Evan Smith

Final Exam - Take-home portion

You are required to propose complete processor/memory architecture to be used in a certain application. Initially, you need to propose a name for your new architecture and explain what application your proposed architecture will be used in.

Overall goal of this processor is to run an integrated sensor in a smart home system. Priorities are low power consumption and simple operations for monitoring sensors and communicating back to the base.

1. Identify a set of processor parameters and structures.

You need to define the components, unique parameters, and instruction set for your proposed processor structure. To determine the instruction set, you will have to create your own set of instructions that your processor can handle.

In your proposed architecture, specify the following:

1.1 Processor architecture and instruction set:

- Determine the processor components and the characteristics for each component.
    - **MIPS-style components:**
        - **ALU with Forwarding Unit**
        - **Register Stack**
        - **Controller**
        - **Accumulator**
        - **Program Counter**
        - **Additional component**
        - **RC port for communication**
- Determine how many registers there are in your computer architecture.
    - **16 registers, R1-R15, with R0 as a zero.**
    - **Registers are 16 bits, to allow for higher precision**
- Determine the instructions that your proposed processor can handle.
    - **ADD, SUB, MUL, DIV, MOD, ABS**
    - **TEM (read temperature into register)**
    - **HUM (read humidity into register)**
    - **MSN (check proximity sensor into register)**
    - **LDW, STW, MOV (move register to register)**
    - **TRN (transmit register over RC)**
    - **CLK (check ticks since last execution)**

- Determine the instruction size.
    - **Fixed instruction size of 24 bits**
    - **Want to maintain a simple architecture with a smaller instruction decoding system to promote efficient operation.**

1.2 Instruction type and format:

- Determine how many instruction types your processor will support. And the format of each type.
    - **All types begin with a 4-bit instruction, followed by different blocks based on type**
    - **R-type: (arithmetic / sensor interactions)**
        - **INST – REG (repeated up to 5 registers)**
        - **REG is 4 bits address of the register**
    - **J-type: (jump)**
        - **INST – ADD - Extra**
        - **ADD is 16 bits**
        - **4 bits unused**
    - **M-type: (memory access and immediate)**
        - **INST – REG – ADD**
        - **REG is 4 bits**
        - **ADD is 16 bits**

1.3 Processor architecture and characteristics

- You need to determine single core vs multicore and the multithread support or not and why and how the multithread, that your proposed processor has, is supported.
    - **This processor is single-threaded for power consumption reasons. The main thread will simply loop over checks on the various sensors, reporting back when needed. No multithreading required.**
- Processor Pipelining stages (how many stages and the functions in each stage), pipelining hazards that may appear in your proposed processor, and how you are going to overcome these hazards.
    - **Standard 5-stage pipelining**
        - **Fetch, Decode, Execute, Memory, Writeback**
    - **Pipelining hazards: standard to the course**
        - **Data, Control, and Structural**
        - **Because of the forwarding unit, we move forward with the discussed approach in the course and use stall cycles when needed.**
        - **Compiler will work to avoid hazards as well**

2. Identify the memory structure for your proposed architecture.

You should define the memory size, unique parameters, and the instructions that deal with the memory. You need to determine the cache levels and how cache works in your proposed structure.

In your proposed architecture, specify the following:

2.1 Determine the memory structure used in your computer architecture.

You may choose to have a unified memory for both data and instructions or you may prefer a separate memory for each. You are required to justify why you chose a specific architecture.

- **We use a separate memory. This is because the variation in data will be very large, since the data will primarily come from analog sensor readings, but the instruction set is very small, and will run in a single main loop that will reuse the same set of instructions.**

2.2 Determine the size of your proposed memory.

- **The Instruction memory will be 4Kb, to enable 1,300 or so instructions, the estimated size of the program scale**
- **The Data memory will be 32Kb, storing 10,600 elements, mostly temperature and proximity information for tracking movement of these values over time. Buffers will be overwritten often, since the sensor will send information back to the hub often**

2.3 Determine what instructions can access the memory and how.

- **Only LDW and STW can access memory, to minimize hazards, and only during the Memory / Writeback steps in the pipeline.**

2.4 Determine the cache levels.

You may choose to have one or more cache levels. In each cache level, you have to determine if this level is a unified or a separate cache for both data and instructions. In addition, you have to select the cache size for each level. You are required to justify why you chose a specific architecture.

- **Separate cache for both data and instructions**
- **Instruction cache is single level, 1Kb. This will be more than sufficient to hold all the instructions used typically in the main monitoring loop. The other unique instructions will trigger only sometimes and will have a smaller memory to traverse.**
- **Data cache is 2 levels, 8Kb each. While there will be some consistent locations, a lot of the data access will be more random based on the analog inputs, so we expect a much lower hit rate regardless of the cache structure.**

2.5 Determine how cache works.

Determine how to locate a block in the cache, and how to choose a block to be replaced from the cache. You are required to justify why you chose a specific architecture.

- **Both caches use direct mapping block placement to simplify and streamline all accesses, going for simplified memory access and more efficient power usage.**
- **The instruction cache will use Least Commonly Used as the replacement strategy. This is because of the overall structure of the code, which will be a loop of the same instructions for the vast majority of executions.**
- **The data cache will use FIFO to take advantage of the potential temporal locality in the sensor data that will often be compared against recent measurements**

2.6 Propose a similar memory sharing protocol similar to the snoopy protocol.

- **This processor will not use a memory-sharing protocol to reduce overhead, and due to the strong locality that should exist in the data and instructions that the processor will handle.**