**CHAPTER 0-3**

**universal programming languages**

* A variety of features are found in modern high-level programming languages.
  + Which of these features are actually necessary?
* Our approach is to describe a simple imperative programming language that is rich enough to allow us to express programs for computing all the Turing-computable functions (and thus all the computable functions).
  + A programming language with this property is called a universal programming language.

**Machine language instructions manipulate bit patterns**

* A computer can only execute ML. Thus, an HLL program must be translated to ML before it can be executed.
* The data definition statements found in HLLs allow programmers the luxury of thinking in terms of data structures and data types such as array of numeric values and strings of alphabetic characters, even though the machine itself does not associate interpretations to the bit patterns representing these objects.
  + These data definition statements merely allow the compiler to prevent programmers from making stupid mistakes.
* ML instructions manipulate bit patterns without knowing what the bit patterns represent.
  + The interpretation of these bit patterns is achieved via various coding systems such as 2's complement for signed numbers, floating point scheme for real numbers, Unicode for characters, etc, using appropriate machine language instructions.
* Thus data definition statements, while nice, are not necessary.

**The Bare Bones universal** **programming language**

* No data definition statements are provided.
  + All variables in Bare Bones will be considered to represent bit patterns.
  + For convenience, we will represent a bit pattern using a nonnegative integer.
  + A Bare Bones programmer can simply introduce a new variable name when it is needed, with the understanding that it refers to a bit pattern.
  + Of course, a translator for our Bare Bones language must be able to distinguish variable names from the other terms. This is done by designing the syntax of Bare Bones so that the role of any term can be identified by syntax alone.
* Data manipulation statements
* There are 3 assignment statements.
  + - clear X

Assign a string of 0s to variable X.

* + - incr X

Increments the value of X (incr can always be done on a TM because of its infinite amount of memory).

* + - decr X

Decrements the value of X. If X is 0 already, then this operation is a null operation.

* There is 1 control structure representing a loop. We adopt the policy of writing only 1 statement per line, and use indentation to mark the body of a loop structure.
  + - while X not 0:

body

The while statement causes any statement or statement sequence indented below the while statement to be repeated as long as the value of the variable X is not zero.

**Programming in Bare Bones**

* Keep in mind that our goal in presenting the language Bare Bones is to investigate what is possible, not what is practical.
* Can we implement the assignment X=constant where X is a variable?
  + The following sequence implements X=3
* Can we implement the assignment Z=X where X and Z are variables?
  + The following sequence implements Z=X
* A Bare Bones program for computing Z=X × Y (X is destroyed)
* Thus, although Bare Bones itself does not have an assignment instruction, we often write Bare Bones programs as though it did.

**The universality of BareBones**

* First, we observe that any program written in Bare Bones can be thought of as directing the computation of a function.
  + The function’s input consists of values assigned to variables prior to executing the program, and the function’s output consists of the values of variables when the program terminates.
* Consider the following BareBones program:

incr X

It computes the successor function that is computed by the Turing machine M discussed earlier.

* Consider the following BareBones program where X and Y are input variables and Z is an output variable:

Z=Y

while X not 0:

incr Z

decr X

What function does it compute?

* Researchers have shown that the Bare Bones programming language can be used to express algorithms for computing all the Turing-computable functions.
  + Combining this with the Church-Turing thesis implies that any computable function can be computed by a program written in Bare Bones.
* Thus Bare Bones is a universal programming language in the sense that, if an algorithm exists for solving a problem, then that problem can be solved by some Bare Bones program.

**HW**

1. Show that the statement invert X (whose action is to convert the value of X to zero if its initial value is nonzero and to 1 if its initial value is zero) can be simulated by a Bare Bones program segment. That is, write a program sequence in Bare Bones that simulates the action of the statement:

if X not 0:

X=0

else:

X=1

1. Show that the if-else structure can be simulated using Bare Bones. That is, write a program sequence in Bare Bones that simulates the action of the statement:

if X not 0:

S1

else:

S2

where S1 and S2 represent arbitrary statement sequences. Hint: use the invert operation.

1. Describe the function computed by the following Bare Bones program assuming the function’s input is represented by X and its output by Z

clear Z

while X not 0:

incr Z

incr Z

decr X

1. Write a program in Bare Bones that assigns 0 to Z if X is even; otherwise assign 1 to Z.
2. Show that even our simple BareBones language contains more statements than necessary by showing that the clear statement can be replaced with combinations of other statements in the language: