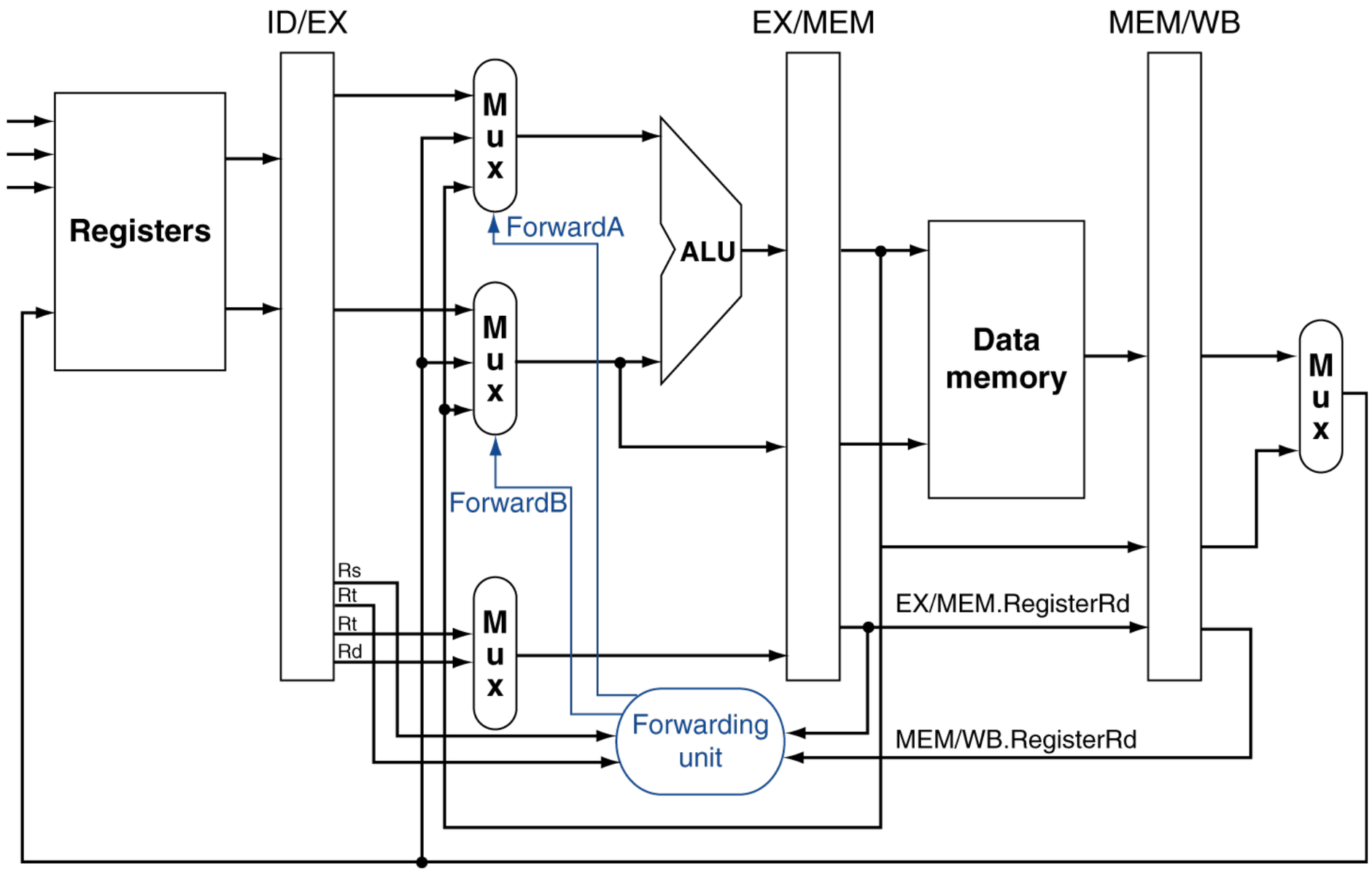
แนวทางการสร้าง Forwarding Network เพื่อจัดการกับ Data Hazard

204324 ปฏิบัติการระบบคอมพิวเตอร์
ภาคต้น 2562

Forwarding Paths



b. With forwarding

Forwarding Conditions

- Forward จาก MEM มาที่ EX
  - if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 10
  - if (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 10
- Forward จาก WB มาที่ EX
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 01

Double Data Hazard

- พิจารณาชุดคำสั่งต่อไปนี้

  add $1,$1,$2
  add $1,$1,$3
  add $1,$1,$4

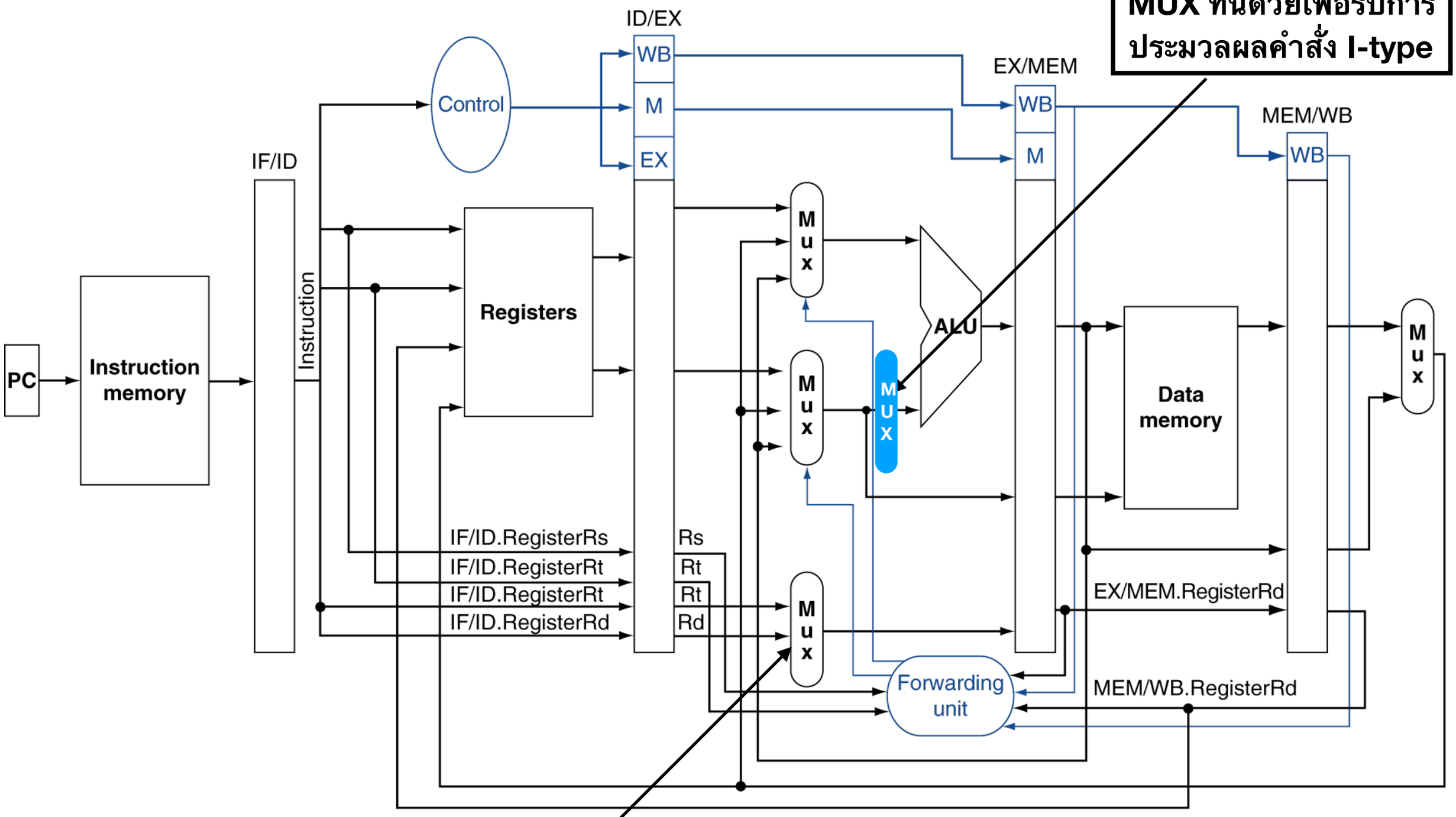
- ปัญหาถ้าใช้ forwarding logic ที่ได้กล่าวมา คือ if ของ forward logic จาก WB มาที่ EX มาทีหลัง จะ overwrite สัญญาณ forwarding ของ if ที่มาก่อน

แก้ไข Forwarding Logic ใหม่

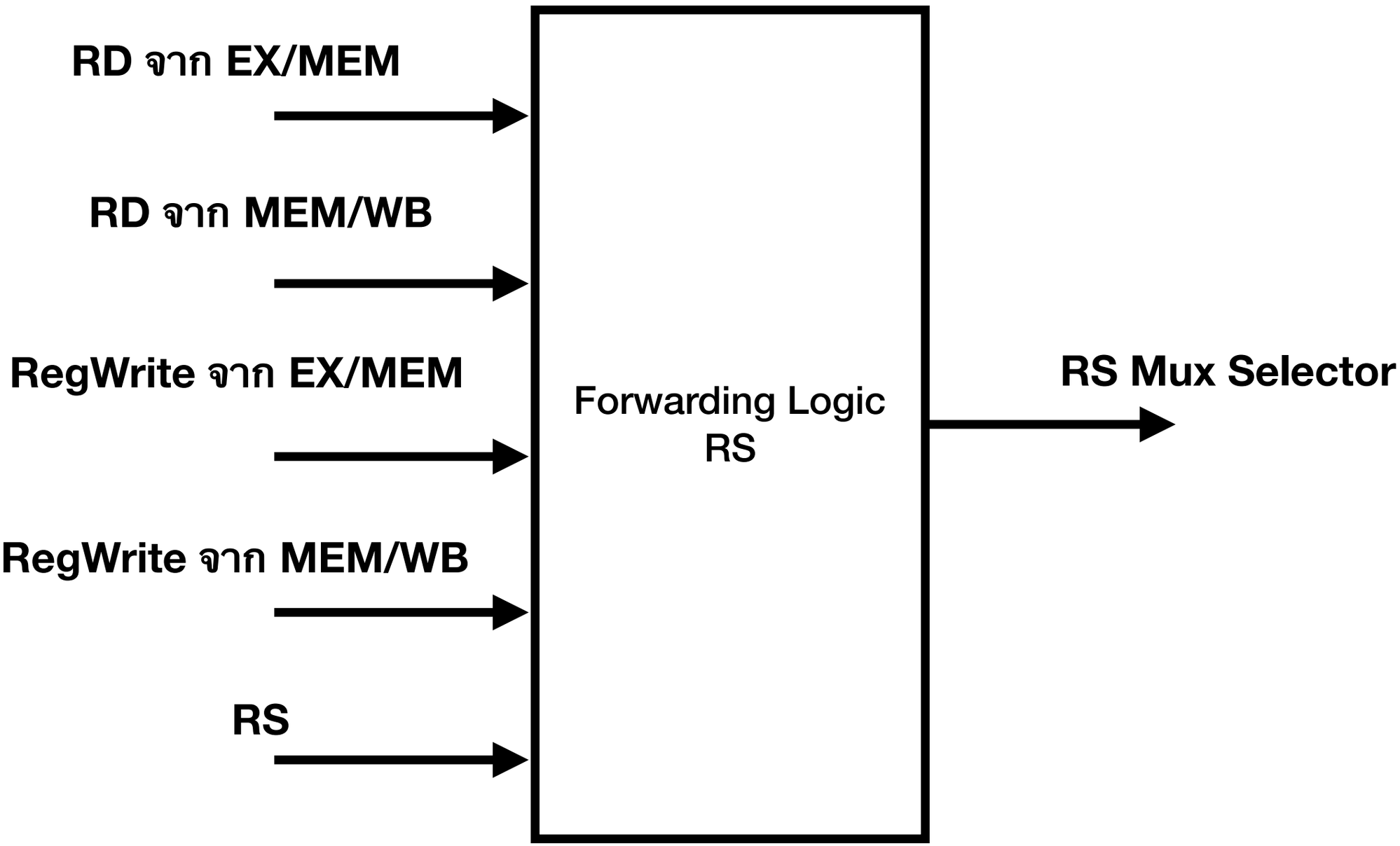
- Forward จาก MEM ก็ต่อเมื่อ ไม่ได้ forward จาก EX
- MEM hazard
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
    and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
    ForwardA = 01
  - if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
    and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0)
    and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
    and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
    ForwardB = 01

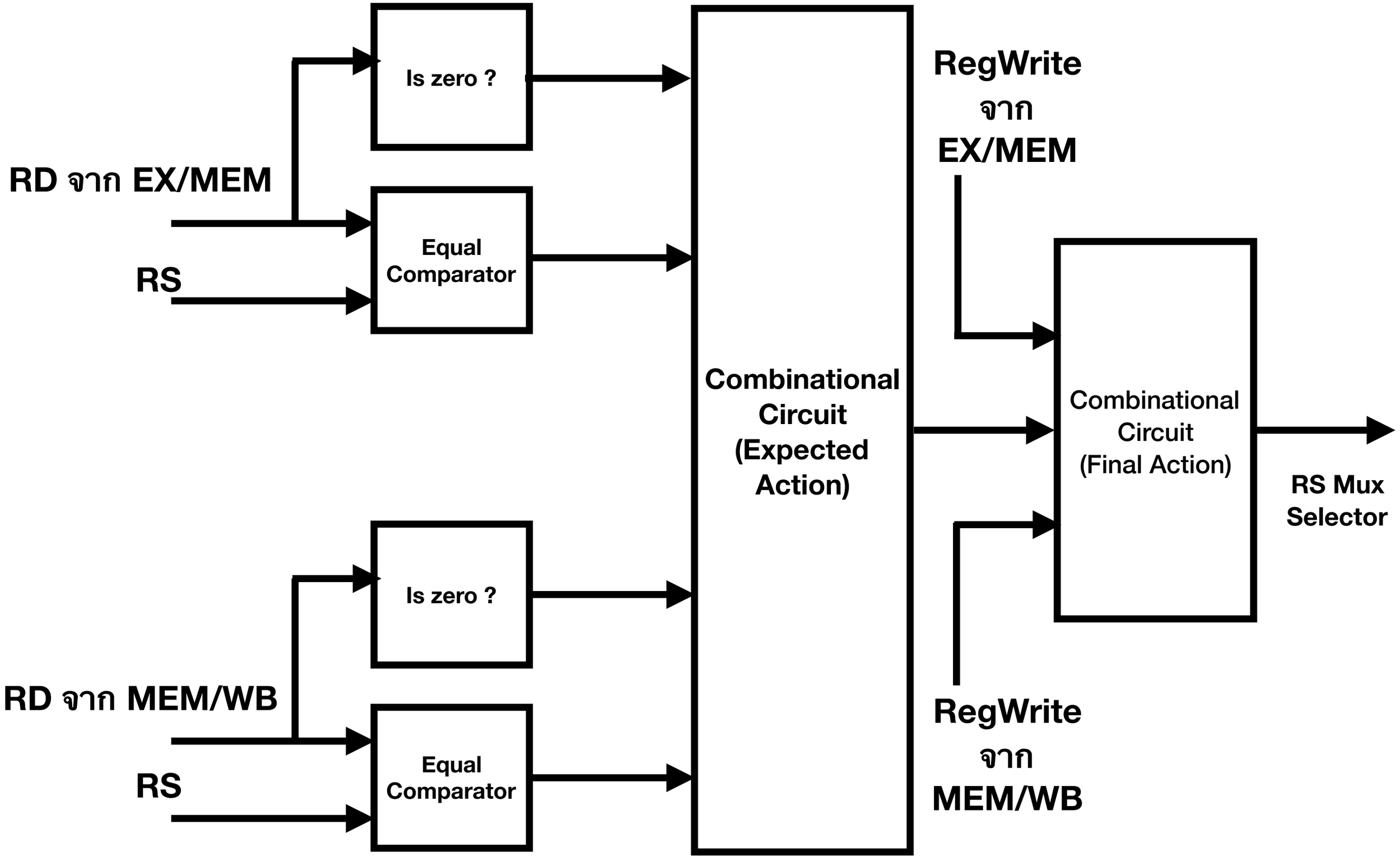
Datapath ที่รวมการทำ Forwarding

อย่าลืมใส่ ALUSrc MUX ที่นี่ด้วยเพื่อรับการประมวลผลคำสั่ง I-type



จะต้องมีสัญญาณควบคุมที่มาเลือก destination register ที่ถูกต้องที่ Mux นี้ด้วย


RD จาก EX/MEM
RD จาก MEM/WB
RegWrite จาก EX/MEM
RegWrite จาก MEM/WB
RS
Forwarding Logic
RS
RS Mux Selector



ภายใน Forwarding Logic RS

MEM/WB equal comp	EX/MEM equal comp	MEM/WB is zero ?	EX/MEM is zero ?	Expected Action
0	0	0	0	do not forward
0	0	0	1	do not forward
0	0	1	0	do not forward
0	0	1	1	do not forward
0	1	0	0	forward from EX/MEM
0	1	0	1	do not forward
0	1	1	0	forward from EX/MEM
0	1	1	1	do not forward
1	0	0	0	forward from MEM/WB
1	0	0	1	forward from MEM/WB
1	0	1	0	do not forward
1	0	1	1	do not forward
1	1	0	0	forward from EX/MEM
1	1	0	1	do not forward (กรณีนี้เป็นไปไม่ได้)
1	1	1	0	do not forward (กรณีนี้เป็นไปไม่ได้)
1	1	1	1	do not forward

Combinational Circuit Logic (Expected Action)

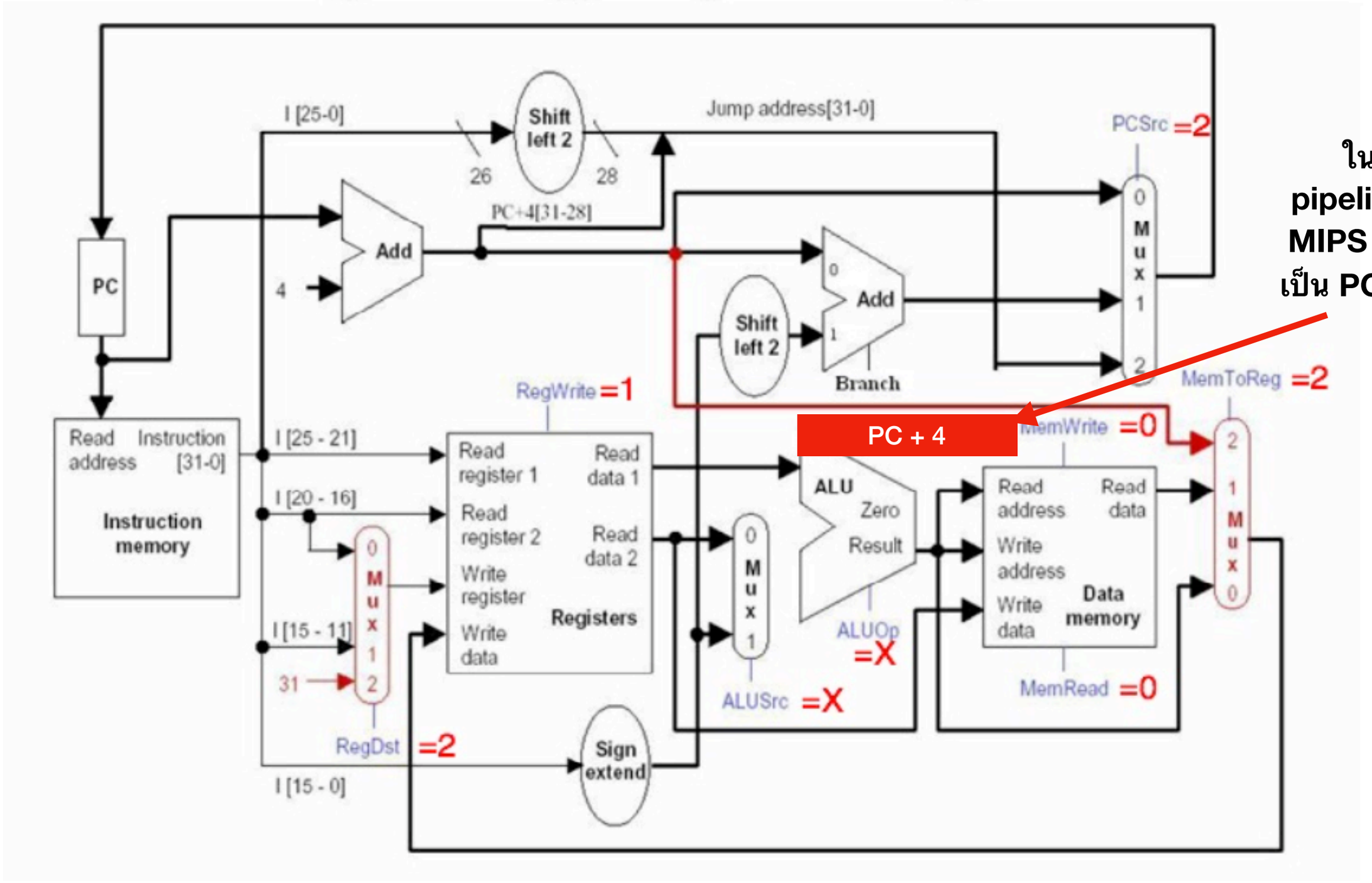
เอาพุทก่อนส่งไปตัดสินใจการ forward ในขั้นสุดท้าย

อย่าลืมสิ่งต่อไปนี้

- สร้าง forwarding logic ของ RS ให้เป็น sub-circuit
- ทำ combinational circuit สำหรับ final action ให้สมบูรณ์แบบ
- เมื่อสร้าง forwarding logic ของ RS ได้สมบูรณ์แล้ว เราจะได้ circuit ที่เป็น forwarding loic ของ RT มาโดยอัตโนมัติ
    - เนื่องจากใช้ logic เดียวกัน อินพุทชุดเดียวกัน ต่างกันที่ปลายทางของสัญญาณเอาพุทที่นำไปควบคุม multiplexer เท่านั้น

การจัดการกับคำสั่ง jal ใน pipelined MIPS

jal - single-cycle datapath



ใน pipelined MIPS ต้องเป็น PC + 8

เพิ่มขยาย Mux สำหรับ RegDst และ MemToReg เหมือนกับใน single-cycle MIPS ตามภาพด้านบน แต่ค่าอินพุทที่เข้ามาที่ MemToReg Mux แทนที่จะเป็น PC + 4 ต้องเป็น PC + 8

อย่าลืมสิ่งต่อไปนี้สำหรับ jal

- ค่า PC + 8 จะต้องถูกคำนวณที่ IF (Fetch) และถูกส่งผ่านไปยัง pipeline register ตีคู่ไปกับตัวคำสั่ง jal ตั้งแต่ IF/ID ID/EXE EXE/MEM และ MEM/WB
- ค่า PC + 8 จะต้องถูกเขียนลง register file ณ ตำแหน่งที่ jal อยู่ที่ WB
- หลังจากเพิ่มขยาย RegDst Mux แล้ว ค่า PC + 8 ที่จะเขียนลง register $ra ก็สามารถนำมา forward ผ่าน forwarding network ที่ไรสร้างขึ้นมาได้ปกติ
    - คำสั่งที่จะใช้ผลลัพธ์จาก jal ที่พบอยู่ประจำคือคำสั่ง jr $ra