

Topic 3 : Controlling the Computer

Language translation and Java approach...
D&L pages 235-241 in 2nd Edition (pages may be slightly different in other editions – but check the index page (“Compilation”, “Interpreters”).

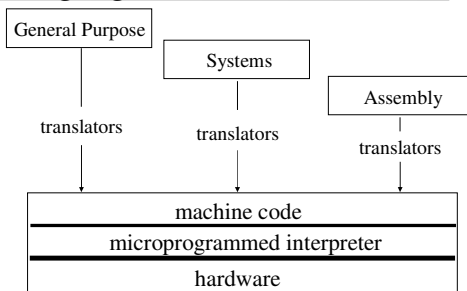
C.Sc. 131: Systems Architecture

Approaches to Language Translation

- We have already seen one example of language translation:
 - translation of machine language into microcode.
- High level languages such as C, Ada etc. must themselves be turned into either machine or assembly language statements.
- Each high level language requires its own translator.

C.Sc. 131: Systems Architecture

Approaches to Language Translation



C.Sc. 131: Systems Architecture

Approaches to Language Translation ... contd.

- There are two basic approaches to language translation:
 - interpretation.
 - compilation.
- Each approach has its own advantages and disadvantages,

C.Sc. 131: Systems Architecture

Interpretation

- Interpreters process a program statement by statement, execution of each statement preceding translation of the next.

start at the beginning of the high-level program
repeat
 translate the next high-level statement
 execute the result
until the end of the program is reached

C.Sc. 131: Systems Architecture

Interpretation ... contd.

- The disadvantage with interpretation is that each high-level statement may be translated many times
- Steps to help minimise this problem involve making the syntax of the language to be interpreted as simple as possible (e.g. Basic)
- Where have we seen an interpreter: we wrote a microcode interpreter to execute machine language programs

C.Sc. 131: Systems Architecture

Interpretation ... contd.

- Interpretation tends to be unsuitable if:-
 - the time taken to translate a statement outweighs the time taken to execute it
 - the program is executed frequently
 - execution speed is paramount
- Alternative approach is to compile the program.

C.Sc. 131: Systems Architecture

Interpretation ... a question.

- Question: why not write microcode interpreters to process high-level languages directly
 - difficulty of writing a complex interpreter in microcode.
 - a separate microcode interpreter would be required for each high-level language.
 - these interpreters would occupy large amounts of expensive micromemory.
 - to make writing such interpreters feasible microinstructions would need to be more powerful, and machine architectures more complex.

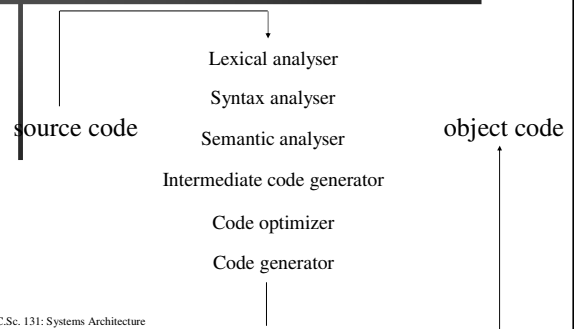
C.Sc. 131: Systems Architecture

Compilation

- Compilation involves the translation of the entire program before execution.
 - start at the beginning of the high-level program
 - repeat
 - translate the next high-level statement
 - until the end of the program is reached
 - execute the result
- Compilation is particularly suitable for programs which are executed multiple times without change.

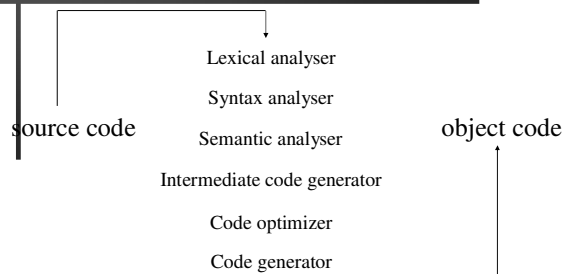
C.Sc. 131: Systems Architecture

Structure of a Translator



C.Sc. 131: Systems Architecture

Structure of a Translator

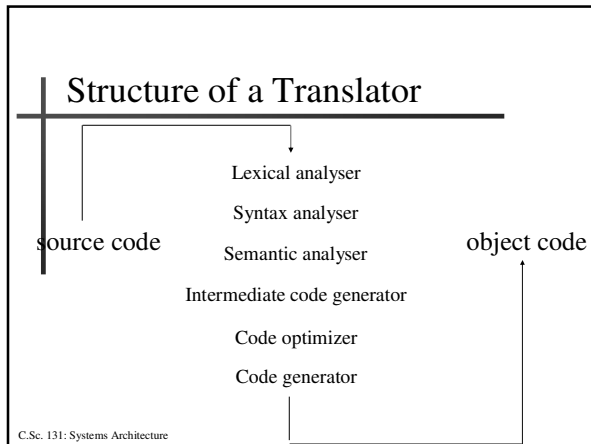


C.Sc. 131: Systems Architecture

Lexical Analyser

- The lexical analyser groups characters from the input file into logical tokens of the language, e.g.:
 - 'WHILE'
 - 'IF'
 - INTEGER
 - <=
 - CHARACTER.

C.Sc. 131: Systems Architecture



Syntax Analyser

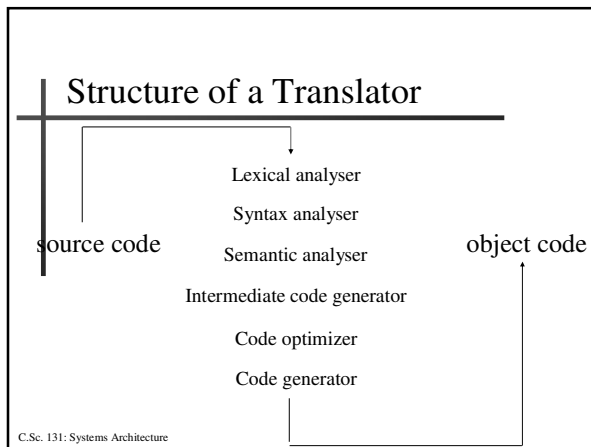
- Groups the tokens produced by the lexical analyser into syntactic structures.
- This is achieved by a process called *parsing*.

```

loop: WHILE ( expression ) { statements }
statements: statement
           statement statements
  
```

- This is covered in much more detail next year.

C.Sc. 131: Systems Architecture



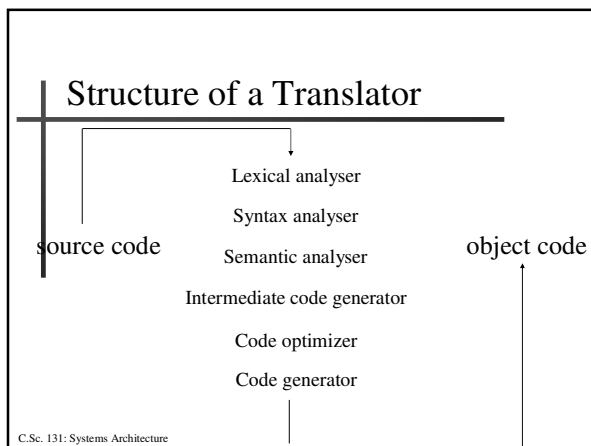
Semantic Analyser

- Analyses the code for 'content'.
- In the example below the expression must return a Boolean value.

```

loop: WHILE ( expression ) { statements }
  
```

C.Sc. 131: Systems Architecture

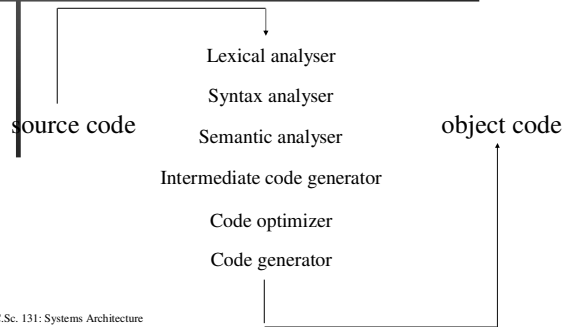


Intermediate Code Generator

- Generates something between machine code and the high-level language.
- Tends not to specify every register to use, etc.
- This code can be easily optimized by the next phase.

C.Sc. 131: Systems Architecture

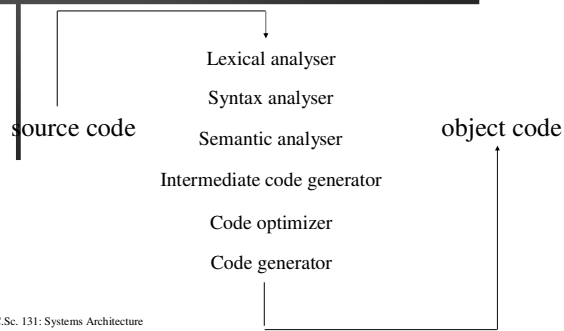
Structure of a Translator



Code Optimizer

- Code optimization tends to try and optimize speed of execution or reduce program size.
 - Examples include:
 - re-arranging loop structures.
 - optimizing expression evaluation.
- C.Sc. 131: Systems Architecture

Structure of a Translator



Code Generator

- This phase actually generates the machine code for the target machine.
 - Specific to the target machine - rest of the compiler is pretty machine independent.
 - Resulting program is ready for execution.
- C.Sc. 131: Systems Architecture

Assembly Language ... Recap.

label:	operation	mode	operand	comment
here:	LDA	#	50	; load 50 into acc
there:	STA		50	; store A into loc. 50
	JMP		there	

C.Sc. 131: Systems Architecture

A Specific Class of Translator: The Assembler

- Assembly language can be converted into machine code using an assembler.
 - Note confusing terminology: the language is called assembler as is the translator.
 - In the case of most assembly languages one statement corresponds to a single machine code statement so the process is relatively simple.
- C.Sc. 131: Systems Architecture

Where does Java fit in to all this?

Reference : <http://www.javasoft.com>

C.Sc. 131: Systems Architecture

The Java Environment

Java Source

C.Sc. 131: Systems Architecture

The Java Environment

Java Source

Java Compiler

C.Sc. 131: Systems Architecture

The Java Environment

Java Source

Java Compiler

Java Bytecode

C.Sc. 131: Systems Architecture

The Java Environment

Java Source

Java Compiler

Java Bytecode

Class Loader & Verifier

C.Sc. 131: Systems Architecture

The Java Environment

Java Source

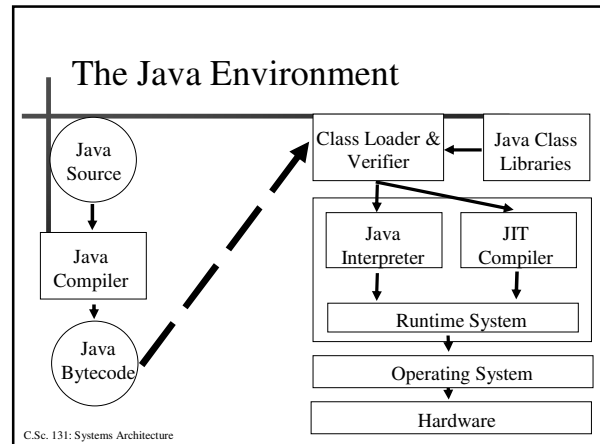
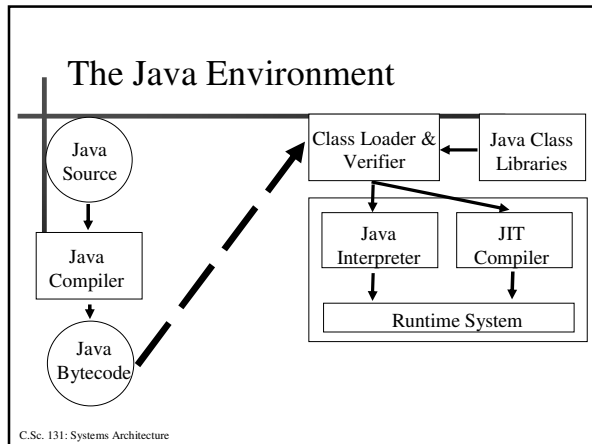
Java Compiler

Java Bytecode

Class Loader & Verifier

Java Class Libraries

C.Sc. 131: Systems Architecture



- ### The Java Virtual Machine
- The Java Virtual Machine is an imaginary machine.
 - It has its own architecture (registers etc.) and machine code (bytecode).
 - These are designed to be easy to map onto any existing machines: hence we can write software emulators of the Java Virtual Machine.
- C.Sc. 131: Systems Architecture

- ### The Java Virtual Machine
- The Java Virtual Machine can be implemented as a standalone emulator,
 - Windows, Unix, EPOC etc.
 - Or, it can be embedded in (for example) a browser such as Netscape.
 - Depending on the context it runs either applications or applets.
- C.Sc. 131: Systems Architecture

- ### This afternoon...
- A look at how the computer can interact with the outside world.
 - Self contained – but see web links.
- C.Sc. 131: Systems Architecture