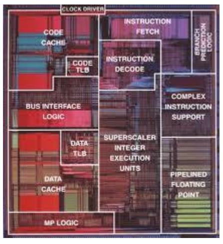
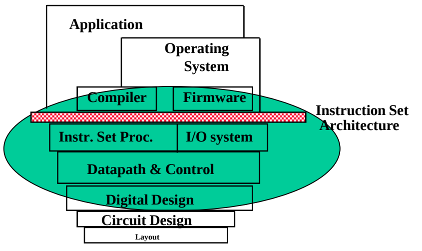
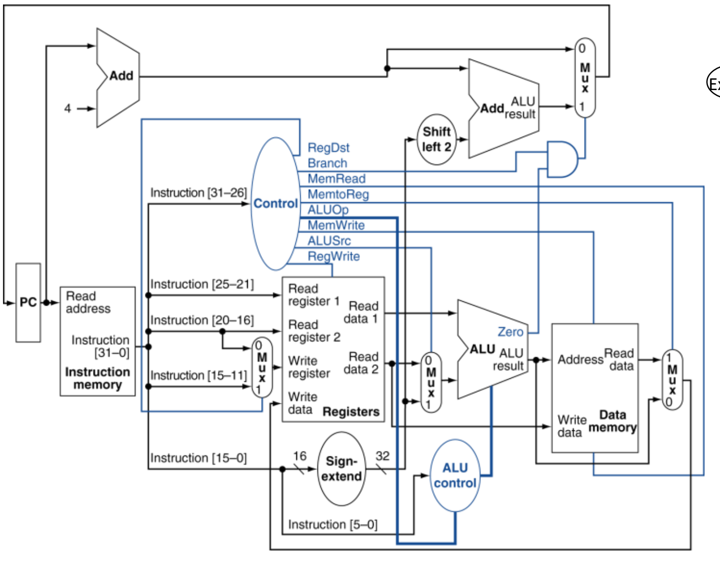
CS/SE 3340
Computer Architecture

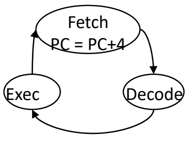




Building the Processor – Data Path & Control

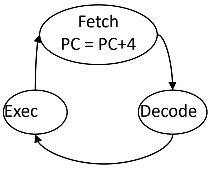
Adapted from slides by Profs. D. Patterson and J. Hennessey



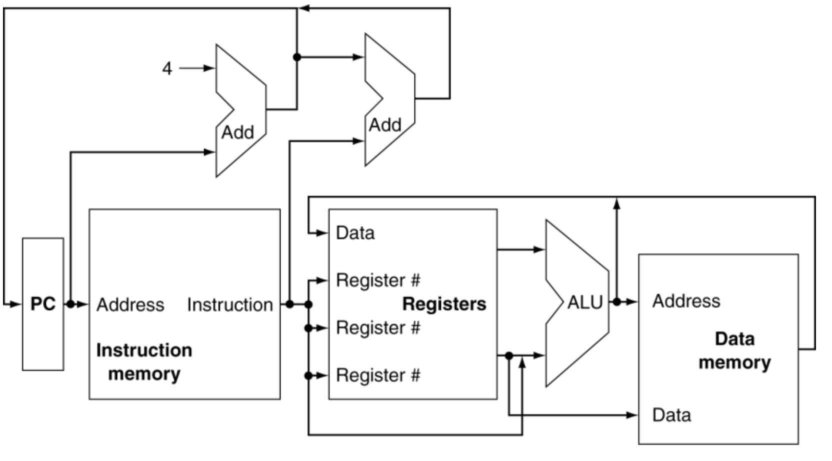


Instruction Execution

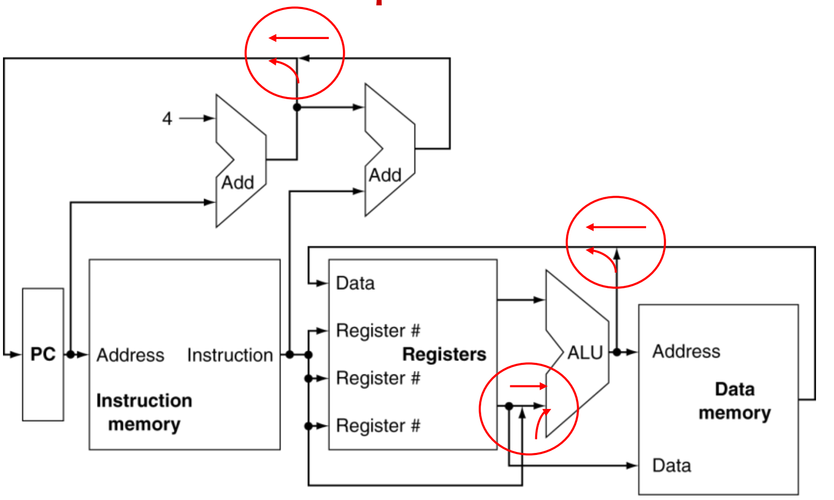
- PC → instruction memory, fetch instruction
  - PC is automatically updated to PC + 4 *(why?)*
- Register numbers → register file, read/write registers
- Depending on instruction type
  - Use ALU to calculate
    - Arithmetic result
    - Memory address for load/store
    - Branch target address
  - Access data memory for load/store
  - PC ← target address or PC + 4: *next instruction*



CPU Overview



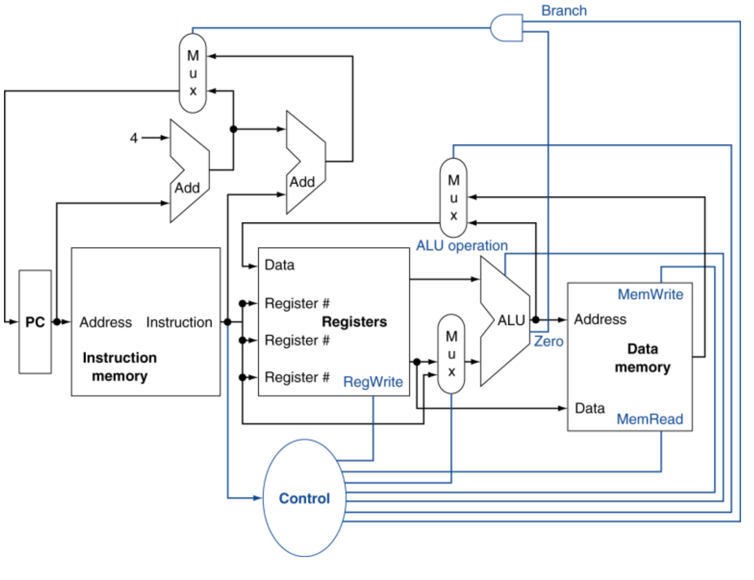
Multiplexers



- Can't just join wires together
  - Must use *multiplexers*

5

Control



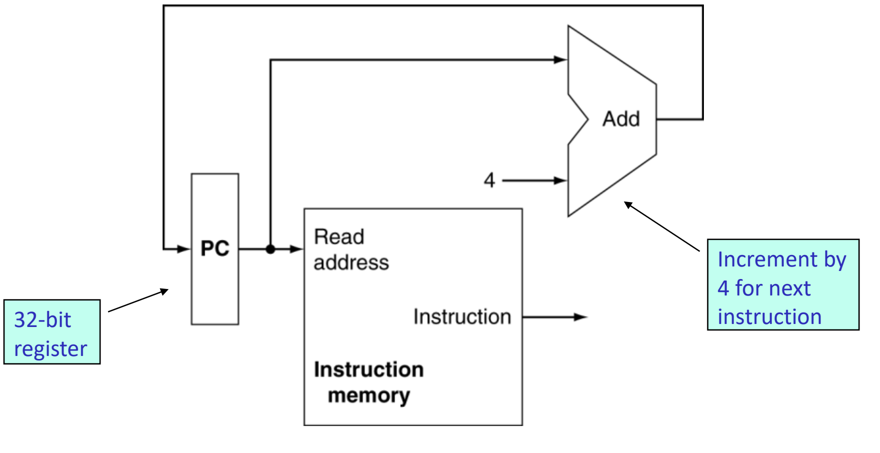
6

Building a Datapath

- Datapath
  - Elements that process data and addresses in the CPU
    - Registers, ALUs, mux's, memories, …
- We will build a MIPS datapath incrementally
  - Refining the overview design

7

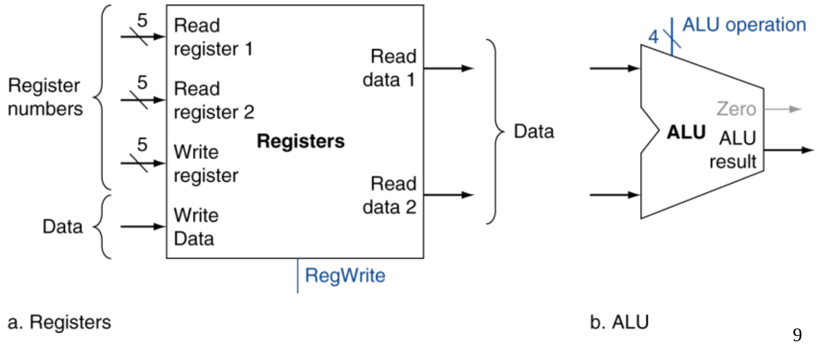
Instruction Fetch



8

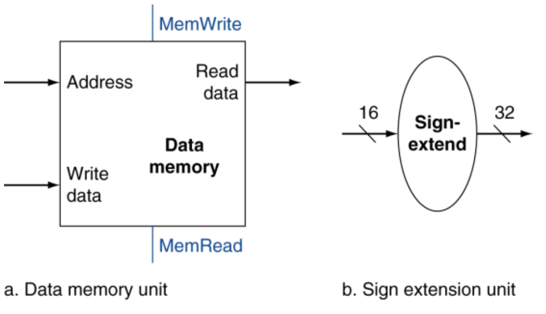
R-Format Instructions

- Read two register operands
- Perform arithmetic/logical operation
- Write register result



Load/Store Instructions

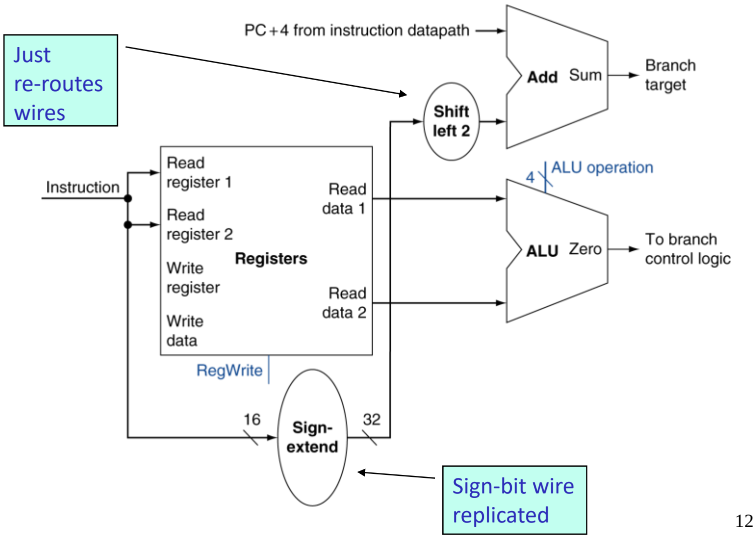
- Read register operands
- Calculate address using 16-bit offset
  - Use ALU, but sign-extend offset
- Load: Read memory and update register
- Store: Write register value to memory



Branch Instructions

- Read register operands
- Compare operands
  - Use ALU, subtract and check Zero output
- Calculate target address
  - Sign-extend displacement
  - Shift left 2 places (word displacement)
  - Add to PC + 4
    - Already calculated by instruction fetch

Branch Instructions

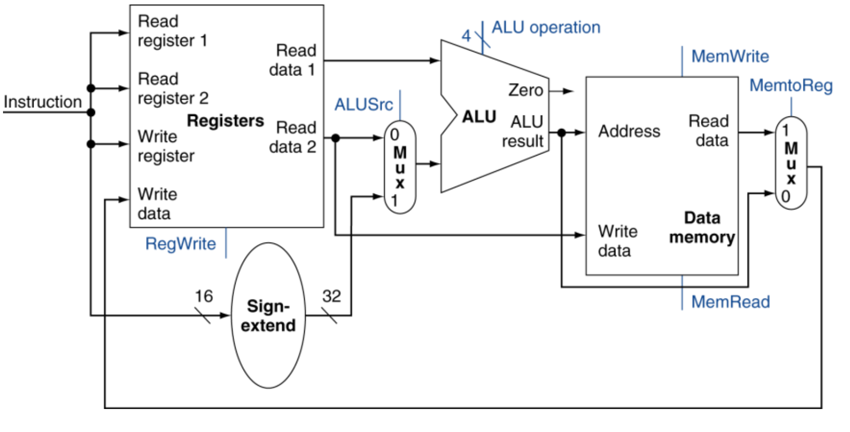


Composing the Elements

- First-cut data path does an instruction in one clock cycle
  - Each datapath element can only do one function at a time
  - Hence, we need separate instruction and data memories
- Use multiplexers where alternate data sources are used for different instructions

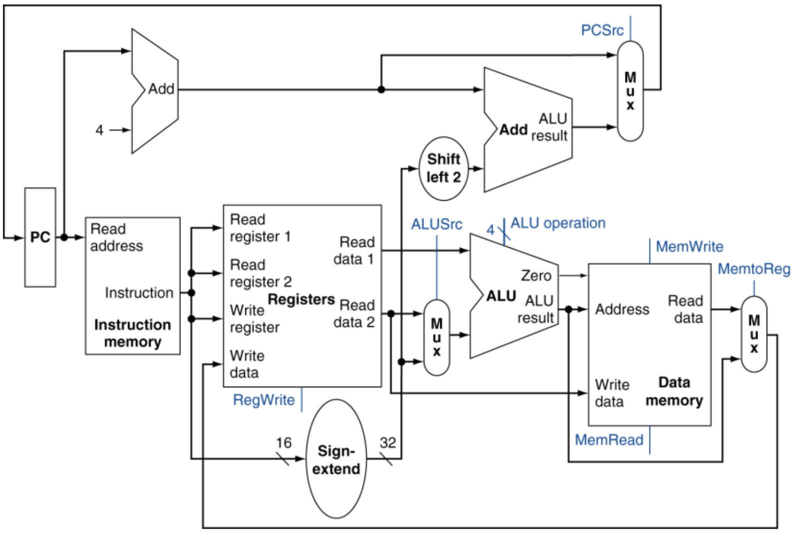
13

R-Type/Load/Store Datapath



14

Full Datapath



15

ALU Control

- ALU is used for
  - Load/Store: F = add
  - Branch: F = subtract
  - R-type: F depends on funct field

ALU control	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set-on-less-than
1100	NOR

16

ALU Control

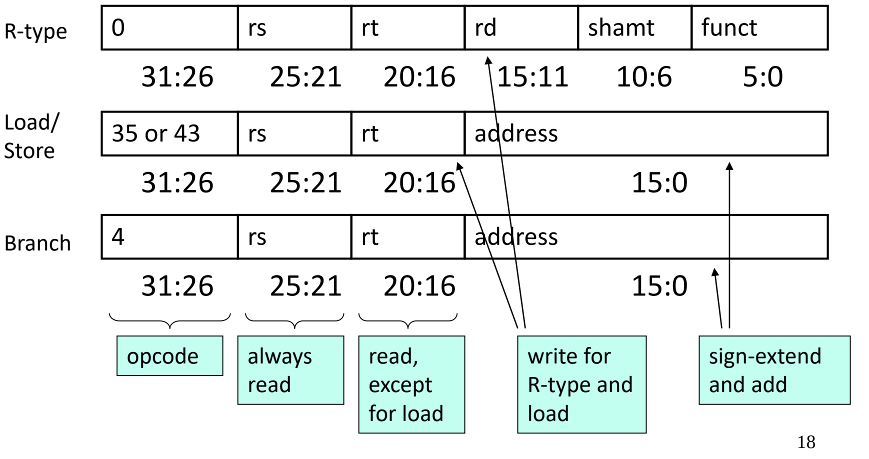
- Assume 2-bit ALUOp derived from opcode
  - Combinational logic derives ALU control

opcode	ALUOp	Operation	funct	ALU function	ALU control
lw	00	load word	XXXXXX	add	0010
sw	00	store word	XXXXXX	add	0010
beq	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
		subtract	100010	subtract	0110
		AND	100100	AND	0000
		OR	100101	OR	0001
		set-on-less-than	101010	set-on-less-than	0111

17

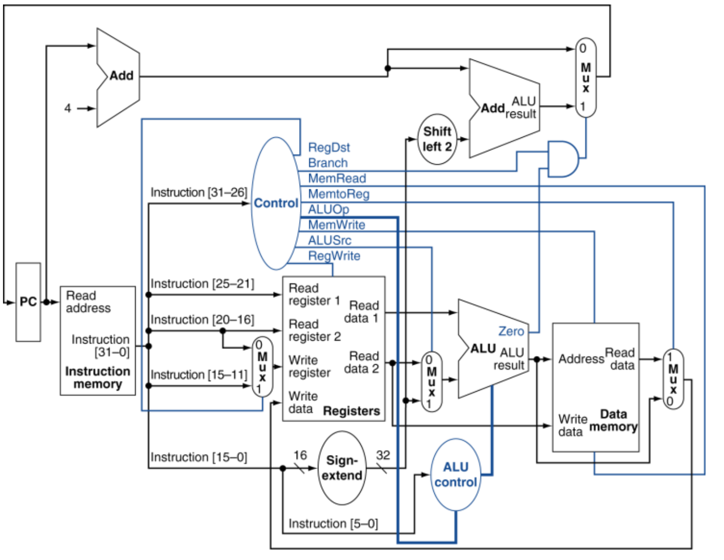
The Main Control Unit

- Control signals derived from instruction



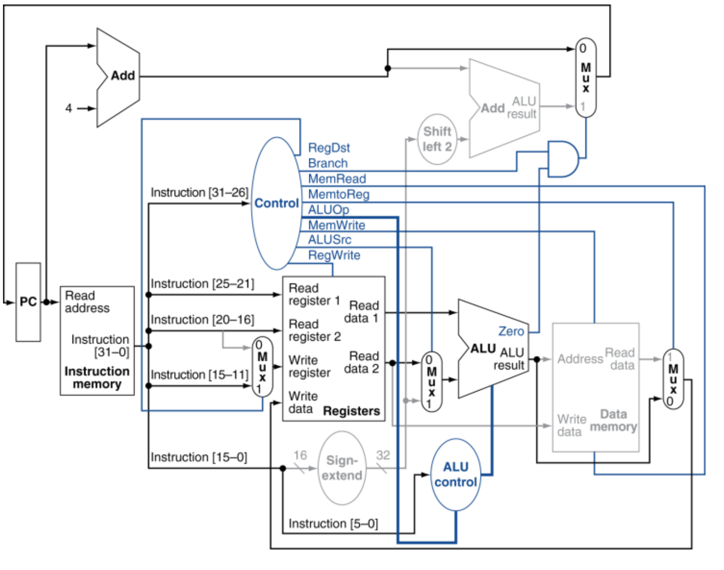
18

Datapath With Control



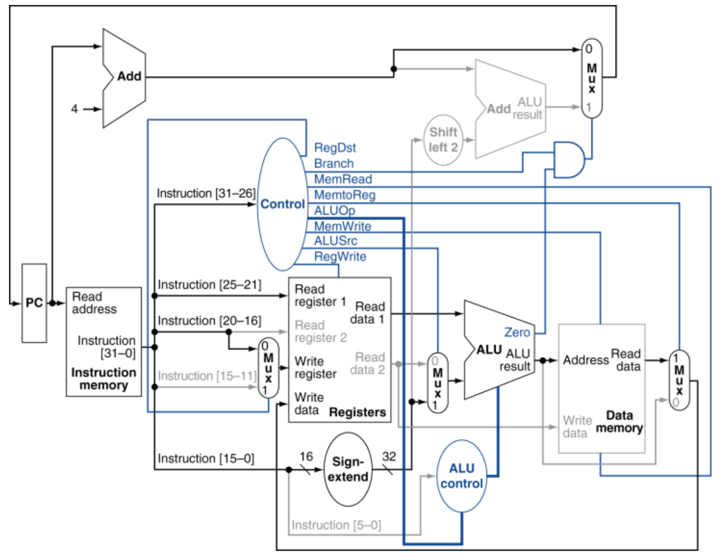
19

R-Type Instruction



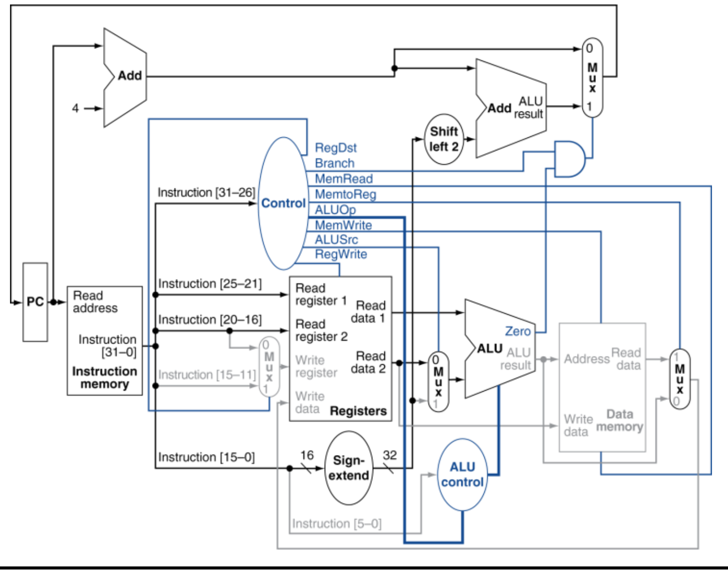
20

Load Instruction



21

Branch-on-Equal Instruction



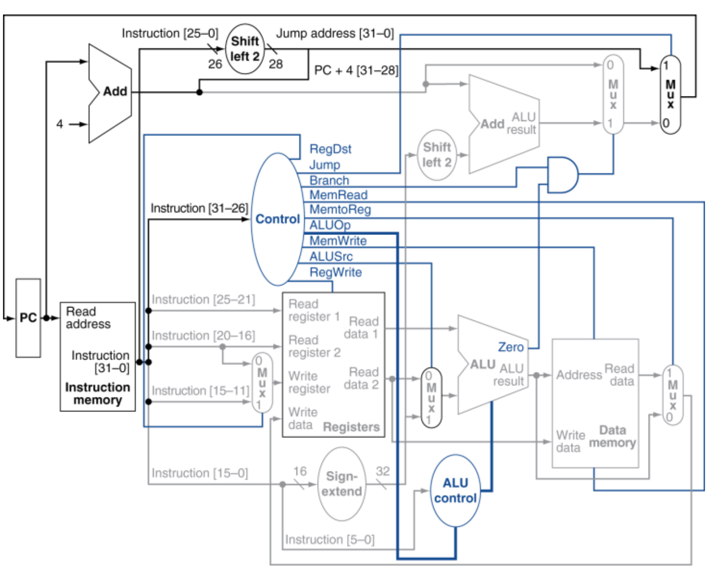
22

Implementing Jumps

Jump	2	address
	31:26	25:0

- Jump uses word address
- Update PC with concatenation of
  - Top 4 bits of old PC
  - 26-bit jump address
  - 00
- Need an extra control signal decoded from opcode

Datapath With Jumps Added



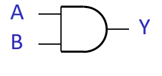
Logic Design Basics

- Information encoded in binary
  - *Low voltage = 0, High voltage = 1*
  - One wire per bit
  - Multi-bit data encoded on multi-wire buses
- Combinational elements
  - Operate on data
  - Output is a function of input
- State (sequential) elements
  - Store information, behavior depends on state

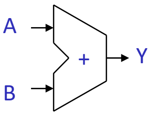
25

Combinational Elements

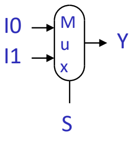
- AND-gate
  - Y = A & B



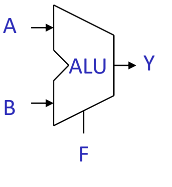
- Adder
  - Y = A + B



- Multiplexer
  - Y = S ? I1 : I0



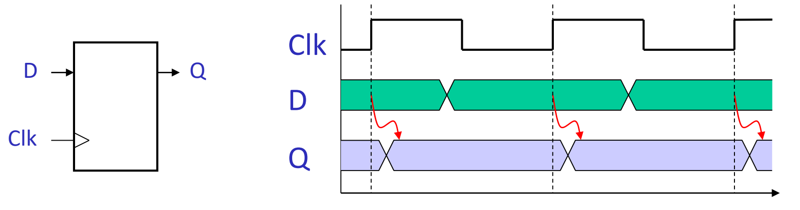
- Arithmetic/Logic Unit
  - Y = F(A, B)



26

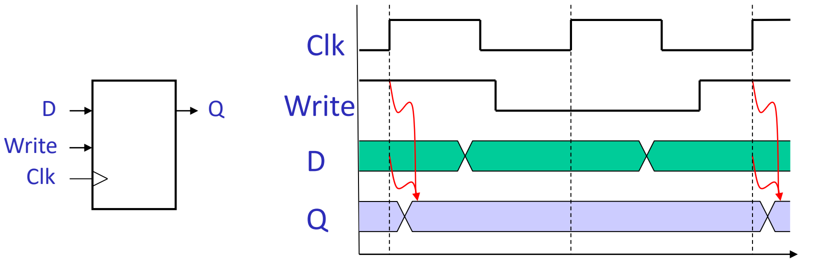
Sequential Elements

- Register: stores data in a circuit
  - Uses a *clock signal* to determine when to update the stored value
  - Edge-triggered: update when Clk changes from 0 to 1



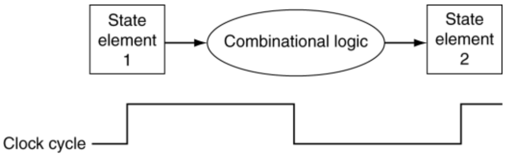
Sequential Elements

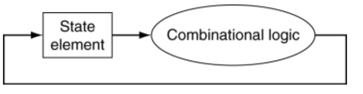
- Register with write control
  - Only updates on clock edge when write control input is 1
  - Used when stored value is required later



Clocking Methodology

- Combinational logic transforms data during clock cycles
  - Between clock edges
  - Input from state elements, output to state element
  - Longest delay determines clock period





29

Summary

- To build a processor we need to build a ***data path*** and a ***control*** unit
- Multiplexers are needed to select ("merge") inputs
- A data path and control for a simple (single cycle) processor that supports most instruction types were examined
- At hardware (circuit) level combinational and sequential logics are needed for this purpose

30