Multi-Cycle CPU
Multi-Cycle CPU

- Combine Functional Units
  - Reuse for different phases of instructions
  - One ALU for
    - PC increment
    - Branch target computation
    - Address computation for memory access
    - R-Type instruction execution
  - One memory unit for both instructions and data

- Multiple but Shorter Clock Cycles
  - Different instructions take different number of cycles
  - Average CPI times cycle time gives better performance
Branch Completion (Branch Taken)
Instruction Fetch after Branch Completion
R-Type Instruction Execution
Instruction Fetch After R-Type
Memory Access for lw
Write Back for lw
Instruction Fetch after lw

ALU Ctrl

ALU Out

Zero

ALU

RegWrite

Mem

Mem

Write

IRWrite

Mem

Read

Write Data

PCWrite

To Control

31-26

Read/Write

Addr

Mem Data

Write Data

Reg

Sign

Ext

Shift

Left

2

4

Instruction Fetch after lw

ADD
Control Lines

PCWriteCond  Write PC conditionally on branch
PCWrite      Write PC for increment or jump
PCSrc        Select source for writing to PC

IorD         Select address for memory read/write
MemRead      Read from memory (instruction or data)
MemWrite     Write to memory (store word)

IRWrite      Write to Instruction Register
Control Lines

**MemtoReg**  
Select memory or ALUOut to write to register

**RegDst**  
Select field to select destination register

**RegWrite**  
Write to selected register

**ALUSrcA**  
Select source for upper ALU input

**ALUSrcB**  
Select source for lower ALU input
Clock Cycles

1. Instruction Fetch, PC Increment
2. Instruction Decode, Register Fetch, Branch Target Computation
3. R-type Execution or Memory Address Computation or Branch Completion
4. R-type Write Back or Memory Access
5. Memory write back
Control Overview

Start

Instruction Fetch → Instruction Decode

Memory Access → R-type → Cond Branch → Uncond Branch

Cond Branch
Control: Finite State Machine

Start

Instruction Fetch 0

MemRead
IorD = 0
IRWrite
ALUSrcA = 0
ALUSrcB = 01
ALU - Add
PCSrc = 00
PCWrite

Instruction Decode 1

ALUSrcA = 0
ALUSrcB = 11
ALU - Add

Idur or stur

R-type cbz

Memory Access FSM

R-type FSM

Cond Branch FSM

Branch FSM

R-type cbz
Memory Access FSM

From State 1
lw or sw

ALUSrcA = 1
ALUSrcB = 10
ALU - Add

Memory Address Comp 2

Memory Access lw 3

MemRead IorD = 1

Memory Access sw 5

MemWrite IorD = 1

Write Back 4

RegWrite MemToReg = 1
RegDst = 0

To State 0
R-type FSM

Execution 6
ALUSrcA = 1
ALUSrcB = 00
ALU- function

Write Back 7
RegDst = 1
RegWrite
MemtoReg = 0

From State 1
R-type

To State 0
Conditional Branch FSM

Branch Completion 8

ALUSrcA = 1
ALUSrcB = 00
ALU - Pass
PCWriteCond
PCSrc = 1

From State 1
cbz

To State 0
(Unconditional) Branch FSM

From State 1

cbz

To State 0

ALUSrcA = 1
ALUSrcB = 00
PCWrite
PCSrc = 1

Branch Completion 8
Complete Finite State Machine

Start

Instruction Fetch 0

MemRead
IorD = 0
IRWrite
ALUSrcA = 0
ALUSrcB = 01
ALU - Add
PCSrc = 00
PCWrite

ALUSrcA = 0
ALUSrcB = 11
ALU - Add

Instruction Decode 1

Idur or stur

R-type

cbz

Unconditional Branch Completion 9

ALUSrcA = 1
ALUSrcB = 00
PCWrite
PCSrc = 1

ALUSrcA = 1
ALUSrcB = 00
ALU - function

Execution 6

Conditional Branch Completion 8

ALUSrcA = 1
ALUSrcB = 00
ALU - Pass
PCWriteCond
PCSrc = 1

RegDst = 1
RegWrite
MemtoReg = 0

Write Back 7

RegWrite
MemToReg = 1
RegDst = 0

MemWrite
IorD = 1

Memory Access
ldur 3

MemRead
IorD = 1

Memory Access
Idur 3

Write Back 4

ALUSrcA = 1
ALUSrcB = 10
ALU - Add

ALUSrcA = 1
ALUSrcB = 10
ALU - Add

Memory Address
Comp 2

Write
MemToReg = 1
RegDst = 0
FSM Implementation as PLA

Opcode from IR

Clock

State Register

PLA

PCWrite
PCWriteCond
PCSrc
IorD
MemRead
MemWrite
IRWrite
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
MemtoReg