Single Cycle CPU Datapath and Control

**Single Cycle CPU Design**

Here we have a single cycle CPU diagram. Answer the following questions:

1. Name each component.
2. Name each datapath stage and explain its functionality.

|  |  |
| --- | --- |
| Stage | Functionality |
| Instruction fetch | Send an address to the instruction memory and then read the instruction. |
| Decode / register read | Generate the control signal values using the opcode & funct fields. Read the register values with the rs & rt fields. Sign / zero extend the immediate. |
| execute | Perform arithmetic / logical operations. |
| memory | Read from / write to the data memory. |
| Register write | Write back the ALU result / the memory load to the register file. |

1. Provide data inputs and control signals to the next PC logic.
2. Implement the next PC logic.

next PC logic

instruction memory

PC

ALU

data memory

register write

memory

execute

decode / register read

instruction fetch



Addr

Read

Data

Write

Data Write Enable

Read Write

Addr1 Data

Read

Data1

Read

Addr2

Write Addr

Read

Data2

Write

Enable

RF[rt]

ALUSrc ALUCtr

Addr

Read

Data

+4

Inst[25:21]

RF[rs]

A

1

0

Inst[20:16]

Out

0

1

0

0

B

Inst

[15:11]

1

1

Inst[15:0]

Sign / Zero

Extended

Inst[31:26]

Inst[5:0]

RegDst ExtOp

RegWr

MemWr

MemToReg

concat

<< 2

0

1

add

<< 2

register file

branch

jump

control unit

Single Cycle CPU Datapath and Control

**Single Cycle CPU Control Logic**

Fill out the values for the control signals from the previous CPU diagram.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instrs. | Control Signals | | | | | | | | |
| Jump | Branch | RegDst | ExtOp | ALUSrc | ALUCtr | MemWr | MemtoReg | RegWr |
| add | 0 | 0 | 1 | X | 0 | 0010 | 0 | 0 | 1 |
| ori | 0 | 0 | 0 | 0 | 1 | 0001 | 0 | 0 | 1 |
| lw | 0 | 0 | 0 | 1 | 1 | 0010 | 0 | 1 | 1 |
| sw | 0 | 0 | X | 1 | 1 | 0010 | 1 | X | 0 |
| beq | 0 | 1 | X | 1 | 0 | 0110 | 0 | X | 0 |
| j | 1 | 0 | X | X | X | XXXX | 0 | X | 0 |

This table shows the ALUCtr values for each operation of the ALU:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Operation | AND | OR | ADD | SUB | SLT | NOR |
| ALUCtr | 0000 | 0001 | 0010 | 0110 | 0111 | 1100 |

# Clocking Methodology

* The input signal to each state element must stabilize before each rising edge.
* Critical path: Longest delay path between state elements in the circuit.
* tclk ≥ tclk-to-q + tCL + tsetup, where tCL is the critical path in the combinational logic.
* If we place registers in the critical path, we can shorten the period by reducing the amount of logic between registers.

# Single Cycle CPU Performance Analysis

The delays of circuit elements are given as follows:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Element | Register clk-to-q | Register Setup | MUX | ALU | Mem Read | Mem Write | RegFile Read | RegFile Setup |
| Parameter | tclk-to-q | tsetup | tmux | tALU | tMEMread | tMEMwrite | tRFread | TRFsetup |
| Delay(ps) | 30 | 20 | 25 | 200 | 250 | 200 | 150 | 20 |

1. Give an instruction that exercises the critical path.

lw

1. What is the critical path in the single cycle CPU?

PC -> instruction memory -> register -> ALU -> data memory -> MUX -> register

1. What are the minimum clock cycle, tclk, and the maximum clock frequency, fclk? Assume the tclk-to-q > hold time.

tclk >= tsetup + tclk-to-q + tMEMread + tMUX + tRFread + tMUX + tALU + tMUX + tRFsetup

1. Why is a single cycle CPU inefficient?

Not all instructions exercise the critical path. It is not parallelized. Each component can be active concurrently

1. How can you improve its performance? What is the purpose of pipelining?

Multicycle paths. Pipelining: Put pipeline registers between two datapath stages. Reduce the clock time.