CSED311 Lab4-2: Pipelined CPU w/ control flow instructions

Byeongho YU

bhyu418postech.ac.kr

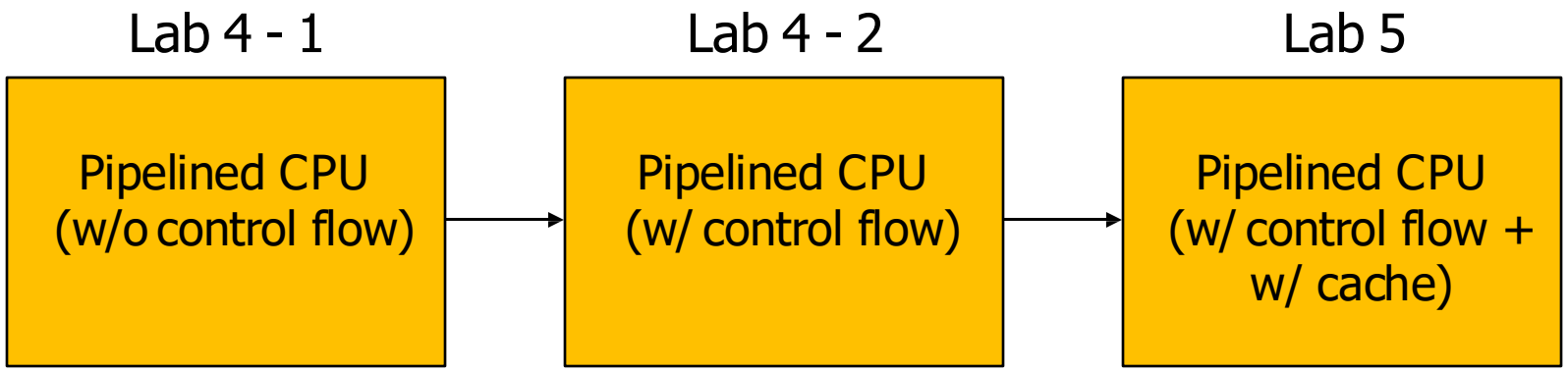
Contact the TAs at cs311-2025ta@postech.ac.kr

Objectives

- Understand and implement a pipelined CPU
- Implement a pipelined CPU with control flows

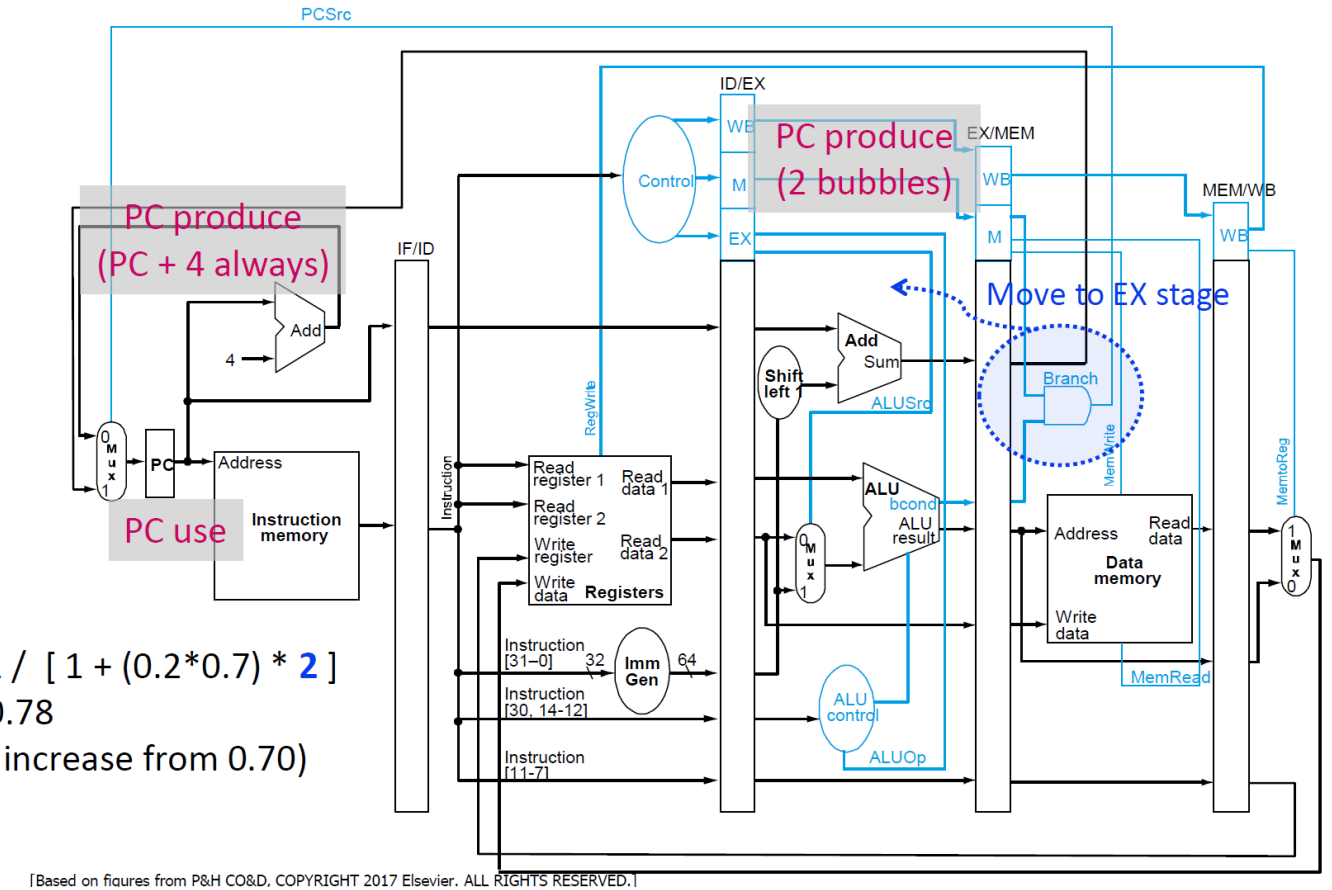
Lab schedule

- You will be implementing a pipelined CPU with control flow instruction support
  - Cache will be implemented in Lab 5



Datapath (w/control flow instructions)

- Resolve at EXE stage (BEQ, BNE, BLT, BGE, JAL, JALR)
  - Miss prediction causes two bubbles



IPC = 1 / [1 + (0.2*0.7) * **2**]
= 0.78
(~11% increase from 0.70)

Submission

- Implementation (Deadline: 5/13 9:00 am)
  - 5-stage pipelined CPU w/ control flow instructions
    - Implement your design over lab4-1 implementation
    - Control hazard
      - Branch prediction (need to flush on misprediction)
        - Always not taken (no extra credit) – Does not require BTB
        - Always taken (partial extra credit +3) – Require BTB (32 entries)
        - 2-bit global prediction (partial extra credit +5) – Require BTB (32 entries)
        - Gshare (full extra credit +7) – Require BTB (32 entries)
        - The BTB must be initialized as empty
  - You need to follow the rules described in lab_guide.pdf

- Report (Deadline: 5/13 18:00)
  - How to handle branch prediction?
  - Describe your design of branch predictor
  - If you implement 2-bit global prediction
    - Compare total cycles of 2-bit global prediction with that of always-taken and always-not-taken
  - If you implement always-taken
    - Compare total cycles of always-taken with that of always-not-taken

Submission

- Implementation file format
  - .zip file name: Lab4-2_{team_num}_{student1_id}_{student2_id}.zip
  - Contents of the zip file (only *.v):
    - cpu.v
    - …
    - Do not include top.v, InstMemory.v, DataMemory, and RegisterFile.v
- Report file format
  - Lab4-2_{team_num}_{student1_id}_{student2_id}.pdf

Questions?

FAQ

- For non control flow instruction, although the next pc value is not stored in the BTB, if the **xor operation** is performed with **BHSR**, there may be cases in which the tag coincides by chance when indexing the BTB, and prediction is made as the next pc value of the control flow instruction.
  So in this case, **miss prediction** will occur.

  - Assuming that such a situation rarely occurs in the lab, you can implement what you learned in class.
  - It doesn’t matter much about the problems that cause conflict. Even if a conflict occurs, you can update it with a new target.

FAQ

- You don't need to match the exact cycle for control flow instructions.
  - It could be different following your branch prediction method
  - You can check the rough cycle number from the result of 5 stage pipeline cpu in "Ripse"