Computer Systems

Midterm Exam

Note) Submit all the source code by zipping it into a single file, including the files I gave you. Your source code should be ready for the TA to run without any manual modifications.

1. (Total 10 points) Converting C Code to RISC-V Instructions

You are given a fragment of C code. Your task is to convert this C code into a corresponding RISC-V instruction stream and ensure it runs correctly on a single-cycle RISC-V CPU. The following steps outline your work:

```
void process_array(int arr[10]) {
    int i, sum = 0;
    for (i = 0; i < 10; i++) {
        if (arr[i] % 2 == 0) {
            sum += arr[i];
        } else {
            sum -= arr[i];
        }
        if (sum > 50) {
            exit(0);
        }
    }
}
```

1.1. Instruction Stream Creation (3 pts): Convert the given C code into a minimized RISC-V instruction stream.

1.2. Instruction Memory Implementation (3 pts): Implement the instruction memory such that the RISC-V processor can access and execute your instruction stream.

1.3. Single-Cycle CPU Modification (4 pts): Modify or create the single-cycle RISC-V CPU code to run your generated instruction stream. Make sure that the CPU correctly handles all the instructions needed to execute the given C code.

Grading Criteria

1.1 Instruction Stream Creation (3 points):

- **Full points if it works correctly.**
- **Half points if there are some functionality issues.**
- **Zero points if it does not work.**

1.2 Instruction Memory Implementation (3 points):

- **Full points if it works correctly.**
- **Half points if there are some functionality issues.**
- **Zero points if it does not work.**

1.3 Single-Cycle CPU Modification (4 points):

- **Full points if it works correctly.**
- **Half points if there are some functionality issues.**
- **Zero points if it does not work.**

2. (Total 10 pts) You are given the following fragment of an existing program represented as an instruction stream:

```
add t0, t1, t2
sub t3, t0, t4
mul t5, t3, t6
and t7, t5, t8
or t0, t7, t9
xor t1, t0, t2
sll t2, t1, t3
srl t4, t2, t5
add t6, t4, t7
sub t8, t6, t9
mul t0, t8, t1
div t3, t0, t2
rem t5, t3, t4
add t7, t5, t6
sub t9, t7, t8
xor t1, t9, t0
and t2, t1, t3
or t4, t2, t5
sll t6, t4, t7
srl t8, t6, t9
add t0, t8, t1
sub t3, t0, t2
mul t5, t3, t4
and t7, t5, t6
or t9, t7, t8
xor t0, t9, t1
sll t2, t0, t3
srl t4, t2, t5
div t6, t4, t7
rem t8, t6, t9
add t1, t8, t0
sub t3, t1, t2
mul t5, t3, t4
xor t7, t5, t6
and t9, t7, t8
or t0, t9, t1
sll t2, t0, t3
srl t4, t2, t5
add t6, t4, t7
sub t8, t6, t9
```

This instruction stream is part of a larger computational program that performs a series of arithmetic, logical, and bitwise operations on data stored in registers. It represents a generic data processing routine, possibly from an algorithm that

involves complex mathematical transformations or manipulations of data, such as digital signal processing, cryptographic operations, or graphics calculations. The operations include arithmetic additions, subtractions, multiplications, divisions, bitwise AND, OR, XOR, and shifts, suggesting the program is transforming input data through multiple stages to produce a desired result.

Your task is to work with this instruction stream and optimize it using RISC-V instructions. The following tasks are required:

2.1. Instruction Stream Optimization Using Standard RISC-V Instructions (5 pts)
Given the provided instruction stream, reduce the number of instructions as much as possible using only the standard RISC-V instructions defined in the RISC-V ISA. Your goal is to minimize the instruction count while maintaining the same functionality. Document your changes and the reasoning behind the optimizations you made.

2.2. Instruction Stream Optimization Using a Custom Instruction (5 pts)
Define a single custom instruction to further optimize the instruction stream. You are required to create a custom instruction that can combine multiple operations from the given instruction stream into a single operation to reduce the instruction count. Describe the custom instruction, how it works, and demonstrate how it helps minimize the instruction count. Modify the CPU datapath and control logic to support your new custom instruction, and provide the updated assembly program that includes this new instruction.

Grading Criteria

2.1 Instruction Stream Optimization Using Standard RISC-V Instructions (5 points):

- **Full points if it works correctly.**
- **Half points if there are some functionality issues.**
- **Zero points if it does not work.**

2.2 Instruction Stream Optimization Using a Custom Instruction (5 points):

- **5 points for the fewest instructions.**
- **4 points for the second fewest.**
- **3 points for the third fewest.**
- **2 points for all others.**
- **Zero points if it does not work.**

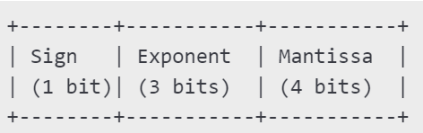
3. Designing and Implementing FP8 Matrix Multiplication (Total 15 pts)

The FP8 (8-bit Floating Point) format is a compact representation for floating-point numbers, often used in machine learning to save memory and improve computational efficiency. FP8 uses a smaller number of bits compared to standard floating-point formats, making it suitable for applications where lower precision is acceptable.

FP8 consists of:

- Exponent Bias: The exponent bias for FP8 is 7, which is used to represent both positive and negative exponents.
- 1-bit sign: Represents the sign of the number (0 for positive, 1 for negative).
- 3-bit exponent: Represents the exponent, used to scale the value.
- 4-bit mantissa: Represents the significant digits (or fraction) of the number.

The diagram below illustrates the FP8 format structure:



Your task is to design and implement matrix multiplication using the FP8 format.

Below is an example of your target program:

```
// Matrix multiplication function for FP8 format
void fp8_matrix_mult(fp8 A[2][2], fp8 B[2][2], fp8 C[2][2]) {
    int i, j, k;
    for (i = 0; i < 2; i++) {
        for (j = 0; j < 2; j++) {
            fp8 sum = 0x00;
             // Assuming 0x00 represents zero in FP8 format
            for (k = 0; k < 2; k++) {
                // Perform multiplication and addition in FP8
                sum += (A[i][k] * B[k][j]);
            }
            C[i][j] = sum;
        }
    }
}


int main()
{
  // Initialize matrices A and B with example floating-point values
   //cast to FP8
    fp8 A[2][2] =
   {
        {(fp8)3.75, (fp8)-4.0},
        {(fp8)4.25, (fp8)3.5}
    };
    fp8 B[2][2] =
   {
        {(fp8)-3.5, (fp8)4.1},
        {(fp8)3.75, (fp8)-3.6}
    };
    fp8 C[2][2];
    fp8_matrix_mult(A, B, C);

    return 0;
}
```

The following tasks are required:

3.1. Instruction Set Extension for FP8 Operations (5 pts)

Define the necessary instructions to perform matrix multiplication using FP8 values. These instructions should include loading FP8 values, performing multiplication and addition, and storing the results. Describe the syntax and functionality of each instruction, and explain how they are used in the matrix multiplication process.

3.2. Datapath and Functional Unit Design (10 pts)

In this program, A, B, and C are 2x2 matrices represented directly in FP8 format. The multiplication and addition operations are performed in FP8 without converting to other floating-point types. We assume that FP8 arithmetic operations are directly supported.

Design the datapath and functional units required to support FP8 matrix multiplication in a single-cycle CPU. This question includes designing the FP8 arithmetic unit that can handle addition and multiplication operations. Provide a detailed diagram of the modified datapath, and explain how the control logic is modified to support the new FP8 operations.

Grading Criteria

3.1 Instruction Set Extension for FP8 Operations (5 points):

- **Full points if it works correctly.**
- **Half points if there are some functionality issues.**
- **Zero points if it does not work.**

3.2 Datapath and Functional Unit Design (10 points):

- **10 points for the shortest cycle.**
- **8 points for the second shortest.**
- **6 points for the third shortest.**
- **5 points for all others.**
- **3 points if there are some functionality issues.**
- **Zero points if it does not work.**