Computer Architecture

Fall, 2025
Week 7
2025.10.14

組別: _ _ _ _ _ **簽名:** _

[group 3]

1. **Question:**

   Let A and B be two 4-bit signed numbers represented in 2's complement, where A = 0110 and B = 0101.

   1. Perform A + B and determine if overflow occurs.
   2. Explain how overflow is detected using CarryIn and CarryOut of the most significant bit.

Ans:

1. A = 0110 (+6), B = 0101 (+5)

   A + B = 1011 (−5) → overflow occurred.

2. Because CarryIn(MSB)=1, CarryOut(MSB)=0, Overflow = 1 ⊕ 0 = 1 (overflow).

[group 4]

2. **Question:**

   For these 4-bit signed numbers, which of them are overflow?

   (a)0010+1010

   (b)1111+1000

   (c)0011+0111

   (d)1010+0100

Ans: b & c

[group 5]

3. **Question:**

   Please select the correct statements:

   A. We can detect overflow in ALU by using XOR gate.

B. We can implement NOR with AND gates and inverters in ALU.

C. ALUOP 0000 is the control signal of add

D. SLT subtracts the two inputs and directly uses the sign bit of (A − B) as the result

Ans:

A,B are correct

C is false, signal of add

D is false, need overflow detection (Sign_Bit ⊕ Overflow_Bit)

[group 10]

4. **Question:**

True or False?

(a) The NOR operation cannot be implemented using only AND gates and inverters in the ALU.

(b) When an exception occurs, the current PC (Program Counter) value is stored in the EPC (Exception Program Counter) register.

(c) The MIPS addu and addiu instructions trigger an overflow exception when the result exceeds the 32-bit range.

(d) The selectors control which functional unit performs the operation, ensuring that only the correct gate is active.

Ans:

(a) False, by De Morgan's law, NOR(A, B) = (NOT A) AND (NOT B). It can be implemented using only AND gates and NOT gates.

(b) True, the EPC saves the address of the instruction that caused the exception, allowing the CPU to return after handling it.

(c) False, addu and addiu ignore overflow by design. Only add and addi raise exceptions on signed overflow.

(d) False, all gates in the ALU operate simultaneously regardless of the control signal. The selector's job is not to activate or deactivate gates, but rather to choose the correct output among all the results.

[group 2]

5. **Queestion:**

Answer the following True and False questions.

(a) In 4-bit two's complement representation, the operation 0101 + 0101 results in an overflow.

(b) When adding operands with different signs, overflow can still occur.

(c) NAND and NOR gates are known as universal gates. Therefore, NAND gate can be used for the zero detection logic without any other changes.

(d) Consider a 3-bit scenario where (A2, A1, A0) = (1, 0, 1) and (B2, B1, B0) = (1, 1, 0). If Cin0 = 1, then according to carry look-ahead theory the value of Cin2 would be 1;

(e) In multi-level CLA, the second-level carry-lookahead unit uses block-level P and G to calculate the final sum outputs for each block.

Ans:

(a) T

(b) F (+, -, moves towards 0)

(c) F (Zero Detection: NOR gate)

(d) T

(e) F (to generate block-level Cin)

[group 6]

6. **Question:**

Why is the ALU control for NOR equal to 1100?

Ans:

For the NOR operation, we have A NOR B = (NOT A) AND (NOT B).

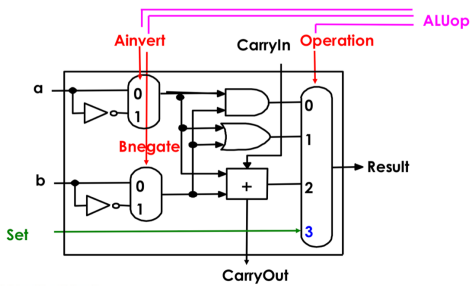
We set Ainvert = 1, Binvert = 1, and Operation = 00 (AND).

Combining these bits gives the ALU control input 1100.

[group 8]

7. **Question:**

Explain the function of the ALUop control signal. If we want to perform A nand B, what would be the corresponding ALU Control signal?



Ans:

ALUop	Function
0000	and
0001	or
0010	add
0110	subtract
0111	SLT
1100	nor

A nand B = A' + B'. Thus, ALUop = 1101

[group 7]

8. **Question:**

Explain the difference between Ripple Carry Adder with Carry Look Ahead Adder?

Ans:

Ripple Carry Adder connects many 1-bit full adders in a chain, each adder takes one-bit of the operands and produces a Sum bit and a Carry-out bit, the carry ou t but then becomes the carry-in for the next adder. Each bit must wait for the ca rry from previous bit to start.

Carry Look Ahead Adder calculate all the carry bits in parallel using two signals which are:

- Generate(G) = A * B

- Propagate(P) = A $\oplus$ B

These signals are sent to the Carry Look ahead unit with the initial Carry in and p redicts all carries for the rest of the adders.

[group 1]

9. **Question:**

In this problem, we will calculate the sum of two 4-bit binary numbers, A and B.

Given

- $A = 1011_2$

- $B = 0111_2$
- The initial carry-in is $c_0 = 0$.

Your goal is to solve this addition without using the standard, column-by-column method. Instead, you will use the internal logic of a Carry Look-ahead Adder.

Ans:

Formulas to Use

1. **Generate**: $g_i = A_i$ AND B_i
2. **Propagate**: $p_i = A_i$ OR B_i
3. **Carry Equation**: $c_{i+1} = g_i$ OR (p_i AND c_i)
4. **Sum Equation**: $S_i = A_i$ XOR B_i XOR c_i

Step 1: Calculate the 'g' and 'p' signals

We calculate the generate (g) and propagate (p) signals for each bit.

- **Bit 0:** A_0=1, B_0=1
  - $g_0 = 1$ AND $1 = 1$
  - $p_0 = 1$ OR $1 = 1$
- **Bit 1:** A_1=1, B_1=1
  - $g_1 = 1$ AND $1 = 1$
  - $p_1 = 1$ OR $1 = 1$
- **Bit 2:** A_2=0, B_2=1
  - $g_2 = 0$ AND $1 = 0$
  - $p_2 = 0$ OR $1 = 1$
- **Bit 3:** A_3=1, B_3=0
  - $g_3 = 1$ AND $0 = 0$
  - $p_3 = 1$ OR $0 = 1$

Step 2: Calculate the Carries

Using $c_0 = 0$ and the signals from Step 1, we find the carry for each position.

- $c_1 = g_0$ OR (p_0 AND c_0) = 1 OR (1 AND 0) = 1 OR 0 = **1**
- $c_2 = g_1$ OR (p_1 AND c_1) = 1 OR (1 AND 1) = 1 OR 1 = **1**
- $c_3 = g_2$ OR (p_2 AND c_2) = 0 OR (1 AND 1) = 0 OR 1 = **1**
- $c_4 = g_3$ OR (p_3 AND c_3) = 0 OR (1 AND 1) = 0 OR 1 = **1**

So, the final carry-out (c_4) is **1**.

Step 3: Calculate the Final Sum

With all the carries known, we can now find the sum bits (S).

- $S_0 = A_0$ XOR B_0 XOR c_0 = 1 XOR 1 XOR 0 = 0 XOR 0 = **0**
- $S_1 = A_1$ XOR B_1 XOR c_1 = 1 XOR 1 XOR 1 = 0 XOR 1 = **1**
- $S_2 = A_2$ XOR B_2 XOR c_2 = 0 XOR 1 XOR 1 = 1 XOR 1 = **0**
- $S_3 = A_3$ XOR B_3 XOR c_3 = 1 XOR 0 XOR 1 = 1 XOR 1 = **0**

The final 4-bit sum S[3:0] is **0010**.

Final Answer: The result of the addition is a sum of 0010 with a carry-out of 1 -> 10010

[group 11]

10. **Question:**

    Compare the gate delay of two 4-bits adder, one implemented with Ripple Carry Adder, the other with Carry Lookahead Adder. (Assume each logic gate has one gate delay)

Ans:

RCA: 4*2 = 8

CLA: 1+2 = 3

[group 13]

11. **Question:**

    How many gate delays do we need to compute a 64 bits number when applying cascaded carry lookahead method with 8 bitwise of each ALU?

Ans:

64/8=8 stages

Each stages need to operate “or” and “and”, which takes two gate delays **8*2=16** gate delays.