Exercise walk-through

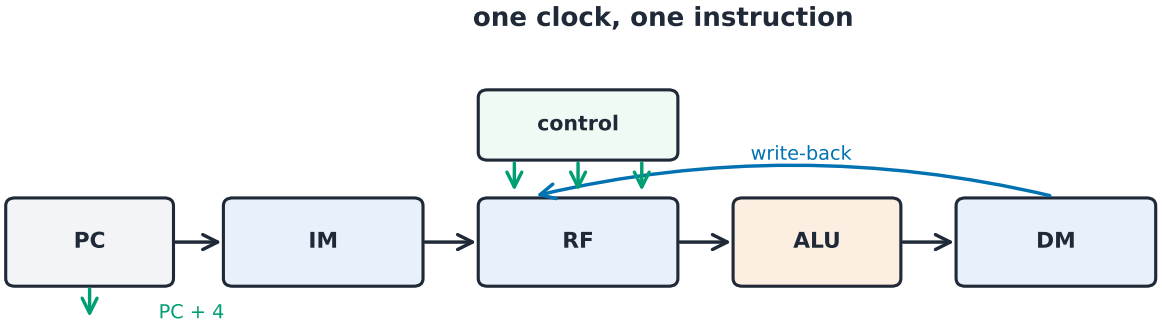
6.2
Datapath and control

University Computer Science

Datapath and control: six bits per instruction

1. **Path** PC to IM to RF to **ALU** to DM to write-back: lw walks all of it - 200 + 100 + 200 + 250 + 150 = 900 ps.
2. **Signals** lw R4, 8(R1): RegWrite 1, MemRead 1, MemToReg 1, ALUSrc 1, Branch 0 - six bits tell the story.
3. **Store** sw flips the port: RegWrite 0, MemWrite 1, and MemToReg becomes a do-not-care.

Single-cycle datapath



control bits: RegWrite · MemRead · MemWrite · MemToReg · ALUSrc · Branch

for lw R4, 8(R1): RegWrite 1 · MemRead 1 · MemToReg 1 · ALUSrc 1 · Branch 0

cycle time = the slowest path (lw: IM + RF + ALU + DM + wiring = 900 ps)

> One clock, one instruction: fetch at PC, read registers, compute at the **ALU**, touch data memory if asked, write back, and choose the next PC.

Exercises 2.1·2.3

2.1 Name the five blocks an `lw` passes through, in order. [3]

2.2 For `sw R5, 12(R1)`, give RegWrite, MemWrite and MemRead. [3]

2.3 State what a don’t-care control bit means. [3]

Exercises 2.1·2.3

2.1 Name the five blocks an `lw` passes through, in order. [3]

2.1

- PC, IM, RF, ALU, DM (then write-back to RF) ✓.

2.2 For `sw R5, 12(R1)`, give RegWrite, MemWrite and MemRead. [3]

2.2

- RegWrite 0, MemWrite 1, MemRead 0 ✓.

2.3 State what a don’t-care control bit means. [3]

2.3

- Either value leaves the instruction’s visible behaviour unchanged ✓.

Exercise 3.1

3.1 Parameters: IM 200 ps, RF read 100 ps, ALU 200 ps, DM 250 ps, RF write 100 ps, wiring 150 ps total.

Compute the `lw` path time and state the clock cycle. Compute the `add` path (IM + RF + ALU + wiring, no DM) and explain why it does not set the cycle. Compute the total time for 100 instructions (mixed), and the new cycle time if DM improves to 150 ps, with the new total. [8]

Exercise 3.1

3.1 Parameters: IM 200 ps, RF read 100 ps, ALU 200 ps, DM 250 ps, RF write 100 ps, wiring 150 ps total.

Compute the `lw` path time and state the clock cycle. Compute the `add` path (IM + RF + ALU + wiring, no DM) and explain why it does not set the cycle. Compute the total time for 100 instructions (mixed), and the new cycle time if DM improves to 150 ps, with the new total. [8]

3.1

- `lw`: $200 + 100 + 200 + 250 + 150 = 900$ ps; cycle $= 900$ ps ✓.
- `add`: $200 + 100 + 200 + 150 = 650$ ps – under the cycle, so it waits; only the longest path sets the clock ✓.
- 100 instructions: $100 \times 900 = 90\,000$ ps ✓.
- DM 150 ps: `lw` path $= 800$ ps – now the new cycle; total $80\,000$ ps ✓.

Exercise 3.2

3.2 For each instruction, give all six control bits and one line of justification: `lw R2, 0(R1)`; `sw R3, 4(R1)`; `beq R1, R0, next`; `add R4, R1, R2`.

State which bits a `beq` sets differently from the handout's `lw` row and what the datapath does when the branch is not taken. [7]

Exercise 3.2

3.2 For each instruction, give all six control bits and one line of justification: `lw R2, 0(R1)`; `sw R3, 4(R1)`; `beq R1, R0, next`; `add R4, R1, R2`.

State which bits a `beq` sets differently from the handout's `lw` row and what the datapath does when the branch is not taken. [7]

3.2

- `lw R2, 0(R1)`: RegWrite 1, MemRead 1, MemWrite 0, MemToReg 1, ALUSrc 1, Branch 0 – data memory feeds write-back ✓.
- `sw R3, 4(R1)`: RegWrite 0, MemWrite 1, MemRead 0, MemToReg x, ALUSrc 1, Branch 0 ✓.
- `beq R1, R0, next`: RegWrite 0, MemRead 0, MemWrite 0, ALUSrc 0, Branch 1 – the ALU compares, memory untouched ✓.
- `add R4, R1, R2`: RegWrite 1, MemRead 0, MemWrite 0, MemToReg 0 (ALU result), ALUSrc 0, Branch 0 ✓.
- Versus `lw`: Branch 0 $\rightarrow$ 1, ALUSrc 1 $\rightarrow$ 0, MemToReg x; not taken means the PC simply takes PC + 4 ✓.