An Overview of Pipelining

DR. RAJIB RANJAN MAITI

CSIS, BITS-PILANI, HYDERABAD

Pipelining

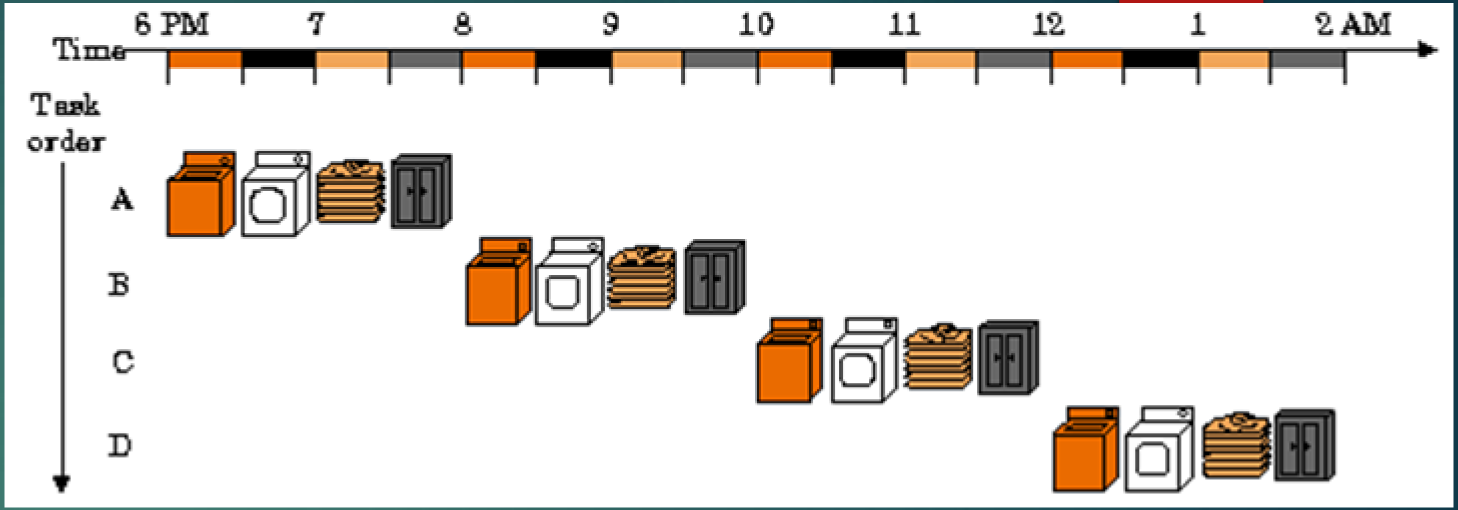
- Multiple instructions are overlapped in execution

- Anyone who has done a lot of laundry has intuitively used pipelining

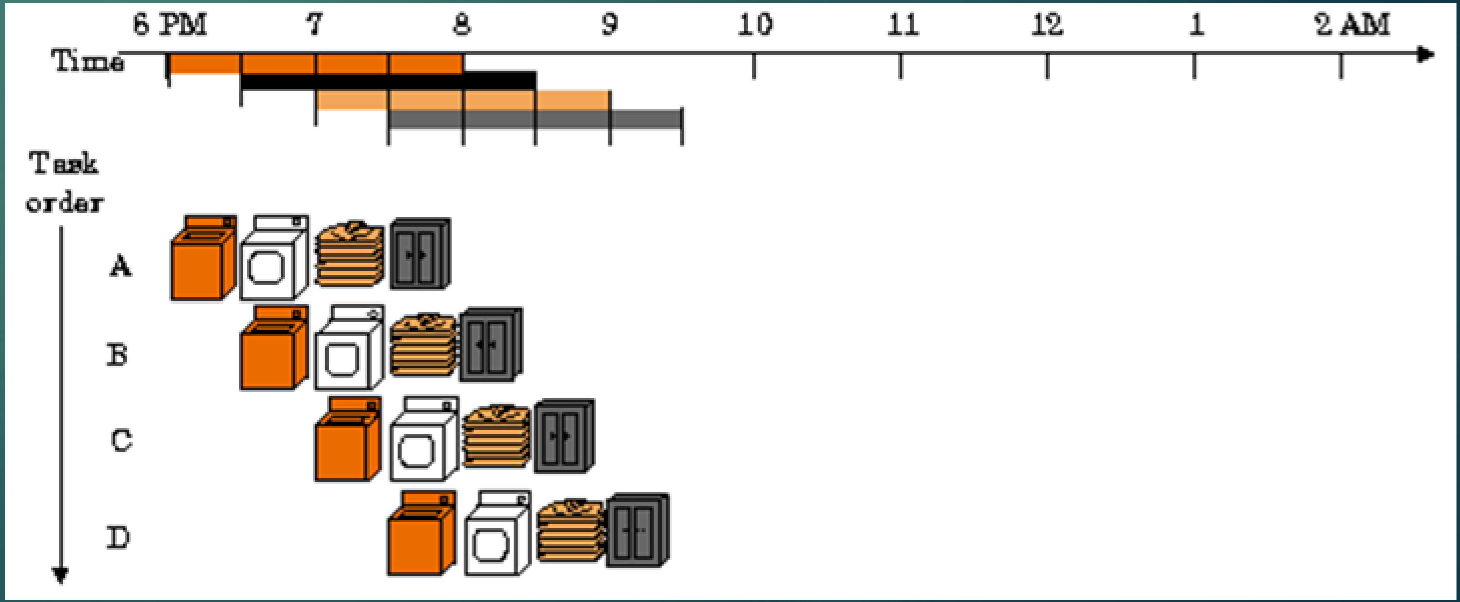
- The ***nonpipelined*** approach to laundry would be as follows:
  - 1. Place one dirty load of clothes in the **washer**
  - 2. When the washer is finished, place the wet load in the **dryer**
  - 3. When the dryer is finished, place the dry load on a **table** and fold
  - 4. When folding is finished, **put** the clothes **away**
- When all these steps are done,
  - start over with the next dirty load

The laundry analogy for pipelining.
Ann, Brian, Cathy, and Don each have dirty clothes to be washed, dried, folded, and put away.

The washer, dryer, "folder," and "storer" each take 30 minutes for their task.



Pipelined approach: use two-dimensional line of the four stages, but we really have just one of each resource.



Sequential laundry takes 8 hours for 4 loads of wash, while pipelined laundry takes just 3.5 hours.

Pipeline possibility in MIPS

- MIPS instructions classically take five steps:
  - 1. Fetch instruction from memory.
  - 2. Read registers while decoding the instruction.
    - The regular format of MIPS instructions allows reading and decoding to occur simultaneously.
  - 3. Execute the operation or calculate an address.
  - 4. Access an operand in data memory.
  - 5. Write the result into a register.

Pipelining speeds up instruction execution

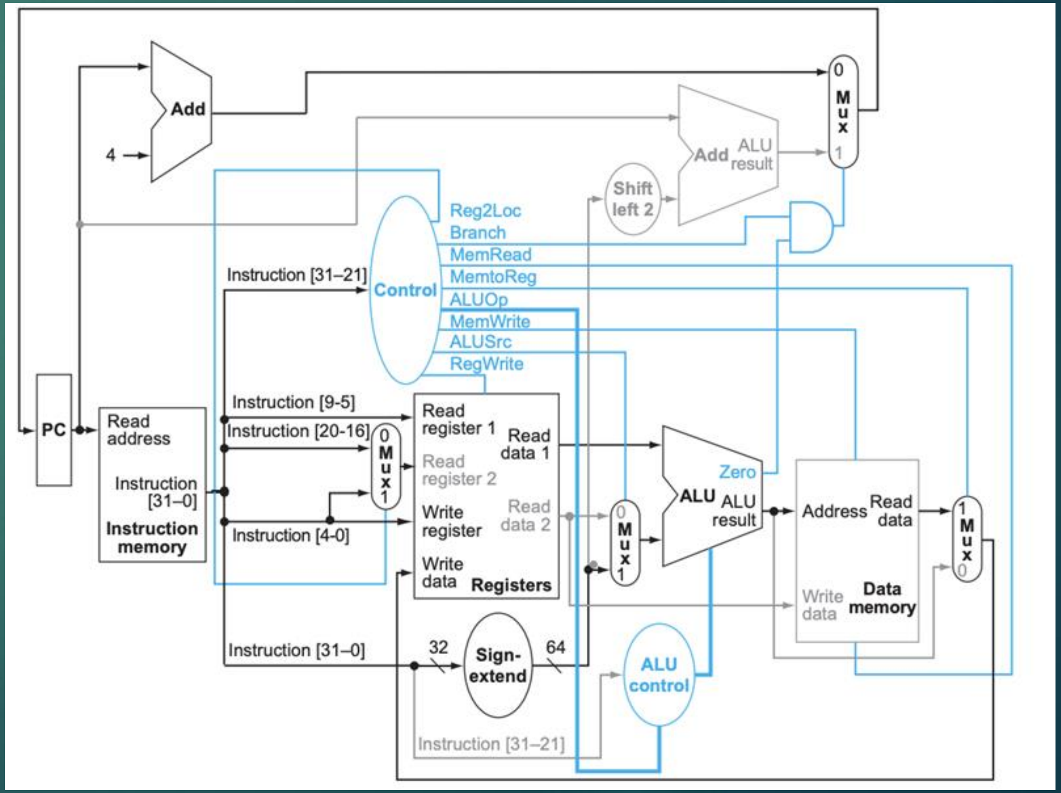
- Let us limit our attention to eight instructions:
  - load word (lw),
  - store word (sw),
  - add (add),
  - subtract (sub),
  - AND (and),
  - OR (or),
  - set less than (slt), and
  - branch on equal (beq)

- The operation times for the major functional units are
  - 200 ps for memory access,
  - 200 ps for ALU operation, and
  - 100 ps for register file read or write

- In the single-cycle model, every instruction takes exactly one clock cycle, so the clock cycle must be stretched to accommodate the slowest instruction
  - it is **lw**
  - CCT: the time required for every instruction is 800 ps

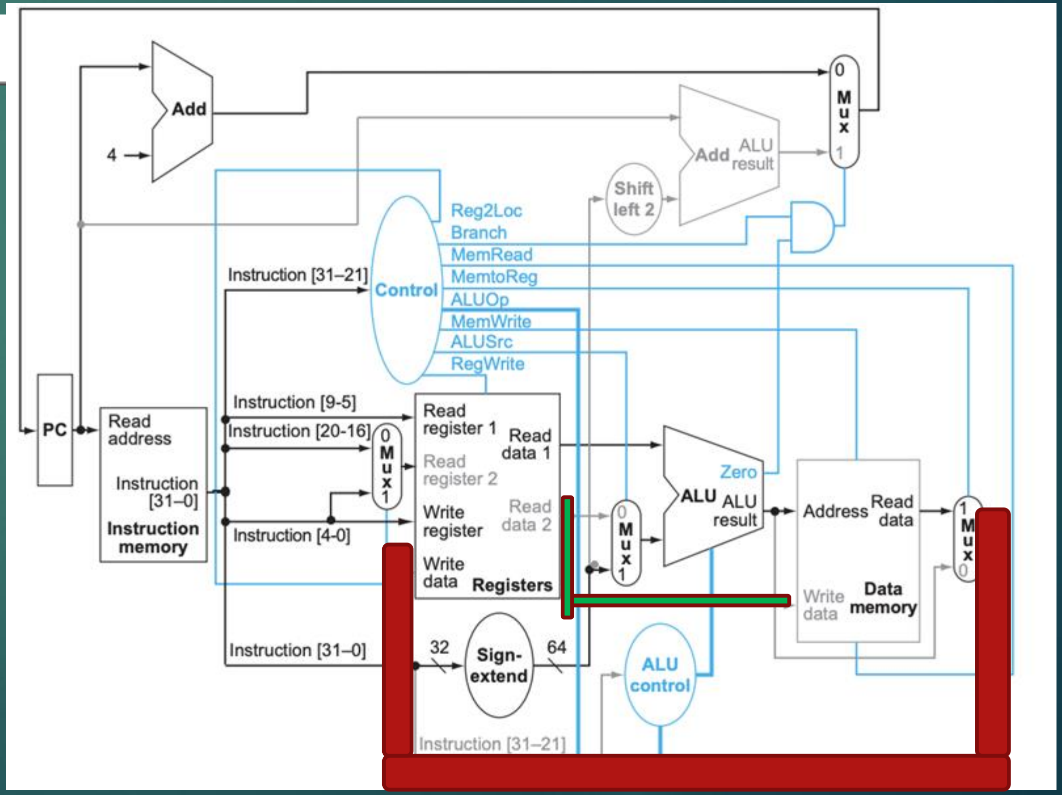
Pipelining speeds up instruction execution

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (`lw`)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps



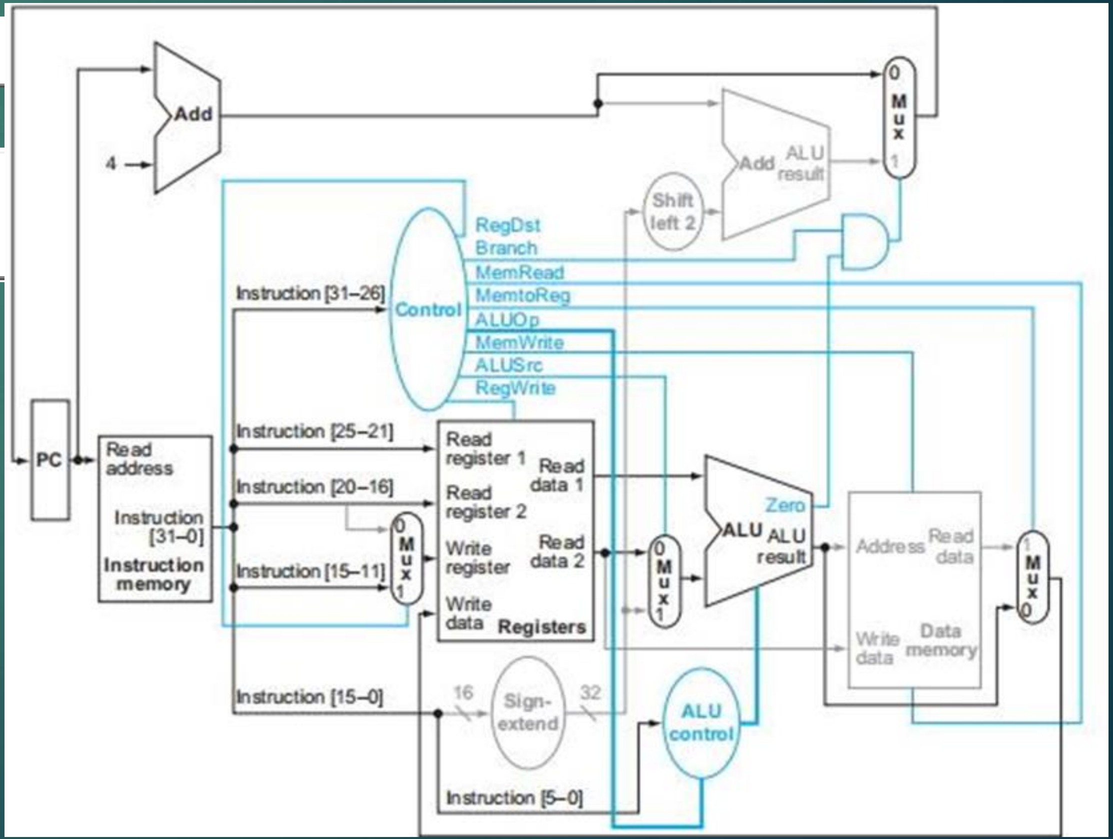
Pipelining speeds up instruction execution

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (sw)	200 ps	100				

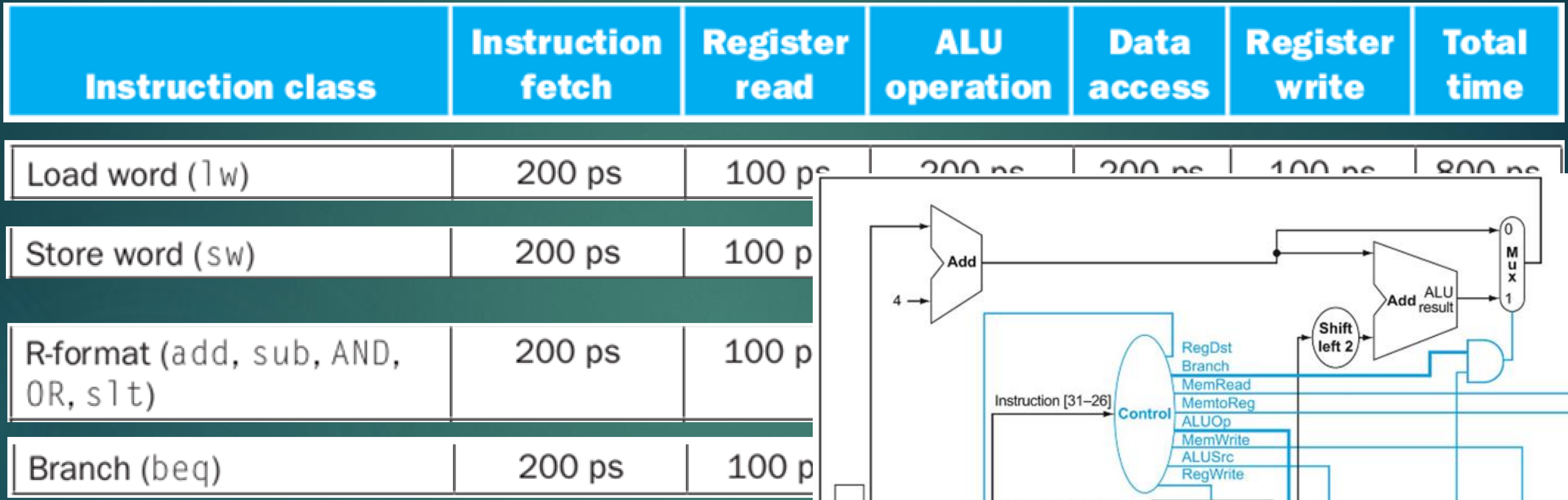


Pipelining speeds up instruction execution

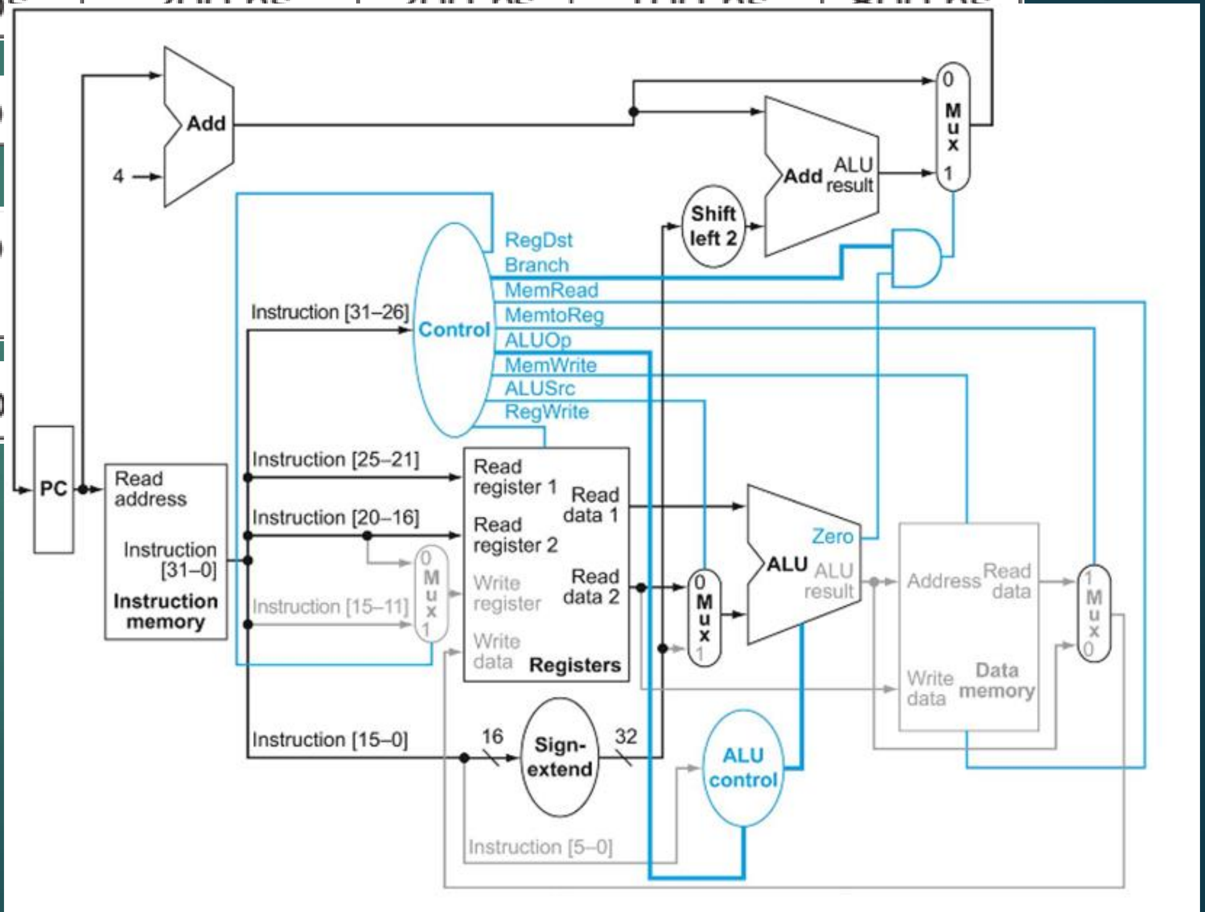
Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (sw)	200 ps	100 ps				
R-format (add, sub, AND, OR, slt)	200 ps	100 ps				



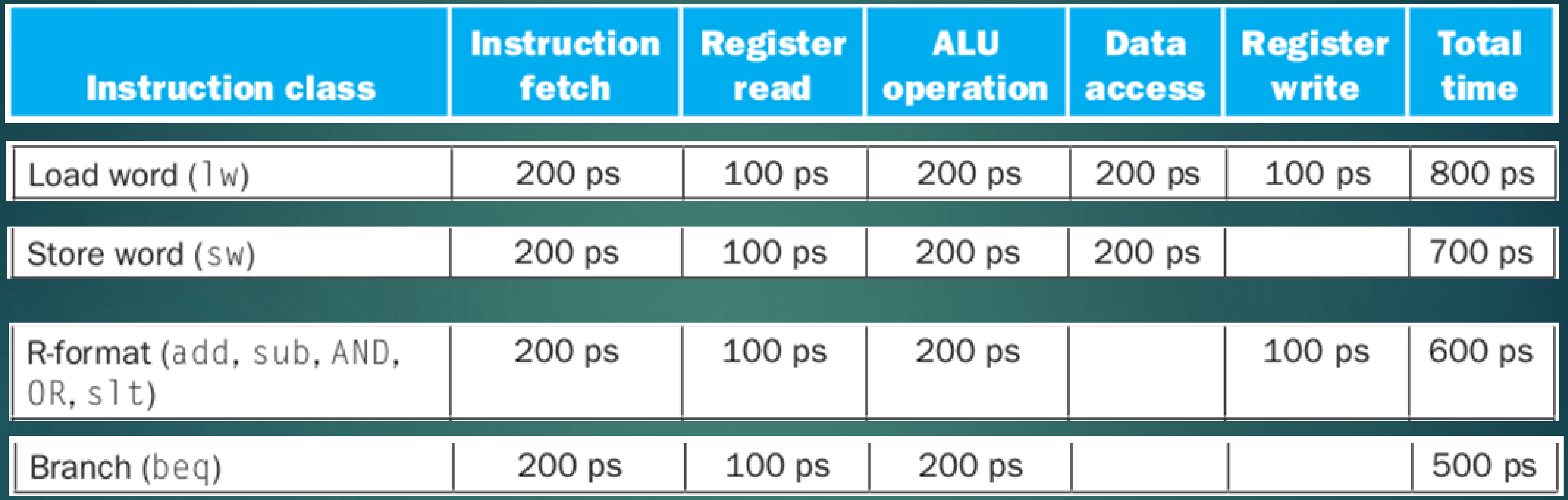
Pipelining speeds up instruction execution



Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (`lw`)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (`sw`)	200 ps	100 p				
R-format (`add`, `sub`, `AND`, `OR`, `slt`)	200 ps	100 p				
Branch (`beq`)	200 ps	100 p				



Pipelining speeds up instruction execution



Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (`lw`)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (`sw`)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (`add`, `sub`, `AND`, `OR`, `slt`)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (`beq`)	200 ps	100 ps	200 ps			500 ps

Pipelining speeds up instruction execution

- the time between the first and fourth instructions in the nonpipelined design is
  - 3×800 ps = 2400 ps

- Each of the pipeline stages take **a single clock cycle**,
  - so the clock cycle must be long enough to accommodate the slowest operation

- Clock cycle time in pipelined execution
  - Must be the worst-case clock cycle of 200 ps,
  - Even though some stages take only 100 ps

- So, the time between the first and fourth instructions is
  - 3×200 ps = 600 ps

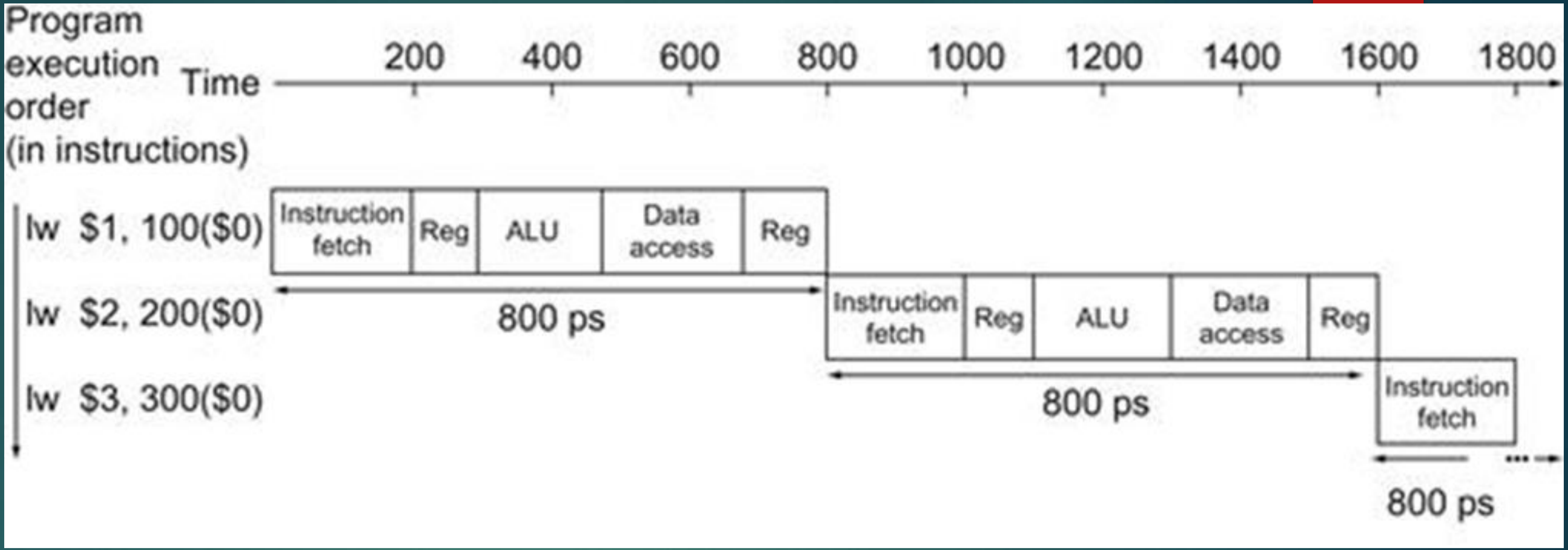
Pipelining speeds up instruction execution

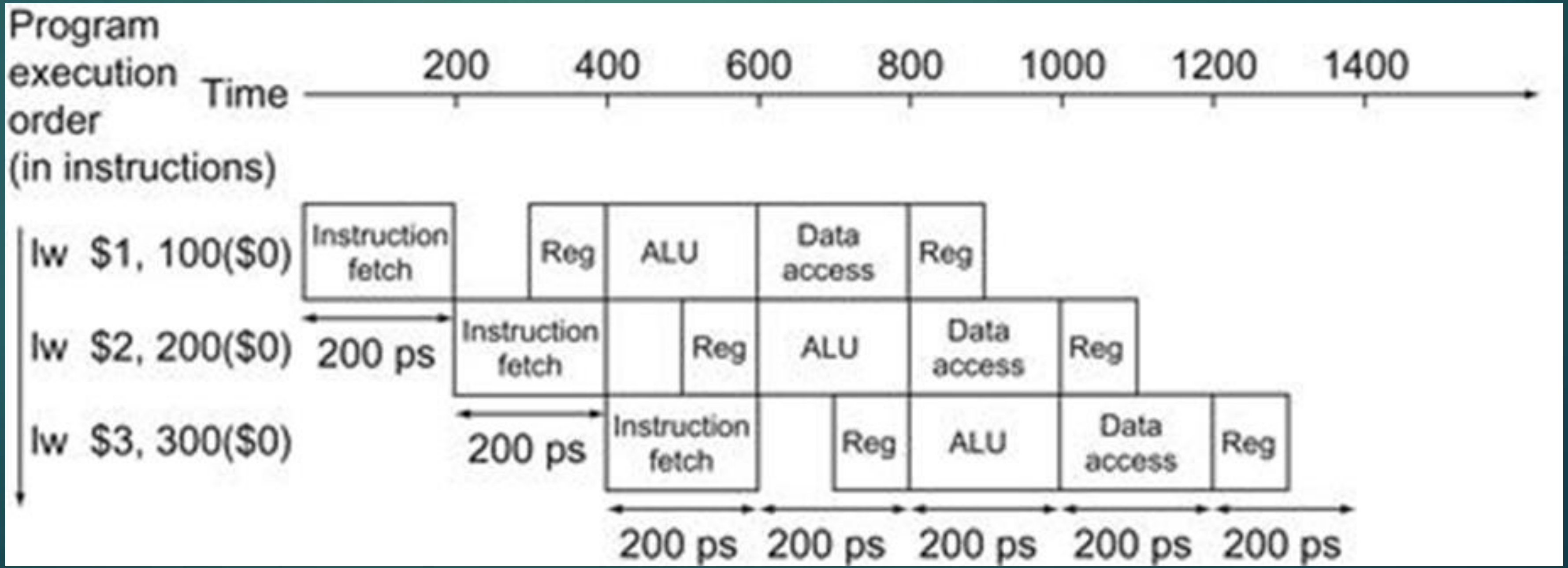
- If the stages are perfectly balanced, then the time between instructions (TBI) on the pipelined processor—assuming ideal conditions—is equal to
  - $TBI_p = \frac{TBI_{np}}{N_P}$,
  - *TBI*: Time between instructions,
  - p: pipelined,
  - np: non-pipelined,
  - N: number of stages
- The formula suggests that
  - a five-stage pipeline should offer nearly a five fold improvement
- Using our example, clock cycle time
  - np time : 800 ps
  - p time : 160 ps

Single Cycle

vs.

Pipeline





Overhead in pipelining

- Pipelining involves some overhead,
  - the **time per instruction** in the pipelined processor will **exceed the minimum possible**, and
  - **speed-up** will be less than the number of pipeline stages
- Pipelining **improves performance** by ***increasing instruction throughput***,
  - *as opposed to decreasing the execution time of **an individual instruction**,*
  - but instruction throughput is the important metric because real programs execute billions of instructions.

Thank You