OCR Computer Science A Level

1.4.1 Data Types

Concise Notes

Specification:

1.4.1 a)

- Primitive data types
  - Integer
  - Real / floating point
  - Character
  - String
  - Boolean

1.4.1 b)

- Represent positive integers in binary

1.4.1 c)

- Negative numbers in binary
  - Sign magnitude
  - Two's complement

1.4.1 d)

- Addition and subtraction of binary integers

1.4.1 e)

- Represent positive integers in hexadecimal

1.4.1 f)

- Convert positive integers between binary, hexadecimal and denary

1.4.1 g)

- Representation and normalisation of floating point numbers in binary

1.4.1 h)

- Floating point arithmetic
  - Positive and negative numbers
  - Addition and subtraction

1.4.1 i)

- Bitwise manipulation and masks
  - Shifts
  - Combining with AND, OR, and XOR

1.4.1 j)

- How character sets are used to represent text
  - ASCII
  - Unicode

Data Types

Integer

- A whole number
- Zero is an integer
- Negative numbers are integers
- Can't have a fractional part
- Useful for counting things

```
-88
0
15
```

Real

- Positive or negative numbers
- Can, but do not necessarily, have a fractional part
- Useful for measuring things
- All integers are real numbers

```
-75.3
5.66
15
```

Character

- A single symbol used by a computer
- The letters A to Z
- The numbers 0 to 9
- Symbols like %, £, and, □

```
R
ほ
```

String

- A collection of characters
- Can be used to store a single character
- Can also be used to store many characters in succession
- Useful for storing text
- Don't cut off leading 0s like numeric types

```
Hello!
07954
```

Boolean

- Restricted to `True` and `False`
- Useful for recording data that can only take two values

```
True
False
```

Representing Positive Integers in Binary

- A single binary digit is called a bit
- Eight binary digits can be combined to form a byte
- Half a byte (four bits) is called a nybble
- The least significant bit of a binary number is the one furthest to the right
- The most significant bit is furthest to the left

Binary Addition

When adding binary, there are four simple rules to remember:

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

Negative Numbers in Binary

- Binary can represent negative numbers using a few different methods, we cover:
  - Sign magnitude
  - Two's complement
- These methods give a special meaning to certain bits

Sign Magnitude

- The equivalent of adding a + or - sign in front of a number
- A leading 1 is added for a negative number
- A leading 0 is added for a positive number

Two's Complement

- Has the added advantage of making binary arithmetic with negative numbers much more simple
- Works by making the most significant bit negative
- Converting to two's complement is as simple as flipping all of the bits in the positive version of a binary number and adding one

Subtracting in Binary using Two's Complement

- Two's complement makes subtraction in binary easy
- Subtracting a number from another is the same as adding a negative number
- To subtract in binary, use binary addition with a negative two's complement number

Hexadecimal

- Hexadecimal is base 16
- The characters `0-9` are as usual
- The characters `A-F` represent `10-15`
- Place values start with 1 (16^0) and go up in powers of 16.

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**

Converting from hexadecimal to binary

- First convert each hexadecimal digit to a decimal number
- Convert these to a binary nybble
- Combine the nybbles to form a single binary number

Converting from hexadecimal to decimal

- First convert to binary, as explained above, and then convert from binary to decimal
- Alternatively, use the place values of hexadecimal to convert directly to decimal

Floating Point Numbers in Binary

- Floating point binary is similar to scientific notation
- Floating point numbers can be split into two parts:
  - Mantissa
  - Exponent
- The mantissa is always taken to have the binary point after the most significant bit
- Next convert the exponent to decimal
- Move the binary point according to the exponent

Normalisation

- Maximises precision in a given number of bits
- To normalise a binary number:
    - Adjust the mantissa so that it starts `01` for a positive number of `10` for a negative number

Addition and Subtraction of Floating Point Numbers

Addition

- To add floating point binary numbers, their exponents need to be the same
- Then add the mantissas
- Finally, normalise the result if required

Subtraction

- Involves converting to two's complement and adding
- The exponents must be the same
- Use binary addition on the mantissas
- Normalise the result if required

Bitwise Manipulation and Masks

Shifts

- A shift performed on binary numbers is called a logical shift
- There are two varieties:
    - Logical shift left
    - Logical shift right
- Involves moving all of the bits in a binary number a specified number of places to the right or to the left
- Can be thought of as adding a number of leading or trailing zeros
- The result is a multiplication (or division if shifting right) by two to the power of the number of places shifted
- A logical shift left by one place has the effect of doubling the initial number

Masks

- Can be applied to binary numbers by combining them with a logic gate

Character Sets for Representing Text

- A published collection of codes and corresponding characters
- Can be used by computers for representing text
- Two widely used character sets are ASCII and Unicode

ASCII

- American Standard Code for Information Interchange
- The leading character set before Unicode
- Uses 7 bits to represent $2^7 = 128$ different characters
- ASCII soon came into trouble when computers needed to represent other languages with different characters

Unicode

- Solves the problem of ASCII's limited character set
- Uses a varying number of bits allowing for over 1 million different characters
- Many characters have yet to be allocated
- Enough capacity to represent a wealth of different languages, symbols, and emoji