University of Engineering and Technology (UET), Peshawar, Pakistan

Lecture 12

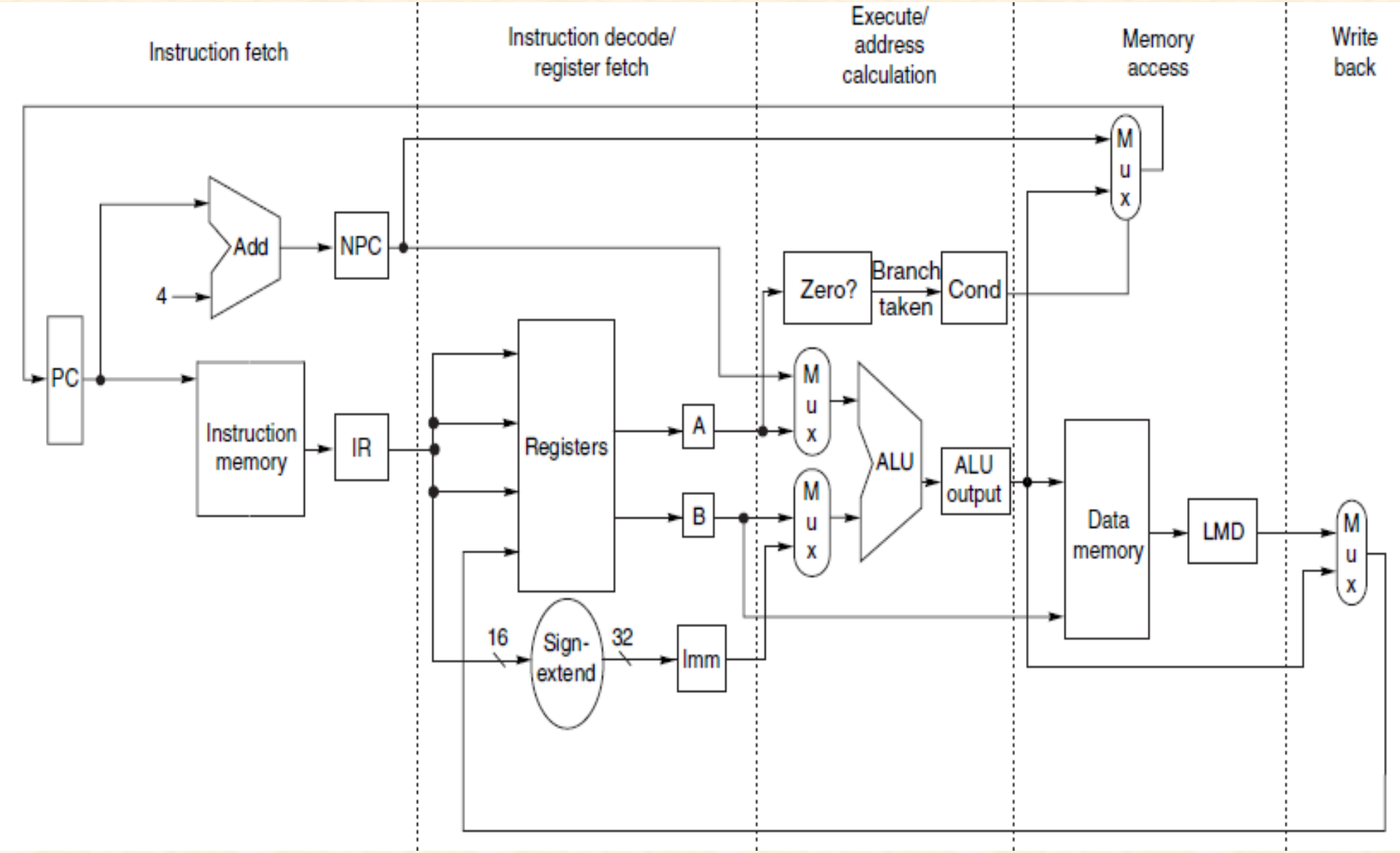
CSE-304: Computer Organization and Architecture

BY:

Dr. Muhammad Athar Javed Sethi

Slides are taken from student resources of William Stallings website

Simple Implementation of MIPS



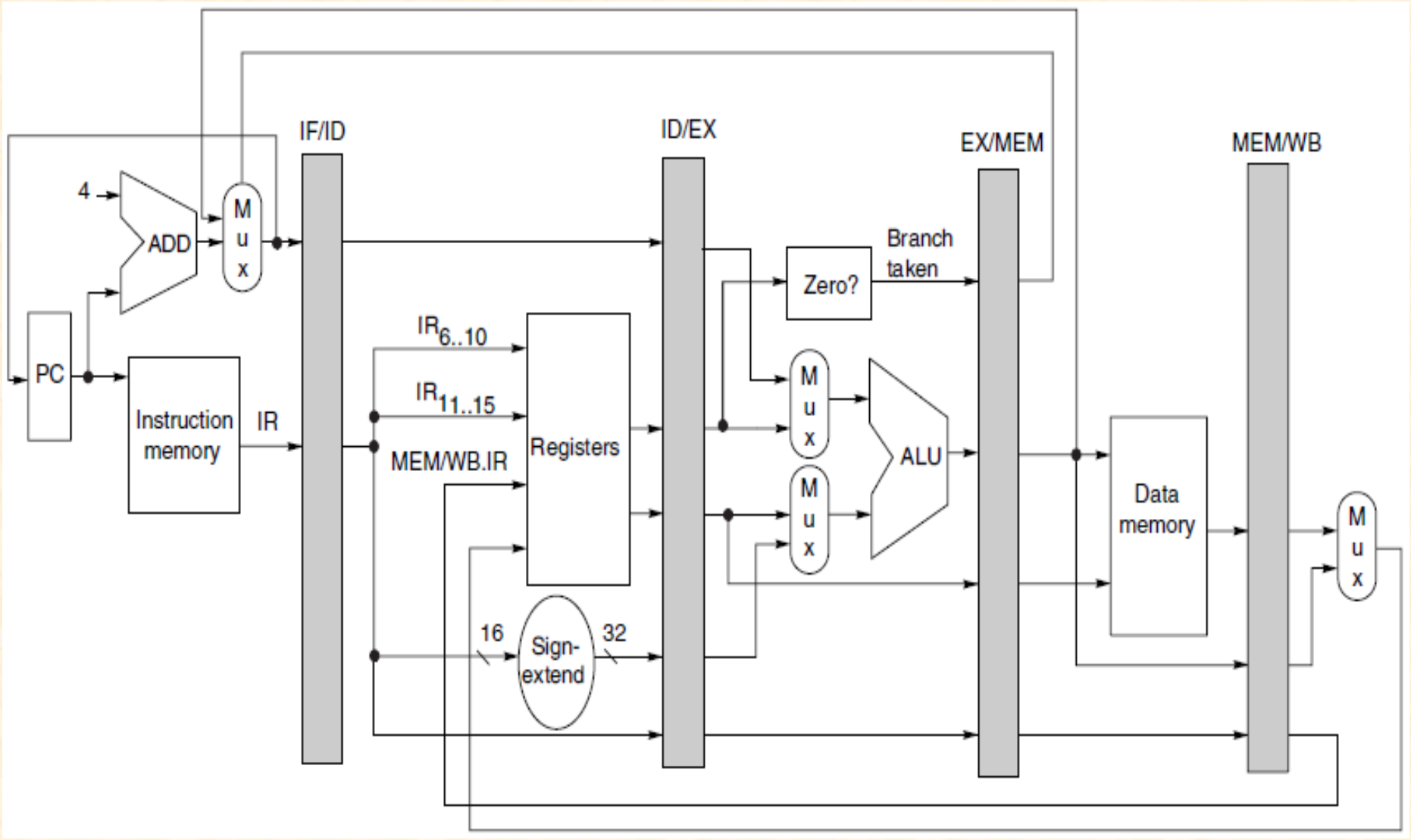
Simple Implementation of MIPS

- **Instruction Fetch cycle (IF)**
  - **IR=Mem[PC]**
  - **NPC=PC+4**
- **Instruction Decode cycle (ID)**
  - **A=Reg[rs]**
  - **B=Reg[rt]**
  - **Imm← sign extended immediate field or IR**
- **Execution cycle (EX)**
  - **Memory Reference**
    - **ALUOutput=A+Imm**
  - **Register Register ALU instruction**
    - **ALUOutput= A func B**
  - **Register Immediate ALU instruction**
    - **ALUOutput= A op Imm**
  - **Branch**
    - **ALUOutput=NPC+Imm**
    - **Cond← (A==0)**

Simple Implementation of MIPS

- **Memory Access cycle (MEM)**
  - **Memory Reference**
    - **LMD=Mem[ALUOutput] or Mem[ALUOutput]=B**
  - **Branch**
    - **If (cond) PC = ALUOutput**
- **Write Back cycle (WB)**
  - **Register Register ALU Instruction**
    - **Regs[rd]=ALUOutput**
  - **Register Immediate ALU Instruction**
    - **Regs[rt]=ALUOutput**
  - **Load Instruction**
    - **Regs[rt]=LMD**

A Basic Pipeline for MIPS



Events of Every Pipe Stage Of the MIPS Pipeline

Stage	Any instruction		
IF	IF/ID.IR ← Mem[PC]; IF/ID.NPC,PC ← (if ((EX/MEM.opcode == branch) & EX/MEM.cond){EX/MEM. ALUOutput} else {PC+4});		
ID	ID/EX.A ← Regs[IF/ID.IR[rs]]; ID/EX.B ← Regs[IF/ID.IR[rt]]; ID/EX.NPC ← IF/ID.NPC; ID/EX.IR ← IF/ID.IR; ID/EX.Imm ← sign-extend(IF/ID.IR[immediate field]);		
	ALU instruction	**Load or store instruction**	**Branch instruction**
EX	EX/MEM.IR ← ID/EX.IR; EX/MEM.ALUOutput ← ID/EX.A func ID/EX.B; or EX/MEM.ALUOutput ← ID/EX.A *op* ID/EX.Imm;	EX/MEM.IR to ID/EX.IR EX/MEM.ALUOutput ← ID/EX.A + ID/EX.Imm; EX/MEM.B ← ID/EX.B;	EX/MEM.ALUOutput ← ID/EX.NPC + (ID/EX.Imm << 2); EX/MEM.cond ← (ID/EX.A == 0);
MEM	MEM/WB.IR ← EX/MEM.IR; MEM/WB.ALUOutput ← EX/MEM.ALUOutput;	MEM/WB.IR ← EX/MEM.IR; MEM/WB.LMD ← Mem[EX/MEM.ALUOutput]; or Mem[EX/MEM.ALUOutput] ← EX/MEM.B;	
WB	Regs[MEM/WB.IR[rd]] ← MEM/WB.ALUOutput; or Regs[MEM/WB.IR[rt]] ← MEM/WB.ALUOutput;	For load only: Regs[MEM/WB.IR[rt]] ← MEM/WB.LMD;	

Implementing Control for the MIPS pipeline

Situation	Example code sequence	Action
No dependence	LD **R1**,45(R2) DADD R5,R6,R7 DSUB R8,R6,R7 OR R9,R6,R7	No hazard possible because no dependence exists on R1 in the immediately following three instructions.
Dependence requiring stall	LD **R1**,45(R2) DADD R5,**R1**,R7 DSUB R8,R6,R7 OR R9,R6,R7	Comparators detect the use of R1 in the DADD and stall the DADD (and DSUB and OR) before the DADD begins EX.
Dependence overcome by forwarding	LD **R1**,45(R2) DADD R5,R6,R7 DSUB R8,**R1**,R7 OR R9,R6,R7	Comparators detect use of R1 in DSUB and forward result of load to ALU in time for DSUB to begin EX.
Dependence with accesses in order	LD **R1**,45(R2) DADD R5,R6,R7 DSUB R8,R6,R7 OR R9,**R1**,R7	No action required because the read of R1 by OR occurs in the second half of the ID phase, while the write of the loaded data occurred in the first half.