ECE 586: COMPUTER ARCHITECTURE

Course Instructor: Prof. Mark Faust

FINAL PROJECT REPORT: PDP8 FLOATING POINT OPERATIONS

Submitted By:

- Kirtan Ketanbhai Mehta
- Mohammad Suheb Zameer
- Chenyang Li
- Vinoth Ganesh

<u>OBJECTIVE</u>:

The main objective of the project is to include Floating Point Operations into existing PDP8 Instruction Set Architecture using IEEE 754 format for Floating point Numbers.

<u>Description</u>:

A PDP8 Assembly code is converted to an object code output (.mem file) by using a PAL assembler. The output .mem file is read using Verilog's *$readmemh()* function. PDP8 uses 12 bit words and IEEE 754 uses 32 bit words. A SystemVerilog code is used to convert the floating point numbers into 3 12-bit word format (each word represented in octal format) and is appended to the .mem file produced by the PAL assembler. Bash scripting is used to automate the simulation process.

The following new instructions are being included into the existing PDP-8 ISA:

1. FPCLAC (opcode 0)
2. FPLOAD (opcode 1)
3. FPSTOR (opcode 2)
4. FPADD (opcode 3)
5. FPMULT (opcode 4)

The above new instructions are represented using ***Input Output Transfer (IOT) Instructions*** using ***Device code 55_8*** for floating point instructions. Three opcode bits are used to represent each floating point operation.

<u>Floating Point Operations:</u>

- **FPCLAC** : The bits 9 to 11 of the IOT are used to represent the opcode. Each Instruction from the memory is stored in a 12-bit Instruction register IR. If IR[9:11] is 0 , then FPCLAC operation is used to clear the contents of the Floating point Accumulator.

- **FPLOAD** : If the IR[9:11] is 1, then *FP_Load* task is invoked which calculates he Effective address and loads the Floating point accumulator with the floating point value fetched from the corresponding memory address location.

- **FPSTOR**: If the IR [9:11] is 2, then *FP_store* task is invoked which stores the content of the Accumulator in the corresponding Memory address location.

- **FPADD** : If the IR [9:11] is 3, *FP_add* task is invoked. Three conditions are checked during foating point addition. If the Exponents of both the operands are equal (exp_1 == exp_2), the output mantissa is the sum of the mantissa's of the input operands. If the (exp_1 > exp_2), the mantissa of the smaller operand is shifted to the right (exp1-exp2) number of times to match both the exponents. The output exponent is the exponent value of the bigger operand and the output mantissa is the sum of the new mantissa obtained by shifting the smaller operand and the mantissa of the bigger operand.

- **FPMULT**: If the IR [9:11] is 4, *FP_mult task* is invoked. The exponent fields are added along with the bias and the output exponent is calculated along with the bias to be represented in IEEE 754 format. The multiplication operation is performed using the Shift + Add operations depending upon the number of 1's in the mantissa of the second operand.

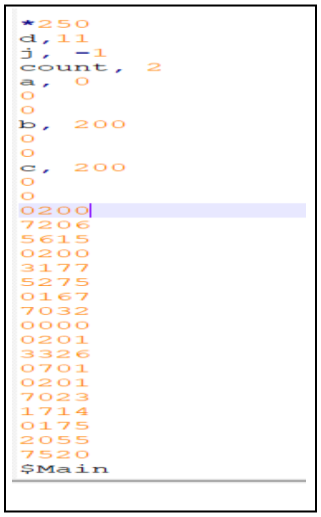
  A normalization factor is included depending upon the MSB bit of the output Mantissa.

  The sign bit is computed using an XOR operation where the output sign bit is 0 (positive) if the sign bit of the operands are same and the output sign bit is 1(negative) if the sign bit of the operands are opposite.

<u>Test Cases:</u>

Some Test cases are generated to test the Floating point operations and the results are verified manually. After Round-off the results in the accumulator are verified to be close to the expected value and the maximum difference between the expected and output result is found to be 1. Test cases were generated manually and also randomized using a C code and the results are verified.

<u>Example Test Case:</u>



```
*250
d,11
j, -1
count, 2
a, 0
0
0
b, 200
0
0
c, 200
0
0
0200
7206
5615
0200
3177
5275
0167
7032
0000
0201
3326
0701
0201
7023
1714
0175
2055
7520
$Main
```

Underflow and Overflow Scenarios:

There are three cases to be considered before detecting an overflow/underflow.

Case 1: If the sign bit of any one of the operand is '1' (negative) and the exponent bits of ay one of the operand are all zero, output is denormalized.

Case 2: If the sign bit of any one of the operand is '0' (positive) and the exponent bits of ay one of the operand are all zero, output in accumulator is Zero.

Case 3:If the exponent bits of any one of the operand are all 1's (255) , Output in accumulator is NAN or infinity depending on the value of significand.

Work Split-up:

1.Kirtan Ketanbhai Mehta: Performed FP_ADD and FP_MUL operations, converted the contents of .mem file to PDP-8 format (3 12-bit word format) , bash Scripting to automate simulation, Wrote Test cases and Verification of FP_MUL operation.

2.Mohammad Suheb Zameer : Handled Overflow/Underflow conditions, Carried out FP_MUL operation , Wrote Test Cases and Verification of FP_ADD and FP_MUL Operations.

3.Chenyang Li: Handled Overflow/Underflow conditions, Carried out FP_ADD ,FPLOAD and FP_STORE Operation , Automated test cases and performed verification of Floating point Add operation.

4. Vinoth Ganesh : Reading operands using Command line , Checked conditions for octal and hex representation, Performed FP_CLAC operation and assisted with Test Case generation and Final Project Report.

References:

1. PDP-8 Emulator Program User's manual.
2. ECE 586- Slides by Mark Faust.
3. http://www.rfwireless-world.com/Tutorials/floating-point-tutorial.html
4. Introduction to System Verilog by Chris spear
5. PAL (PDP-8 assembler) from Resource Page of Course Website.
6. Mark A Erle, BrianJ. Hickmann, Michael J. Schulte, IEEE " Decimal Floating Point Multiplication",DOI : 10.1109/TC.2008.218, IS SN: 0018-9370.