Ben Frey and Joe Lambrecht

Dr. Joseph Myre

CISC 340

21 April 2021

Rip Van saWinkle – Simulator Documentation

UST-3400 Specifications

32-Bit Architecture

Memory: 65536 words

Registers: 8

Pipelined Design

The Rip Van saWinkle CPU architecture is a pipelined implementation of the previous generation of UST-3400 single-cycle simulator. Being a pipelined design, this generation of the UST-3400 architecture has a series of buffers such that 5 instructions are in flight at any one time. This allows for more efficient processing, as portions of multiple instructions are being processed within one cycle, rather than just one instruction being run per cycle as in previous implementations.

To begin, a program of an N amount of words is read into the first N words of UST-3400 instruction memory (as well as the data memory, which is assumed to be an updated duplicate of instruction memory to avoid structural hazards). Two states are created for this simulator. The first, state, contains the values at the beginning of a cycle. This state is not altered within each cycle. The second state, newstate, stores the changed parameters as they are determined within the cycle. At the end of the cycle, newstate is stored into state. This allows us to interpret that the stages of any given cycle take place simultaneously. To begin, the pc is initialized to zero, the pipeline buffers are populated

with NOOP instructions, and the simulator is run with state and newstate as inputs. In each cycle, the following stages occur:

Instruction Fetch

An instruction is read from instruction memory and pc is incremented. The total number of fetched instructions in incremented, as is the number of retired instructions (this is corrected later on if necessary). The newstate IFID buffer stores the instruction and the pc+1 value.

Instruction Decode

The instruction is decoded into its parts- the register values and offset. These values and the pc+1 value are stored in the newstate IDEX buffer.

Execute

First, data hazards are checked (see Hazards below). If necessary, data forwarding and/or load stalling occurs. Otherwise, the aluresult is calculated for each of the instructions accordingly (adding the register values for ADD, inverting the AND of the register values for NAND, adding register B and the offset for LW and SW, and subtracting the registers for BEQ). The aluresult is stored in the newstate EXMEM buffer alongside the branch target (pc+1 + offset), register A, and the instruction.

Memory

If the instruction in memory is a BEQ, increment the total number of branches executed. If the branch is taken, the total number of branches taken is incremented and the pc is set to the branch target, then control hazards are checked for (see Hazards below). For R-Type instructions, the alu result is designated to be written back into the register file. For a LW, the memory is accessed at the address of the alu result and the value at that address is designated to be written back into the register file. For SW instructions, the memory is accessed in a similar fashion, but the value of register A is written to the address accessed. Finally, if it is a halt

instruction, the state statistics of fetched and retired are decremented to offset the unnecessary instructions read in after it. The instruction and the data designated for write back are passed to the Write Back stage.

Write Back

For R-type instructions and LW instructions, the destination register is extracted from the instruction code and the writeback data is written to that register in the register file. As the final step, the state is set to equal the newstate such that all actions in this cycle take effect simultaneously.

Hazards

1. Data Hazards
    a. This simulator checks for data hazards at the start of the execute stage. Data forwarding is checked for by comparing the register values in the IDEX buffer against the stored destination registers for any R-Type or LW instructions in the WBEND, MEMWB, and EXMEM buffers, in that order. If a further instruction would change a register being using by the instruction in the IDEX buffer, the value from that instruction is used instead. If the instruction is a LW in the EXMEM buffer, the value to use instead is not known yet, so a load stall must take place. A load stall “pauses” the PC of the CPU and inserts a bubble into the EXMEM buffer. A bubble, simply a noop along with data parameters of 0, allows for the load word to pass through the memory stage and access a necessary address from data memory. Buffers past EX/MEM are allowed to process (move down the pipeline).
2. Control Hazards

a. This simulator checks for control hazards in the memory stage. If a branch was taken, the IFID, IDEX, and EXMEM buffers are flushed, that is, all the fields in those buffers are set to 0 and the instructions are set to NOOP. These instructions will not make it to the end stage, so the retired statistic must be decremented equal to the number of instructions flushed. Because this simulator always predicts that the branch will not be taken, the amount of mispredictions must be incremented as well.

Development Considerations

The development of the UST-3400 Rip Van saWinkle second-generation CPU presented one significant issue during development. Calculating the fetch instruction when a halt was present was very tricky to overcome. The time spent on testing for, identifying, and solving this issue took longer than the total amount of time on the rest of the simulator. Nevertheless, the issue was found and fixed, and the fetch statistic now computes correctly in all cases.

There is one test case in the autograder (Multihazard #3) in which the cycle count for our simulator was incorrect. We were unable to replicate this in any test case and currently consider it an unidentified corner case.