OCR Computer Science AS Level

1.2.3 Introduction to Programming

Concise Notes

Specification:

1.2.3 a)

- **Procedural programming language techniques:**
  - Program flow
  - Variables and constants
  - Procedures and functions
  - Arithmetic, Boolean and assignment operators
  - String handling
  - File handling

1.2.3 b)

- **Assembly language**
  - Following and writing simple LMC programs

Procedural programming language techniques

Program Flow

- In procedural programming, the program flow is given by three main structures:
  - Sequence
    Code is executed line-by-line, from top to bottom
  - Selection
    A certain block of code is run if a specific condition is met, using IF, ELSE IF and ELSE statements
  - Iteration
    A block of code is executed a certain number of times or while a condition is met. This uses FOR, WHILE or REPEAT UNTIL loops.

Variables and Constants

- Named locations in memory where data is stored.
- The contents of this location can be changed while the program is being executed
- Assigned using the = sign which is called an assignment operator
- Value of constants cannot be edited by the program during execution
- Used to prevent values from being accidentally changed

Procedures and Functions

- Both named blocks of code that perform a specific task
- While procedures do not have to return a value, functions must always return one, single value

Arithmetic, Boolean and assignment operators

- Arithmetic operators are used to carry out mathematical functions such as +, -. * and /
- `**` is used for exponentiation which is when a number is raised to a power
- `DIV or //` calculates the whole number of times a number goes into another, which is called integer division
- `MOD` or `%` is used to find the remainder when a number is divided by another
- Relational operators are used to make comparisons between two values, such as >, <, =, >= and <=
- One additional operator is the 'not equal to' operator denoted by '!=='
- `==` is used to check whether a value is identical to another
- These can be combined with Boolean operators to check whether multiple conditions are met within a single statement
- Boolean operators include AND, OR and NOT

String handling

- There are various operations that can be performed on strings and that you need to be aware of

To get the length of a string:

```
stringname.length
```

```
text="physics and maths tutor"
```

`text.length` will produce 23.

To get a substring (a section within a string):

```
stringname.subString(startingPosition, numberOfCharacters)
```

```
text="physics and maths tutor"
print(text.substring(2,4)
```

will produce 'ysic'.

File handling

- You also need to be able to use pseudocode to handle files.

To open a file to read:

```
myFile = openRead("filename.txt")
```

To read a line from a file:

```
fileContent = myFile.readLine()
```

To close a file:

```
myFile.close()
```

To open a file to write:

```
myFile = openWrite("nameoffile.txt")
```

To write a line to a file:

```
myFile.writeLine("Physics and Maths Tutor")
```

The end of the file is given by:

```
endOfFile()
```

Assembly Language

- Low level language that is the next level up from machine code
- Uses mnemonics, which are abbreviations for machine code instructions
- Commands used are processor-specific
- Each line in assembly language is equivalent to one line of machine code

Below is a list of the mnemonics you need to be aware of and be able to use:

Mnemonic	Instruction	Function
ADD	Add	Add the value at the given memory address to the value in the Accumulator
SUB	Subtract	Subtract the value at the given memory address from the value in the Accumulator
STA	Store	Store the value in the Accumulator at the given memory address
LDA	Load	Load the value at the given memory address into the Accumulator
INP	Input	Allows the user to input a value which will be held in the Accumulator
OUT	Output	Prints the value currently held in the Accumulator
HLT	Halt	Stops the program at that line, preventing the rest of the code from executing.
DAT	Data	Creates a flag with a label at which data is stored.
BRZ	Branch if zero	Branches to a given address if the value in the Accumulator is zero. This is a conditional branch.
BRP	Branch if positive	Branches to a given address if the value in the Accumulator is positive. This is a conditional branch.
BRA	Branch always	Branches to a given address no matter the value in the Accumulator. This is an unconditional branch.