NEAT/3

This is a High Level Assembler implemented on the NCR Century computers, a family of computers designed primarily for data processing. NEAT/3 was designed, in 1966– 68, with two goals in mind:

1. To make it easy to write short to medium size programs, without having to go through a COBOL compiler (a slow and complex process at that time).

2. To make it easy to write an efficient COBOL compiler for the Century computers.

The NCR literature does not mention the name 'high-level assembler', nor does it mention the term 'higher-level language'. It simply refers to NEAT/3 as a programming language, and to the translator, as the NEAT/3 compiler. However, on examining the language, it is easy to see that it does not have the type of powerful statements and control structures one expects to find in a highe r-level language. The statements are simple, and resemble typical assembler instructions. This is why they are easy to translate, and this is why the NEAT/3 translator was easier to write than a COBOL compiler. Most of the time, a NEAT/3 statement is translated into one machine instruction. Only when conversion between data types is necessary, the translator (we will call it a translator, not a compiler or an assembler) generates more machine instructions.

The main feature that makes NEAT/3 look like a higher-level language is the data definitions. The way data items and files are declared in NEAT/3 closely resembles COBOL. Concepts such as working storage area, constants area, data records divided into fields, and data types, are all borrowed from COBOL, which makes it easy to write a COBOL compiler in NEAT/3. Even editing masks (for data to be printed) are supported and use the same characters '9','Z','$','+', ' –','.' as in COBOL.

The data types are: Characters (coded in USASI, not ASCII), signed & unsigned decimal, signed & unsigned packed decimal, binary, and hex.

The following is an example of a record declared in NEAT/3:

Name	Code	Location	Length	Type	Picture
D SALESRED	R	50			
D STORECODE	F	0	3	X	
D MANAGER	F	3	8	X	
D DATE	F	11	6	X	
D DAY	F	11	2	X	
D MONTH	F	13	2	X	
D YEAR	F	15	2	X	
D GROCERY	F	17	6	U	
D PRODUCE	F	23	5	U	
D MEAT	F	28	6	U	
D DAIRY	F	34	5	U	
D MISC	F	39	5	U	
D DAILYTOTAL	F	44	6	U	

Note the following:

1. The code field can have values of: R for a record, F for a field, and A, for an area.

2. Field DAY is located at the same position as DATE.Thus DAY is a subfield of DATE (and, as a result, also MONTH, YEAR) which is how a record becomes a tree structure, same as in COBOL.

3. The D at the beginning of each line stands for data declaration.

4. No 'picture' is shown in this example, but pictures are heavily used in NEAT/3 and are virtually identical to the COBOL picture clause.

5. The possible types are:

Code Data Type

X or blankCharacter
S Generated spaces
Z Generated zeros
U Unsigned decimal
D Signed decimal
B Binary
H Hex
P Signed packed decimal
K Unsigned packed decimal
E Editing mask

Next we examine some of the NEAT/3 instructions (or, as the NCR literature calls them, statements). This is the area were one can really justify the name high-level assembler. Most of the instructions are simple, resemble typical assembler instructions, and are easy to translate. The main difference in translation is the automatic conversion between data types, which NEAT/3 supports, but a typical assembler does not.

What are typical valid and invalid type conversions in languages such as COBOL and NEAT/3?

1. Comparisons. The instruction 'COMP A,B' compares its two operands and sets status flags like any typical comparison in machine language. The only difference is that the operands can be of different types, and the translator takes care of type conversion.

2. Conditional Branches. The 'BRE CALCTAX' instruction is executed by testing the status flags and branching to CALCTAX on 'equal'. There are several such branches, each translated into one machine instruction. They are the only way to make decisions in NEAT/3, except for the simple IF described below.

3. Test and branch. The if alphabetic (IFAL) instruction tests its first operand and branches to the second operand (a label) if the first operand is alphabetic (of type character).

4. Moves. Those instructions workthe same as in COBOL. In the simplest case, a move is translated into one machine instruction but, if it also involves type conversion and editing, it is translated into several instructions.

5. End of execution (FINISH). This instruction is translated into machine instructions that close all open files and perform a software interrupt to the operating system.

In summary, NEAT/3 justifies the name high-level assembler language, but also the name low-level programming language. Its translator may be called a high-level assembler but also a compiler. It seems to lie somewhere in between assembler language and COBOL.