8. Program construction

Assemblers

An assembler is a program which coverts the low level assembly programming language into machine code. The assembler does this by converting the one-word assembly instructions into an opcode, e.g. converting AND to 0010. It also allocates memory to variables, often resulting in an operand.

Assembly code	Assembler conversion	Opcode	Operand
AND A	→	0010	0001
LOD B	→	0110	0010

Interpreters

Before high level programming languages can be run, code is converted by an interpreter, one line at a time, into machine code, which is then executed by the CPU.

Compilers

A compiler is used when high-level programming languages are converted into machine code, ready to be executed by the CPU. There are four main stages of compilation:

Lexical analysis

- Comments and unneeded spaces are removed.
- Keywords, constants and identifiers are replaced by 'tokens'.
- A symbol table is created which holds the addresses of variables, labels and subroutines.

Syntax analysis

- Tokens are checked to see if they match the spelling and grammar expected, using standard language definitions. This is done by parsing each token to determine if it uses the correct syntax for the programming language.
- If syntax errors are found, error messages are produced.

Semantic analysis

- Variables are checked to ensure that they have been properly declared and used.
- Variables are checked to ensure that they are of the correct data type, e.g. real values are not being assigned to integers.
- Operations are checked to ensure that they are legal for the type of variable being used, e.g. you would not try to store the result of a division operation as an integer.

Code generation

- Machine code is generated.
- Code optimisation may be employed to make it more efficient/faster/less resource intense.

Translators

A translator changes (translates) a program written in one language into an equivalent program written in a different language. For example, a program written using the PASCAL programming language may be translated into a program written in one of the C programming languages using a translator.

Types of error that may occur in programming code

Error	Description	Example
Syntax	An error that occurs when a command does not follow the expected syntax of the language, e.g. when a keyword is incorrectly spelt	• **Incorrect:** IF A ADN B Then • **Correct:** IF A AND B Then
Runtime/ execution	An error that only occurs when the program is running and is difficult to foresee before a program is compiled and run	• Program requests more memory when none is available, so the program *crashes*
Logical	An error that causes a program to output an incorrect answer (that does not necessarily crash the program)	• An algorithm that calculates a person's age from their date of birth, but ends up giving negative numbers
Linking	An error that occurs when a programmer calls a function within a program and the correct library has not been linked to that program	• When the square root function is used and the library that calculates the square root has not been linked to the program
Rounding	Rounding is when a number is approximated to nearest whole number/tenth/hundredth, etc.	• 34.5 rounded to nearest whole number is 35, an error of +0.5
Truncation	Truncating is when a number is approximated to a whole number/tenth/hundredth, etc. nearer zero	• 34.9 truncated to whole number is 34, an error of -0.9

INTERESTING FACT

In 1947, Grace Murray Hopper, an admiral in the U. S. Navy and a computer programming pioneer, documented the first actual computer bug when a moth got trapped in a computer.