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
Instruction Decode/Register Fetch/Branch Target
Instruction Fetch after Branch Completion
R-Type Instruction Execution
Instruction Fetch After R-Type
Instruction Decode/Register Fetch/Branch Target

Memory IR Registers

ALU Out

Zero

PCWrite

IRWrite

Mem Read Mem Write

Memory

IR

Mem Data Reg

RegWrite

Read Reg 1

Read Reg 2

Write Reg

Write Data

Read Data 1

Read Data 2

ALU

D

E

ALU Op

Add

ALU Ctrl

Sign Ext

Shift Left 2

Res

Addr
Memory Access for lw
Write Back for lw
Instruction Fetch after `lw`
<table>
<thead>
<tr>
<th>Control Lines</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCWriteCond</td>
<td>Write PC conditionally on branch</td>
</tr>
<tr>
<td>PCWrite</td>
<td>Write PC for increment or jump</td>
</tr>
<tr>
<td>PCSrc</td>
<td>Select source for writing to PC</td>
</tr>
<tr>
<td>IorD</td>
<td>Select address for memory read/write</td>
</tr>
<tr>
<td>MemRead</td>
<td>Read from memory (instruction or data)</td>
</tr>
<tr>
<td>MemWrite</td>
<td>Write to memory (store word)</td>
</tr>
<tr>
<td>IRWrite</td>
<td>Write to Instruction Register</td>
</tr>
</tbody>
</table>
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
ALUOp     Select ALU operation or set to function code:  00 = Add, O1 = Subtract,
           10 = use funct field (bits 0-5)
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
Control: Finite State Machine

Instruction Fetch 0
MemRead
IorD = 0
IRWrite
ALUSrcA = 0
ALUSrcB = 01
ALUOp = 00
PCSrc = 00
PCWrite

Instructions

Instruction Decode 1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00

Branch FSM

Memory Access FSM

R-type FSM

Cond Branch FSM

Start

ldur or stur

R-type

cbz

b
Memory Access FSM

Memory Address Comp 2

ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00

From State 1
lw or sw

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
ALUOp = 10

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
ALUOp = 01
PCWriteCond
PCSrc = 1

From State 1
cbz

To State 0
(Unconditional) Branch FSM

Branch Completion
8

From State 1
cbz

To State 0

ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWrite
PCSrc = 1
Complete Finite State Machine

- **Start**
  - **Instruction Fetch 0**
    - MemRead
    - IorD = 0
    - IRWrite
    - ALUSrcA = 0
    - ALUSrcB = 01
    - ALUOp = 00
    - PCSrc = 00
    - PCWrite
  - **MemRead**
    - IorD = 0
    - IRWrite
    - ALUSrcA = 0
    - ALUSrcB = 01
    - ALUOp = 00
    - PCSrc = 00
    - PCWrite
  - **Instruction Decode 1**
    - ALUSrcA = 0
    - ALUSrcB = 11
    - ALUOp = 00
  - **Instruction Decode 1**
    - ALUSrcA = 0
    - ALUSrcB = 11
    - ALUOp = 00
  - **Idur or stur**
  - **R-type**
    - cbz
    - b
    - Unconditional Branch Completion 9
  - **Execution 6**
    - ALUSrcA = 1
    - ALUSrcB = 00
    - ALUOp = 10
  - **Conditional Branch Completion 8**
    - ALUSrcA = 1
    - ALUSrcB = 00
    - ALUOp = 01
    - PCWriteCond
    - PCSrc = 1
  - **Write Back 7**
    - RegDst = 1
    - RegWrite
    - MemtoReg = 0
  - **Write Back 7**
    - RegDst = 1
    - RegWrite
    - MemtoReg = 0
  - **Unconditional Branch Completion 9**
    - ALUSrcA = 1
    - ALUSrcB = 00
    - ALUOp = 01
    - PCWrite
    - PCSrc = 1
  - **Write Back 4**
    - RegWrite
    - MemToReg = 1
    - RegDst = 0
  - **Write Back 4**
    - RegWrite
    - MemToReg = 1
    - RegDst = 0
  - **Memory Access Idur 3**
    - MemRead
    - IorD = 1
    - IRWrite
    - ALUSrcA = 0
    - ALUSrcB = 01
    - ALUOp = 00
    - PCSrc = 00
    - PCWrite
  - **Memory Access Idur 3**
    - MemRead
    - IorD = 1
    - IRWrite
    - ALUSrcA = 0
    - ALUSrcB = 01
    - ALUOp = 00
    - PCSrc = 00
    - PCWrite
  - **Start**

---

**Detailed States**

- **Start**
  - **Instruction Fetch 0**
    - MemRead
    - IorD = 0
    - IRWrite
    - ALUSrcA = 0
    - ALUSrcB = 01
    - ALUOp = 00
    - PCSrc = 00
    - PCWrite
  - **MemRead**
    - IorD = 0
    - IRWrite
    - ALUSrcA = 0
    - ALUSrcB = 01
    - ALUOp = 00
    - PCSrc = 00
    - PCWrite
  - **Instruction Decode 1**
    - ALUSrcA = 0
    - ALUSrcB = 11
    - ALUOp = 00
  - **Idur or stur**
  - **R-type**
    - cbz
    - b
    - Unconditional Branch Completion 9
  - **Execution 6**
    - ALUSrcA = 1
    - ALUSrcB = 00
    - ALUOp = 10
  - **Conditional Branch Completion 8**
    - ALUSrcA = 1
    - ALUSrcB = 00
    - ALUOp = 01
    - PCWriteCond
    - PCSrc = 1
  - **Write Back 7**
    - RegDst = 1
    - RegWrite
    - MemtoReg = 0
  - **Write Back 7**
    - RegDst = 1
    - RegWrite
    - MemtoReg = 0
  - **Unconditional Branch Completion 9**
    - ALUSrcA = 1
    - ALUSrcB = 00
    - ALUOp = 01
    - PCWrite
    - PCSrc = 1
  - **Write Back 4**
    - RegWrite
    - MemToReg = 1
    - RegDst = 0
  - **Write Back 4**
    - RegWrite
    - MemToReg = 1
    - RegDst = 0
  - **Memory Access Idur 3**
    - MemRead
    - IorD = 1
    - IRWrite
    - ALUSrcA = 0
    - ALUSrcB = 01
    - ALUOp = 00
    - PCSrc = 00
    - PCWrite
  - **Memory Access Idur 3**
    - MemRead
    - IorD = 1
    - IRWrite
    - ALUSrcA = 0
    - ALUSrcB = 01
    - ALUOp = 00
    - PCSrc = 00
    - PCWrite
FSM Implementation as PLA

- Opcode from IR
- Clock
- State Register
- PLA
  - PCWrite
  - PCWriteCond
  - PCSrc
  - IorD
  - MemRead
  - MemWrite
  - IRWrite
  - ALUOp
  - ALUSrcB
  - ALUSrcA
  - RegWrite
  - RegDst
  - MemtoReg