

## Levels in Processor Design

- **Circuit design**
  - Keywords: transistors, wires etc. Results in gates, flip-flops etc.
- **Logical design**
  - Putting gates (AND, NAND, ...) and flip-flops together to build basic blocks such as registers, ALU's etc (cf. CSE 370)
- **Register transfer**
  - Describes execution of instructions by showing data flow between the basic blocks
- **Processor description (the ISA)**
- **System description**
  - Includes memory hierarchy, I/O, multiprocessing etc

4/16/2004

CSE378 Single cycle implementation.

1

## Register transfer level

- Two types of components (cf. CSE 370)
  - *Combinational*: the output is a function of the input (e.g., adder)
  - *Sequential*: state is remembered (e.g., register)

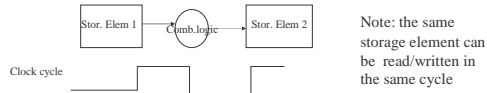
4/16/2004

CSE378 Single cycle implementation.

2

## Synchronous design

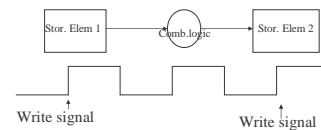
- Use of a periodic clock
  - *edge-triggered* clocking determines when signals can be read and when the output of circuits is stable
  - Values in storage elements can be updated only at clock edges
  - Clock tells when events can occur, e.g., when signals sent by control unit are obeyed in the ALU



4/16/2004

CSE378 Single cycle implementation.

3



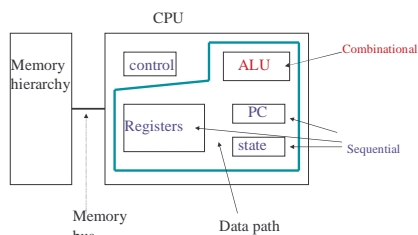
Logic may need several cycles to propagate values  
True in designs today with very high clock frequency

4/16/2004

CSE378 Single cycle implementation.

4

## Processor design: data path and control unit



4/16/2004

CSE378 Single cycle implementation.

5

## Processor design

- **Data path**
  - How does data flow between various basic blocks
  - What operations can be performed when data flows
  - What can be done in one clock cycle
- **Control unit**
  - Sends signals to data path elements
  - Tells what data to move, where to move it, what operations are to be performed
- **Memory hierarchy**
  - Holds program and data

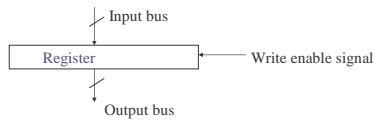
4/16/2004

CSE378 Single cycle implementation.

6

## Data path basic building blocks. Storage elements

- Basic building block (at the RT level) is a register
- In our mini-MIPS implementation registers will be 32-bits
- A register can be read or written



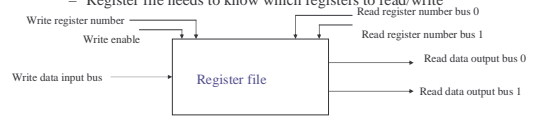
4/16/2004

CSE378 Single cycle implementation.

7

## Register file

- Array of registers (32 for the integer registers in MIPS)
- ISA tells us that we should be able to:
  - read 2 registers, write one register in a given instruction (at this point we want one instruction per cycle)
  - Register file needs to know which registers to read/write



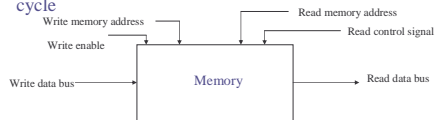
4/16/2004

CSE378 Single cycle implementation.

8

## Memory

- Conceptually, like register file but much larger
- Can only read one location or write to one location per cycle



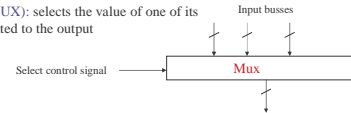
4/16/2004

CSE378 Single cycle implementation.

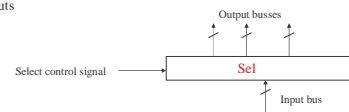
9

## Combinational elements

Multiplexor (MUX): selects the value of one of its inputs to be routed to the output



Demultiplexor (deMUX or SEL): routes its input to one of its outputs



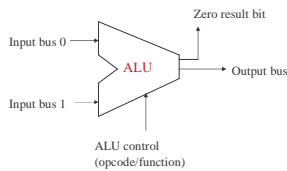
4/16/2004

CSE378 Single cycle implementation.

10

## Arithmetic and Logic Unit (ALU - combinational)

- Computes (arithmetic or logical operation) output from its two inputs

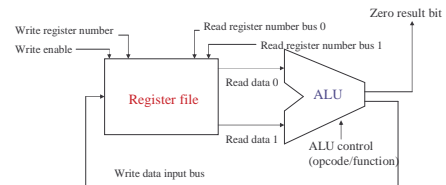


4/16/2004

CSE378 Single cycle implementation.

11

## Putting basic blocks together (skeleton of data path for arith/logical operations)

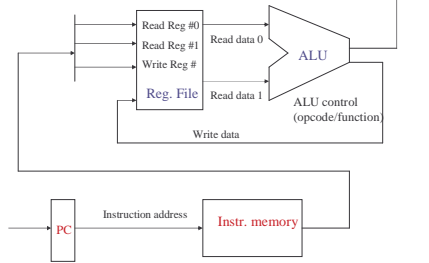


4/16/2004

CSE378 Single cycle implementation.

12

### Introducing instruction fetch

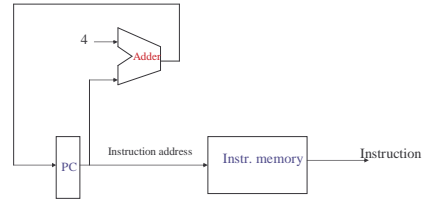


4/16/2004

CSE378 Single cycle implementation.

13

### PC has to be incremented (assume no branch)

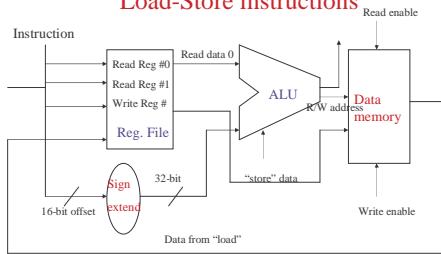


4/16/2004

CSE378 Single cycle implementation.

14

### Load-Store instructions

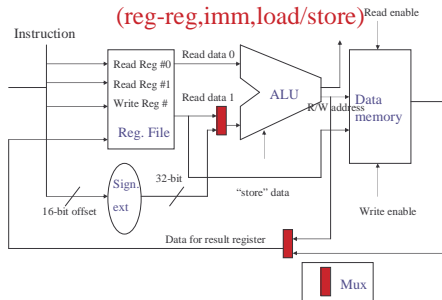


4/16/2004

CSE378 Single cycle implementation.

15

### Data path for straight code (reg-reg,imm,load/store)

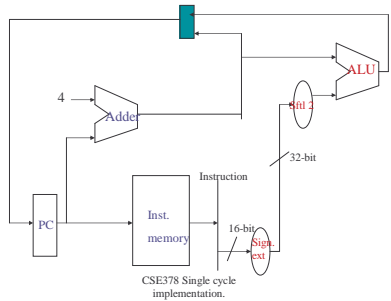


4/16/2004

CSE378 Single cycle implementation.

16

### Branch data path



4/16/2004

CSE378 Single cycle implementation.

17