Grade 12 A/L ICT · Lesson 3

3.3 Binary Arithmetic and Bitwise Logic: Add, Subtract, AND, OR, XOR, NOT

Syllabus 3.3. The ALU inside the CPU does all its work with bits. In this part you learn **binary addition** (with carries, even when adding several numbers), **binary subtraction** (with borrowing, and with 2's complement), what **overflow** is, and the **bitwise logical operations NOT, AND, OR and XOR**.

1. The rules of binary addition

Binary addition works **exactly like decimal addition**, column by column from the **right**. The only difference: in binary, **2 is written as 10**, so you write 0 and **carry 1** to the next column.

Bits added එකතු කරන බිටු	Sum bit එකතු බිටුව	Carry රැගෙන යාම
0 + 0 0 + 0	**0** 0	**0** 0
0 + 1 0 + 1	**1** 1	**0** 0
1 + 1 1 + 1	**0** 0	**1** 1
1 + 1 + 1 1 + 1 + 1	**1** 1	**1** 1

The four rules of binary addition

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 1 = 10 → write **0**, carry **1**
- 1 + 1 + 1 = 11 → write **1**, carry **1**

🧱 **Worked example: Add $1011_2 + 0110_2$.**
Right column: 1 + 0 = 1. Next: 1 + 1 = 10, write 0 carry 1. Next: 0 + 1 + 1 = 10, write 0 carry 1. Next: 1 + 0 + 1 = 10, write 0 carry 1. Last carry goes down. Answer **10001_2**. Check: 11 + 6 = 17 ✔

	2^4	2^3	2^2	2^1	2^0

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

Carry	1	1	1		
	0	1	0	1	1
	0	0	1	1	0
Sum	**1**	**0**	**0**	**0**	**1**

🧮 **Worked example: Add three numbers: 1011 + 1101 + 0111.**
In every column three or four 1s may meet. Count the 1s (plus the carry): 3 = 11, so write **1** and carry **1**. Doing this for each column gives **11111_2**. Check: 11 + 13 + 7 = 31 ✔

	2^4	2^3	2^2	2^1	2^0
Carry	1	1	1	1	
	0	1	0	1	1
	0	1	1	0	1
	0	0	1	1	1
Sum	**1**	**1**	**1**	**1**	**1**

💡 **Many numbers?** If a column adds up to 4 (100 in binary), write 0 and carry 1 **two columns** to the left. Or add two numbers first, then add the third to the answer – this is safer in an exam.

2. Binary subtraction with borrowing

- 0 − 0 = 0, 1 − 0 = 1, 1 − 1 = 0
- 0 − 1: **borrow 1** from the next column on the left. The borrowed 1 is worth **2** here, so 2 − 1 = **1**.
- The column that lent the 1 is now **one less**.

🧮 **Worked example: Subtract $11010_2 - 01011_2$.**
Right column: 0 − 1, borrow → 2 − 1 = 1. Next: 1 − 1 − 1 (paying back the borrow) is below 0, so borrow again: 2 + 1 − 1 − 1 = 1. Continue the same way; the table shows where each borrow was paid. Answer **01111_2**. Check: 26 − 11 = 15 ✔

	2^4	2^3	2^2	2^1	2^0
Borrow paid	−1	−1	−1	−1	
	1	1	0	1	0
−	0	1	0	1	1
Difference	**0**	**1**	**1**	**1**	**1**

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

🧮 **Worked example: Subtract $10000_2 - 00001_2$ (a long chain of borrows).**
The right column needs a borrow, but its neighbours are 0, so the borrow travels to the leftmost 1. That 1 becomes 0 and every 0 on the way becomes 1. Answer **01111_2** (16 − 1 = 15).

	2^4	2^3	2^2	2^1	2^0
Borrow paid	−1	−1	−1	−1	
	1	0	0	0	0
–	0	0	0	0	1
Difference	**0**	**1**	**1**	**1**	**1**

3. Subtraction with 2's complement

A CPU does not need a separate subtract circuit. To work out **A − B** it adds: **A + (2's complement of B)**. Any carry out of the left end (the 9th bit for 8-bit numbers) is **thrown away**.

🧮 **Worked example: Work out 45 − 20 using 8-bit 2's complement.**
+45 = 0010 1101. −20 = 1110 1100 (flip 00010100 and add 1). Add them: 100011001. Throw away the 9th bit → **0001 1001** = 25 ✔

	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Carry	1	1	1		1	1			
	0	0	0	1	0	1	1	0	1
	0	1	1	1	0	1	1	0	0
Sum	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**1**

4. Overflow

An 8-bit 2's complement number can only hold −128 to +127. If the true answer is **outside this range**, the result is wrong. This is called **overflow**. A sign check finds it: **adding two positives gives a negative**, or **adding two negatives gives a positive**.

🧮 **Worked example: Add 100 + 50 in 8-bit 2's complement. What goes wrong?**
0110 0100 + 0011 0010 = **1001 0110**. The MSB is 1, so the computer reads **−106**, not 150. Two positives gave a negative: **overflow**, because 150 > 127.

5. Bitwise logical operations

A **bitwise** operation works on **each pair of bits on its own**, column by column. There is **no carry**.

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

A	B	A AND B	A OR B	A XOR B	NOT A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

AND
1 only if **both** bits are 1.

OR
1 if **at least one** bit is 1.

XOR
1 if the bits are **different**.

NOT
Flips every bit (one input).

🧱 **Worked example: A = 11001010 and B = 10101100. Find A AND B, A OR B, A XOR B and NOT A.**
Work column by column using the truth table: AND = **10001000**, OR = **11101110**, XOR = **01100110**, NOT A = **00110101**.

	b7	b6	b5	b4	b3	b2	b1	b0
A	1	1	0	0	1	0	1	0
B	1	0	1	0	1	1	0	0
A AND B	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
A OR B	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**0**
A XOR B	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**
NOT A	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**1**

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved



Bitwise operations on A and B

6. Using bitwise operations (masks)

A **mask** is a bit pattern used to pick out or change certain bits. **AND** with a mask **checks or clears** bits, **OR sets** bits to 1, and **XOR toggles** (flips) bits.

🧮 **Worked example: Is bit 3 of A = 11001010 set? Use a mask.**
11001010 AND 00001000 = **00001000**. The answer is not 0, so bit 3 **is 1**.

🧮 **Worked example: Change the ASCII letter "A" (01000001) to "a" using one bitwise operation.**
Small letters are 32 more, and 32 = 00100000. 01000001 OR 00100000 = **01100001** = 97 = "a".

NEW 2026 **Bitwise AND finds a network.** A computer with IP address 192.168.1.130 and subnet mask 255.255.255.0 does 192.168.1.130 AND 255.255.255.0 to get its network address **192.168.1.0**. Your home router does this for every packet.

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

🚀 **Beyond the syllabus (2026 update)** *Not required for the exam – read for interest.* **Beyond the syllabus – 2's complement in every CPU.** Every modern processor (Intel, AMD, Apple M-series, ARM chips in phones) stores signed integers in 2's complement. The C++20 and C23 language standards now **require** it. The ALU sets a **carry flag** for unsigned overflow and an **overflow flag** for signed overflow. XOR is also the heart of encryption such as AES: data XOR key hides the data, and XOR with the same key again brings it back.

Key Words

Carry – The 1 moved to the next column when a column adds up to 2 or more.

Borrow – Taking 1 from the next column on the left when 0 − 1 is needed.

Overflow – When the true result is too big or too small for the number of bits.

Sum bit – The bit written in a column after adding.

Bitwise operation – An operation done on each pair of bits separately, with no carry.

AND – Gives 1 only when both bits are 1.

OR – Gives 1 when at least one bit is 1.

XOR – Gives 1 when the two bits are different.

NOT – Flips every bit: 0 becomes 1 and 1 becomes 0.

Mask – A bit pattern used with AND, OR or XOR to check or change chosen bits.

ALU – The part of the CPU that does arithmetic and logic.

Truth table – A table that shows the output for every input combination.

Sample Q&A

1. State the four rules of binary addition.

> **Answer:** 0 + 0 = 0; 0 + 1 = 1; 1 + 1 = 0 carry 1; 1 + 1 + 1 = 1 carry 1.

2. Add 1011 + 1101 + 0111.

> **Answer:** 11111 (11 + 13 + 7 = 31).

3. Subtract 11010 − 01011.

> **Answer:** 01111 (26 − 11 = 15).

4. How does a CPU subtract using only an adder?

> **Answer:** It finds the 2's complement of the number being subtracted and adds it. The carry out of the left end is ignored. Example: 45 − 20 = 00101101 + 11101100 = 00011001 = 25.

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

5. What is overflow? Give an example.

> **Answer:** Overflow happens when the answer does not fit in the bits. In 8-bit 2's complement, 100 + 50 = 150 is more than 127, so the result 10010110 reads as −106.

6. Find 11001010 XOR 10101100 and say what XOR shows.

> **Answer:** 01100110. XOR shows 1 wherever the two bits are different.

MCQ

1. What is 1011 + 0110?

A. 1101
B. 10001
C. 10101
D. 11001
E. 10011

2. 1 + 1 + 1 in binary gives:

A. Sum 0, carry 1
B. Sum 0, carry 0
C. Sum 1, carry 1
D. Sum 1, carry 0
E. Sum 3, no carry

3. What is 1011 + 1101 + 0111?

A. 11011
B. 111111
C. 11101
D. 11111
E. 10111

4. What is 11010 − 01011?

A. 01110
B. 01111
C. 01101
D. 10001
E. 10111

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

5. What is 10000 − 00001?

A. 11111
B. 01110
C. 10001
D. 00001
E. 01111

6. In binary subtraction, 0 − 1 needs:

A. A carry to the next column
B. Nothing, the answer is 0
C. An overflow flag
D. A NOT operation
E. A borrow from the next column

7. To find A − B a CPU usually calculates:

A. A + 2's complement of B
B. A XOR B
C. A AND NOT B
D. B − A then NOT
E. A + B

8. In 8-bit 2's complement, 00101101 + 11101100 (after dropping the 9th bit) is:

A. 10011001
B. 11100011
C. 00011001
D. 00011000
E. 100011001

9. Adding 100 + 50 in 8-bit 2's complement gives a negative answer. This is:

A. Overflow
B. A mask
C. A borrow
D. Underflow of a float
E. Correct, 150 is negative

10. 11001010 AND 10101100 = ?

A. 10001010
B. 10001000
C. 01100110
D. 11101110
E. 01110111

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

11. 11001010 OR 10101100 = ?

A. 11101110
B. 01100110
C. 10001000
D. 11101111
E. 11111110

12. 11001010 XOR 10101100 = ?

A. 10011001
B. 01100111
C. 10001000
D. 11101110
E. 01100110

13. NOT 11001010 = ?

A. 11001010
B. 11001011
C. 01010011
D. 00110101
E. 00110100

14. Which operation gives 1 only when both bits are 1?

A. ADD
B. OR
C. NOT
D. AND
E. XOR

15. Which operation with mask 00100000 changes “A” into “a”?

A. NOT
B. 2’s complement
C. OR
D. AND
E. Subtraction

16. IP 192.168.1.130 AND mask 255.255.255.0 gives:

A. 192.168.1.0
B. 192.168.1.255
C. 0.0.0.130
D. 255.255.255.130
E. 192.168.1.130

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

17. Which term means: “The 1 moved to the next column when a column adds up to 2 or more”?

A. ALU
B. Overflow
C. Carry
D. XOR
E. AND

18. Which term means: “Taking 1 from the next column on the left when 0 − 1 is needed”?

A. OR
B. NOT
C. Sum bit
D. Borrow
E. Truth table

19. Which term means: “When the true result is too big or too small for the number of bits”?

A. Mask
B. Overflow
C. Bitwise operation
D. Carry
E. XOR

20. Which term means: “The bit written in a column after adding”?

A. Borrow
B. AND
C. NOT
D. ALU
E. Sum bit

Structured Questions

1. (a) Add 1011 + 0110 showing the carries. (b) Check your answer in decimal. *(3 marks)*

> (a) carries in three columns; sum 10001; (b) 11 + 6 = 17.

2. Add the three binary numbers 1011, 1101, 0111 and give the answer in binary and decimal. *(3 marks)*

> Each column: count the 1s plus carry; sum 11111; decimal 31.

3. (a) Subtract 11010 − 01011 using borrowing. (b) Do 45 − 20 in 8-bit 2’s complement. *(3 marks)*

> (a) 01111; (b) 00101101 + 11101100 = 100011001, drop the 9th bit; 00011001 = 25.

© 2026 schoolict.net (Ruwan Mangala Suraweera) · All rights reserved

4. (a) What is overflow? (b) Show that 100 + 50 overflows in 8-bit 2's complement. *(4 marks)*

> (a) result outside the range the bits can hold; (b) 01100100 + 00110010 = 10010110; MSB 1 so reads −106; two positives gave a negative.

5. A = 11001010, B = 10101100. Find (a) A AND B (b) A OR B (c) A XOR B (d) NOT A (e) one use of AND with a mask. *(5 marks)*

> (a) 10001000; (b) 11101110; (c) 01100110; (d) 00110101; (e) checking a bit or finding a network address with a subnet mask.

Summary

- Binary addition: 1 + 1 = 0 carry 1, 1 + 1 + 1 = 1 carry 1; work from the right.
- For many numbers, count the 1s in each column plus the carry.
- Subtraction: 0 − 1 borrows 1 (worth 2) from the left column.
- A CPU subtracts by adding the 2's complement and dropping the extra carry.
- Overflow: the true answer is outside the range of the bits (8-bit: −128 to +127).
- Bitwise: AND both 1, OR any 1, XOR different, NOT flips; masks check, set or toggle bits.

MCQ answer key – 3.3

1. **B** – 11 + 6 = 17 = 10001.
2. **C** – 3 = 11.
3. **D** – 11 + 13 + 7 = 31.
4. **B** – 26 − 11 = 15.
5. **E** – 16 − 1 = 15.
6. **E** – Borrow 1 (worth 2).
7. **A** – Subtraction becomes addition.
8. **C** – 45 + (−20) = 25 = 00011001.
9. **A** – 150 > 127.
10. **B** – 1 only where both are 1.
11. **A** – 1 wherever at least one is 1.
12. **E** – 1 where bits differ.
13. **D** – Flip every bit.
14. **D** – AND truth table.
15. **C** – OR sets bit 5 to 1 (+32).
16. **A** – 130 AND 0 = 0.
17. **C** – Carry – The 1 moved to the next column when a column adds up to 2 or more.
18. **D** – Borrow – Taking 1 from the next column on the left when 0 − 1 is needed.
19. **B** – Overflow – When the true result is too big or too small for the number of bits.
20. **E** – Sum bit – The bit written in a column after adding.

© 2026 schoolict.net (Ruwan Mangala Suraweera). All rights reserved.