CSE 141 – Computer Architecture
Spring 2005

Lecture 6
Single Cycle CPU Part 1
April 13, 2005

Pramod V. Argade

CSE141: Introduction to Computer Architecture

Instructor: Pramod V. Argade (p2argade@cs.ucsd.edu)
Office Hour:
Mon. 5:00 - 6:00 PM (AP&M 5218)

TAs:

Baris Arslan: barslan@cs.ucsd.edu
Chris Roedel: croedel@cs.ucsd.edu
Raid Ayoub: rayoub@cs.ucsd.edu
Leo Porter: leporter@cs.ucsd.edu

Textbook: Computer Organization & Design
The Hardware Software Interface, 3rd Edition.
Authors: Patterson and Hennessy

Web-page: http://www.cse.ucsd.edu/classes/sp05/cse141

Announcements

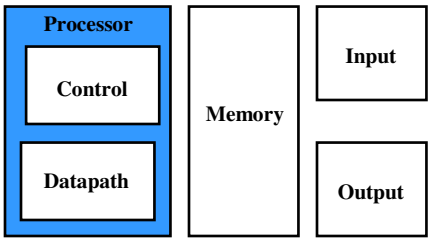
- **Reading Assignment**
  - Chapter 3. Arithmetic for Computers
    Sections B6, In More Depth 3.23, 3.4 - 3.8
  - Chapter 5. The Processor: Datapath and Control
    Sections 5.1 - 5.3, Appendix B7 - B11
- **Homework 2: Due Mon., April 18 in class**
  3.13, 3.18, 3.19, 3.27, 3.30, 3.35, 3.38
  Multiply -11x-10 using Booth's algorithm and 5-bit 2-complement representation of multiplicand and multiplier.
  5.1
- **Quiz**
  When: Mon., April 18th, First 10 minutes of the class
  Topic: Arithmetic for Computers, Chapter 3 **Need:** Paper, pen

Course Schedule

Lecture #	Date	Day	Lecture Topic	Quiz Topic	Homework Due
1	3/28	Monday	Introduction, Ch. 1	-	-
2	3/30	Wednesday	Performance, Ch. 4	-	-
3	4/4	Monday	ISA, Ch. 2	Performance	#1
4	4/6	Wednesday	Arithmetic, Ch. 3	-	-
5	4/11	Monday	Arithmetic, Ch. 3	ISA	#2
6	4/13	Wednesday	Single cycle CPU, Ch. 5	-	-
7	4/18	Monday	Single cycle CPU, Ch. 5	Arithmetic	#3
8	4/20	Wednesday	Multi-cycle CPU, Ch. 5	-	-
9	4/25	Monday	Multi-cycle CPU, Ch. 5	Single Cycle CPU	#4
10	4/27	Wednesday	Review for the Midterm	-	-
	5/2	**Monday**	**Mid-term Exam**	-	-
11	5/4	Wednesday	Exceptions, Ch. 5 and Pipelining, Ch. 6	-	-
12	5/9	Monday	Pipelining, Ch. 6	-	-
13	5/11	Wednesday	Data and control hazards, Ch. 6	-	-
14	5/16	Monday	Data and control hazards, Ch. 6	Pipeline Hazards	#5
15	5/18	Wednesday	Memory & cache design, Ch. 7	-	-
16	5/23	Monday	Memory & cache design, Ch. 7	Cache	#6
17	5/25	Wednesday	Virtual Memory & cache design, Ch. 7	-	-
No Class	5/30	Monday	Memorial Day Holiday	-	-
18	6/1	Wednesday	Course Review	-	-
	6/10	**Friday**	**Final Exam**		

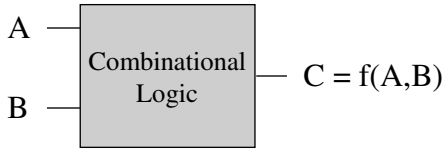
Datapath and Control Design

- The Five Classic Components of a Computer

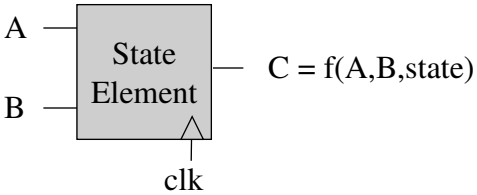


Two Types of Logic Components

- Combinational
  - Elements that operate on data values
  - Produces same output if given same inputs

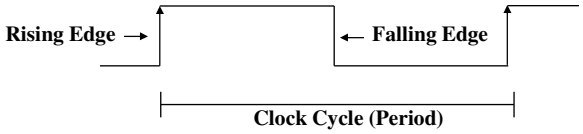


- State Elements
  - Contains internal storage
  - May produce different output if given same inputs
  - Clock is used to determine when a state element(s) should be written



Clock

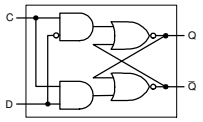
- Clock is a free running signal
  - Fixed cycle time (period)
  - Frequency = 1/(cycle time)
  - Duty Cycle: (% high)/(%low), e.g. 50/50 Duty Cycle below
  - Jitter: Uncertainty/wobble in rising or falling edge

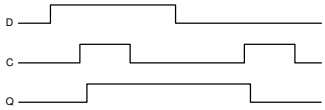


Storage Elements

D Latch

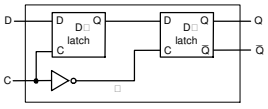
- Two inputs:
  - the data value to be stored (D)
  - the clock signal (C) indicating when to read & store D
- Two outputs:
  - the value of the internal state (Q) and it's complement

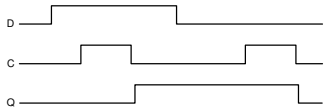




Falling edge triggered D flip-flop

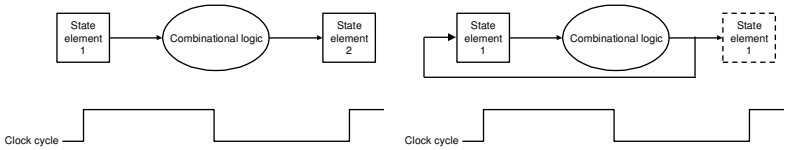
- Output changes only on the clock edge





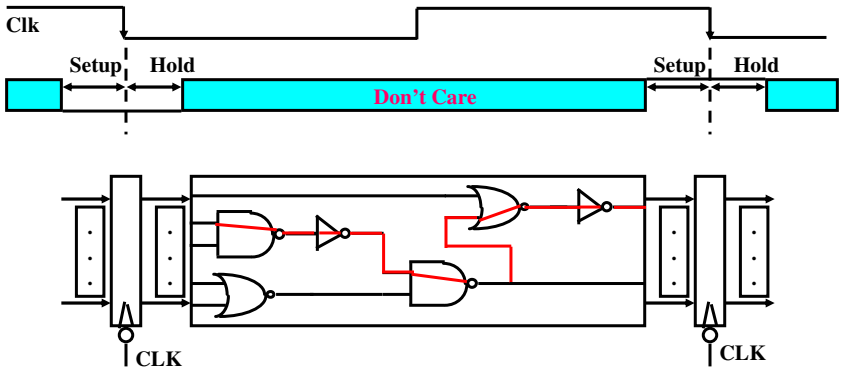
Edge-triggered Clocking

- Values stored in the machine are updated on a clock edge
  - The clock edge can be either rising or falling



- By default a state element is written every clock edge
  - An explicit write control signal is required otherwise.
- Edge triggered methodology allows, in the same clock cycle to:
  - read the contents of a register
  - send the value through some combinational logic, and
  - write the contents of the same or another register
- Possible to have the same state element as input and output

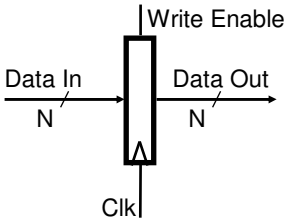
CPU: Clocking



- All storage elements are clocked by the same clock edge

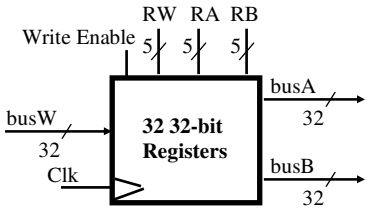
Register: A Storage Element

- 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)

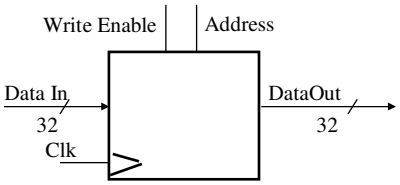


Register File

- Register File consists of (32) registers:
  - Two 32-bit output busses: busA and busB
  - 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)

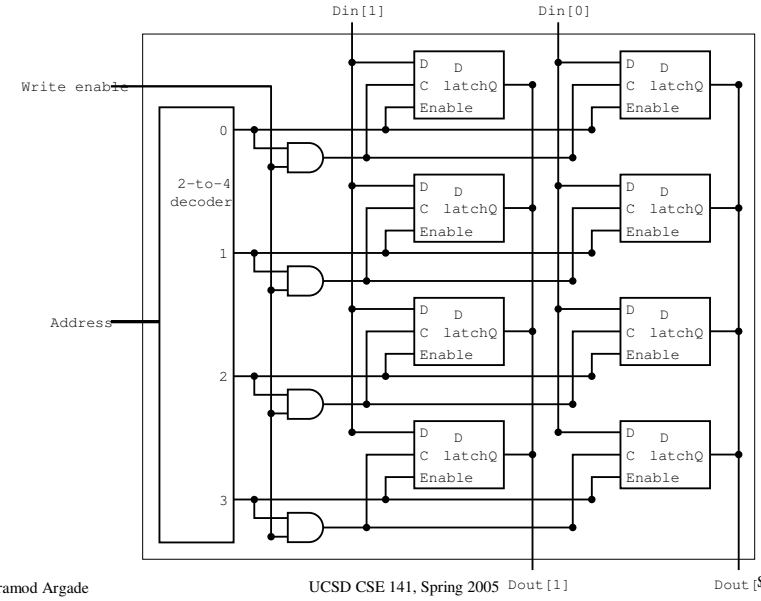


Memory



- 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."

Basic 4 x 2 Static RAM



Register Transfer Language (RTL)

- Mechanism for describing the movement and manipulation of of data between storage elements.
- Example:

  R[rd] ← R[rs] + R[rt]

  PC ← PC + 4

  R[rt] ← Mem[R[rs] + immediate]

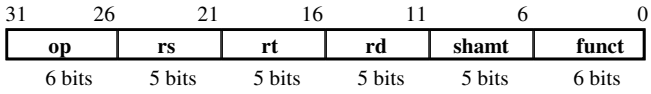
A Simple Implementation of MIPS CPU

- Simplified to contain only:
  - Memory-reference instructions: lw, sw
  - Arithmetic-logical instructions: add, sub, and, or, slt
  - Control flow instructions: beq, j
- Execution Time = Instructions * CPI * Cycle Time
- Processor design (datapath and control) will determine:
  - Clock cycle time
  - Clock cycles per instruction
- We will design a single cycle processor:
  - Advantage: One clock cycle per instruction
  - Disadvantage: long cycle time

Arithmetic Instructions (R-Type)

- ADD, SUB, AND, OR, SLT
- Example

  add rd, rs, rt

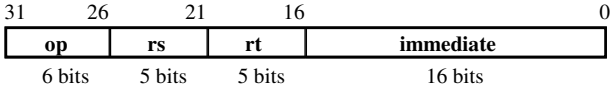


Load/Store Instructions (I-Type)

- LW, SW
- Examples

  lw rt, rs, imm16

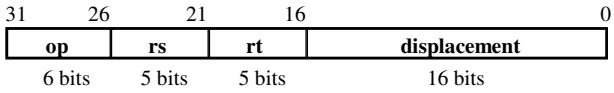
  sw rt, rs, imm16



Branch (I-Type)

- Beq
- Example

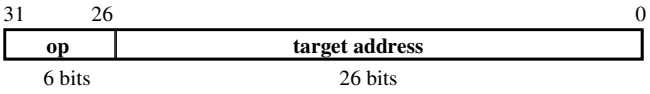
  beq rs, rt, imm16



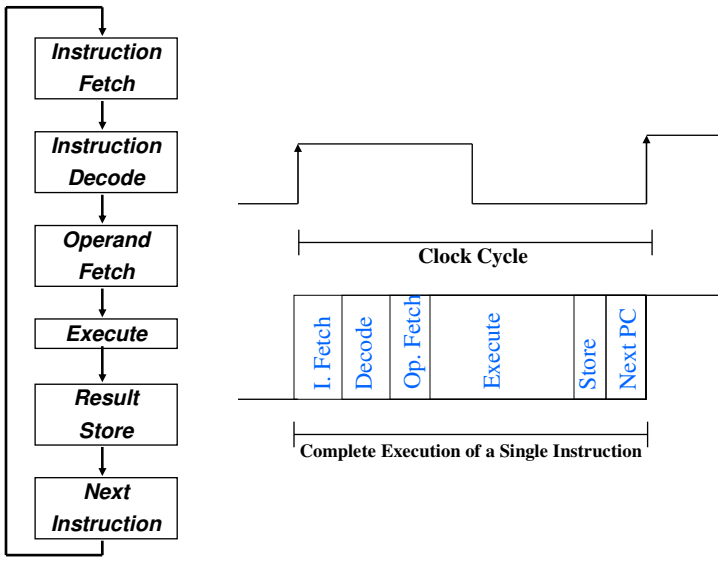
Jump (J-Type)

- J
- Example

  J Label



Single Cycle Implementation ⇨ Datapath and Control

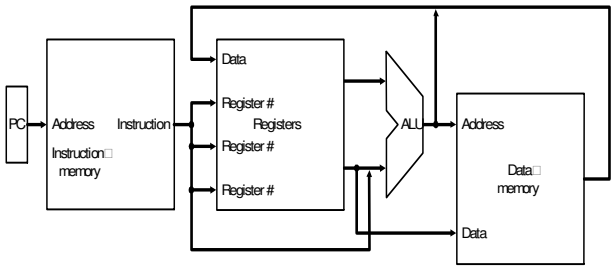


Components Required to implement the ISA

- Next PC generation
  - Add 4 or extended 16-bit immediate to PC
- Memory
  - Instruction read
  - Data read/write
- Registers (32 x 32-bit)
  - Read register rs
  - Read register rt
  - Write register rt or rd
- Sign extend immediate operand
- ALU to operate on the operands

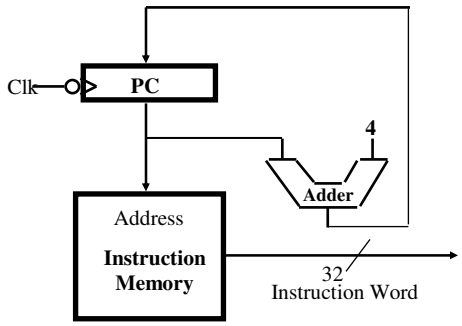
Datapath

- Abstract / Simplified View:



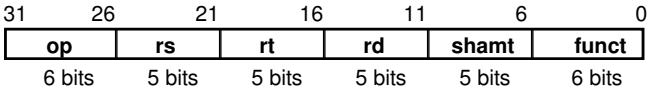
CPU: Instruction Fetch

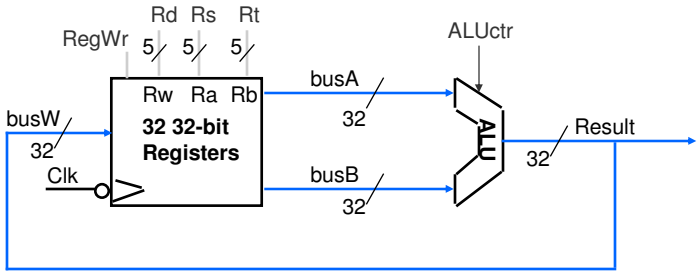
- RTL version of the instruction fetch step:
  - Fetch the Instruction: mem[PC]
  - – Update the program counter:
    - Sequential Code: PC ← PC + 4
    - Branch and Jump: PC ← "something else"



CPU: Register-Register Operations (Add, Subtract etc.)

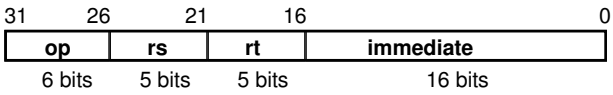
- R[rd] ← R[rs] op R[rt] Example: addU rd, rs, rt
  - Ra, Rb, and Rw come from instruction's rs, rt, and rd fields
  - ALUctr and RegWr: control logic after decoding the instruction

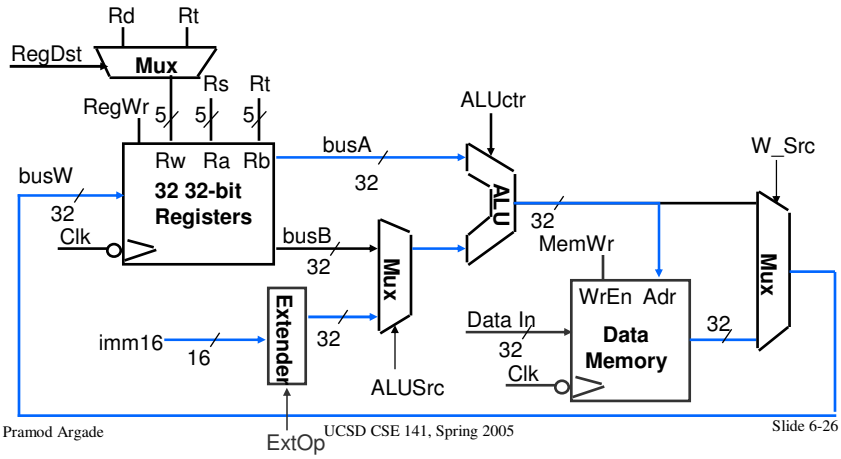




CPU: Load Operations

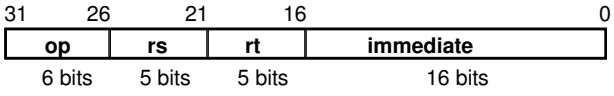
- **R[rt] ← Mem[R[rs] + SignExt[imm16]] Example: lw rt, rs, imm16**

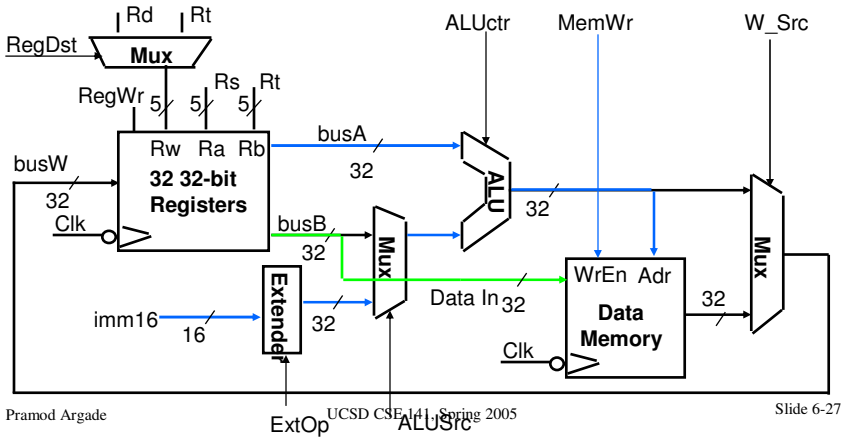




CPU: Store Operations

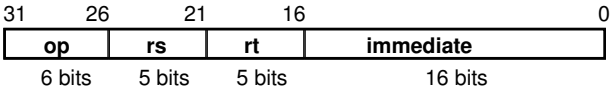
- Mem[R[rs] + SignExt[imm16] ← R[rt]] Example: sw rt, rs, imm16

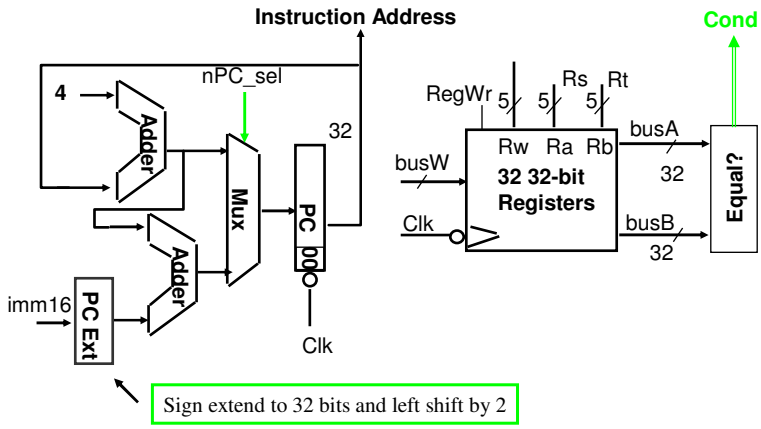




CPU: Datapath for Branching

- beq rs, rt, imm16 Datapath generates condition (equal)



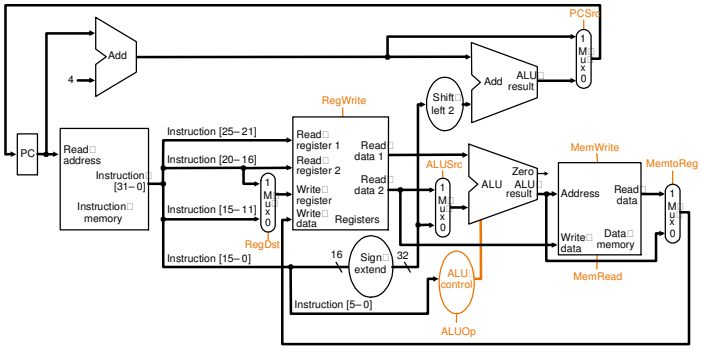


CPU: Binary arithmetic for PC

- In theory, the PC is a 32-bit byte address into the instruction memory:
  - Sequential operation: PC<31:0> = PC<31:0> + 4
  - Branch operation: PC<31:0> = PC<31:0> + 4 + SignExt[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
- In other words:
  - 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<31:2>:
  - Sequential operation: PC<31:2> = PC<31:2> + 1
  - Branch operation: PC<31:2> = PC<31:2> + 1 + SignExt[Imm16]
  - In either case: Instruction Memory Address = PC<31:2> concat “00”

Single Cycle Implementation

- Putting it all together
  - Supports Arithmetic, Load/Store and Branch instructions



Announcements

- **Reading Assignment**
  - Chapter 3. Arithmetic for Computers
    Sections B6, In More Depth 3.23, 3.4 - 3.8
  - Chapter 5. The Processor: Datapath and Control
    Sections 5.1 - 5.3, Appendix B7 - B11
- **Homework 2: Due Mon., April 18 in class**
  3.13, 3.18, 3.19, 3.27, 3.30, 3.35, 3.38
  Multiply -11x-10 using Booth's algorithm and 5-bit 2-complement representation of multiplicand and multiplier.
  5.1,
- **Quiz**
  When: Mon., April 18th, First 10 minutes of the class
  Topic: Arithmetic for Computers, Chapter 3 **Need:** Paper, pen