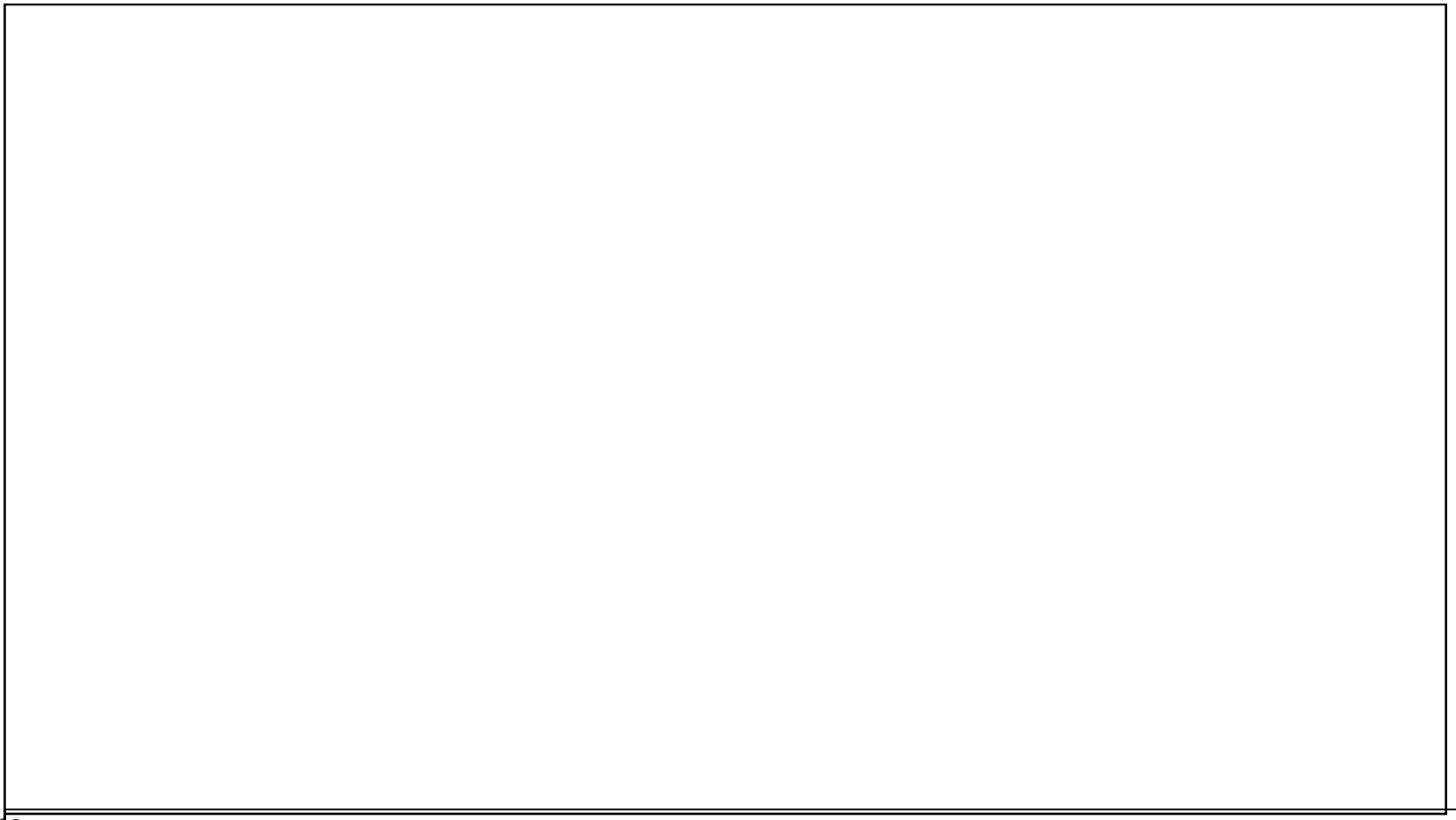
INSTRUCTION FETCH AND DECODE



3:12 / 3:12 1.0x

Download transcript .txt

Help

Show Discussion New Post

1. CHECK YOUR UNDERSTANDING (1/1 point)

Which of the following is an **incorrect** statement regarding the first state of Instruction Fetch?

- LD.PC = 1 in order to update the PC register with the incremented PC.
- All other LD signals are 0. ✔
- GatePC = 1 and all other Gate signals are 0.
- It doesn't matter if values are read from the Register File.

EXPLANATION

The PC is a loadable register that gets updated when LD.PC = 1. The update is necessary to change the PC to point to the

next instruction in Memory. The change will happen in the next rising edge of the clock, simultaneously with transitioning to the second Instruction Fetch state, but LD.PC must be 1 when that clock tick occurs for that load to happen.

LD.MAR = 1 in order to load the (pre-incremented) PC into the MAR register.

GatePC = 1 allows the contents of the PC register to be placed on the bus. Since this is a shared set of wires, only one of the drivers must drive the bus at one time. Therefore, all other Gate signals must be 0.

Since reading is non-destructive, it doesn't matter if we read the Register File. In fact, so long as power is being applied to the circuit, it is always being read in our design.

Help

Final Check | **Save** | **Hide Answer** *You have used 1 of 2 submissions*

Show Discussion

New Post

2. CHECK YOUR UNDERSTANDING (1/1 point)

Which of the following is an **incorrect** statement regarding the second state of Instruction Fetch?

- LD.MDR = 1 in order to update the MDR register with the instruction that is read from Memory.
- The FSM will remain in this state until R = 1.
- LD.MAR must remain at 1 so that the correct value of the address remains in the register. ✔
- It doesn't matter what operation is being performed by the ALU.

Help

EXPLANATION

The MDR is a loadable register that gets updated when LD.MDR = 1. The update is necessary to load the instruction that is read from Memory. The change will happen in the next rising edge of the clock, simultaneously with transitioning to the third Instruction Fetch state, but MD.MDR must be 1 when that clock tick occurs for that load to happen.

The R signal accounts for slower memories that cannot be read in a single cycle. The FSM needs to remain in the second state until the number of cycles required to read the Memory. When the instruction is ready at the output, the Memory asserts R = 1. Since the MAR was loaded with the correct address in the first state, it can become 0 in the second state and the register will retain its value. There is no need to repeatedly load MAR during this state. Since the ALU is not used in this state, the operation that it performs is irrelevant.

Hide Answer *You have used 2 of 2 submissions*

Show Discussion New Post

3. CHECK YOUR UNDERSTANDING (1/1 point)

Which of the following is an **incorrect** statement regarding the third state of Instruction Fetch?

Help

- LD.IR = 1 in order to update the IR register with the instruction that was read from Memory in the second state.
- The value of LD.CC does not matter since we are not updating the CC register. ✔
- The bus is used during this cycle.
- LD.PC must be 0 in order to retain the current value of the PC.

EXPLANATION

The IR is a loadable register that gets updated when LD.IR = 1. The update is necessary to load the instruction that is sent over the bus from MDR. The change will happen in the next rising edge of the clock, simultaneously with transitioning to the Decode state, but LD.IR must be 1 when that clock tick occurs for that load to happen.

LD.CC must be 0 in order to avoid inadvertently overwriting the CC register. If the current instruction is a branch, its action (taken or not taken) will depend on the value of the CC register from the last instruction that wrote the register file. That CC register value must be preserved in order for the branch to take the intended action.

The bus is used to transfer the value in MDR to IR.

The PC currently points to the next instruction in memory. Asserting LD.PC = 1 will change that value, causing the program to jump to a different location in memory. Thus, LD.PC must be 0.

Final Check | **Save** | **Hide Answer** | *You have used 1 of 2 submissions*

Show Discussion | New Post

Help

4. CHECK YOUR UNDERSTANDING (1 point possible)

Which of the following is an **incorrect** statement regarding the Instruction Decode state?

- LD.CC will be 1 for Branch instructions and 0 otherwise. ✔
- The BEN signal can be generated using three 2-input NAND gates followed by a 3-input NAND gate, without any NOT gates. ✖
- The bus is not used during this cycle.
- The next state that will be entered after this one is solely dependent on the value of the instruction opcode.

EXPLANATION

LD.CC = 1 only in the clock cycle when an instruction is updating the register file. Since a Branch instruction does not write a register, it never sets LD.CC = 1. LD.CC must be 0 during Instruction Decode in order to avoid inadvertently overwriting the CC register.

BEN can be implemented with three 2-input AND gates feeding a 3-input OR gate. From DeMorgan's Law, the AND and OR

gates can be replaced with NAND gates. The use of non-inverting values of IR[11:9] and the CC register obviates the need for NOT gates.

No bus transfers occur for Instruction Decode.

From the state diagram, the value of IR[15:12], the opcode, solely determines the next state.

Help

Hide Answer

You have used 2 of 2 submissions

Show Discussion

New Post

business, chemistry, computer science, economics, finance, electronics, engineering, food and nutrition, history, humanities, law, literature, math, medicine, music, philosophy, physics, science, statistics and more. EdX is a non-profit online initiative created by founding partners Harvard and MIT.

Help

Terms of Service and Honor Code

Privacy Policy (Revised 10/22/2014)

POWERED BY OPENedX

Contact

FAQ

edX Blog

Donate to edX

Jobs at edX

LinkedIn

Google+

Tumblr

Meetup

Reddit

Youtube