Computer Architecture

Fall, 2020

Week 9

2020.11.09

組別：_____ 簽名：_____

[group11]

1. The followings are the steps for how to design a processor. Please write down the steps in the correct order.

(A) Select set of datapath components and establish clocking methodology

(B) Analyze instruction set

(C) Assemble the control logic

(D) Analyze implementation of each instruction to determine setting of control points effecting register transfer

(E) Assemble datapath meeting the requirements

Ans: (B) -> (A) -> (E) -> (D) -> (C)

[group14] (對抗賽)

2. Which statements are true?

(A) In Combinational elements, we use the concept of edge-triggered to update the Clk changes from 0 to 1.

(B) Combinational logic transforms data during clock cycles.

(C) Write enable is a control signal that lets an input write to a register if certain conditions are met (asserted).

(D) Branch operation (beq/bne) calculates the branch condition by subtracting the register selected by the $rt field from the register selected by the $rs field.

(E) An edge-triggered methodology allows a state element to be read and written in a different clock cycle.

Answer1

(B), (C), (D), (E)

(A) State (sequential) elements

[group8]

3. 按 Instruction 執行步驟的先後次序排序

(1) Access data memory

(2) Instruction fetch

(3) Register access

ANS: (2)>(3)>(1)　　ch4_1 : ppt.p3

[group10] (對抗賽)

4. True or false for the following statement?

(a) We can update the value of sequential element at any time when control input is 1

(b) Register file always has 1 input bus and 1 output bus.

(c) memory needs read control signal to decide what time can read a value from memory.

(d) During read, register file behaves as a combinational circuit.

Ans:

(a) false, only updates on clock edge when write control input is 1

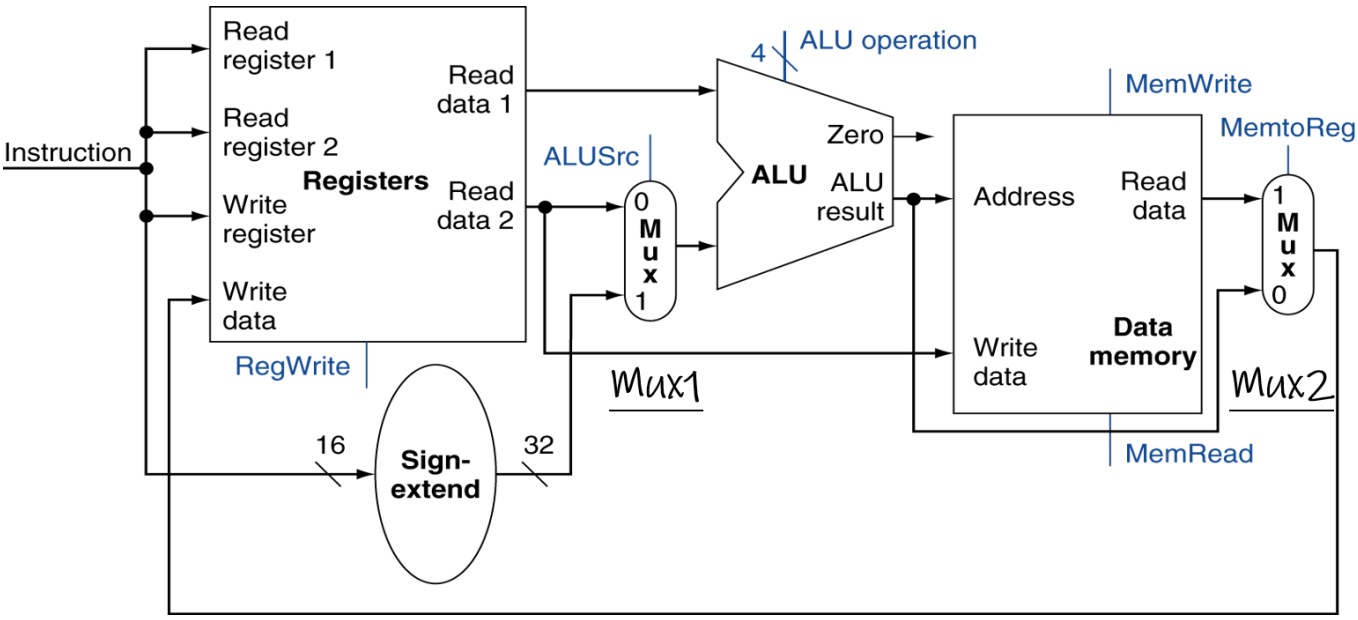
(b) false, register file can have 1 input bus and 2 output busses

(c) false, memory don’t need for read control.

(d) True

[group5] (對抗賽)

5.



(1) R-Type : Mux1=___ , Mux2=___

(2) Load/Store : Mux1=___ , Mux2=___

A :

(1) R-Type : Mux1= 0 , Mux2= 0

(2) Load/Store : Mux1= 1 , Mux2= 1

[group9] (對抗賽)

6. What is the difference between Combinational and Sequential elements, and which one will we choose to make register?

Ans:

1.Sequential can store data in a circuit. (meaning that output depends not only on the present value of its input signals but on the sequence of past inputs).

This is in contrast to combinational, whose output is a function of only the present input.

2.Sequential elements

[group3] (對抗賽)

7. Please find whether the followings are true or false, if false, correct it.

(A) All instructions start by fetching the instructions, read registers, then use ALU

(B) When we write `R[rt]<-MEM[R[rs]+sign_ext(imm16)]`, it means that we store data to memory.

(C) Combinational elements are used to store information and controlled by clock.

(D) Writing enable is more decisive than clock in storage element.

(E) The shortest delay determines clock period.

Ans：

(A) False, jump don’t use ALU

(B) False, it means that we load data to memory.

(C) False, state elements are used to store information and controlled by clock.

(D) True

(E) False, longest delay determine clock period.

[group12] (對抗賽)

8. 下列哪些選項正確，如果錯誤請說明原因

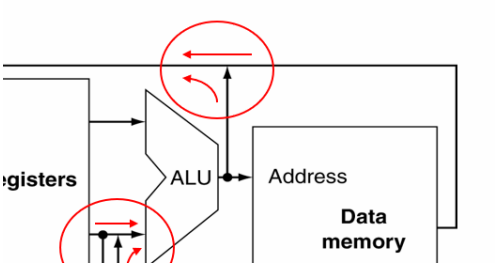
(A) When executing an instruction, before doing the next instruction, we should let Program Counter <- Program Counter + 4.

(B) Register is a sequential component.

(C) The CLK input is a factor ONLY during a write operation. During read, it behaves as a combinational circuit.

(D) Longest delay of combinational logic determines clock period.

(E) Consider the figure below, when two paths meet, we use multiplexers to select.



Ans：

(A)、(B)、(C)、(D)、(E)

[group13] (對抗賽)

9. Please describe the register transfer and the register type of the following instructions:

1. `add`
2. `lw`
3. `slt`
4. `addi`
5. `j`

Ans:

1. R-type: `R[rd] <- R[rs] + R[rt]; PC <- PC + 4;`
2. I-type: `R[rt] <- MEM[R[rs] + sign_ext(imm16)]; PC <- PC + 4;`
3. R-type: `if(R[rs] < R[rt]) then R[rd] <- sign_ext(+1), else R[rd] <- sign_ext(0); PC <- PC + 4;`
4. I-type: `R[rt] <- R[rs] + sign_ext(imm16); PC <- PC + 4;`
5. J-type: `PC <- PC + 4 + (PC[31:28] || address || 00);`