Hardware & Software

Electronic Operations of a Processor

- When a program is running on a computer the processor is constantly performing very many tiny electronic operations.
- For example, one such operation reads one byte of data from main memory into the processor. Another operation tests if one of the bits in a byte is a 1 bit or a 0 bit. Most processors are able to perform several thousand types of tiny operations like these, and can perform billions of them per second.
- Those are the only things that a processor can do. It has a set of tiny electronic operations that it can to perform, and that is all. These tiny electronic operations are performed one at a time. But billions of them are performed per second, and billions of small operations can add up to a large and useful action.
- Everything that a processor does is built out of these tiny operations! Luckily, you don't need to know the details of these operations to write programs in Java. The purpose of a *high-level language* like Java is to organize the tiny electronic operations into large, useful units represented by program statements.

Machine Instructions

- Users and programmers of computers usually don't think about the billions of tiny electronic operations that go on each second. The situation is (very roughly) similar to when you are driving your car. You think about the "big operations" it can perform, such as "accelerate", "turn left", "brake", and so on. You don't think about tiny operations, such as the valves in your engine opening and closing 24,000 times per minute or the crankshaft spinning at 3000 revolutions per minute.

- A machine instruction consists of several bytes in memory that tell the processor to perform one machine operation. The processor looks at machine instructions in main memory one after another, and performs one machine operation for each machine instruction. A collection of machine instructions in main memory is called a machine language program or (more commonly) an executable program.

Machine Language Program

- Actual processors have many more machine instructions and the instructions are much more detailed. A typical processor has a thousand or more different machine instructions.
- A machine language program is a sequence of machine language instructions in main memory.
- A machine instruction consists of one or more bytes (in this example, only one).
- The processor runs a program one machine instruction at a time.
- All the little machine operations add up to something useful.

High Level Programming Languages

- It is rare for programmers to write programs in machine language. The executable files (the directly runnable machine language programs) for most applications contain hundreds of thousands (or even millions) of machine language instructions. It would be very hard to create something like that from scratch.
- Most programs are created using a high level programming language such as Java, C, C++, or BASIC. With a high **level language**, a programmer creates a program using powerful, "big" operations which will later be converted into many little machine operations.
- For example, here is a line from a program in the language C:

```
int sum = 0;
```

- This declares and initializes a variable to zero (a big operation).

Source Programs

- Usually a source program is translated into a machine language program. An application program called a **translator** takes a source program as input and produces a machine language program as output.

- A machine language program is also called an executable program, executable file, or sometimes, just executable.

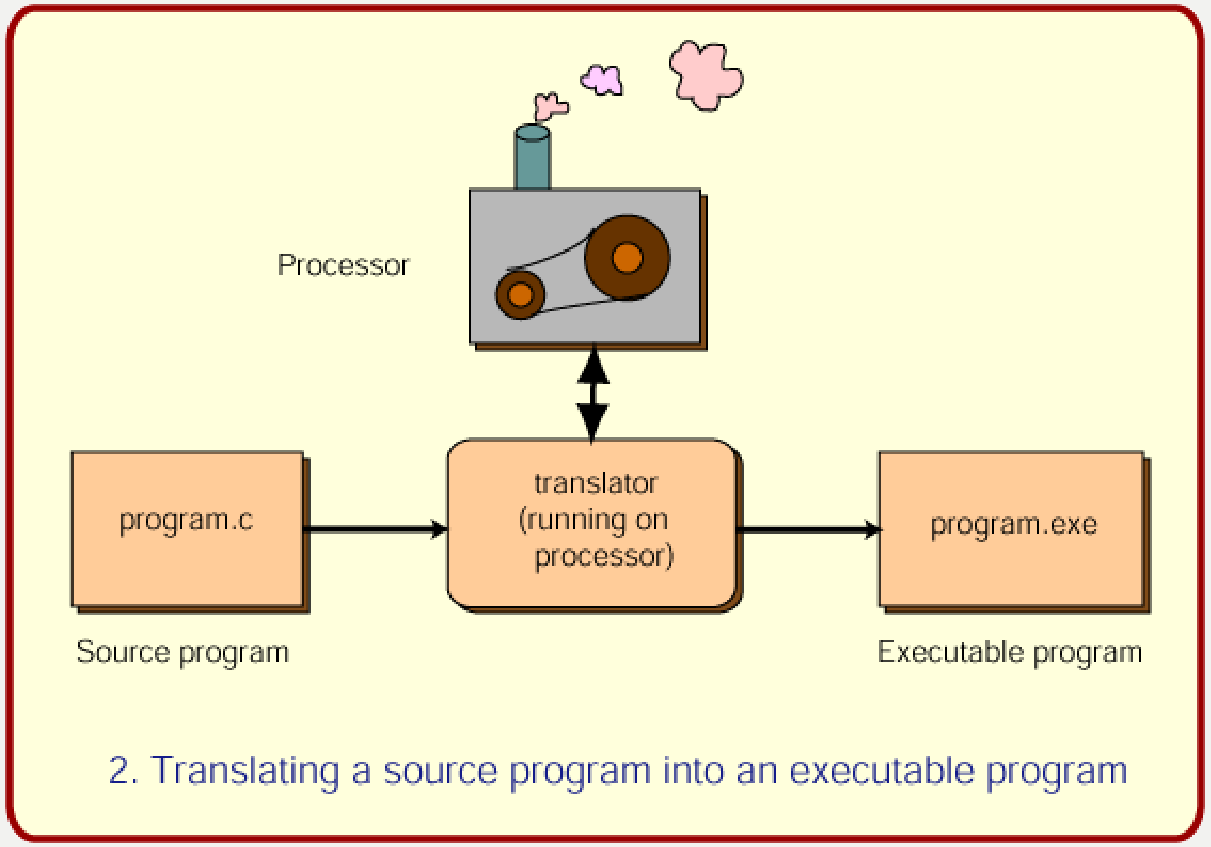
- For example, the C program addup.c could be translated into an executable program. The executable program might be called addup.exe and can be saved on the hard disk. Now the executable version of the program can be copied into main memory and executed.

- The word **compile** means the same thing as translate. So one can say that a source program is compiled into an executable program.

Program Translation

- The source program is created using a text editor.
- It contains instructions in a high level language.
- It contains bytes that represent characters.
- The source program is kept on the hard disk.
- The source program can not be run by the processor.
- A translator (compiler) program translates the source program into an executable program.
- The source program remains unchanged; a new executable program is created.
- The executable program contains machine instructions.
- A translator translates from a specific high level language (like C) into machine instructions for a specific processor type (like Pentium).
- The executable program is also kept on hard disk.
- The machine language program is copied from disk into main memory.
- The processor directly executes these machine language instructions.

Program Translation



2. Translating a source program into an executable program

Program Execution

- For commercial software like games and word processors, the machine code is the product that is sold to the user. The user does not get a copy of the source program.

- A student learning programming, or a programmer developing an application, creates source programs and translates them (with a compiler) into executable programs.

- The above is what goes on with languages like Ada, Pascal, C, C++, FORTRAN and others. Java adds a few more steps, which will be discussed in the next chapter.

Portability

- Ideally, only one program needs to be written in the high level language. That source file can then be translated into several executable files, each containing the correct machine instructions for its intended processor. This is how the same game can be made for desktop computers and game machines.

- The idea of using one source file for executable programs that run on different processors is called software portability. You would like to write a program just once (in a high level language) and then to run it on any computer system by translating it into that system's machine language.

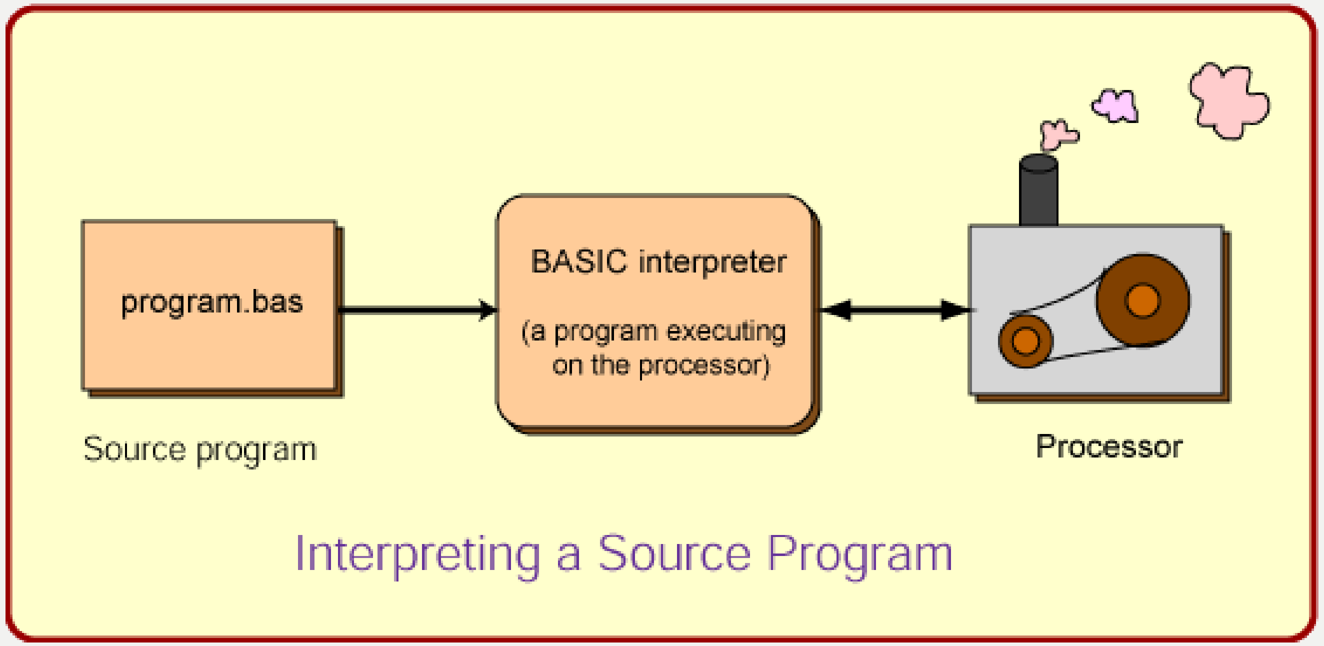
- Usually, unfortunately, things do not work out that nicely. There are enough little problems so that it takes a substantial amount of human effort to get a source program running on a several different systems. Sometimes it is months before a game program that has been released for PCs is released for game consoles.

- One of the big advantages of Java is that it is automatically portable between computer systems that have Java support. No human effort is involved at all.

Interpreter

- Another way to execute a source program is to use an interpreter for the language. An interpreter is an executable program that runs directly on the processor. An interpreter reads through a source program written in a high level language and performs the actions that the source program asks for.
- The BASIC interpreter works by reading in commands from the BASIC source program one by one. Each time it reads in a command, the interpreter does what the command asks. A BASIC command might ask to add two numbers together. That is fine. The BASIC interpreter is a program, and programs can easily add together two numbers. The BASIC program might then ask to write the sum to the monitor. Still fine. The BASIC interpreter can easily do that.
- The BASIC interpreter uses the fundamental machine operations of the processor to perform the actions requested in the BASIC source program. But the source program itself is not translated into machine language.
- This is like a cook who can perform basic cooking operations but can't read. How can the cook follow a recipe? The cook hires an assistant who knows how to read. The assistant reads the instructions of a recipe one by one to the cook, who performs them one by one as they are read.

Interpreter



Interpreting a Source Program

Virtual Machine

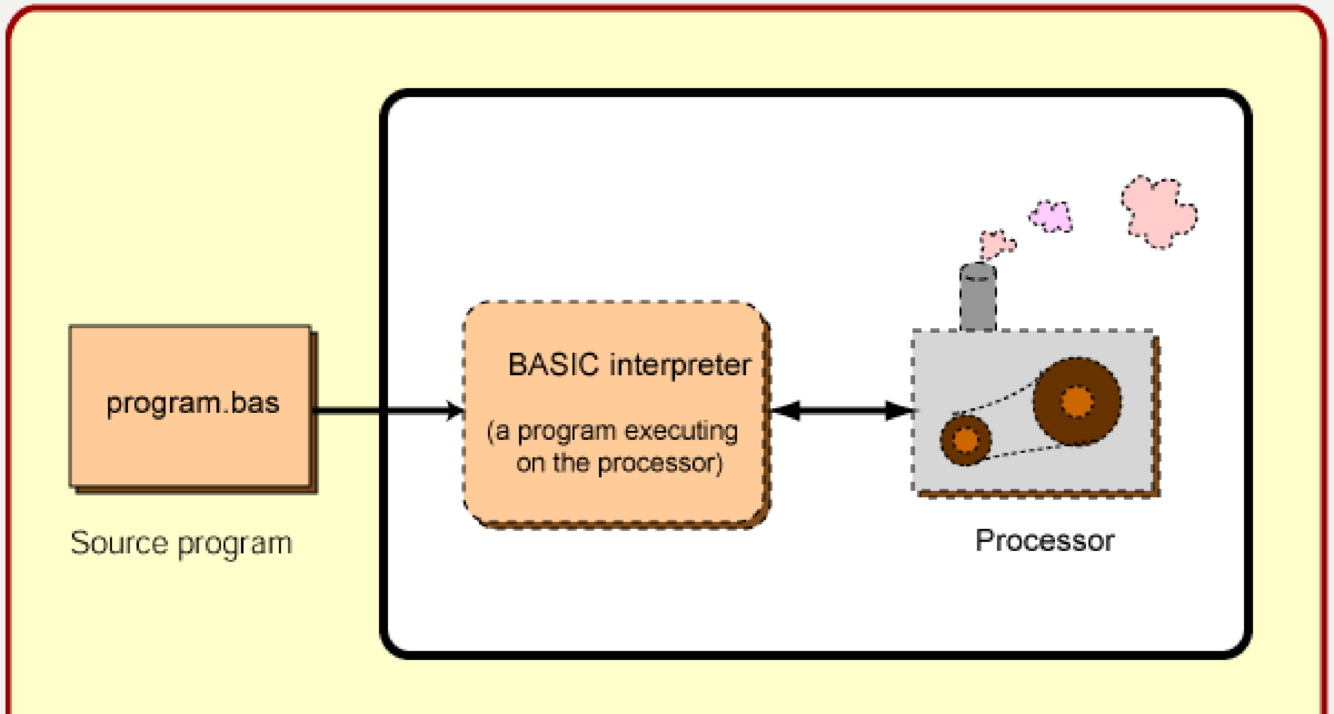
- When an interpreter is running a BASIC source program, both the interpreter and the source program are in main memory. The interpreter consists of machine instructions that the hardware can execute directly. The BASIC source program consists of commands that the interpreter can perform.

- From the perspective of the BASIC program, it looks like the commands in BASIC are being directly executed by some sort of machine. The figure has been modified to show this.

- This is really the same as the previous figure, but now a box has been drawn around the actual (hardware) processor and the interpreter that it is executing. The combination looks like a machine that can directly execute BASIC commands. It is as if BASIC commands are the machine language for combination of processor and interpreter.

- The word virtual is used in situations where software has been used to make something that looks like the real thing. In this case it looks like we have a machine that can directly execute BASIC, so we can say that we have a BASIC virtual machine.

Virtual Machine



Interpreting a Source Program on a Virtual Processor

Speed

- The situation with computer languages is somewhat like that with human languages:
- **Translator:** takes a complete document in one language and translates it into a complete document in a second language, which can then be used by a reader of the second language.
- **Interpreter:** acts as an intermediate between a speaker of one language and a speaker of another language. Usually an interpreter works one sentence at a time. Immediately after a sentence is spoken in the first language, the translator converts it into the second language. You and your French translator (say) could in combination be regarded as a "virtual French speaker".
- Using a human interpreter as an intermediate is slower than conversing directly in a particular language. The same is true with computer language interpreters. The interpreter has to do quite a bit of work to deal with the language it is interpreting. The extra work makes it look like the virtual processor is much slower than the real one.