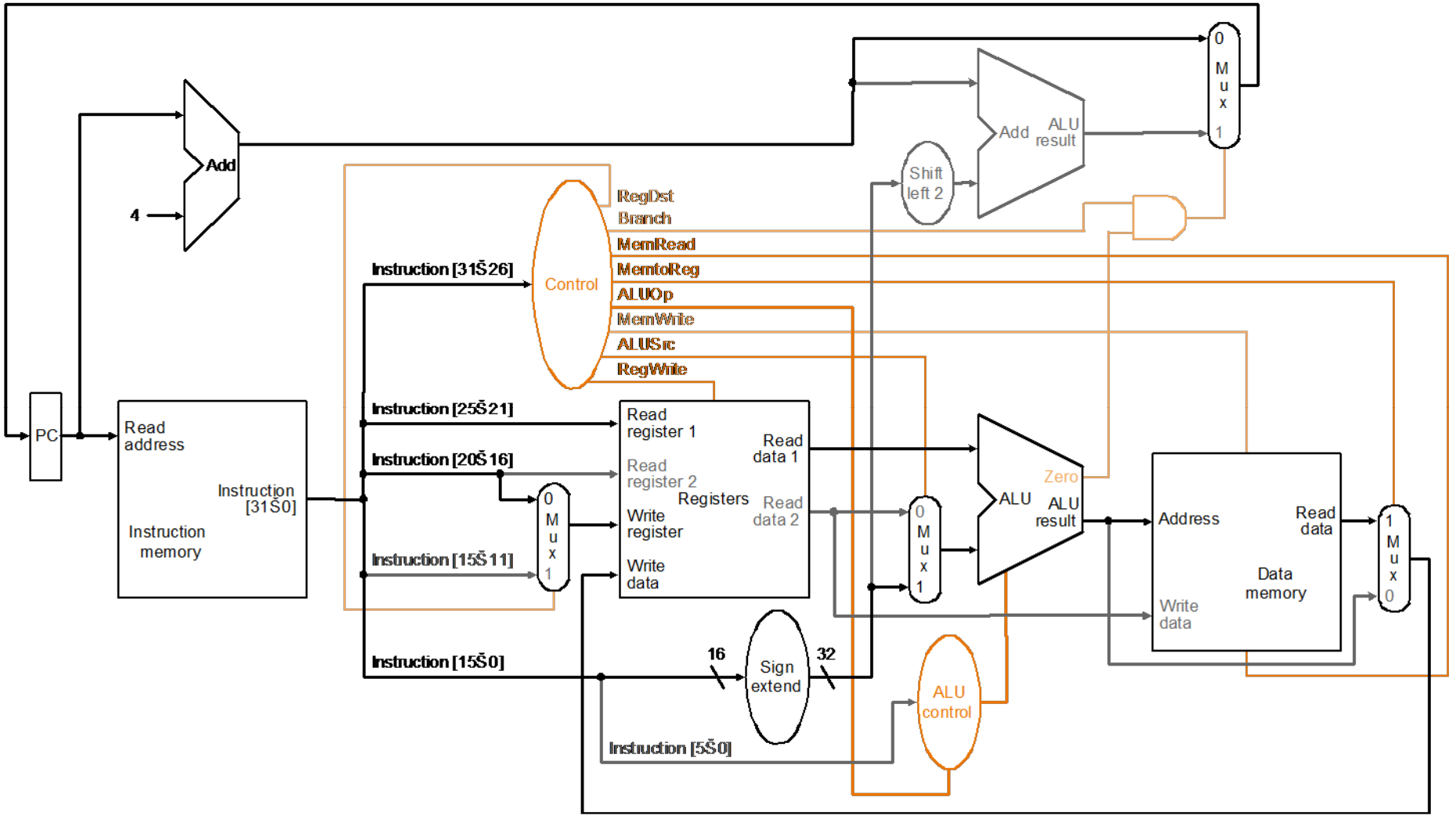
EECE 321: Computer Organization

Mohammad M. Mansour
Dept. of Electrical and Compute Engineering
American University of Beirut

Lecture 20: MIPS Single-Cycle Processor Implementation

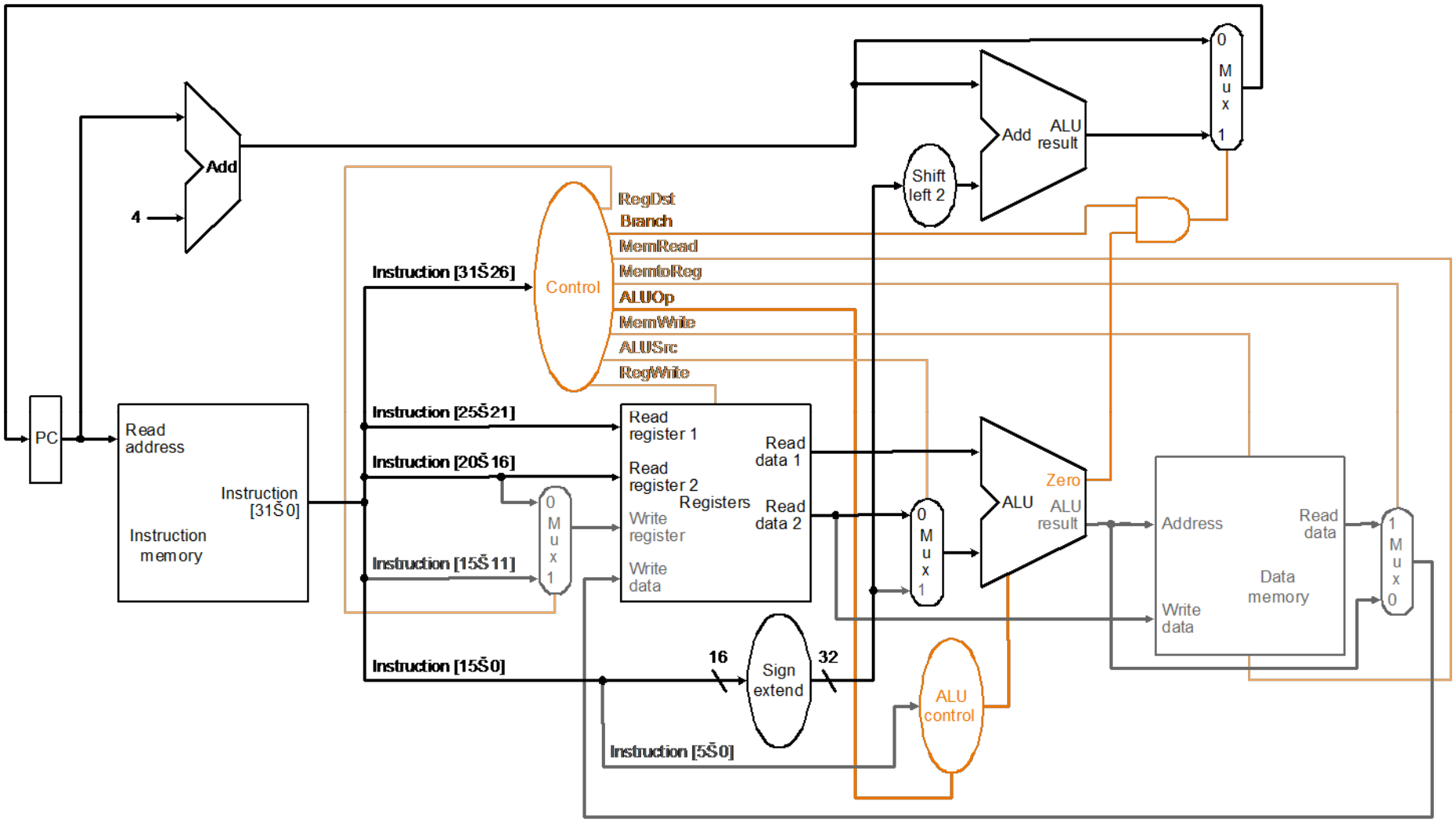
Example: Execution of `lw $t1,offset($t2)` on Datapath

- Divide into 5 steps: inst fetch, op fetch, addition, mem read, write-back.



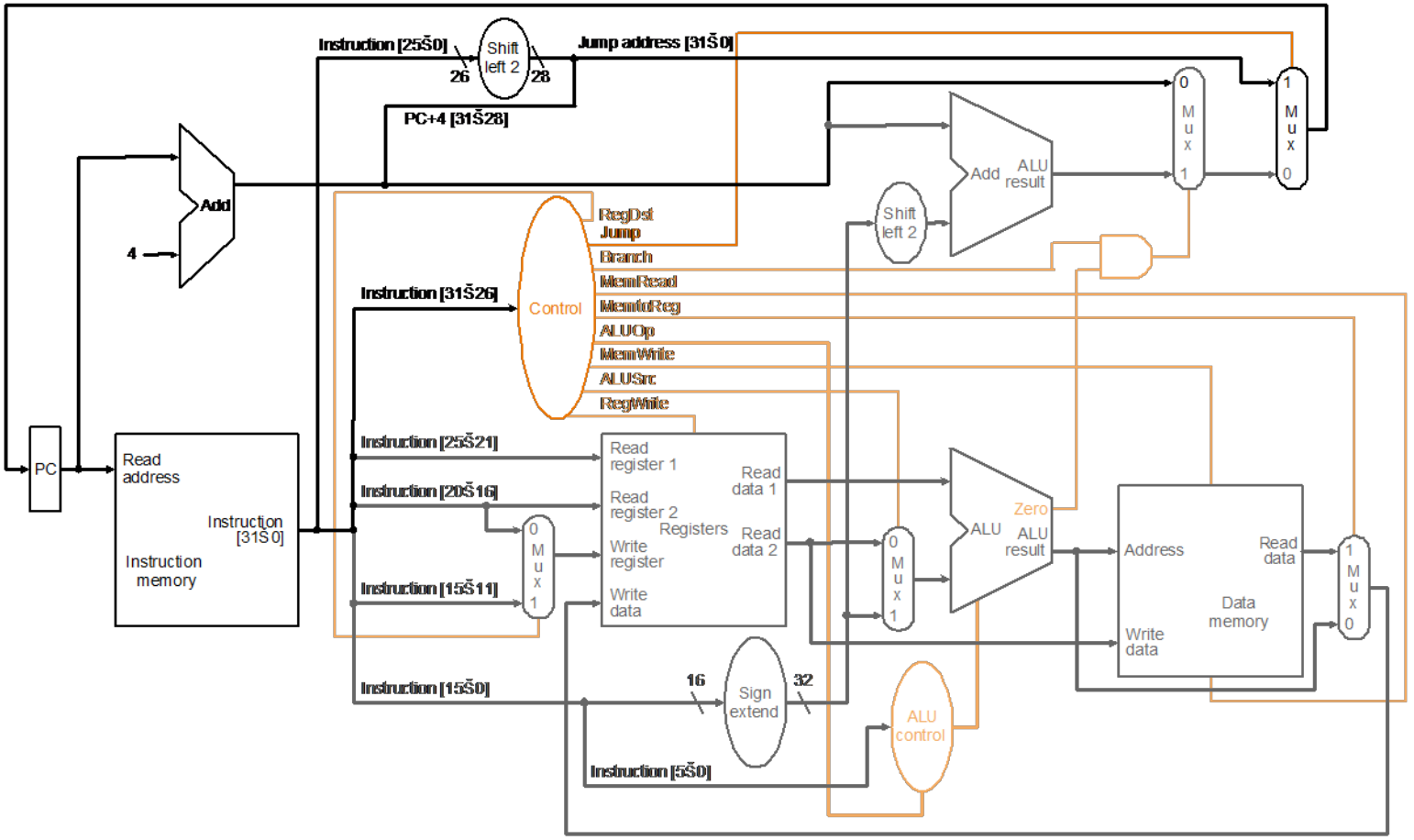
Example: Execution of `beq $t1,$t2,offset` on Datapath

- Divide into 4: inst fetch, op fetch, branch target address computation, branch decision.



Implementing Jumps

- We'll extend the datapath to include jump instructions: J Label
  - Jump is similar to branch but computes the target PC differently and is unconditional.
  - Jump address: 4 MSBs from PC || 26 bits from Label field || 00
  - Need an additional control signal called 'jump' and an additional MUX



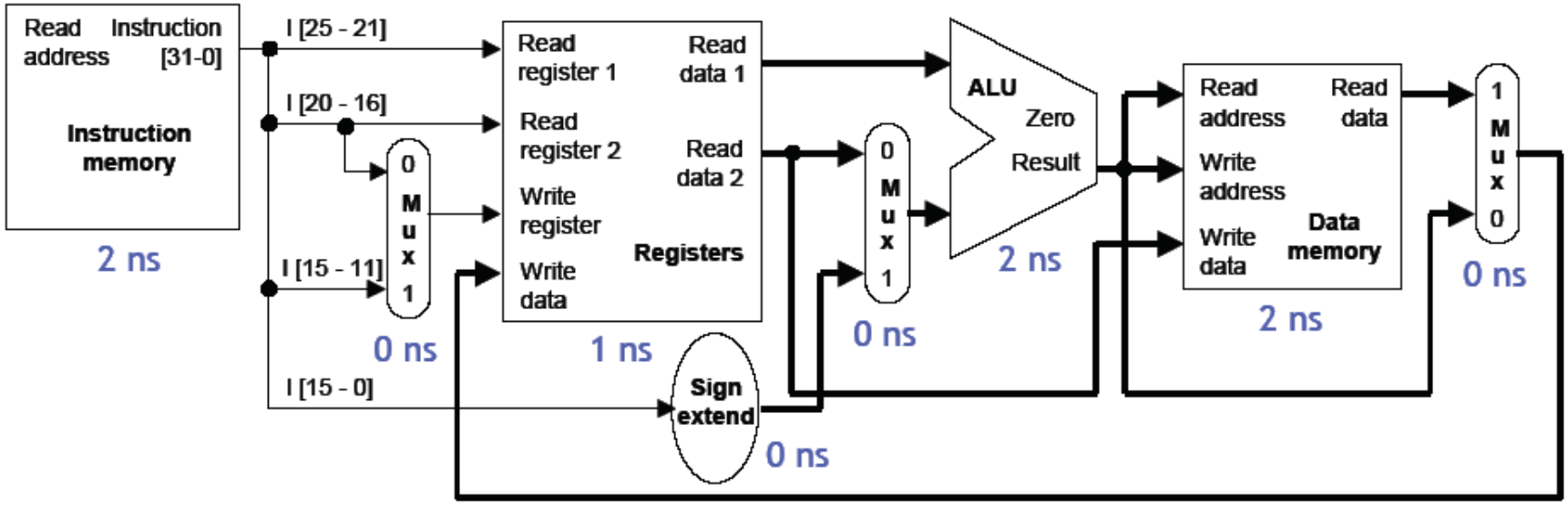
Performance of Single-Cycle Machines

- Single-Cycle operations recap:
  - On a positive clock edge, the PC is updated with a new address.
  - A new instruction can then be loaded from memory. The control unit sets the datapath signals appropriately so that:
    - Registers are read,
    - ALU output is generated,
    - Data memory is read or written, and
    - Branch target addresses are computed.
  - Several things happen on the next positive clock edge.
    - The register file is updated for arithmetic or lw instructions.
    - Data memory is written for a sw instruction.
    - The PC is updated to point to the next instruction.
- In a single-cycle datapath operations in Step 2 must complete within one clock cycle, before the next positive clock edge.
  - Assume that the operation time for the major functional units are:
  - memory (2ns,ultra-optimisitic), ALU and adders (2ns), register file access (1ns)
- Ignore delay of all other units
- If all instructions must complete within one clock cycle, then the cycle time has to be large enough to accommodate the slowest instruction.

The Slowest Instruction

- For example, lw $t0,–4($sp) needs 8ns:

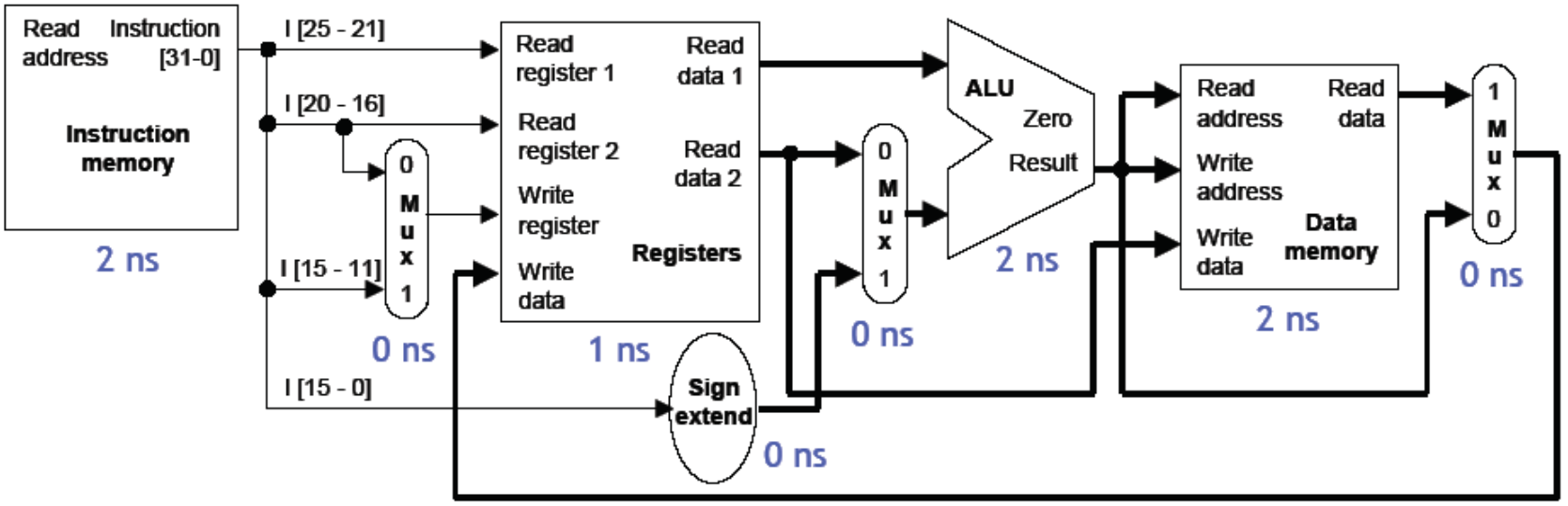
reading the instruction memory	2ns	8ns
reading the base register $sp	1ns	
computing memory address $sp-4	2ns	
reading the data memory	2ns	
storing data back to $t0	1ns	



The Slowest Instruction Determines the Clock-Cycle Time

- If we make the cycle time 8ns then every instruction will take 8ns, even if they don't need that much time.
- For example, the instruction add $s4, $t1, $t2 really needs just 6ns.

reading the instruction memory	2ns	6ns
reading registers $t1 and $t2	1ns	
computing $t1 + $t2	2ns	
storing the result into $s0	1ns	



Impact on Performance

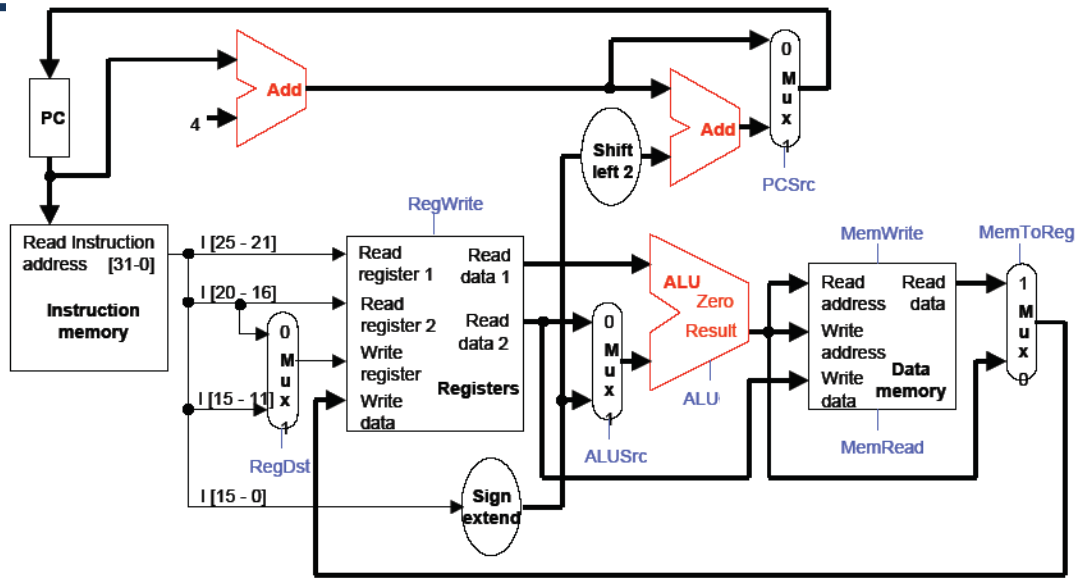
- With these same component delays, a sw instruction would need 7ns, and beq would need just 5ns.
- Let's consider the 'gcc' instruction mix:

Instruction	Frequency
Arithmetic	48%
Loads	22%
Stores	11%
Branches	19%

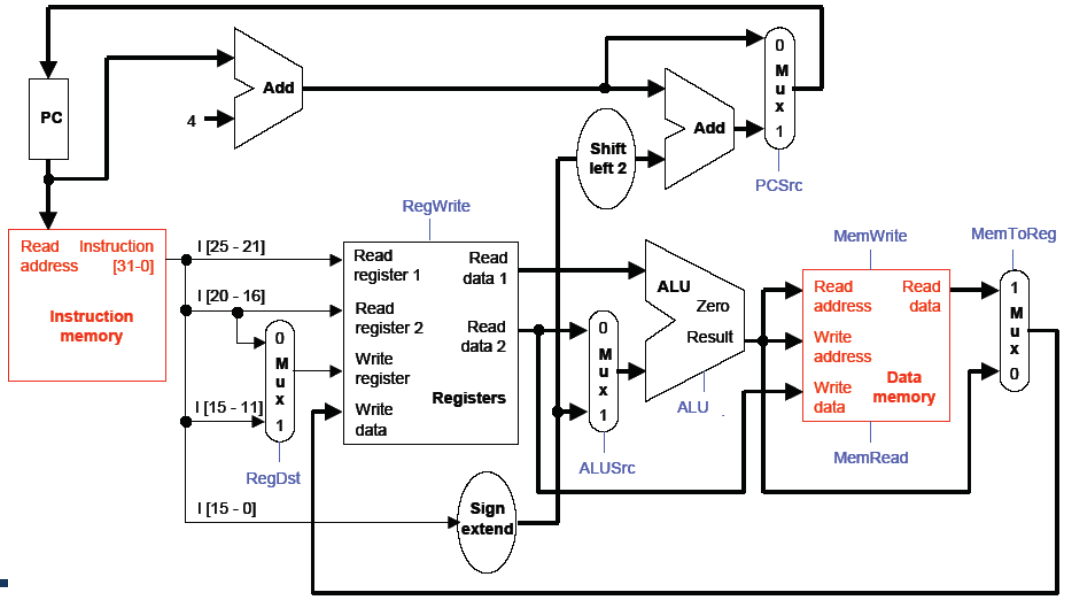
- With a single-cycle datapath, each instruction would require 8ns.
- But if we could execute instructions as fast as possible, the average time per instruction for gcc would be:
  - (48% x 6ns) + (22% x 8ns) + (11% x 7ns) + (19% x 5ns) = 6.36ns
- The single-cycle datapath is about 1.26 times slower!
- We've made very optimistic assumptions about memory latency:
  - Main memory accesses on modern machines is >50ns.
    - For comparison, an ALU on the Pentium4 takes ~0.3ns.
- Our worst case cycle (loads/stores) includes 2 memory accesses
  - A modern single cycle implementation would be stuck at <10Mhz.
  - Caches will improve common case access time, not worst case.
- Tying frequency to worst case path violates first law of performance!!

What About Hardware-Efficiency?

- A single-cycle datapath also uses extra hardware—one ALU is not enough, since we must do up to three calculations in one clock cycle for a beq.

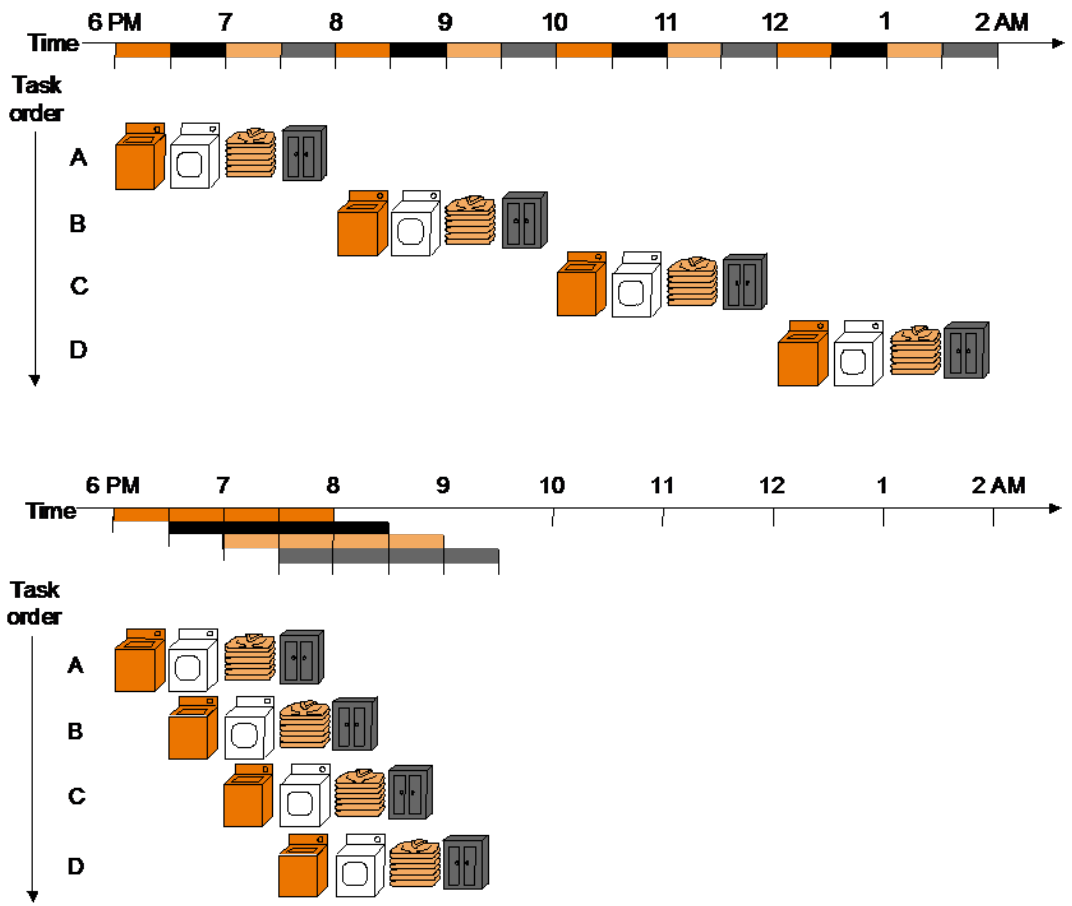


- Remember we had to use a Harvard architecture with two memories to avoid requiring a memory that can handle two accesses in one cycle.



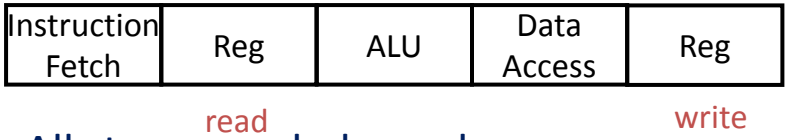
Introduction to Pipelining

- Pipelining is an implementation technique in which multiple instructions are overlapped in execution.
- Today, pipelining is key to making processors fast.
- Analogy to laundry:



Processor Pipelining

- Same principles can be applied to processors where we pipeline inst. execution.
  - For MIPS, the 5 stages are pipelined.
- Designate the 5 stages as follows:
  - Instruction fetch (IF)
  - Instruction decode and operand fetch: Reg
  - ALU operation execution: ALU
  - Access an operand in data memory: Data access
  - Write the result into a register: Reg
- An instruction executes by doing the appropriate work in each stage.
- Ex: load



- Convention: All stages are balanced
  - Register writes occur during first half of the stage, reads in the second half.

- R-Format instruction: doesn't use data access stage

Processor Pipelining

- Example: Assume we pipeline the 5 steps of executing instructions in MIPS (`lw, sw, add, sub, and, or, slt, beq`). Assume access of all functional units is 2ns, except register file which is 1ns.