SC3050
Advanced Computer Architecture

Lab 4 briefing

College of Computing and Data Science

Nanyang Technological University

Objectives

1. Implementation and functional verification of five-stage pipeline architecture for R-type, D type, CB type and B type instructions.

Instructions and its opcode (from define.v)

```
`define ADD 11'b00000000000
`define SUB 11'b00000000001
`define AND 11'b00000000010
`define XOR 11'b00000000011
`define ORR 11'b00000000100
`define LDUR 11'b00000000101
`define STUR 11'b00000000110
`define CBZ 8'b00000111
`define B 6'b001000
```

Points to be noted (5–stage pipeline)

- Instruction memory available in ‘imemory.v’.
- The ‘imemory.v’ uses file operation (“imem_test0.txt”)and the instructions are accessed from a file.
- Data memory available in ‘dmemory.v’. .
- The ‘dmemory.v’ uses file operation (“dmem_test0.txt”) and the instructions are accessed from a file.

(Note that memory is clocked and hence we have a clock cycle delay for getting the output from memory.)

- Other files available from lab 3 are Alu.v, Control.v, Regfile.v, PC.v and the pipeline registers along with the 5 stage datapath file which combines all the rest.
- The ‘IF_IDstage.v’ is the pipeline register between IF and ID for [passing the PC value alone, ‘ID_EXEstage.v’ is the pipeline register between decode and execute section , “EXE_MEM stage.v” is the pipeline register between execute and memory section and “MEM_WB stage.v” is the pipeline register between meory and write back.

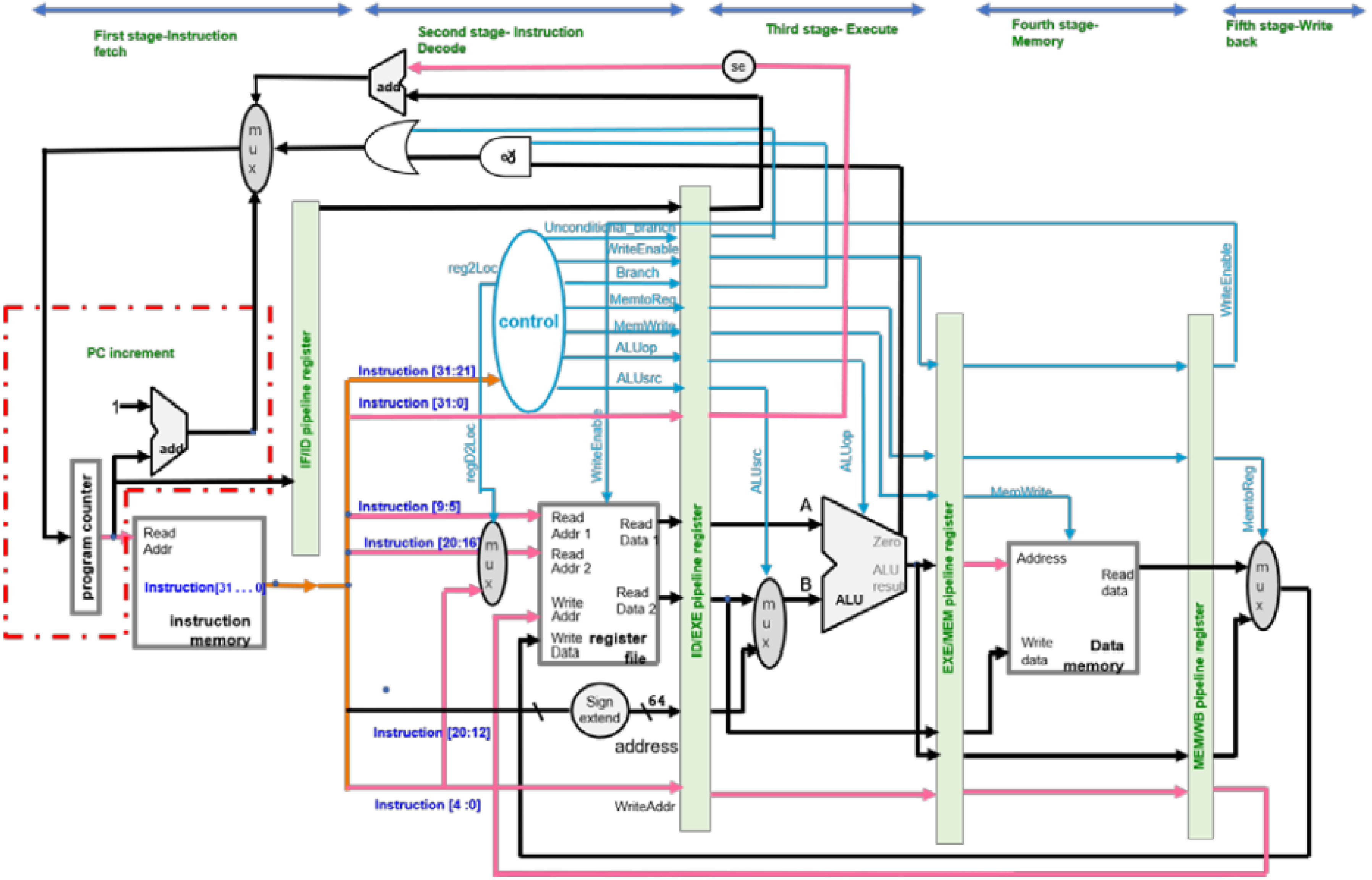

Fig. 1 Five-stage Pipelined CPU implementation diagram for R, D, CB and B type instructions

- Please note that, we are adding both “CBZ” and “B” instructions to the data path along with the previous set of instructions covered in lab4 based on the five-stage pipelined CPU. As both the instructions change the ordering of instructions by changing the content of Program Counter (PC), we need to be careful about the control hazards introduced by these instructions

- Need to be mindful about data and control hazards

- Data hazards: Insert two NOPs to remove RAW dependencies between two consecutive instructions.

- Insert two NOPs after every CBZ instruction to remove the control hazard

- Insert two NOPs after every B instruction to remove the control hazard