

# Designing a Single Cycle Datapath

or

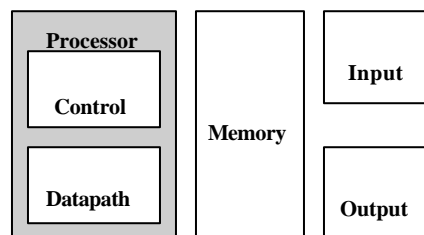
*The Do-It-Yourself CPU Kit*

CSE 141

Allan Snively

## The Big Picture: Where are We Now?

- The Five Classic Components of a Computer



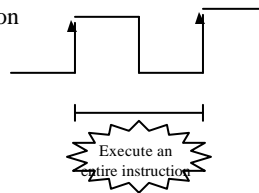
- Today's Topic: Datapath Design, then Control Design

CSE 141

Allan Snively

## The Big Picture: The Performance Perspective

- $ET = Insts * CPI * Cycle\ Time$
- Processor design (datapath and control) will determine:
  - Clock cycle time
  - Clock cycles per instruction
- Starting today:
  - Single cycle processor:
    - Advantage: One clock cycle per instruction
    - Disadvantage: long cycle time



CSE 141

Allan Snively

## The Processor: Datapath & Control

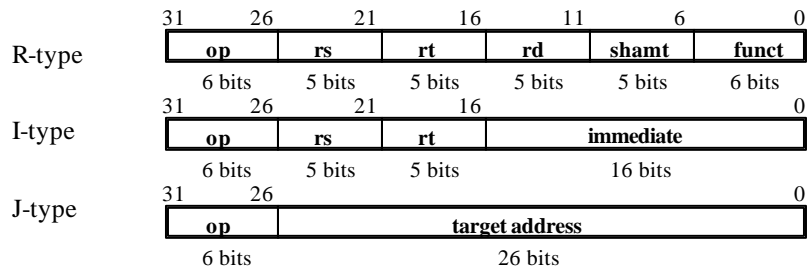
- We're ready to look at an implementation of the MIPS
- Simplified to contain only:
  - memory-reference instructions: `lw`, `sw`
  - arithmetic-logical instructions: `add`, `sub`, `and`, `or`, `sll`
  - control flow instructions: `beq`
- Generic Implementation:
  - use the program counter (PC) to supply instruction address
  - get the instruction from memory
  - read registers
  - use the instruction to decide exactly what to do
- All instructions use the ALU after reading the registers
  - memory-reference? arithmetic? control flow?

CSE 141

Allan Snively

## Review: The MIPS Instruction Formats

- All MIPS instructions are 32 bits long. The three instruction formats:

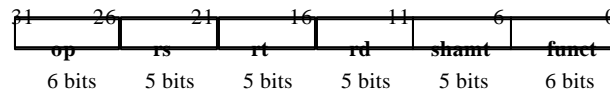


CSE 141

Allan Savely

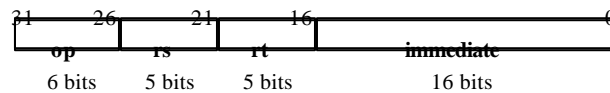
## The MIPS Subset

- R-type
  - *add rd, rs, rt*
  - *sub, and, or, slt*



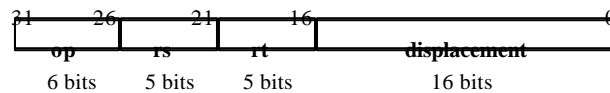
- LOAD and STORE

- *lw rt, rs, imm16*
- *sw rt, rs, imm16*



- BRANCH:

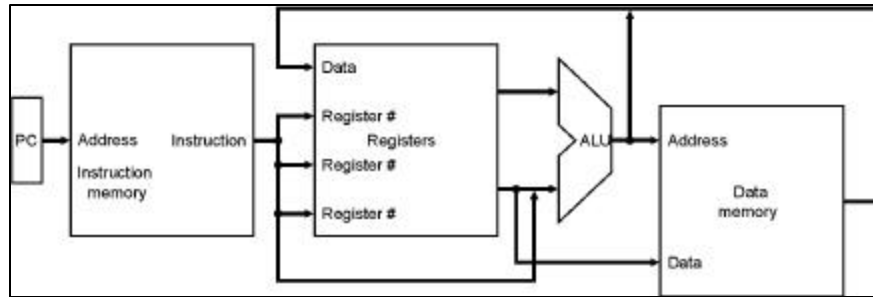
- *beq rs, rt, imm16*



CSE 141

Allan Savely

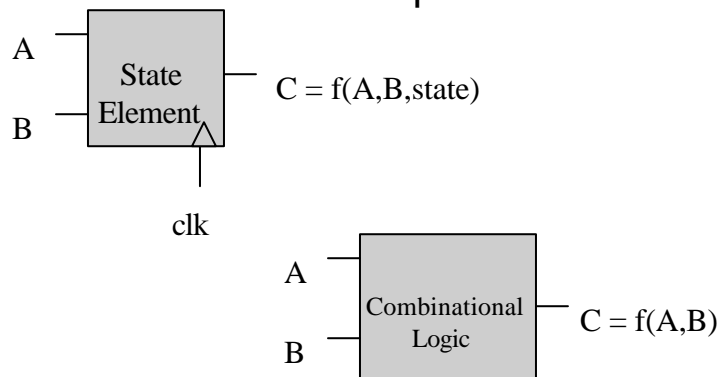
## Where We're Going – The High-level View



CSE 141

Allan Snively

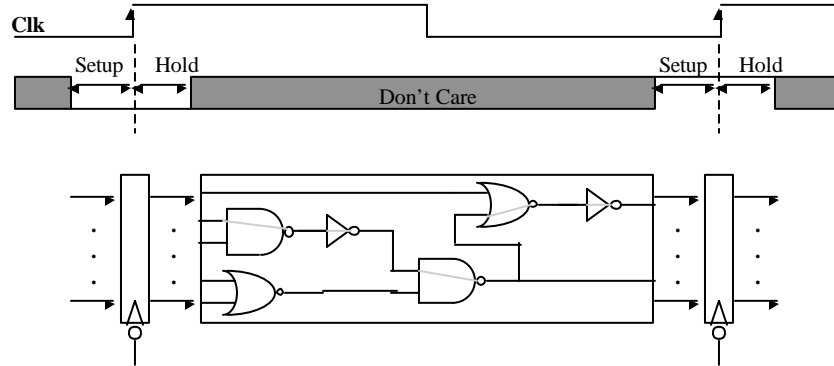
## Review: Two Types of Logic Components



CSE 141

Allan Snively

## Clocking Methodology



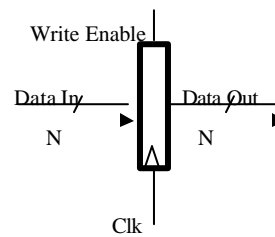
- All storage elements are clocked by the same clock edge

CSE 141

Allan Snively

## Storage Element: Register

- Register
  - Similar to the D Flip Flop except
    - N-bit input and output
    - Write Enable input
  - Write Enable:
    - 0: Data Out will not change
    - 1: Data Out will become Data In (on the clock edge)

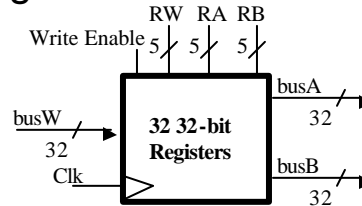


CSE 141

Allan Snively

## Storage Element: Register File

- Register File consists of (32) registers:
  - Two 32-bit output buses:
  - One 32-bit input bus: busW
- Register is selected by:
  - RA selects the register to put on busA
  - RB selects the register to put on busB
  - RW selects the register to be written via busW when Write Enable is 1
- Clock input (CLK)

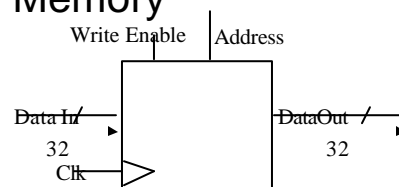


CSE 141

Allan Snively

## Storage Element: Memory

- Memory
  - One input bus: Data In
  - One output bus: Data Out
- Memory word is selected by:
  - Address selects the word to put on Data Out
  - Write Enable = 1: address selects the memory word to be written via the Data In bus
- Clock input (CLK)
  - The CLK input is a factor ONLY during write operation
  - During read operation, behaves as a combinational logic block:
    - Address valid => Data Out valid after "access time."



CSE 141

Allan Snively

## Register Transfer Language (RTL)

- is a mechanism for describing the movement and manipulation of data between storage elements:

$R[3] \leftarrow R[5] + R[7]$

$PC \leftarrow PC + 4 + R[5]$

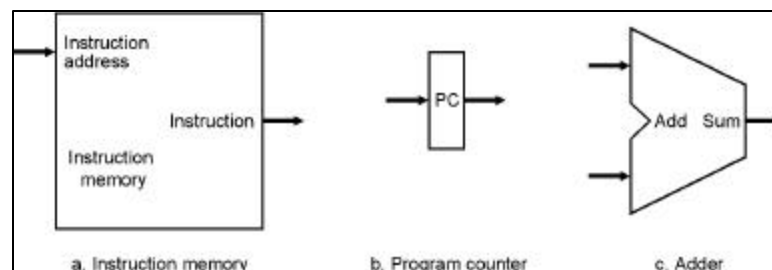
$R[rd] \leftarrow R[rs] + R[rt]$

$R[rt] \leftarrow Mem[R[rs] + immed]$

*CSE 141*

*Allan Snively*

## Program Counter Management

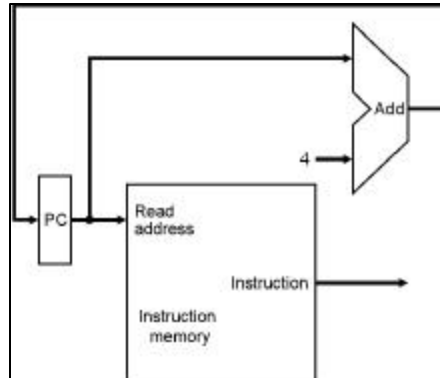


*CSE 141*

*Allan Snively*

## Overview of the Instruction Fetch Unit

- The common RTL operations
  - Fetch the Instruction:  $inst \leftarrow mem[PC]$
  - Update the program counter:
    - Sequential Code:  $PC \leftarrow PC + 4$
    - Branch and Jump  $PC \leftarrow \text{"something else"}$

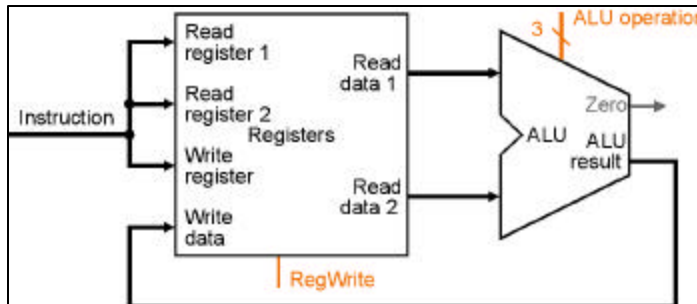
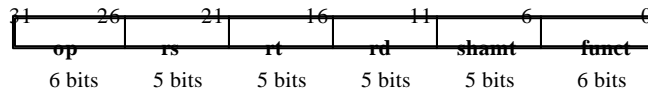


CSE 141

Allan Snively

## Datapath for Register-Register Operations

- $R[rd] \leftarrow R[rs] \text{ op } R[rt]$  Example: *add rd, rs, rt*
  - Ra, Rb, and Rw comes from instruction's rs, rt, and rd fields
  - ALUctr and RegWr: control logic after decoding the instruction



CSE 141

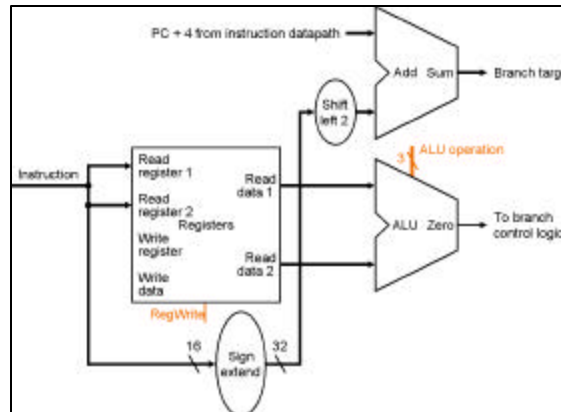
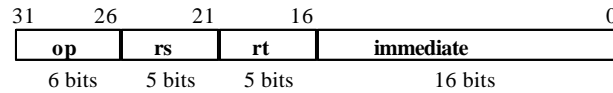
Allan Snively



## Datapath for Branch Operations

*beq* *rs, rt, imm16*

We need to compare Rs and Rt



CSE 141

Allan Snively

## Binary Arithmetic for the Next Address

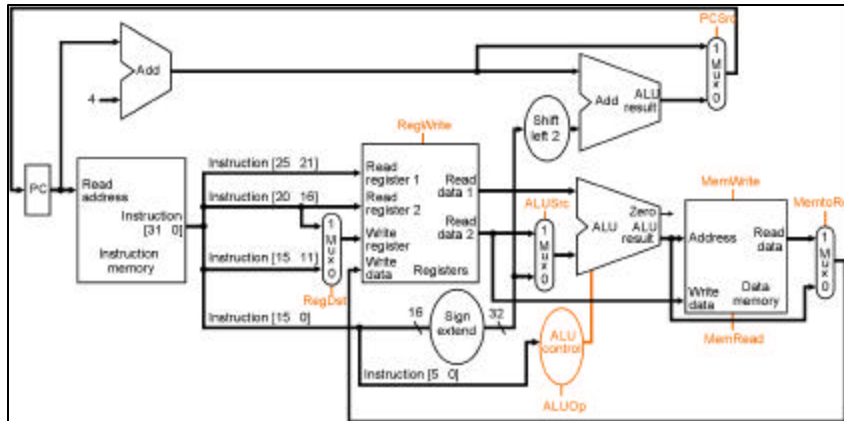
- In theory, the PC is a 32-bit byte address into the instruction memory:
  - Sequential operation:  $PC\langle 31:0 \rangle = PC\langle 31:0 \rangle + 4$
  - Branch operation:  $PC\langle 31:0 \rangle = PC\langle 31:0 \rangle + 4 + \text{SignExt}[\text{Imm16}] * 4$
- The magic number “4” always comes up because:
  - The 32-bit PC is a byte address
  - And all our instructions are 4 bytes (32 bits) long
  - The 2 LSBs of the 32-bit PC are always zeros
  - There is no reason to have hardware to keep the 2 LSBs
- In practice, we can simplify the hardware by using a 30-bit  $PC\langle 31:2 \rangle$ :
  - Sequential operation:  $PC\langle 31:2 \rangle = PC\langle 31:2 \rangle + 1$
  - Branch operation:  $PC\langle 31:2 \rangle = PC\langle 31:2 \rangle + 1 + \text{SignExt}[\text{Imm16}]$
  - In either case: Instruction Memory Address =  $PC\langle 31:2 \rangle$  concat “00”

CSE 141

Allan Snively



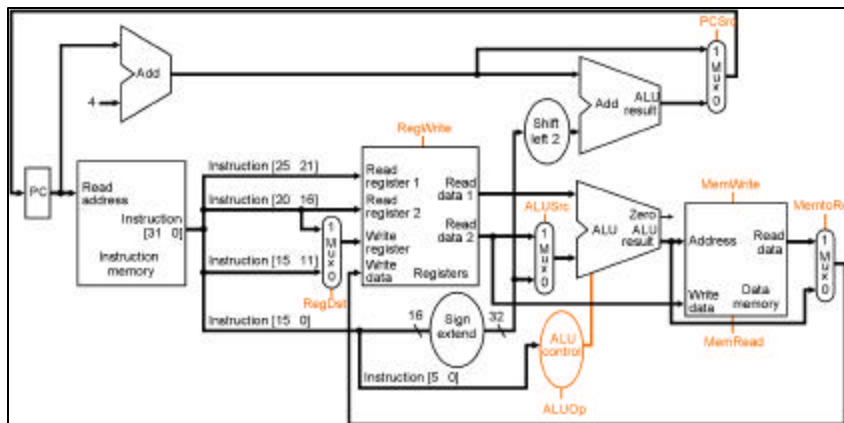
## The Load Datapath



CSE 141

Allan Snively

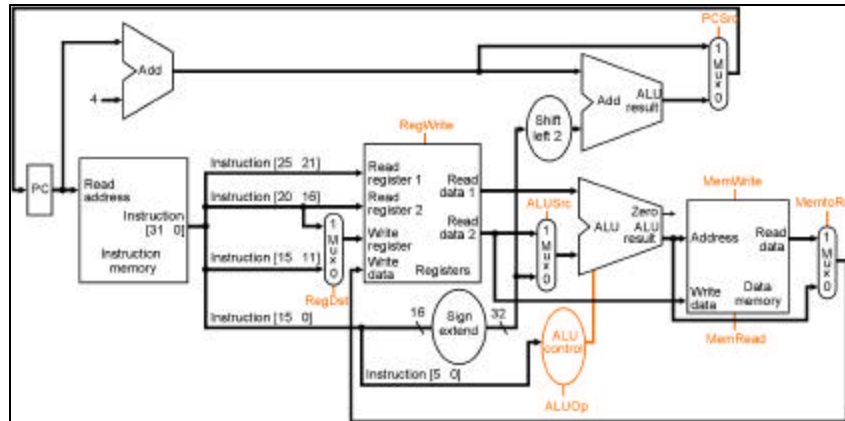
## The store Datapath



CSE 141

Allan Snively

## The beq Datapath



CSE 141

Allan Savely

## Key Points

- CPU is just a collection of state and combinational logic
- We just designed a very rich processor, at least in terms of functionality
- Performance = Insts \* CPI \* Cycle Time
  - where does the single-cycle machine fit in?

CSE 141

Allan Savely