AQA Computer Science A-Level

4.5.4 Binary number system

Concise Notes

Specification:

4.5.4.1 Unsigned binary:

Know the difference between unsigned binary and signed binary

Know that in unsigned binary the minimum and maximum values for a given number of bits, *n*, are 0 and 2^n -1 respectively

4.5.4.2 Unsigned binary arithmetic:

Be able to:

- add two unsigned binary integers
- multiply two unsigned binary integers

4.5.4.3 Signed binary using two's complement:

Know that signed binary can be used to represent negative integers and that one possible coding scheme is two's complement.

Know how to:

- represent negative and positive integers in two's complement
- perform subtraction using two's complement
- calculate the range of a given number of bits, *n*.

4.5.4.4 Numbers with a fractional part:

Know how numbers with a fractional part can be represented in:

- fixed point form in binary in a given number of bits
- floating point form in binary in a given number of bits

Be able to convert for each representation from:

- decimal to binary of a given number of bits
- binary to decimal of a given number of bits

4.5.4.5 Rounding errors:

Know and be able to explain why both fixed point and floating point representation of decimal numbers may be inaccurate.

4.5.4.6 Absolute and relative errors:

Be able to calculate the absolute error of numerical data stored and processed in computer systems.

Be able to calculate the relative error of numerical data stored and processed in computer systems.

Compare absolute and relative errors for large and small magnitude numbers, and numbers close to one.

4.5.4.7 Range and precision:

Compare the advantages and disadvantages of fixed point and floating point forms in terms of range, precision and speed of calculation.

4.5.4.8 Normalisation of floating point form:

Know why floating point numbers are normalised and be able to normalise un-normalised floating point numbers with positive or negative mantissas.

4.5.4.9 Underflow and overflow:

Explain underflow and overflow and describe the circumstances in which they occur.

Signed and unsigned binary

- Binary numbers can be either signed or unsigned
- A computer has to be told whether a number is signed or unsigned before working with it. There is no way to tell from the number's appearance.
- Unsigned binary numbers can only represent positive numbers
- Signed binary allows for the representation of negative numbers using binary

Range of unsigned numbers

- The range of numbers that can be represented by an unsigned binary number depends on the number of bits available
- There is a pattern to the range of numbers that can be represented by a given number of bits
- For n bits, there are 2^n possible permutations of the bits
- For n bits, a range of decimal numbers from 0 to $2^n - 1$ can be represented

Unsigned binary arithmetic

Adding two unsigned binary integers

- There are four important rules to remember:

1. 0 + 0 + 0 = 0
2. 0 + 0 + 1 = 1
3. 0 + 1 + 1 = 10
4. 1 + 1 + 1 = 11

Note

The process of adding binary numbers is covered in the A* and B notes.

- After carrying out binary addition, it's a good idea to check your answer by converting to decimal if you have time

Multiplying two unsigned binary integers

- Write out one of the two numbers starting under each occurrence of a 1 in the second number
- Then add the contents of the columns
- Binary multiplication can be checked by converting to decimal

Note

The process of multiplying binary numbers is covered in the A* and B notes.

Signed binary with two's complement

- Two's complement allows for the representation of both positive and negative numbers in binary
- The most significant bit of a number is given a negative place value

Subtraction using two's complement

- Computers work by adding numbers
- To perform subtraction, computers add negative numbers

Range of two's complement numbers

- The range of a two's complement signed binary number includes both positive and negative values
- With *n* bits, the range of a two's complement signed binary number is from $2^{n-1}-1$ to -2^{n-1}

Numbers with a fractional part

- Binary can be used to represent numbers with a fractional part
- There are two ways to do this, fixed point form and floating point form

Fixed point binary

- A specified number of bits are placed before a binary point
- The remaining bits fall behind the binary point
- Standard binary place values are used for columns before the binary point
- Columns behind the binary point start at ½, then ¼ and ⅛ etc.

Floating point binary (binary to decimal)

- Comparable to scientific notation
- A number is represented as a mantissa and an exponent
- In exam questions, both the mantissa and exponent will be represented using two's complement signed binary
- A number of bits are allocated to the mantissa the remaining bits form the exponent.
- In order to convert from floating point form to decimal, first convert the exponent to decimal
- Next, treat the number as if there were a binary point between the first and second digits of the mantissa, move the binary point the number of positions specified by the exponent
- Now treat the mantissa as a fixed point binary number
- Finally, convert from binary to decimal using the usual method

Floating point binary (decimal to binary)

- First convert your decimal number to fixed point binary
- Next, normalise the number so that it starts with `01` (for a positive number) or `10` (for a negative number)
- When converting from floating point to decimal, the binary point is assumed to be between the first two digits in the mantissa. Move the binary point until this is achieved.
- The number of positions through which the binary point is moved forms the exponent, which must be converted to binary
- Combine the mantissa and exponent to form a normalised floating point number

Rounding errors

- There are some decimal numbers that cannot possibly be represented exactly in binary
- Binary can only approximately represent these numbers
- For this reason, both fixed point and floating point representations of decimal numbers may be inaccurate

Absolute and relative errors

- You can calculate absolute and relative errors to see how close a particular number is to an actual value

Absolute error calculation

- The actual amount by which a value is inaccurate
- Can be calculated by finding the difference between the given value and the actual value

Relative error calculation

- A measure of uncertainty in a given value compared to the actual value
- Relative errors are relative to the size of the given value
- Can be calculated using the formula:

$$relative\ error = \frac{absolute\ error}{actual\ value}$$

- A percentage can be calculated if the result is multiplied by 100

Errors in relation to magnitude

- An absolute error of 0.1cm in a measurement of 50m results in a very small relative error of 0.002%
- The same absolute error of 0.1cm in a measurement of 1cm results in a much larger relative error of 10%

Fixed point vs floating point

- Both fixed point and floating point perform the same function of representing numbers with fractional parts in binary
- Floating point allows for the representation of a greater range of numbers with a given number of bits than fixed point
- The number of bits allocated to each part of a floating point number affects the numbers that can be represented
    - A large exponent and a small mantissa allows for a large range but little precision
    - A small exponent and a large mantissa allows for good precision but only a small range
- The placement of the binary point in fixed point notation determines the range and precision of the numbers that can be represented
    - A binary point close to the left of a number gives good precision but only a small range of numbers
    - A binary point close to the right gives good range but poor precision

Normalisation

- Floating point numbers are normalised to provide the maximum level of precision for a given number of bits
- Involves ensuring that the a floating point numbers starts with `01` (for a positive number) or `10` (for negative numbers)

Underflow and overflow

- Two types of error that can occur when working with binary

Underflow
- Occurs when very small numbers are to be represented but there are not enough bits available

Overflow
- Occurs when a number is too large to be represented with the available bits
- Particularly important when using signed binary where overflow can cause positive numbers to give negative results