HW #5: Optimization of the Pipelined CPU

國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

Chun-Jen Tsai
NYCU
05/24/2023

HW #5: Optimization of Pipelined CPU

- Goal: optimize the pipelined CPU in HW#4
  - You should try to reduce hazard induced pipeline stalls
    - Data hazard can be resolved by forwarding
      - Load-use hazard cannot be avoided by in-order processor, so you do not have to handle this
    - Control hazard can be resolved by a branch predictor
  - Your grade will be based on how fast you can execute a sample program provided by the TAs
- The deadline of the HW is on 6/8, by 5:00pm.

Guidelines for HW#5

- You probably should start with adding the forwarding unit to your pipeline first (see textbook section 4.8)
- For the branch prediction unit, you can try the bimodal branch predictor (see textbook section 4.9)
  - You may want to begin with a 1-bit bimodal predictor instead of the 2-bit bimodal predictor described in the textbook

Some Suggestions on BPU (1/2)

- A 1-bit predictor records the previous branch decision (taken or not taken) of a `beq` instruction in the branch history table (BHT)
  - Each BHT entry has the PC of the `beq` instruction, the precious decision, and the target PC
  - The decision bits of the BHT can all be initialized to 1's (taken)
  - You can implement the table using either direct-mapping (need a large BHT, e.g. 256 entries) or fully-associative memory (can use a small BHT, e.g. 16 entries).
- For simplicity, you can use a single BHT to handle all branch instructions (including `j`, `jal`, and `jr`)
  - Although for `j` and `jr`, the decisions are always "taken", we still need to store their target PC somewhere → BHT is an ideal place

Some Suggestions on BPU (2/2)

- The Fetch stage will check the BHT table to see if the current PC points to a branch instruction in the BHT
  - If yes, use the previous decision
    - Decision is 1: PC ← PC target
    - Decision is 0: PC ← PC + 4
  - If not, update the entry in the BHT
    - For direct-mapping BHT, you don't use TAG bits to different branch instructions that share the same entry
    - For fully-associative BHT, you can use FIFO replacement policy
- The Execute stage will update the BHT (set the decision and the target PC) when we have the real decision of a `beq` instruction

HW #5 Grading Guide

- You should extend the size of the instruction and data memory to 64 words each
  - The TAs will use a larger program to test your CPU this time
  - There will be both data and control hazards in the program
  - The fewer cycles your CPU uses to run the program, the better grade you will get
- You should upload your code to E3 by the deadline