CSCE 3301 – Computer Architecture

Spring 2024

Project 2: femTomas Tomasulo Algorithm Simulation

Report

Raef Hany: 900213194

Tarek Kassab: 900213491

Shaza Ali: 900213391

Dr. Salama Cherif

This report details the implementation of a simplified processor simulator in C++. The simulator models the core components of a processor's execution pipeline, including instructions, reservation stations, registers, and memory.

A Brief Description of Our Implementation :

The simulator utilizes a modular approach, encapsulating functionality within distinct classes. It simulates the execution of instructions through a pipeline, where instructions progress through stages:

- **Issue:** Instructions are fetched from an instruction list and placed into available reservation stations.
- **Execute:** Instructions are executed in the reservation stations, checking for operand availability and advancing their state.
- **WriteBack:** Upon completion, results are written back to registers or memory.

The simulator implements data dependencies between instructions, ensuring that an instruction cannot execute until its operands are ready. This is managed through reservation stations, which allow for out-of-order execution while preserving data dependencies.

Global Variables

- **instructions (vector):** Stores the list of instructions to be executed.
- **stations (vector):** Represents the functional units (reservation stations) of the processor.
- **registers (vector):** Models the processor's register file.
- *loadStore (vector<Instruction>):* Keeps track of load and store instructions for memory address conflict checking.
- *finished (vector<Instruction>):**Stores the finished instructions.
- **mem (Memory):** Represents the main memory of the simulated system.
- **time (int):** Tracks the simulation time.
- **PC (int):** Represents the program counter, pointing to the current instruction.
- **branchesInFlight (int):** Tracks the number of branch instructions currently in the pipeline.
- *LastBranch (Instruction):* Stores the most recently issued branch instruction.

Functions

- **Simulate():** Advances the simulation by one clock cycle, calling WriteBack(), Execute(), and Issue().
- **WriteBack():** Simulates writing results back to registers or memory after an instruction finishes execution.

- **Execute():** Updates the state of instructions in reservation stations, decrementing execution cycles and checking for operand availability.
- **Issue():** Fetches instructions from the instruction list and places them into available reservation stations.
- **allDone():** Checks if all instructions have finished executing.
- **printdata():** Displays the results of the simulation, including execution times and final register/memory values.

Main Function

- Creates a Simulator object, passing the defined instructions and reservation stations.
- The constructor of Simulator runs the simulation until all instructions are finished or the program counter reaches the end of the instruction list.

Bonus Features: Variable Hardware and Multiple Issue

This section details the implementation of two bonus features that enhance the realism and configurability of the processor simulator:

3. Support a Variable Hardware Organization

The simulator now supports a customizable hardware organization, allowing users to specify:

- **Number of Reservation Stations:** Users can define the number of reservation stations for each class of instructions (load, store, branch, etc.). This is accomplished by modifying the main.cpp file, where the user creates a vector of ReservationStation objects. Each ReservationStation object is created with a specific list of instruction types it can handle.
- **Execution Cycles:** The execution cycles required by each functional unit type (represented by a ReservationStation) can be set by specifying the speed and offspeed parameters. speed defines the cycles needed for non-load/store instructions, while offspeed specifies the cycles required for load/store instructions to calculate the memory address.

Example Code (from main.cpp):

```
    vector<ReservationStation> Ss;
Ss.push_back(ReservationStation({0}, 4, 2));   // Load, speed = 4, offspeed = 2
Ss.push_back(ReservationStation({1}, 4, 2));   // Store, speed = 4, offspeed = 2
Ss.push_back(ReservationStation({2}, 1));      // Branch, speed = 1
// ... other reservation stations with specific instruction types and speeds
```

Benefits:

- **Flexibility:** The simulator allows users to experiment with various hardware configurations. They can explore the impact of different numbers of reservation stations and execution cycles for different instruction classes.
- **Realistic Modeling:** This feature enables users to model different processor architectures with varying hardware resources.

4. Support a Multiple-Issue Hardware Organization

The simulator now supports multiple-issue execution, where multiple instructions can be issued and potentially executed in parallel during each clock cycle.

- **Pipeline Width:** The Simulator constructor accepts an Issues parameter, which specifies the pipeline width (the number of instructions issued per clock cycle).
- **Issuing Instructions:** The Issue() function now loops issues times to try issuing multiple instructions per cycle, if available. This allows the simulator to model processors that can issue multiple instructions concurrently.

Example Code (from main.cpp):

```
Simulator simulator(Is, Ss, 2); // 2 instructions issued per cycle
```

Benefits:

- **Performance Simulation:** Multiple-issue execution is a key performance-enhancing technique in modern processors. This feature allows users to model and analyze the performance of processors with different pipeline widths.
- **Realistic Modeling:** The simulator can now represent the behavior of modern superscalar and superpipelined processors.

Impact of Bonus Features:

These bonus features significantly enhance the simulator's functionality and realism. They enable users to:

- **Explore the performance trade-offs of different hardware configurations:** Compare performance with different numbers of reservation stations and varying execution cycles.
- **Analyze the impact of pipeline width:** Study the performance gains achievable with multiple-issue execution.
- **Model a broader range of processor architectures:** Simulate processors with different organizational features, including variable hardware organization and multiple-issue capabilities.

These bonus features, combined with other elements of the simulator, provide a valuable tool for studying and understanding the complexities of processor design and performance.

The results obtained from the simulation of each assembly program provided (Test cases)

Test Cases

Test Case 1: testing everything

Program:

```
    LOAD R1, 10(R0)   ; Load value from memory location 10 into R1
ADD R2, R1, R1    ; Add R1 to R1 and store the result in R2
NAND R3, R2, R1   ; Perform bitwise NAND between R2 and R1, store in R3
ADDI R4, R3, -5    ; Add immediate value -5 to R3, store in R4
MUL R5, R4, R2    ; Multiply R4 by R2, store in R5
STORE R5, 20(R0)  ; Store value of R5 into memory location 20
CALL label1       ; Call the function 'label1'
STORE R5, 40(R0) ; Store value of R5 into memory location 40
label1:
   LOAD R6, 40(R0)  ; Load value from memory location 30 into R6
   BEQ R6, R0, 2    ; Branch to the previous instruction if R6 is equal to 0
   STORE R6, 50(R0) ; Store value of R6 into memory location 40
   RET              ; Return from the function
```

Data:

- Memory location 10: Value = 1
- Memory location 20: Initialized to 2
- Memory location 30: Value = 0
- Memory location 40: Initialized to 0

Expected Output:

- **Instruction Table:**

Instruction	Issued	Exec.	Written
LOAD R1, 10(R0)	0	6	7

ADD R2, R1, R1	1	8	9
NAND R3, R2, R1	2	9	10
ADDI R4, R3, -5	3	11	12
MUL R5, R4, R2	4	19	20
STORE R5, 20(R0)	5	25	27
CALL label1	6	7	8
LOAD R6, 30(R0)	8	25	26
BEQ R6, R0, -2	9	26	28

- **Total Execution Time:** 29 cycles
- **IPC:** 0.31
- **Branch Misprediction Percentage:** 100%

This case tests all supported instructions, including call/return, conditional branches, and various arithmetic/logic operations. It also covers dependencies and instruction reordering.

Test Case 2: Loops

Program:

```
LOAD R3, 10(R0)   ;
LOAD R2, 20(R0)   ;
Call Label 2
Label 2:
   ADD R3, R3, R2   ;
   ADDI R2, R2, -1   ;
   BQE R2, R0, 2   ;
```

RET

Data:

- Memory location 10: Value = 1
- Memory location 20: Value = 2
- Memory location 30: (Initialized to 0)

Expected Output:

Instruction	Issued	Exec.	Written
LOAD R3, 10(R0)	0	6	7
LOAD R2, 20(R0)	1	7	8
Call Label 2	2	3	4
ADD R3, R3, R2	4	9	10
ADDI R2, R2, -1	5	9	11
BQE R2, R0, 2	6	11	12
RET	7	12	13
ADD R3, R3, R2	13	15	16
ADDI R2, R2, -1	14	16	17
BQE R2, R0, 2	15	17	18

- **Total Execution Time:** 19 cycles
- **IPC:** 0.52
- **Branch Misprediction Percentage:** 75%

This test case examines the simulator's ability to handle repetitive code structures with conditional branches. It also evaluates the branch prediction accuracy, which is crucial for efficient loop execution.

Test Case 3: Multiple Loops

ADDI R4 R0 20
ADDI R3 R0 0
Call Label 3
Label 3:
ADDI R3 R3 5
BEQ R3 R3 1
RET

Expected Output:

- **Instruction Table:**

Instruction	Issued	Exec.	Written
ADDI R4 R0 20	0	2	3
ADDI R3 R0 0	1	3	5
Call Label 3	2	3	4
ADDI R3 R3 5	4	6	7
BEQ R3 R3 1	5	7	8
RET	6	8	9
ADDI R3 R3 5	9	11	12
BEQ R3 R3 1	10	12	13
RET	11	13	14
ADDI R3 R3 5	14	16	17

BEQ R3 R3 1	15	17	18
RET	16	18	19
ADDI R3 R3 5	19	21	22
BEQ R3 R3 1	20	22	23

Total Execution Time: 24 cycles

- **IPC:** 0.625
- **Branch Misprediction Percentage:** 62.5%

Test Results

Test 1:

```
Total Execution Time = 29 Cycles
IPC = 0.310345 Instructions per cycles
Branch Misprediction Ratio = 1
Instruction Data:
Instruction: Load R1, 10(R0)    Issuing Time = 0        Execution Start Time = 1        Execution End Time = 6  Write Time = 7
Instruction: Call label 1 (to Load)     Issuing Time = 6        Execution Start Time = 7        Execution End Time = 7 Write Time = 8
Instruction: ADD R2 R1 R1       Issuing Time = 1        Execution Start Time = 7        Execution End Time = 8  Write Time = 9
Instruction: NAND R3 R2 R1      Issuing Time = 2        Execution Start Time = 9        Execution End Time = 9  Write Time = 10
Instruction: ADDI R4 R3 -5      Issuing Time = 3        Execution Start Time = 10       Execution End Time = 11 Write Time = 12
Instruction: MUL R5 R4 R2       Issuing Time = 4        Execution Start Time = 12       Execution End Time = 19 Write Time = 20
Instruction: Load R6, 40(R0)    Issuing Time = 8        Execution Start Time = 9        Execution End Time = 25 Write Time = 26
Instruction: Store R5, 20(R0)   Issuing Time = 5        Execution Start Time = 20       Execution End Time = 25 Write Time = 27
Instruction: BEQ R6, R0, 2      Issuing Time = 9        Execution Start Time = 26       Execution End Time = 26 Write Time = 28

D:\AUC\CSCE\Computer Architecture\Project 2\Nothing\x64\Debug\Nothing.exe (process 26832) exited with code 0.
Press any key to close this window . . .
```

Test 2:

```
Total Execution Time = 19 Cycles
IPC = 0.526316 Instructions per cycles
Branch Misprediction Ratio = 0.75
Instruction Data:
Instruction: Call label 2 (to ADD R3 R3 R2)     Issuing Time = 2        Execution Start Time = 3        Execution End Time = 3  Write Time = 4
Instruction: Load R3, 10(R0)    Issuing Time = 0        Execution Start Time = 1        Execution End Time = 6  Write Time = 7
Instruction: Load R2, 20(R0)    Issuing Time = 1        Execution Start Time = 2        Execution End Time = 7  Write Time = 8
Instruction: ADD R3 R3 R2       Issuing Time = 4        Execution Start Time = 8        Execution End Time = 9  Write Time = 10
Instruction: ADDI R2 R2 -1      Issuing Time = 5        Execution Start Time = 8        Execution End Time = 9  Write Time = 11
Instruction: BEQ R2, R0, 2      Issuing Time = 6        Execution Start Time = 11       Execution End Time = 11 Write Time = 12
Instruction: RET        Issuing Time = 7        Execution Start Time = 12       Execution End Time = 12 Write Time = 13
Instruction: ADD R3 R3 R2       Issuing Time = 13       Execution Start Time = 14       Execution End Time = 15 Write Time = 16
Instruction: ADDI R2 R2 -1      Issuing Time = 14       Execution Start Time = 15       Execution End Time = 16 Write Time = 17
Instruction: BEQ R2, R0, 2      Issuing Time = 15       Execution Start Time = 17       Execution End Time = 17 Write Time = 18

D:\AUC\CSCE\Computer Architecture\Project 2\Nothing\x64\Debug\Nothing.exe (process 21140) exited with code 0.
Press any key to close this window . . .
```

Test 3:

```
Total Execution Time = 24 Cycles
IPC = 0.583333 Instructions per cycles
Branch Misprediction Ratio = 0.625
Instruction Data:
Instruction: ADDI R4 R0 20      Issuing Time = 0        Execution Start Time = 1        Execution End Time = 2  Write Time = 3
Instruction: Call label 3 (to ADDI R3 R3 5)     Issuing Time = 2        Execution Start Time = 3        Execution End Time = 3  Write Time = 4
Instruction: ADDI R3 R0 0       Issuing Time = 1        Execution Start Time = 2        Execution End Time = 3  Write Time = 5
Instruction: ADDI R3 R3 5       Issuing Time = 4        Execution Start Time = 5        Execution End Time = 6  Write Time = 7
Instruction: BEQ R4, R3, 1      Issuing Time = 5        Execution Start Time = 7        Execution End Time = 7  Write Time = 8
Instruction: RET        Issuing Time = 6        Execution Start Time = 8        Execution End Time = 8  Write Time = 9
Instruction: ADDI R3 R3 5       Issuing Time = 9        Execution Start Time = 10       Execution End Time = 11 Write Time = 12
Instruction: BEQ R4, R3, 1      Issuing Time = 10       Execution Start Time = 12       Execution End Time = 12 Write Time = 13
Instruction: RET        Issuing Time = 11       Execution Start Time = 13       Execution End Time = 13 Write Time = 14
Instruction: ADDI R3 R3 5       Issuing Time = 14       Execution Start Time = 15       Execution End Time = 16 Write Time = 17
Instruction: BEQ R4, R3, 1      Issuing Time = 15       Execution Start Time = 17       Execution End Time = 17 Write Time = 18
Instruction: RET        Issuing Time = 16       Execution Start Time = 18       Execution End Time = 18 Write Time = 19
Instruction: ADDI R3 R3 5       Issuing Time = 19       Execution Start Time = 20       Execution End Time = 21 Write Time = 22
Instruction: BEQ R4, R3, 1      Issuing Time = 20       Execution Start Time = 22       Execution End Time = 22 Write Time = 23

D:\AUC\CSCE\Computer Architecture\Project 2\Nothing\x64\Debug\Nothing.exe (process 19648) exited with code 0.
Press any key to close this window . . .
```