Unpipelined MIPS

IF | ID | EX | MEM | WB
---|----|----|-----|-----
• Fetch instruction from Mem[PC]
• PC=PC+4

In parallel:
• Decode the instruction
  • Read source registers, compute registers for possible branch
  • Sign-extend the offset field if needed, compute branch target address by adding sign-extended offset to PC+4 (branch can now be completed)
  Note: we have two options for branch target calculation: in ID or in EX stage!

If instruction is:
• Memory reference: add base register and offset to form memory address
  • Store: store data from register to memory
• ALU: perform the operation; for immediate sign-extend the second operand.

ID stage

IF | ID | EX | MEM | WB
---|----|----|-----|-----
• Fetch instruction from Mem[PC]
• PC=PC+4

In parallel:
• Decode the instruction
  • Read source registers, compute registers for possible branch
  • Sign-extend the offset field if needed, compute branch target address by adding sign-extended offset to PC+4

EX stage

IF | ID | EX | MEM | WB
---|----|----|-----|-----
• Fetch instruction from Mem[PC]
• PC=PC+4

In parallel:
• For memory reference: add base register and offset to form memory address
  • ALU: perform the operation; for immediate sign-extend the second operand. Compute branch target by adding sign-extended offset to PC+4

MEM stage

IF | ID | EX | MEM | WB
---|----|----|-----|-----
• Fetch instruction from Mem[PC]
• PC=PC+4

In parallel:
• For memory reference: add base register and offset to form memory address
  • Store: store data from register to memory

WB stage

IF | ID | EX | MEM | WB
---|----|----|-----|-----
• Fetch instruction from Mem[PC]
• PC=PC+4

In parallel:
• Write data into register file either for LOAD or for ALU instruction
### Pipeline Hazards

- Conflicts that prevent the instruction A from executing within its pipeline stage during a given clock cycle:
  1. **Structural hazards** – instruction A requires a resource occupied by some previous instruction
  2. **Data hazards** – a source operand of instruction A is the output of some previous instruction and is not ready
  3. **Control hazards** – pipelining branches and other instructions that change PC conflicts with execution of later instructions (since we don’t know where to read these instructions from)

### Pipelined MIPS - Resources

- Separate instruction and data memory
- Memory must deliver 5 times the bandwidth
- Register file must support two reads and one write
  - We will perform them in half-cycles, first write and then read
- We need adder to increment PC and to perform branch target calculation, if we are doing this in ID stage
  - If we do branch target calculation in EX stage, we just need a simple adder in ID stage to increment PC

### Example

Consider the unpipelined processor and assume that it has a 1 ns clock cycle and that it uses 4 cycles for ALU operations and branches and 5 cycles for memory operations. Assume that the relative frequencies of these operations are 40%, 20% and 40%, respectively. Suppose that due to clock skew and setup, pipelining the processor adds 0.2 ns of overhead to the clock. Ignoring any latency impact:

- How many stages has this pipeline?
- How much speedup will we gain from a pipeline?
Performance Degradation Due to Hazards

\[
\text{Speedup} = \frac{T_{\text{per instruction, unpipelined}}}{T_{\text{per instruction, pipelined}}} = \frac{\text{CPI}_{\text{unpipelined}} + \text{CC}_{\text{unpipelined}}}{\text{CPI}_{\text{pipelined}} + \text{CC}_{\text{pipelined}}}
\]

\[n \quad \text{Stall/Cycles}_{\text{per instruction}}\]

Structural Hazards

- Resource conflicts:
  - E.g. using same memory for data and instructions will create a hazard whenever we have STORE or LOAD. Can you think of another structural hazard?
  - Instructions are stalled until resource becomes available
  - Conflicts can be avoided through resource duplication
  - If structural hazards are not that frequent it may be cheaper to allow them

Example

Suppose that data references constitute 40% of the mix, and that the ideal CPI of the pipelined processor (if we didn’t have the hazard) is 1. Assume that the processor with the structural hazard (data and instructions are stored in the same memory) has a clock rate that is 1.05 times higher than the clock rate of the processor where we avoid the hazard through resource duplication.

Is the pipeline with or without structural hazard faster, and by how much?

Data Hazards

- Source operand of the current instruction is the output of a previous instruction and is not ready
  
  DADD R1, R2, R3
  DSUB R4, R1, R5
  AND R6, R1, R7
  OR R8, R1, R9
  XOR R10, R1, R11
Forwarding can be generalized as passing the result from one functional unit to another unit that needs it. This is done through pipeline registers, not directly.
Data Hazards Requiring Stalls

- Load followed by ALU instructions that use the result
  - LD R1, 0(R2)
  - DSUB R4, R1, R5
  - AND R6, R1, R7
  - OR R8, R1, R9

A piece of hardware called pipeline interlock is added to detect a hazard and stall the pipeline.