

# Single Cycle Datapath Design

Lecturer: Dr. Hui Annie Guo  
huig@cse.unsw.edu.au  
K17-501F (ext. 57136)

## Overview

- Steps of processor design
- Single-cycle datapath

## Typical Steps of Processor Design

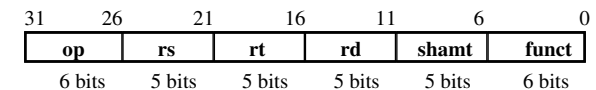
1. Analyse instruction set datapath requirements
  - the meaning of each instruction is given by the register transfers
  - datapath must support each register transfer
2. Select set of datapath components and establish clocking methodology
3. Assemble datapath to meet the requirements
4. Analyse implementation of each instruction to determine a set of control signals that affect the register transfer.
5. Assemble the control logic

## The MIPS-lite Subset

- We demonstrate processor design for the following “reduced” set of instructions

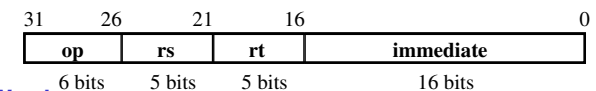
- ADD and SUB

- addU rd, rs, rt
- subU rd, rs, rt



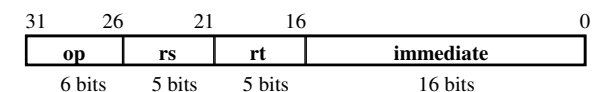
- OR Immediate:

- ori rt, rs, imm16



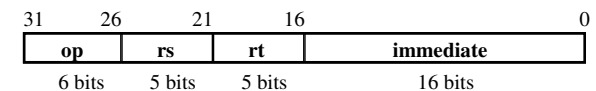
- LOAD and STORE Word

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



- BRANCH:

- beq rs, rt, imm16



## Step 1a: RTL Specifications

- **Register Level Language (RTL) used to describe the execution of instructions**

- All instructions start by fetching the instruction

$MEM[PC] = op | rs | rt | rd | shamt | funct$  or  
 $op | rs | rt | Imm16$

- Then different operations

### inst Register Transfers

**ADDU**  $R[rd] \leftarrow R[rs] + R[rt]; PC \leftarrow PC + 4;$   
**SUBU**  $R[rd] \leftarrow R[rs] - R[rt]; PC \leftarrow PC + 4;$   
**ORi**  $R[rt] \leftarrow R[rs] \vee zero\_ext(Imm16); PC \leftarrow PC + 4;$   
**LOAD**  $R[rt] \leftarrow MEM[ R[rs] + sign\_ext(Imm16) ]; PC \leftarrow PC + 4;$   
**STORE**  $MEM[ R[rs] + sign\_ext(Imm16) ] \leftarrow R[rt]; PC \leftarrow PC + 4;$   
**BEQ** if (  $R[rs] == R[rt]$  ) then  $PC \leftarrow PC + 4 + sign\_ext(Imm16) || 00$   
 else  $PC \leftarrow PC + 4;$

COMP3211/9211

2011S1 wk1\_2 P5

## Step 1b: Requirements of the Instruction Set

- **Memory**
  - instruction & data
- **Registers (let's say 32 x 32)**
  - read RS
  - read RT
  - Write RT or RD
- **PC**
- **Extender**
- **Add/Sub/Or**
  - register +/- register
  - Register xor extended immediate
- **Add**
  - $PC + 4$
  - $PC + 4 + extended\ immediate$

COMP3211/9211

2011S1 wk1\_2 P6

## Step 2: Components of the Datapath

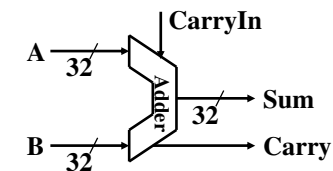
- **Combinational Elements**
- **Storage Elements**
  - Clocking methodology

COMP3211/9211

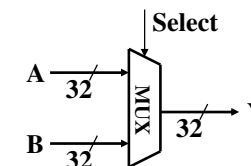
2011S1 wk1\_2 P7

## Combinational Logic Elements (Building Blocks)

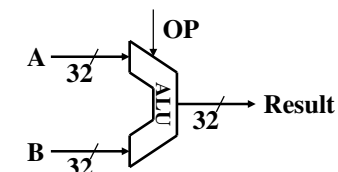
- **Adder**



- **MUX**



- **ALU**



COMP3211/9211

2011S1 wk1\_2 P8

## Storage Elements: Register (Building Block)

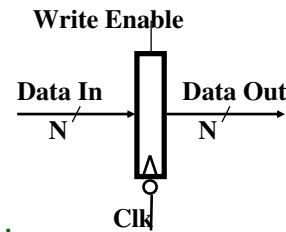
- **Register**

- Similar to the D Flip Flop except

- N-bit input and output
- Write Enable input

- Write Enable:

- negated (0): Data Out will not change
- asserted (1): Data Out will become Data In



COMP3211/9211

2011S1 wk1\_2\_P9

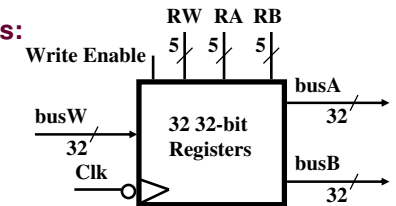
## Storage Elements: Register File

- **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 (number) selects the register to put on busA (data)
- RB (number) selects the register to put on busB (data)
- RW (number) selects the register to be written via busW (data) when Write Enable is 1

- **Clock input (CLK)**

- The CLK input is used for write operations
- During read operation, behaves like a combinational logic block:
  - RA or RB valid  $\Rightarrow$  busA or busB valid after “access time.”

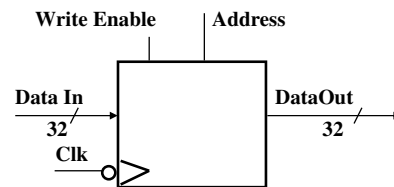
COMP3211/9211

2011S1 wk1\_2\_P10

## Storage Element: Memory

- **Memory (idealized)**

- 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

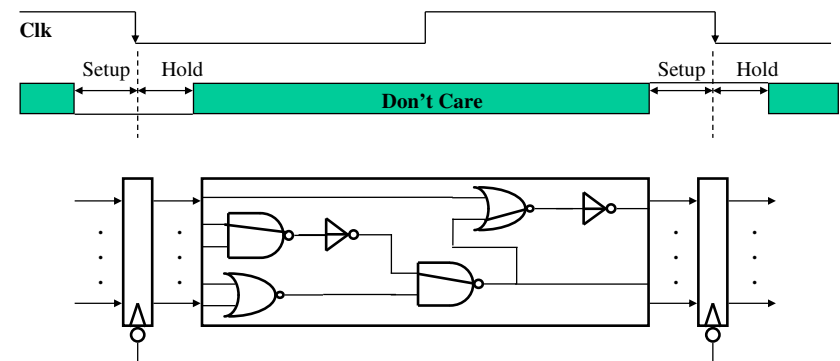
- **Clock input (CLK)**

- The CLK input is often only used for write operations
- During read operation, behaves like a combinational logic block:
  - Address valid  $\Rightarrow$  Data Out valid after “access time.”

COMP3211/9211

2011S1 wk1\_2\_P11

## Clocking Methodology



All storage elements are clocked by the same clock edge.

We therefore need to ensure that the following 2 conditions are met:

1. Cycle Time  $\geq$  CLK-to-Q + Longest Delay Path + Setup + Clock Skew
  - Clock skew = difference in arrival time of clock edges
2. (CLK-to-Q + Shortest Delay Path - Clock Skew)  $>$  Hold Time

COMP3211/9211

2011S1 wk1\_2\_P12

### Step 3

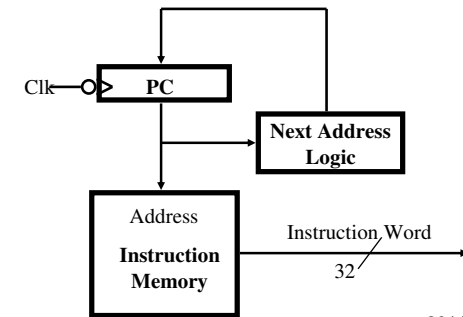
- Register Transfer Requirements  
→ Datapath Assembly
- Instruction Fetch
- Read Operands and Execute Operation

COMP3211/9211

2011S1 wk1\_2 P13

### 3a: Overview of the Instruction Fetch Unit

- Common RTL operations:
  - At start of cycle, fetch the instruction:  $mem[PC]$
  - At end of cycle, update the program counter:
    - Sequential Code:  $PC \leftarrow PC + 4$
    - Branch and Jump:  $PC \leftarrow$  “something else”

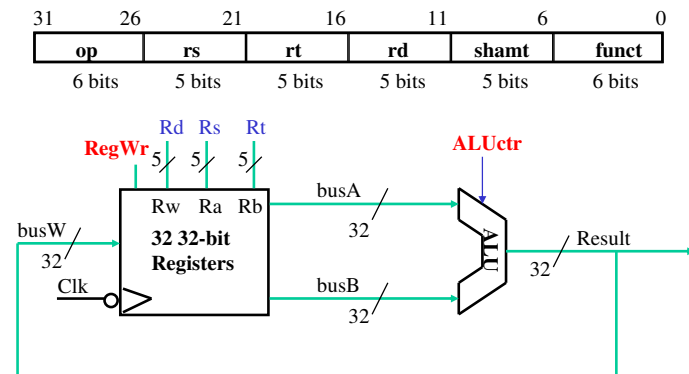


COMP3211/9211

2011S1 wk1\_2 P14

### 3b: Add & Subtract

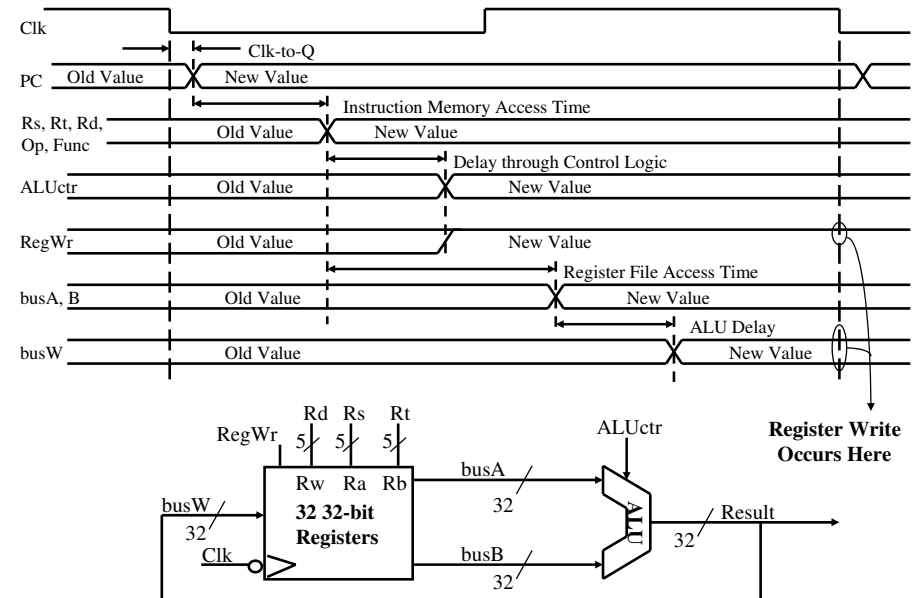
- $R[rd] \leftarrow R[rs] \text{ op } R[rt]$  *Example: addU rd, rs, rt*
- Ra, Rb, and Rw come from instruction's rs, rt, and rd fields
  - ALUctr and RegWr: the control logic after decoding the instruction



COMP3211/9211

2011S1 wk1\_2 P15

### Register-Register Timing

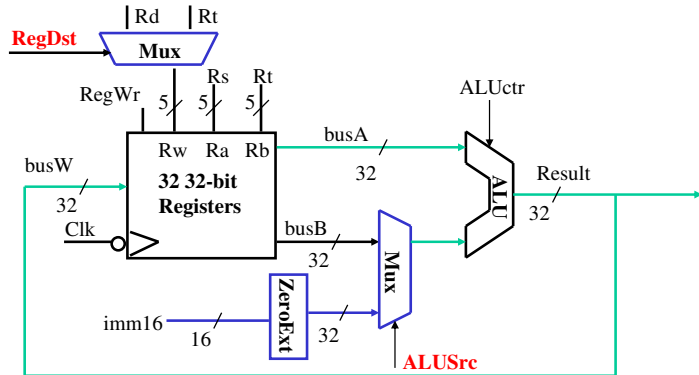
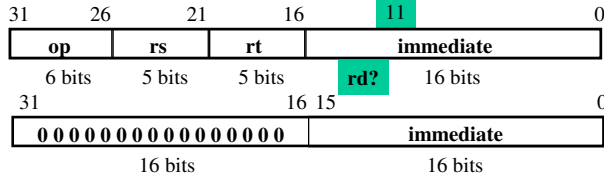


COMP3211/9211

2011S1 wk1\_2 P16

### 3c: Logical Operations with Immediate

$$R[rt] \leftarrow R[rs] \text{ op ZeroExt}[imm16]$$

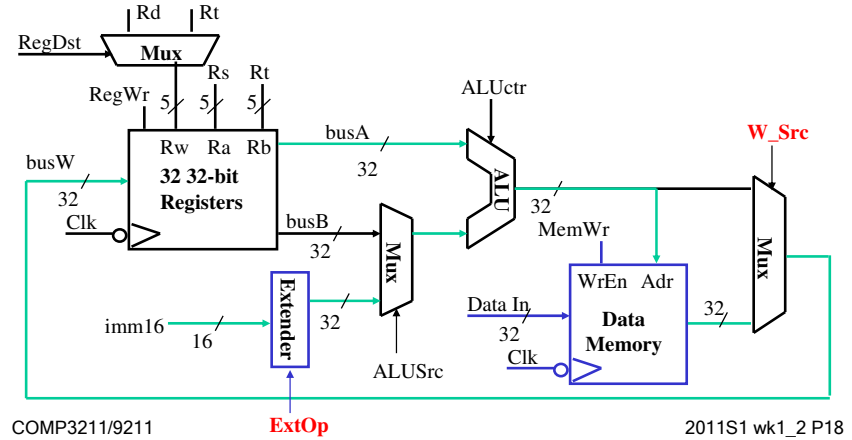
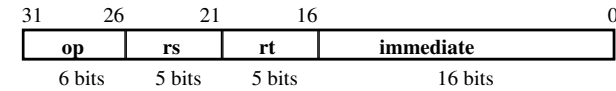


COMP3211/9211

2011S1 wk1\_2\_P17

### 3d: Load Operations

$$R[rt] \leftarrow \text{Mem}[R[rs] + \text{SignExt}[imm16]] \quad \text{Example: } lw \text{ } rt, rs, imm16$$

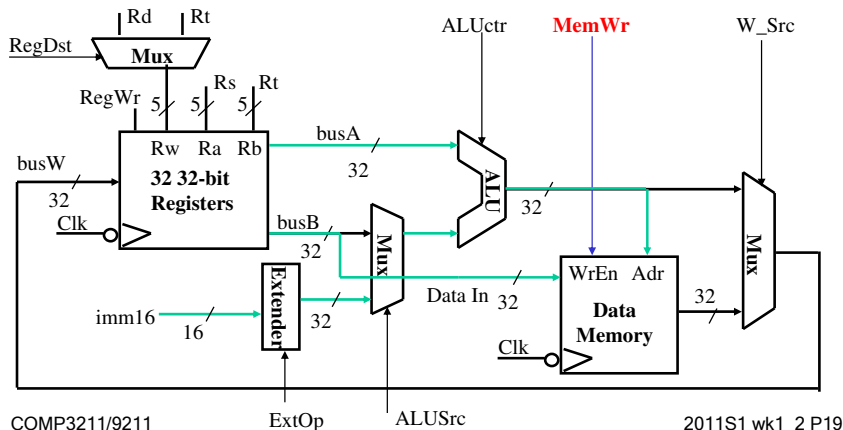
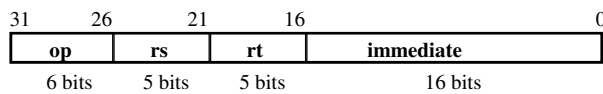


COMP3211/9211

2011S1 wk1\_2\_P18

### 3e: Store Operations

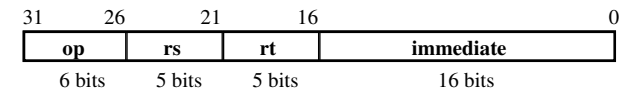
$$\text{Mem}[R[rs] + \text{SignExt}[imm16]] \leftarrow R[rt] \quad \text{Example: } sw \text{ } rt, rs, imm16$$



COMP3211/9211

2011S1 wk1\_2\_P19

### 3f: The Branch Instruction



$$beq \text{ } rs, rt, imm16$$

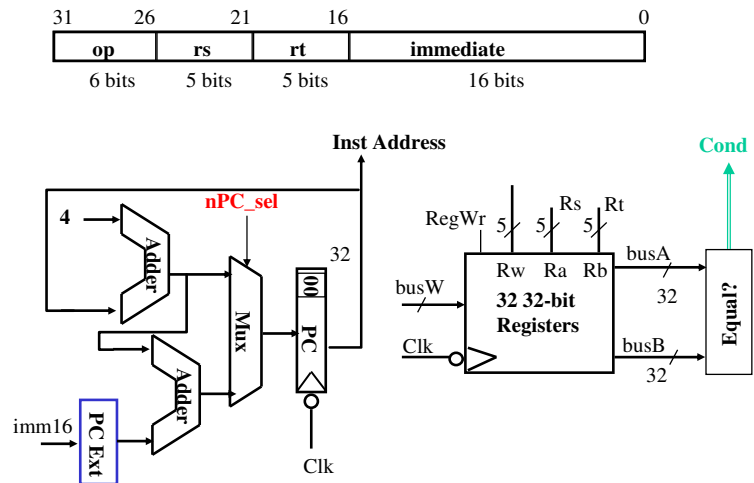
- mem[PC] Fetch the instruction from memory
- Equal  $\leftarrow R[rs] == R[rt]$  Calculate the branch condition
- if (COND eq 1) Calculate the next instruction's address
  - $PC \leftarrow PC + 4 + (\text{SignExt}(imm16) \times 4)$
- else
- $PC \leftarrow PC + 4$

COMP3211/9211

2011S1 wk1\_2\_P20

# Datapath for Branch Operations

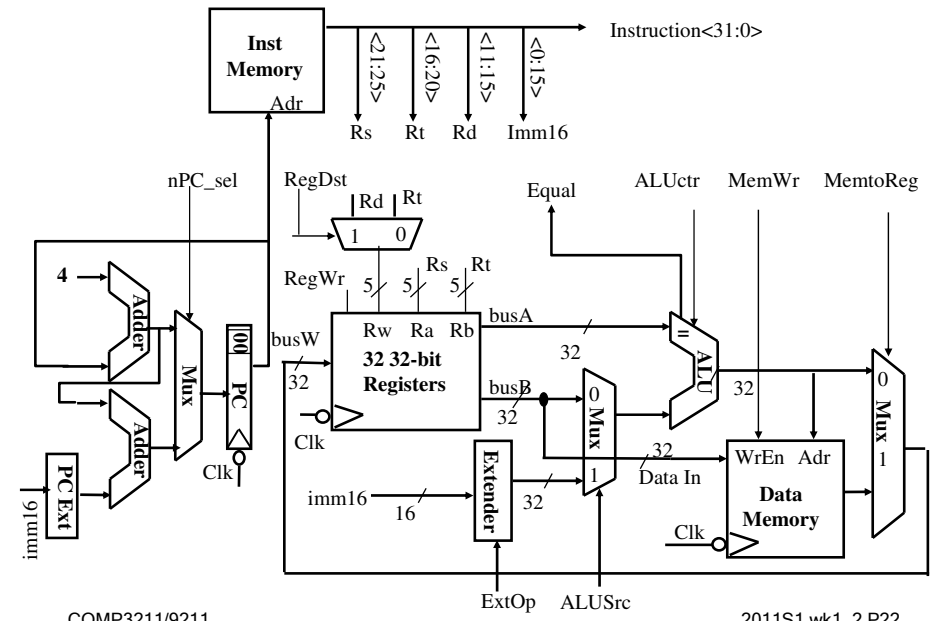
**beq rs, rt, imm16**      Datapath generates condition (equal)



COMP3211/9211

2011S1 wk1\_2\_P21

# Putting it All Together: A Single Cycle Datapath



COMP3211/9211

2011S1 wk1\_2\_P22