Datapath Design

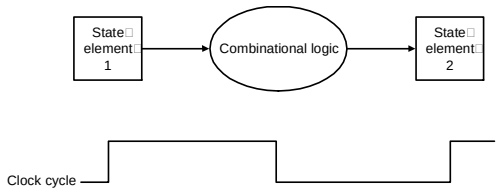
Chapter 5 P & H

Introduction

- Designing an implementation which contains subset of core MIPS instruction set:
  - Memory reference instructions (*lw* & *sw*)
  - Arithmetic-logic instructions (*add*, *sub*, *and*, *or* and *slt*)
  - Branch and jump instructions (*beq*, *j*)

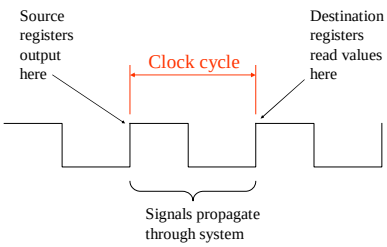
Processor Design

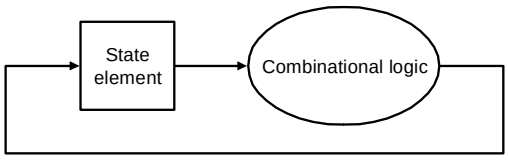
- Clocking Methodology
  - Defines when signals can be read and written
  - Going to assume an edge triggered clocking methodology



Control Step

- Each control step takes one clock cycle

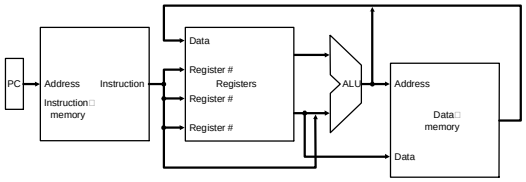




Overview of Implementation

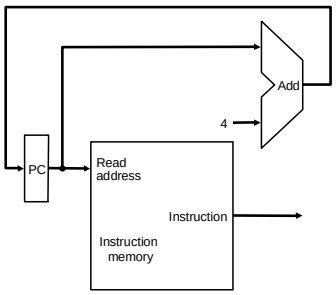
- Consider execution of an instruction
- First two steps identical
  - Use PC to fetch an instruction from memory
  - Read 1 or 2 registers as specified in instruction
- Rest of steps dependent on instruction class
  - Most will use ALU
  - Many write value back to register file

Overview of Implementation



- Will start with simple single cycle implementation
- Implementation will comprise datapath plus control

Instruction Fetch

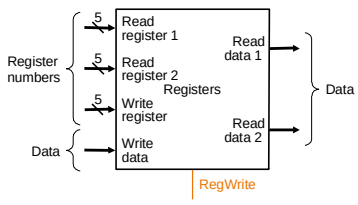


R-type Instructions

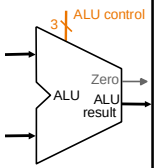
OP (6 bits)	RS (5 bits)	RT (5 bits)	RD (5 bits)	SHAMT (5 bits)	FUNC (6 bits)

- All R-type instructions:
  - Read two registers
  - Perform some ALU operation
    - add, sub, slt, and, or, etc.
  - Write the result back to the register file

Components

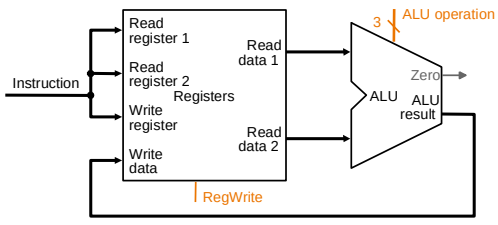


a. Registers



b. ALU

Datapath for R-Type Instructions



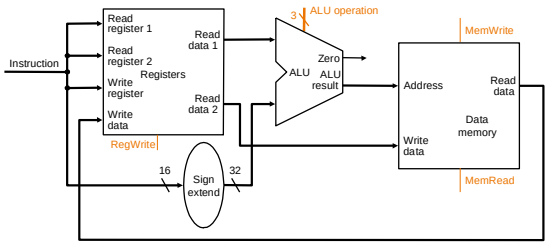
lw and sw Instructions

- `lw $7, offset($8)` or `sw $7, offset($8)`

OP (6 bits)	RS (5 bits)	RT (5 bits)	IMMEDIATE (16 bits)

- These instructions
  - Compute the memory address (16 bit signed offset + base register)
  - If sw then value must also be loaded from reg file
  - If lw the value read from memory must be stored to reg file
  - rs contains base, rt contains src/dst register
- Need to sign-extend offset to 32-bit value
- OP, RS, RT are same format/place as w/ R-type
  - Simplicity favours regularity

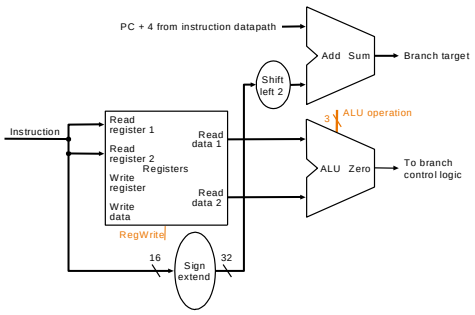
Datapath for lw and sw



beq Instruction

```
• beq $7, $8, offset
• if ($7 == $8) then
     pc <= pc + 4 + (offset << 2)
  else
     pc <= pc + 4
```

beq datapath



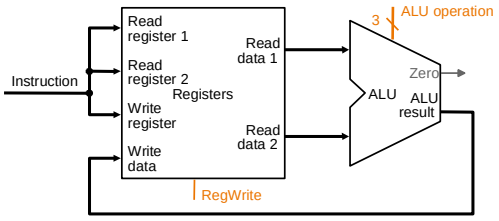
Simple Implementation Scheme

- All instructions execute in single clock cycle:
  - No data path resource used more than once per clock cycle
  - Components of different instruction classes may be shared if no conflicts occur
    - May require multiple connections to same input
    - Multiplexer used to select appropriate input

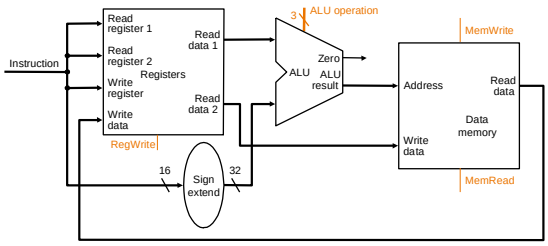
Combining datapath for R-type and memory instructions

- Datapaths very similar
- Two main differences
  - Second input to ALU is a register (for R-type) or sign-extended lower half of instruction (lw and sw)
  - Value stored in dest register comes from ALU (R-type) or memory (lw)

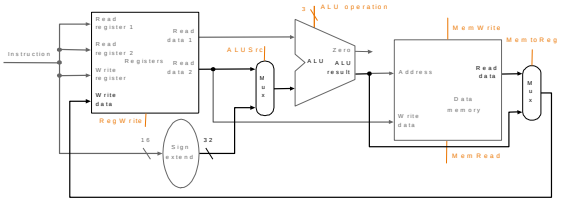
Datapath for R-Type Instructions



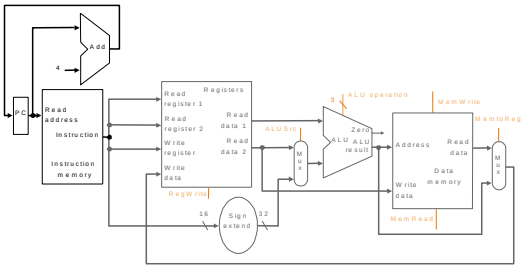
Datapath for lw and sw



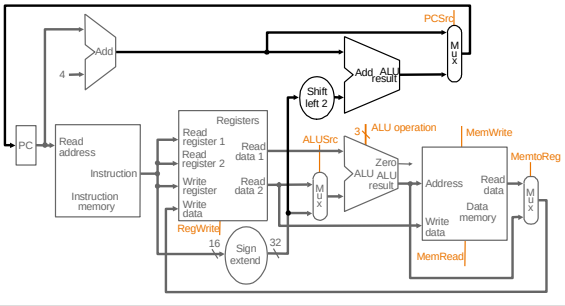
Combined Datapath



Adding Datapath for instruction fetch



Add in beq components



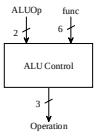
ALU Control

ALU control input	Function
000	AND
001	OR
010	ADD
110	Subtract
111	Set on less than
ALUOp	**operation**
00	Add
01	Subtract
10	Use func field

For R-Type instructions need to perform operation dependent of function field

For load instructions use ALU to compute memory address by addition

For branch instructions ALU used for subtraction



Truth Table for ALU control

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

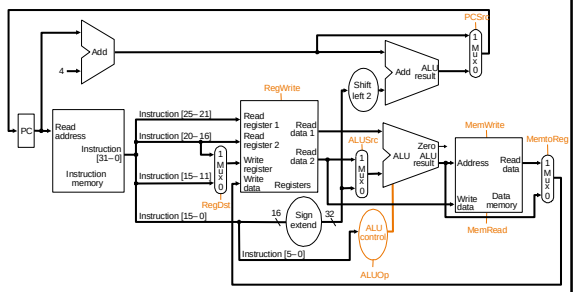
- Note multiple levels of control
  - Main control unit generates ALUOp
  - ALU control generates operation

Main Control Unit Design

0	RS	RT	RD	SHAMT	FUNC
35 or 43	RS	RT	IMMEDIATE		
4	RS	RT	IMMEDIATE		

- Opcode fields always bits 31 – 26 (Op[5-0])
- Two regs to be read are always rs and rt
- Base reg for loads and stores always rs
- 16 bit offset always bits 15 – 0
- Destination register is in one of two places
  - r-type instructions RD
  - lw it is RT

Datapath



Datapath with Control Unit