[From: *Information retrieval and machine translation,* ed. Allen Kent. International conference… September 6-12, 1959, Cleveland, Ohio; part 2 (New York: Interscience Publishers, 1961)]

CHAPTER 25

An Instruction Code for Language Processing

ANTHONY V. BANES, HOWARD L. ENGEL, DON R. SWANSON

Ramo-Wooldridge, a Division of Thompson Ramo Wooldridge Inc., Los Angeles, California

A general purpose computer program for automatic Russian to English translation has been under development at Ramo-Wooldridge during the past several years.* Approximately 60,000 words of Russian text have been machine translated and analyzed in detail. The output translation in its present form is rough in quality, and certainly many important language and linguistic problem areas need additional work; nevertheless to varying depths essentially all major problems of Russian-English automatic translation have at least been encountered and studied. As a by-product of this general purpose computer program a preliminary study has been carried out of the required attributes of a stored program general purpose computer instruction code especially adapted to language translation.

This instruction code analysis involves only operations performed in the internal high speed memory of a computer. It is assumed that a large capacity external memory (e.g. for dictionary storage) is available, but no specifications are placed on access time or nature of readout. Thus it is immaterial for our purposes here whether the external memory has high speed random access or whether it requires sequential search such as does magnetic tape.

A 9,000-instruction program for Russian-English translation was used as the principal source of requirements and guidance for development of the instruction code. Auxiliary programs for collecting and organizing language data to facilitate analysis, and other programs for testing hypothetical translation rules, also provided valuable input to the study. In addition we considered, though not in detail, other approaches to mechanical translation and certain operations involving information retrieval.

In parallel with the design of an instruction code we have also initiated work on the logical design of a stored program computer especially adapted to these instructions. Thus the instruction code which we shall outline here could be used for two main purposes: (a) as a compiler or interpretive routine to enable programming to be carried out in a pseudo-code adapted to language processing, and (b) as a

*M20-8U13 " Experimental Machine Translation of Russian to English," December 1958, Ramo-Wooldridge.

basis for design of a stored program computer for language translation and processing.

We shall discuss briefly the nature of the processes performed in our language translation program and outline our present ideas on what some of the instructions ought to be like. This work continues and we again point out the preliminary nature of our results.

Language translation can be conveniently broken down into processes of word lookup, stem affix stripping, idiom lookup, rules of syntax, and rules of meaning. In addition, as we have mentioned, a number of processes associated with the research procedure itself should be included.

Word lookup in a dictionary involves alphabetic sequencing of variable length Russian words (from one to twenty characters each) search in the high speed memory for a given word in an alphabetized sequence, and miscellaneous transfer and input-output operations. The stripping of stems from affixes involves comparison and matching of from one to six characters, a table lookup operation involving about a thousand six-character entries, and numerous alphanumeric compare (=) and branch operations. Formation of idioms is characterized by table lookup, code matching, transfer of variable length records, and editing of sequences of several words. To implement rules of syntax and meaning, a variety of operations are performed which we shall not describe here; however, some general observations can usefully be made. The processing of one to five digit codes associated with words of the sentence occurs with high frequency in our translation program. Searching a sentence for words and codes belonging to specified categories and then branching on the results of the search characterizes many syntactic operations. However the extent of the sentence search is almost invariably contingent upon encountering punctuation, symbols, or certain other categories of codes and words. The facility for direct specification of such search conditions and contingencies, the ease with which one, two, and three character codes can be transferred and compared, and the convenience of alphanumeric table lookup with essentially instantaneous readout are all important requirements for an instruction code adapted to syntactic processing. These operations are frequently performed throughout the 9,000-step R-W computer program for translation.

To represent the numerous special routines for testing hypotheses and in other respects mechanizing portions of the language research process, let us consider here an artificial example which incorporates many of the typical processes performed. Suppose we were to select from a large quantity of Russian text all occurrences of instrumental nouns together with modifying adjectives and participles and at the same time keep count of the number of times that these nouns are modified by one, two, or three adjectives or participles. In this event the required operations would include table lookup of a three-digit alphanumeric code denoting grammatical information, transferring and readout of groups of variable length words (one to twenty characters each), and finally a capability for keeping running counts in perhaps 15 or 20 different registers.

Let us now consider a hypothetical set of instructions which largely meet the requirements we have outlined. Many of the proposed instructions are quite similar to those used in present digital computers, and will not be further discussed except to note that they are of the following types: (a) read in messages of variable lengths from input devices, (b) read out variable length messages to output devices, (c) control of index registers, (d) conditional or unconditional program jumps. The following instructions are typical of those tailored to the requirements of language translation. We shall here briefly describe what they are intended to accomplish.

1. Variable Length Memory Readout

This instruction transfers from the external memory a specified number of successive characters. If the number of characters indicated in the instruction is a symbol other than a numeral, the computer transfers characters until that same symbol is encountered in the memory. In this way words or phrases of length unknown to the programmer can be read out by means of a single instruction.

2. Variable Length Transfer

This instruction permits the transfer of any number of characters (up to 99) from any series of successive storage locations to any other series of successive locations. If the instruction symbol indicating the number of characters to be transferred is non-numerical, the transfer continues until a symbol of this kind is encountered. This instruction, like the first, is useful in transferring words or word groups of unknown length. Typical of the translation processes for which it is especially convenient is the selection of one English equivalent from among several choices presented by the dictionary.

3. Word-String Interchange

A "sentence store word" we define here as a unit of memory that may contain a word of Russian, English equivalents, syntactic symbols, and records of decisions made during the translation program. This instruction permits the interchange of up to 9 successive sentence store words with another group of up to 9 successive words anywhere else in the sentence store. This instruction is especially useful for reordering words within sentences, in editing for printout, and in removing phrases from sentences for separate analysis. The two sets of sentence store words to be interchanged need not be equal in number; intervening words are automatically shifted to permit the longer set of words to be inserted at the location of the shorter.

4. Single Character Sentence Search

This instruction determines the number of sentence store words which precede or follow a given sentence store word) that must be searched to find the first occurrence (in a specified character position of a sentence store word) of a symbol specified by the program-

mer. It is well adapted to certain syntactic analysis routines wherein a phrase boundary may be determined by the location of a word carrying a specified part-of-speech code.

5. Single Character Table Lookup Test

A "single character interrogation store" is provided to store 16 groups of six characters each. The symbols in each must be specitied by the programmer. This instruction then permits a branch in the program depending on whether or not a sentence store word selected by the programmer has a symbol at a specified position that is the same as any of a given group of symbols in the single character interrogation store. A branch of this kind is especially valuable, for example, to specify alternate action whenever punctuation is encountered. The single character interrogation store is in this case loaded with various punctuation symbols.

6. Six-Character Table Lookup

A "six-character interrogation store" provides storage for 256 six-character groups. This interrogation store is especially adapted to table lookup and to determining words common to two lists of words. The programmer may with a single instruction interrogate this memory to obtain the address of any specified group of characters in the six-character interrogation store. This instruction is especially appropriate to the stem-affix splitting procedure.

7. One to Six-Character Test and Branch

Any n successive characters ($n \leq 6$) in a sentence store word may be compared for identity with n characters in the instruction, and a branch initiated on the outcome. This instruction is useful for comparing semantic or syntactic codes attached to words within the sentence.

8. Alphanumeric Inequality Comparison

This instruction permits branching upon the result of comparison of a specified number of successive characters in any two sentence store words to determine which precedes the other alphabetically.

The foregoing instructions have been used to program typical portions of our present language translation and research procedures. This trial programming effort has not been extensive but we believe that a few of our early impressions are probably reliable. First, the code is easy to use and it is likely that linguists for example could become proficient in using this code in a far shorter time than they could in using existing general purpose computer instruction codes. We estimate very roughly that programming time and required internal high speed storage can be improved by a factor of five or more over the corresponding parameters for present general purpose computers.