000 || R[01]: F0005678000F4040FEB40000000F4240 ||
||                                       ||                                        || FD: F0005678000F4040FEB40000000F4240 || AO: 0004000800040002000B000000040003 ||                                       ||
||=======================================||========================================||===================================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 18
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||===================================================================================||=======================================||
|| instruction: 18600A6                  || opcode: 1858485                        || function: 1828C24                                                                 || field: 1818022                       ||
||                                       || R[04]: 0000000000000000000000000000000 || M1: F0005678000F4040FEB40000000F4240 || A1: F0005678000F4040FEB40000000F4240 || WE: 1                            ||
```

```
||                                    || R[01]: F0005678000F4040FEB40000000F4240 || M2: 5678000056780000567800056780000 || A2: 5678000056780000567800056780000 || RD: 02                                    ||
||                                    || R[11]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[02]: 0004000800040002000B000000040003 ||
||                                    ||                                         || FD: 0004000800040002000B000000040003 || AO: F6785678567F4040FEFC0000567F4240 ||                                          ||
||====================================||=========================================||=============================================================================||==========================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 19
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK           ||
||====================================||=========================================||=============================================================================||==========================================||
|| instruction: 18594C7                  || opcode: 18600A6                        || function: 1858485                                                        || field: 1828C24                         ||
||                                    || R[05]: 00000000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: F6785678567F4040FEFC0000567F4240 || WE: 1                               ||
||                                    || R[00]: 12345678000000000000000000000000 || M2: F0005678000F4040FEB40000000F4240 || A2: F0005678000F4040FEB40000000F4240 || RD: 04                               ||
||                                    || R[12]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[04]: F6785678567F4040FEFC0000567F4240 ||
||                                    ||                                         || FD: F6785678567F4040FEFC0000567F4240 || AO: F0005678000F4040FEB40000000F4240 ||                                          ||
||====================================||=========================================||=============================================================================||==========================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 20
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK           ||
||====================================||=========================================||=============================================================================||==========================================||
|| instruction: 18394C7                  || opcode: 18594C7                        || function: 18600A6                                                        || field: 1858485                         ||
||                                    || R[06]: 00000000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: F0005678000F4040FEB40000000F4240 || WE: 1                               ||
||                                    || R[05]: F0005678000F4040FEB40000000F4240 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 05                               ||
||                                    || R[11]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[05]: F0005678000F4040FEB40000000F4240 ||
||                                    ||                                         || FD: F0005678000F4040FEB40000000F4240 || AO: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF ||                                          ||
||====================================||=========================================||=============================================================================||==========================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 21
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK           ||
||====================================||=========================================||=============================================================================||==========================================||
|| instruction: 1800000                  || opcode: 18394C7                        || function: 18594C7                                                        || field: 18600A6                         ||
||                                    || R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || M1: 00000000000000000000040000000000 || A1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || WE: 1                               ||
||                                    || R[05]: F0005678000F4040FEB40000000F4240 || M2: F0005678000F4040FEB40000000F4240 || A2: F0005678000F4040FEB40000000F4240 || RD: 06                               ||
||                                    || R[07]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF ||
||                                    ||                                         || FD: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || AO: 00000000000000000000000000000000 ||                                          ||
||====================================||=========================================||=============================================================================||==========================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 22
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK           ||
||====================================||=========================================||=============================================================================||==========================================||
|| instruction: 18414C8                  || opcode: 1800000                        || function: 18394C7                                                        || field: 18594C7                         ||
||                                    || R[00]: 12345678000000000000000000000000 || M1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || A1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || WE: 1                               ||
||                                    || R[00]: 12345678000000000000000000000000 || M2: F0005678000F4040FEB40000000F4240 || A2: F0005678000F4040FEB40000000F4240 || RD: 07                               ||
||                                    || R[00]: 12345678000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[07]: 00000000000000000000000000000000 ||
||                                    ||                                         || FD: 00000000000000000000000000000000 || AO: 0FFFA987000F4040014BFFFF000F4240 ||                                          ||
||====================================||=========================================||=============================================================================||==========================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 23
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK           ||
||====================================||=========================================||=============================================================================||==========================================||
|| instruction: 18180C9                  || opcode: 18414C8                        || function: 1800000                                                        || field: 18394C7                         ||
||                                    || R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || M1: 12345678000000000000000000000000 || A1: 12345678000000000000000000000000 || WE: 1                               ||
||                                    || R[05]: F0005678000F4040FEB40000000F4240 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 07                               ||
||                                    || R[08]: 00000000000000000000000000000000 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[07]: 0FFFA987000F4040014BFFFF000F4240 ||
||                                    ||                                         || FD: 0FFFA987000F4040014BFFFF000F4240 || AO: 00000000000000000000000000000000 ||                                          ||
||====================================||=========================================||=============================================================================||==========================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 24
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK           ||
||====================================||=========================================||=============================================================================||==========================================||
|| instruction: 186A4CA                  || opcode: 18180C9                        || function: 18414C8                                                        || field: 1800000                         ||
||                                    || R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || M1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || A1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || WE: 0                               ||
||                                    || R[00]: 12345678000000000000000000000000 || M2: F0005678000F4040FEB40000000F4240 || A2: F0005678000F4040FEB40000000F4240 || RD: 00                               ||
||                                    || R[03]: 5678000056780000567800056780000  || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[00]: 00000000000000000000000000000000 ||
||                                    ||                                         || FD: 00000000000000000000000000000000 || AO: F0005678FFF0BFBFFEB40000FFF0BDBF ||                                          ||
||====================================||=========================================||=============================================================================||==========================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA
```

```
====================
      Cycle 25
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 180A4CB                  || opcode: 186A4CA                        || function: 18180C9                                                           || field: 18414C8                       ||
||                                       || R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || M1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || A1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || WE: 1                            ||
||                                       || R[09]: 00000000000000000000000000000000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 08                            ||
||                                       || R[13]: 00000000000000000000000000000000 || M3: 56780000567800005678000056780000 || A3: 56780000567800005678000056780000 || R[08]: F0005678FFF0BFBFFEB40000FFF0BDBF ||
||                                       ||                                        || FD: F0005678FFF0BFBFFEB40000FFF0BDBF || AO: 000C0008000C000E00050010000C000D ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 26
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 00356BF                  || opcode: 180A4CB                        || function: 186A4CA                                                           || field: 18180C9                       ||
||                                       || R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || M1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || A1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || WE: 1                            ||
||                                       || R[09]: 000C0008000C000E00050010000C000D || M2: 00000000000000000000000000000000 || A2: 000C0008000C000E00050010000C000D || RD: 09                            ||
||                                       || R[01]: F0005678000F4040FEB40000000F4240 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[09]: 000C0008000C000E00050010000C000D ||
||                                       ||                                        || FD: 000C0008000C000E00050010000C000D || AO: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 27
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 050001F                  || opcode: 00356BF                        || function: 180A4CB                                                           || field: 186A4CA                       ||
||                                       || R[21]: 00000000000000000000000000000000 || M1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || A1: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || WE: 1                            ||
||                                       || R[21]: 00000000000000000000000000000000 || M2: 000C0008000C000E00050010000C000D || A2: 000C0008000C000E00050010000C000D || RD: 10                            ||
||                                       || R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || M3: F0005678000F4040FEB40000000F4240 || A3: F0005678000F4040FEB40000000F4240 || R[10]: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 ||
||                                       ||                                        || FD: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 || AO: 00070054007F005F0000007F007F005E ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 28
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 08FFFFF                  || opcode: 050001F                        || function: 00356BF                                                           || field: 180A4CB                       ||
||                                       || R[00]: 00000000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000000000000 || WE: 1                            ||
||                                       || R[00]: 12345678000000000000000000000000 || M2: 00000000000000000000000000000000 || A2: 00000000000000000000000000000000 || RD: 11                            ||
||                                       || R[00]: 12345678000000000000000000000000 || M3: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || A3: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || R[11]: 00070054007F005F0000007F007F005E ||
||                                       ||                                        || FD: 00070054007F005F0000007F007F005E || AO: 00000000000000000000000001AB5 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 29
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 0C000BF                  || opcode: 08FFFFF                        || function: 050001F                                                           || field: 00356BF                       ||
||                                       || R[31]: 00000000000000000000000001AB5 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000001AB5 || WE: 1                            ||
||                                       || R[31]: 00000000000000000000000001AB5 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 31                            ||
||                                       || R[31]: 00000000000000000000000001AB5 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[31]: 00000000000000000000000001AB5 ||
||                                       ||                                        || FD: 00000000000000000000000001AB5 || AO: 00000000000000000000800000001AB5 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 30
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 0124C7E                  || opcode: 0C000BF                        || function: 08FFFFF                                                           || field: 050001F                       ||
||                                       || R[05]: 00000000000000000000800000001AB5 || M1: 00000000000000000000000001AB5 || A1: 00000000000000000000800000001AB5 || WE: 1                            ||
||                                       || R[00]: 12345678000000000000000000000000 || M2: 00000000000000000000000001AB5 || A2: 00000000000000000000000001AB5 || RD: 31                            ||
||                                       || R[00]: 12345678000000000000000000000000 || M3: 00000000000000000000000001AB5 || A3: 00000000000000000000000001AB5 || R[31]: 00000000000000000000800000001AB5 ||
||                                       ||                                        || FD: 00000000000000000000800000001AB5 || AO: 00000000000007FFF0000800000001AB5 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA


====================
      Cycle 31
```

```
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 050001E                  || opcode: 0124C7E                        || function: 0C000BF                                                           || field: 08FFFFF                       ||
||                                       || R[03]: 00000000000000000000000000000000 || M1: 00000000000000000000800000001AB5 || A1: 00000000000007FFF0000800000001AB5 || WE: 1                             ||
||                                       || R[19]: 00000000000000000000000000000000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 31                            ||
||                                       || R[04]: F6785678567F4040FEFC0000567F4240 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[31]: 00000000000007FFF0000800000001AB5 ||
||                                       ||                                        || FD: 00000000000007FFF0000800000001AB5 || AO: 00000000500007FFF0000800000001AB5 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 32
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 08000BE                  || opcode: 050001E                        || function: 0124C7E                                                           || field: 0C000BF                       ||
||                                       || R[00]: 00000000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000000000000 || WE: 1                             ||
||                                       || R[00]: 12345678000000000000000000000000 || M2: 00000000000000000000000000000000 || A2: 00000000000000000000000000000000 || RD: 31                            ||
||                                       || R[00]: 12345678000000000000000000000000 || M3: F6785678567F4040FEFC0000567F4240 || A3: F6785678567F4040FEFC0000567F4240 || R[31]: 00000000500007FFF0000800000001AB5 ||
||                                       ||                                        || FD: 00000000500007FFF0000800000001AB5 || AO: 00000000000000000000000000009263 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 33
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 0C000FE                  || opcode: 08000BE                        || function: 050001E                                                           || field: 0124C7E                       ||
||                                       || R[05]: 00000000000000000000000000009263 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000000009263 || WE: 1                             ||
||                                       || R[00]: 12345678000000000000000000000000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 30                            ||
||                                       || R[00]: 12345678000000000000000000000000 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[30]: 00000000000000000000000000009263 ||
||                                       ||                                        || FD: 00000000000000000000000000009263 || AO: 00000000000000000000800000009263 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 34
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 1827BFD                  || opcode: 0C000FE                        || function: 08000BE                                                           || field: 050001E                       ||
||                                       || R[07]: 00000000000000000000800000009263 || M1: 00000000000000000000000000009263 || A1: 00000000000000000000800000009263 || WE: 1                             ||
||                                       || R[00]: 12345678000000000000000000000000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 30                            ||
||                                       || R[00]: 12345678000000000000000000000000 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[30]: 00000000000000000000800000009263 ||
||                                       ||                                        || FD: 00000000000000000000800000009263 || AO: 00000000000000005000800000009263 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 35
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 0ECA87C                  || opcode: 1827BFD                        || function: 0C000FE                                                           || field: 08000BE                       ||
||                                       || R[31]: 00000000500007FFF0000800000001AB5 || M1: 00000000000000000000800000009263 || A1: 00000000000000005000800000009263 || WE: 1                             ||
||                                       || R[30]: 00000000000000005000800000009263 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 30                            ||
||                                       || R[04]: F6785678567F4040FEFC0000567F4240 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[30]: 00000000000000005000800000009263 ||
||                                       ||                                        || FD: 00000000000000005000800000009263 || AO: 00000007000000005000800000009263 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 36
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 0C4203C                  || opcode: 0ECA87C                        || function: 1827BFD                                                           || field: 0C000FE                       ||
||                                       || R[03]: 00000000000000000000000000000000 || M1: 00000000500007FFF0000800000001AB5 || A1: 00000000500007FFF0000800000001AB5 || WE: 1                             ||
||                                       || R[10]: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 || M2: 00000000000000005000800000009263 || A2: 00000007000000005000800000009263 || RD: 30                            ||
||                                       || R[25]: 00000000000000000000000000000000 || M3: F6785678567F4040FEFC0000567F4240 || A3: F6785678567F4040FEFC0000567F4240 || R[30]: 00000007000000005000800000009263 ||
||                                       ||                                        || FD: 00000007000000005000800000009263 || AO: 0000000C00007FFF0000800000000AD18 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 37
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 06FFFFC                  || opcode: 0C4203C                        || function: 0ECA87C                                                           || field: 1827BFD                       ||
```

```
||                                    || R[01]: 0000000000000000000000000000000 || M1: 0000000000000000000000000000000 || A1: 0000000000000000000000000000000 || WE: 1                                  ||
||                                    || R[08]: F0005678FFF0BFBFFEB40000FFF0BDBF || M2: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 || A2: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 || RD: 29                                 ||
||                                    || R[08]: F0005678FFF0BFBFFEB40000FFF0BDBF || M3: 0000000000000000000000000000000 || A3: 0000000000000000000000000000000 || R[29]: 0000000C00007FFF0000800000000AD18 ||
||                                    ||                                        || FD: 0000000C00007FFF0000800000000AD18 || AO: 6543000000000000000000000000000 ||                                       ||
||====================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 38
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||======================================||=======================================||=============================================================================||=======================================||
|| instruction: 05FFFFC                 || opcode: 06FFFFC                       || function: 0C4203C                                                        || field: 0ECA87C                       ||
||                                      || R[31]: 6543000000000000000000000000000 || M1: 0000000000000000000000000000000 || A1: 6543000000000000000000000000000 || WE: 1                          ||
||                                      || R[31]: 00000000500007FFF0000800000001AB5 || M2: F0005678FFF0BFBFFEB40000FFF0BDBF || A2: F0005678FFF0BFBFFEB40000FFF0BDBF || RD: 28                         ||
||                                      || R[31]: 00000000500007FFF0000800000001AB5 || M3: F0005678FFF0BFBFFEB40000FFF0BDBF || A3: F0005678FFF0BFBFFEB40000FFF0BDBF || R[28]: 6543000000000000000000000000000 ||
||                                      ||                                       || FD: 6543000000000000000000000000000 || AO: 6543210100000000000000000000000 ||                                       ||
||======================================||=======================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 39
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||======================================||=======================================||=============================================================================||=======================================||
|| instruction: 020019C                 || opcode: 05FFFFC                       || function: 06FFFFC                                                        || field: 0C4203C                       ||
||                                      || R[31]: 6543210100000000000000000000000 || M1: 6543000000000000000000000000000 || A1: 6543210100000000000000000000000 || WE: 1                          ||
||                                      || R[31]: 00000000500007FFF0000800000001AB5 || M2: 00000000500007FFF0000800000001AB5 || A2: 00000000500007FFF0000800000001AB5 || RD: 28                         ||
||                                      || R[31]: 00000000500007FFF0000800000001AB5 || M3: 00000000500007FFF0000800000001AB5 || A3: 00000000500007FFF0000800000001AB5 || R[28]: 6543210100000000000000000000000 ||
||                                      ||                                       || FD: 6543210100000000000000000000000 || AO: 654321010000000007FFF000000000 ||                                       ||
||======================================||=======================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 40
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||======================================||=======================================||=============================================================================||=======================================||
|| instruction: 000001C                 || opcode: 020019C                       || function: 05FFFFC                                                        || field: 06FFFFC                       ||
||                                      || R[12]: 6543210100000007FFF000000000000 || M1: 6543210100000000000000000000000 || A1: 654321010000000007FFF000000000 || WE: 1                          ||
||                                      || R[00]: 12345678000000000000000000000000 || M2: 00000000500007FFF0000800000001AB5 || A2: 00000000500007FFF0000800000001AB5 || RD: 28                         ||
||                                      || R[00]: 12345678000000000000000000000000 || M3: 00000000500007FFF0000800000001AB5 || A3: 00000000500007FFF0000800000001AB5 || R[28]: 654321010000000007FFF000000000 ||
||                                      ||                                       || FD: 654321010000000007FFFFFF00000000 || AO: 654321010000000007FFFFFF000C0000 ||                                       ||
||======================================||=======================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 41
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||======================================||=======================================||=============================================================================||=======================================||
|| instruction: 182E79B                 || opcode: 000001C                       || function: 020019C                                                        || field: 05FFFFC                       ||
||                                      || R[00]: 654321010000000007FFFFFFF00000000 || M1: 654321010000000007FFF000000000 || A1: 654321010000000007FFFFFF00000000 || WE: 1                          ||
||                                      || R[00]: 12345678000000000000000000000000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 28                         ||
||                                      || R[00]: 12345678000000000000000000000000 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[28]: 654321010000000007FFFFFF00000000 ||
||                                      ||                                       || FD: 654321010000000007FFFFFF00000000 || AO: 654321010000000007FFFFFF000C0000 ||                                       ||
||======================================||=======================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 42
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||======================================||=======================================||=============================================================================||=======================================||
|| instruction: 02001FB                 || opcode: 182E79B                       || function: 000001C                                                        || field: 020019C                       ||
||                                      || R[28]: 654321010000000007FFFFFFF000C0000 || M1: 654321010000000007FFFFFF00000000 || A1: 654321010000000007FFFFFF000C0000 || WE: 1                          ||
||                                      || R[25]: 00000000000000000000000000000000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 28                         ||
||                                      || R[05]: F0005678000F4040FEB40000000F4240 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[28]: 654321010000000007FFFFFF000C0000 ||
||                                      ||                                       || FD: 654321010000000007FFFFFF000C0000 || AO: 654321010000000007FFFFFF000C0000 ||                                       ||
||======================================||=======================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 43
====================

||            STAGE 1 - FETCH           ||            STAGE 2 - DECODE           ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||======================================||=======================================||=============================================================================||=======================================||
|| instruction: 008481B                 || opcode: 02001FB                       || function: 182E79B                                                        || field: 000001C                       ||
||                                      || R[15]: 0000000000000000000000000000000 || M1: 654321010000000007FFFFFF000C0000 || A1: 654321010000000007FFFFFF000C0000 || WE: 1                          ||
||                                      || R[00]: 12345678000000000000000000000000 || M2: 00000000000000000000000000000000 || A2: 00000000000000000000000000000000 || RD: 28                         ||
||                                      || R[00]: 12345678000000000000000000000000 || M3: F0005678000F4040FEB40000000F4240 || A3: F0005678000F4040FEB40000000F4240 || R[28]: 654321010000000007FFFFFF000C0000 ||
||                                      ||                                       || FD: 654321010000000007FFFFFF000C0000 || AO: 654321010000000007FFFFFF000C0000 ||                                       ||
||======================================||=======================================||=============================================================================||=======================================||
```

R[RS] = RS DATA M = MUX INPUT DATA FD = FORWARD DATA A = ALU INPUT DATA AO = ALU OUTPUT WE = WRITE ENABLE RD = WRITE DESTINATION R[RD] = WRITE DATA

==========
Cycle 44
==========

STAGE 1 - FETCH	STAGE 2 - DECODE	STAGE 3 - EXECUTE	STAGE 4 - WRITEBACK
instruction: 181737A	opcode: 008481B	function: 02001FB	field: 182E79B
	R[00]: 65432101000000007FFFFFFF000C0000	M1: 00000000000000000000000000000000	A1: 65432101000000007FFFFFFF000C0000	WE: 1
	R[18]: 00000000000000000000000000000000	M2: 12345678000000000000000000000000	A2: 12345678000000000000000000000000	RD: 27
	R[16]: 00000000000000000000000000000000	M3: 12345678000000000000000000000000	A3: 12345678000000000000000000000000	R[27]: 65432101000000007FFFFFFF000C0000
		FD: 65432101000000007FFFFFFF000C0000	AO: 65432101000000007FFFFFFF000F0000	

R[RS] = RS DATA M = MUX INPUT DATA FD = FORWARD DATA A = ALU INPUT DATA AO = ALU OUTPUT WE = WRITE ENABLE RD = WRITE DESTINATION R[RD] = WRITE DATA

==========
Cycle 45
==========

STAGE 1 - FETCH	STAGE 2 - DECODE	STAGE 3 - EXECUTE	STAGE 4 - WRITEBACK
instruction: 184982C	opcode: 181737A	function: 008481B	field: 02001FB
	R[27]: 65432101000000007FFFFFFF000F0000	M1: 65432101000000007FFFFFFF000C0000	A1: 65432101000000007FFFFFFF000F0000	WE: 1
	R[28]: 65432101000000007FFFFFFF000C0000	M2: 00000000000000000000000000000000	A2: 00000000000000000000000000000000	RD: 27
	R[02]: 00040008000400020000000040003	M3: 00000000000000000000000000000000	A3: 00000000000000000000000000000000	R[27]: 65432101000000007FFFFFFF000F0000
		FD: 65432101000000007FFFFFFF000F0000	AO: 65432101000000007FFFFFFF000F4240	

R[RS] = RS DATA M = MUX INPUT DATA FD = FORWARD DATA A = ALU INPUT DATA AO = ALU OUTPUT WE = WRITE ENABLE RD = WRITE DESTINATION R[RD] = WRITE DATA

==========
Cycle 46
==========

STAGE 1 - FETCH	STAGE 2 - DECODE	STAGE 3 - EXECUTE	STAGE 4 - WRITEBACK
instruction: 02000B1	opcode: 184982C	function: 181737A	field: 008481B
	R[01]: F0005678000F4040FEB40000000F4240	M1: 65432101000000007FFFFFFF000F0000	A1: 65432101000000007FFFFFFF000F4240	WE: 1
	R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF	M2: 65432101000000007FFFFFFF000C0000	A2: 65432101000000007FFFFFFF000C0000	RD: 27
	R[09]: 000C0008000C000E00050010000C000D	M3: 00040008000400020000000040003	A3: 00040008000400020000000040003	R[27]: 65432101000000007FFFFFFF000F4240
		FD: 65432101000000007FFFFFFF000F4240	AO: CA864202000000000FFFFFFFE001B4240	

R[RS] = RS DATA M = MUX INPUT DATA FD = FORWARD DATA A = ALU INPUT DATA AO = ALU OUTPUT WE = WRITE ENABLE RD = WRITE DESTINATION R[RD] = WRITE DATA

==========
Cycle 47
==========

STAGE 1 - FETCH	STAGE 2 - DECODE	STAGE 3 - EXECUTE	STAGE 4 - WRITEBACK
instruction: 1830231	opcode: 02000B1	function: 184982C	field: 181737A
	R[05]: 00000000000000000000000000000000	M1: F0005678000F4040FEB40000000F4240	A1: F0005678000F4040FEB40000000F4240	WE: 1
	R[00]: 12345678000000000000000000000000	M2: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF	A2: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF	RD: 26
	R[00]: 12345678000000000000000000000000	M3: 000C0008000C000E00050010000C000D	A3: 000C0008000C000E00050010000C000D	R[26]: CA864202000000000FFFFFFFE001B4240
		FD: CA864202000000000FFFFFFFE001B4240	AO: 3942D148301FAFC000000000311AADC0	

R[RS] = RS DATA M = MUX INPUT DATA FD = FORWARD DATA A = ALU INPUT DATA AO = ALU OUTPUT WE = WRITE ENABLE RD = WRITE DESTINATION R[RD] = WRITE DATA

==========
Cycle 48
==========

STAGE 1 - FETCH	STAGE 2 - DECODE	STAGE 3 - EXECUTE	STAGE 4 - WRITEBACK
instruction: 0000070	opcode: 1830231	function: 02000B1	field: 184982C
	R[17]: 00000000000000000000000000000000	M1: 00000000000000000000000000000000	A1: 00000000000000000000000000000000	WE: 1
	R[00]: 12345678000000000000000000000000	M2: 12345678000000000000000000000000	A2: 12345678000000000000000000000000	RD: 12
	R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF	M3: 12345678000000000000000000000000	A3: 12345678000000000000000000000000	R[12]: 3942D148301FAFC000000000311AADC0
		FD: 3942D148301FAFC000000000311AADC0	AO: 00000000000000000000000000050000	

R[RS] = RS DATA M = MUX INPUT DATA FD = FORWARD DATA A = ALU INPUT DATA AO = ALU OUTPUT WE = WRITE ENABLE RD = WRITE DESTINATION R[RD] = WRITE DATA

==========
Cycle 49
==========

STAGE 1 - FETCH	STAGE 2 - DECODE	STAGE 3 - EXECUTE	STAGE 4 - WRITEBACK
instruction: 1830210	opcode: 0000070	function: 1830231	field: 02000B1
	R[03]: 00000000000000000000000000000000	M1: 00000000000000000000000000000000	A1: 00000000000000000000000000050000	WE: 1
	R[00]: 12345678000000000000000000000000	M2: 12345678000000000000000000000000	A2: 12345678000000000000000000000000	RD: 17
	R[00]: 12345678000000000000000000000000	M3: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF	A3: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF	R[17]: 00000000000000000000000000050000
		FD: 00000000000000000000000000050000	AO: 00050000000500000005000000050000	

R[RS] = RS DATA M = MUX INPUT DATA FD = FORWARD DATA A = ALU INPUT DATA AO = ALU OUTPUT WE = WRITE ENABLE RD = WRITE DESTINATION R[RD] = WRITE DATA

==========

```
      Cycle 50
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 138C60F                   || opcode: 1830210                         || function: 0000070                                                              || field: 1830231                          ||
||                                        || R[16]: 00000000000000000000000000000000 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000000000000 || WE: 1                                ||
||                                        || R[00]: 12345678000000000000000000000000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 17                               ||
||                                        || R[06]: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[17]: 00050000000500000005000000050000 ||
||                                        ||                                         || FD: 00050000000500000005000000050000 || AO: 00000000000000000000000000000003 ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 51
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 118C612                   || opcode: 138C60F                         || function: 1830210                                                              || field: 0000070                          ||
||                                        || R[16]: 00000000000000000000000000000003 || M1: 00000000000000000000000000000000 || A1: 00000000000000000000000000000003 || WE: 1                                ||
||                                        || R[17]: 00050000000500000005000000050000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 16                               ||
||                                        || R[17]: 00050000000500000005000000050000 || M3: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || A3: 0FFFA987FFF0BFBF014BFFFFFFF0BDBF || R[16]: 00000000000000000000000000000003 ||
||                                        ||                                         || FD: 00000000000000000000000000000003 || AO: 00000003000000030000000300000003 ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 52
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 118C612                   || opcode: 118C612                         || function: 138C60F                                                              || field: 1830210                          ||
||                                        || R[16]: 00000003000000030000000300000003 || M1: 00000003000000030000000300000003 || A1: 00000003000000030000000300000003 || WE: 1                                ||
||                                        || R[17]: 00050000000500000005000000050000 || M2: 00050000000500000005000000050000 || A2: 00050000000500000005000000050000 || RD: 16                               ||
||                                        || R[17]: 00050000000500000005000000050000 || M3: 00050000000500000005000000050000 || A3: 00050000000500000005000000050000 || R[16]: 00000003000000030000000300000003 ||
||                                        ||                                         || FD: 00000003000000030000000300000003 || AO: 00000016000000160000001600000016 ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 53
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 1294A0D                   || opcode: 118C612                         || function: 118C612                                                              || field: 138C60F                          ||
||                                        || R[16]: 00000003000000030000000300000003 || M1: 00000003000000030000000300000003 || A1: 00000003000000030000000300000003 || WE: 1                                ||
||                                        || R[17]: 00050000000500000005000000050000 || M2: 00050000000500000005000000050000 || A2: 00050000000500000005000000050000 || RD: 15                               ||
||                                        || R[17]: 00050000000500000005000000050000 || M3: 00050000000500000005000000050000 || A3: 00050000000500000005000000050000 || R[15]: 00000016000000160000001600000016 ||
||                                        ||                                         || FD: 00000016000000160000001600000016 || AO: 0000001C0000001C0000001C0000001C ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 54
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 1853153                   || opcode: 1294A0D                         || function: 118C612                                                              || field: 118C612                          ||
||                                        || R[16]: 00000003000000030000000300000003 || M1: 00000003000000030000000300000003 || A1: 00000003000000030000000300000003 || WE: 1                                ||
||                                        || R[18]: 0000001C0000001C0000001C0000001C || M2: 00050000000500000005000000050000 || A2: 00050000000500000005000000050000 || RD: 18                               ||
||                                        || R[18]: 0000001C0000001C0000001C0000001C || M3: 00050000000500000005000000050000 || A3: 00050000000500000005000000050000 || R[18]: 0000001C0000001C0000001C0000001C ||
||                                        ||                                         || FD: 0000001C0000001C0000001C0000001C || AO: 0000001C0000001C0000001C0000001C ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 55
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
|| instruction: 14841F4                   || opcode: 1853153                         || function: 1294A0D                                                              || field: 118C612                          ||
||                                        || R[10]: 870FFA9FEFFFFC2FFFF014BEDFFFF85 || M1: 00000003000000030000000300000003 || A1: 00000003000000030000000300000003 || WE: 1                                ||
||                                        || R[12]: 3942D148301FAFC000000000311AADC0 || M2: 0000001C0000001C0000001C0000001C || A2: 0000001C0000001C0000001C0000001C || RD: 18                               ||
||                                        || R[10]: 870FFA9FEFFFFC2FFFF014BEDFFFF85 || M3: 0000001C0000001C0000001C0000001C || A3: 0000001C0000001C0000001C0000001C || R[18]: 0000001C0000001C0000001C0000001C ||
||                                        ||                                         || FD: 0000001C0000001C0000001C0000001C || AO: 0000030D0000030D0000030D0000030D ||                                         ||
||========================================||=========================================||===================================================================================||=========================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 56
====================

||             STAGE 1 - FETCH            ||             STAGE 2 - DECODE            ||                                 STAGE 3 - EXECUTE                                 ||           STAGE 4 - WRITEBACK           ||
||========================================||=========================================||===================================================================================||=========================================||
```

```
|| instruction: 14841F5                || opcode: 14841F4                          || function: 1853153                                                                                  || field: 1294A0D                          ||
||                                     || R[15]: 0000001600000016000000 1600000016 || M1: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 || A1: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 || WE: 1                                ||
||                                     || R[16]: 0000000300000003000000 0300000003 || M2: 3942D148301FAFC000000000311AADC0 || A2: 3942D148301FAFC000000000311AADC0 || RD: 13                               ||
||                                     || R[16]: 0000000300000003000000 0300000003 || M3: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 || A3: 870FFFA9FEFFFFC2FFFF014BEDFFFF85 || R[13]: 0000030D0000030D0000030D0000030D ||
||                                     ||                                          || FD: 0000030D0000030D0000030D0000030D || AO: 870F0057FEFF003E00000000EDFF007B ||                                         ||
||=====================================||==========================================||=====================================================================================||=========================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 57
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 040000F                 || opcode: 14841F5                          || function: 14841F4                                                           || field: 1853153                        ||
||                                      || R[15]: 0000001600000016000000 1600000016 || M1: 0000001600000016000000 1600000016 || A1: 0000001600000016000000 1600000016 || WE: 1                              ||
||                                      || R[16]: 0000000300000003000000 0300000003 || M2: 0000000300000003000000 0300000003 || A2: 0000000300000003000000 0300000003 || RD: 19                             ||
||                                      || R[16]: 0000000300000003000000 0300000003 || M3: 0000000300000003000000 0300000003 || A3: 0000000300000003000000 0300000003 || R[19]: 870F0057FEFF003E00000000EDFF007B ||
||                                      ||                                          || FD: 870F0057FEFF003E00000000EDFF007B || AO: 000000160000001F000000160000001F ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 58
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 0C0000F                 || opcode: 040000F                          || function: 14841F5                                                           || field: 14841F4                        ||
||                                      || R[00]: 0000001600000016000000 1600000016 || M1: 0000001600000016000000 1600000016 || A1: 0000001600000016000000 1600000016 || WE: 1                              ||
||                                      || R[00]: 1234567800000000000000 0000000000 || M2: 0000000300000003000000 0300000003 || A2: 0000000300000003000000 0300000003 || RD: 20                             ||
||                                      || R[00]: 1234567800000000000000 0000000000 || M3: 0000000300000003000000 0300000003 || A3: 0000000300000003000000 0300000003 || R[20]: 000000160000001F000000160000001F ||
||                                      ||                                          || FD: 000000160000001F000000160000001F || AO: 000000160000001F000000160000001F ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 59
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 16AD5F6                 || opcode: 0C0000F                          || function: 040000F                                                           || field: 14841F5                        ||
||                                      || R[00]: 0000001600000016000000 1600000016 || M1: 0000001600000016000000 1600000016 || A1: 0000001600000016000000 1600000016 || WE: 1                              ||
||                                      || R[00]: 1234567800000000000000 0000000000 || M2: 1234567800000000000000 0000000000 || A2: 1234567800000000000000 0000000000 || RD: 21                             ||
||                                      ||                                          || M3: 000000160000001F000000160000001F || A3: 000000160000001F000000160000001F || R[21]: 000000160000001F000000160000001F ||
||                                      ||                                          || FD: 0000001600000016000000 0000000016 || AO: 0000000000000016000000 0000000016 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 60
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 17AD5F7                 || opcode: 16AD5F6                          || function: 0C0000F                                                           || field: 040000F                        ||
||                                      || R[15]: 0000001600000016000000 0000000016 || M1: 0000001600000016000000 1600000016 || A1: 0000001600000016000000 0000000016 || WE: 1                              ||
||                                      || R[21]: 000000160000001F000000 160000001F || M2: 1234567800000000000000 0000000000 || A2: 1234567800000000000000 0000000000 || RD: 15                             ||
||                                      || R[21]: 000000160000001F000000 160000001F || M3: 1234567800000000000000 0000000000 || A3: 1234567800000000000000 0000000000 || R[15]: 0000001600000016000000 0000000016 ||
||                                      ||                                          || FD: 0000001600000016000000 0000000016 || AO: 0000000000000016000000 0000000016 ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 61
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 1875DF8                 || opcode: 17AD5F7                          || function: 16AD5F6                                                           || field: 0C0000F                        ||
||                                      || R[15]: 0000000000000016000000 0000000016 || M1: 0000001600000016000000 1600000016 || A1: 0000000000000016000000 0000000016 || WE: 1                              ||
||                                      || R[21]: 000000160000001F000000 160000001F || M2: 000000160000001F000000 160000001F || A2: 000000160000001F000000 160000001F || RD: 15                             ||
||                                      || R[21]: 000000160000001F000000 160000001F || M3: 000000160000001F000000 160000001F || A3: 000000160000001F000000 160000001F || R[15]: 0000000000000016000000 0000000016 ||
||                                      ||                                          || FD: 0000000000000016000000 0000000016 || AO: 00000000000003AB00000000000003AB ||                                       ||
||=======================================||========================================||=============================================================================||=======================================||
     R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 62
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||========================================||=============================================================================||=======================================||
|| instruction: 1800000                 || opcode: 1875DF8                          || function: 17AD5F7                                                           || field: 16AD5F6                        ||
||                                      || R[15]: 0000000000000016000000 0000000016 || M1: 0000000000000016000000 1600000016 || A1: 0000000000000016000000 0000000016 || WE: 1                              ||
||                                      || R[23]: 0000000000000000000000 0000000000 || M2: 000000160000001F000000 160000001F || A2: 000000160000001F000000 160000001F || RD: 22                             ||
||                                      || R[14]: 0000000000000000000000 0000000000 || M3: 000000160000001F000000 160000001F || A3: 000000160000001F000000 160000001F || R[22]: 00000000000003AB00000000000003AB ||
||                                      ||                                          || FD: 00000000000003AB00000000000003AB || AO: 00000000000001CE00000000000001CE ||                                       ||
```

```
||=======================================||=======================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 63
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||=======================================||=============================================================================||=======================================||
|| instruction: 1800000                  || opcode: 1800000                        || function: 1875DF8                                                        || field: 17AD5F7                       ||
||                                       || R[00]: 12345678000000000000000000000000 || M1: 00000000000000160000000000000016 || A1: 00000000000000160000000000000016 || WE: 1                             ||
||                                       || R[00]: 12345678000000000000000000000000 || M2: 000000000000000000000000000001CE || A2: 000000000000001CE00000000000001CE || RD: 23                            ||
||                                       || R[00]: 12345678000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[23]: 00000000000001CE00000000000001CE ||
||                                       ||                                        || FD: 000000000000001CE00000000000001CE || AO: 000000000000001B8000000000001B8 ||                                      ||
||=======================================||=======================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 64
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||=======================================||=============================================================================||=======================================||
|| instruction: 1872459                  || opcode: 1800000                        || function: 1800000                                                        || field: 1875DF8                       ||
||                                       || R[00]: 12345678000000000000000000000000 || M1: 12345678000000000000000000000000 || A1: 12345678000000000000000000000000 || WE: 1                             ||
||                                       || R[00]: 12345678000000000000000000000000 || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 24                            ||
||                                       || R[00]: 12345678000000000000000000000000 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[24]: 000000000000001B8000000000000001B8 ||
||                                       ||                                        || FD: 000000000000001B8000000000001B8 || AO: 00000000000000000000000000000000 ||                                      ||
||=======================================||=======================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 65
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||=======================================||=============================================================================||=======================================||
|| instruction: 1872459                  || opcode: 1872459                        || function: 1800000                                                        || field: 1800000                       ||
||                                       || R[02]: 0004000800040002000B000000040003 || M1: 12345678000000000000000000000000 || A1: 12345678000000000000000000000000 || WE: 0                             ||
||                                       || R[09]: 000C0008000C000E00050010000C000D || M2: 12345678000000000000000000000000 || A2: 12345678000000000000000000000000 || RD: 00                            ||
||                                       || R[14]: 00000000000000000000000000000000 || M3: 12345678000000000000000000000000 || A3: 12345678000000000000000000000000 || R[00]: 00000000000000000000000000000000 ||
||                                       ||                                        || FD: 00000000000000000000000000000000 || AO: 00000000000000000000000000000000 ||                                      ||
||=======================================||=======================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 66
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||=======================================||=============================================================================||=======================================||
|| instruction: 1872459                  || opcode: 1872459                        || function: 1872459                                                        || field: 1800000                       ||
||                                       || R[02]: 0004000800040002000B000000040003 || M1: 0004000800040002000B000000040003 || A1: 0004000800040002000B000000040003 || WE: 0                             ||
||                                       || R[09]: 000C0008000C000E00050010000C000D || M2: 000C0008000C000E00050010000C000D || A2: 000C0008000C000E00050010000C000D || RD: 00                            ||
||                                       || R[14]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[00]: 00000000000000000000000000000000 ||
||                                       ||                                        || FD: 00000000000000000000000000000000 || AO: 00080000000800OC0005FFF00008000A ||                                      ||
||=======================================||=======================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



====================
      Cycle 67
====================

||            STAGE 1 - FETCH            ||            STAGE 2 - DECODE            ||                              STAGE 3 - EXECUTE                              ||          STAGE 4 - WRITEBACK          ||
||=======================================||=======================================||=============================================================================||=======================================||
|| instruction: 1872459                  || opcode: 1872459                        || function: 1872459                                                        || field: 1872459                       ||
||                                       || R[02]: 0004000800040002000B000000040003 || M1: 0004000800040002000B000000040003 || A1: 0004000800040002000B000000040003 || WE: 1                             ||
||                                       || R[09]: 000C0008000C000E00050010000C000D || M2: 000C0008000C000E00050010000C000D || A2: 000C0008000C000E00050010000C000D || RD: 25                            ||
||                                       || R[14]: 00000000000000000000000000000000 || M3: 00000000000000000000000000000000 || A3: 00000000000000000000000000000000 || R[25]: 000800000008000C0005FFF00008000A ||
||                                       ||                                        || FD: 00080000000800OC0005FFF00008000A || AO: 00080000000800OC0005FFF00008000A ||                                      ||
||=======================================||=======================================||=============================================================================||=======================================||
    R[RS] = RS DATA     M = MUX INPUT DATA     FD = FORWARD DATA      A = ALU INPUT DATA     AO = ALU OUTPUT     WE = WRITE ENABLE     RD = WRITE DESTINATION     R[RD] = WRITE DATA



R0  12345678000000000000000000000000
R1  F0005678000F4040FEB40000000F4240
R2  0004000800040002000B000000040003
R3  56780000567800005678000056780000
R4  F6785678567F4040FEFC0000567F4240
R5  F0005678000F4040FEB40000000F4240
R6  0FFFA987FFF0BFBF014BFFFFFFF0BDBF
R7  0FFFA987000F4040014BFFFF000F4240
R8  F0005678FFF0BFBFFEB40000FFF0BDBF
R9  000C0008000C000E00050010000C000D
R10 870FFFA9FEFFFFC2FFFF014BEDFFFF85
R11 00070054007F005F0000007F007F005E
R12 3942D148301FAFC000000000311AADC0
R13 0000030D0000030D0000030D0000030D
R14 00000000000000000000000000000000
R15 00000000000000160000000000000016
```

```
R16 0000000300000003000000300000003
R17 0005000000050000000500000050000
R18 0000001C0000001C0000001C0000001C
R19 870F0057FEFF003E00000000EDFF007B
R20 0000001600000001F000000160000001F
R21 0000001600000001F000000160000001F
R22 00000000000003AB00000000000003AB
R23 00000000000001CE00000000000001CE
R24 00000000000001B800000000000001B8
R25 0008000000008000C0005FFF00008000A
R26 CA86420200000000FFFFFFFE001B4240
R27 654321010000000 7FFFFFFF000F4240
R28 6543210100000007FFFFFFF000C0000
R29 0000000C00007FFF000080000000AD18
R30 0000000700000005000080000000 9263
R31 0000000500007FFF0000800000001AB5
```