Goes with DCL-71

Digital Computer Laboratory
Massachusetts Institute of Technology
211 Massachusetts Avenue
Cambridge 39, Massachusetts

DIC 6345

January 6, 1955

Dear Sir:

The ONR-sponsored computation group at the MIT Digital Computer Laboratory has long been actively developing translating, compiling, and interpreting routines aimed at simplifying and accelerating the preparation of programs for the Whirlwind I computer. Recently interest has centered on universal languages permitting more efficient communication between different computer groups and directly between different computers.

At the same time Project DIC 7138 at the MIT Servomechanisms Laboratory has been investigating some special data reduction problems by developing experimental programs on WWI. Anticipating the use of an ERA 1103 computer for the final solution of these problems, they have been especially interested in the establishment of various programming aids for the 1103. As a start a cooperative program has been undertaken by the two laboratories, with the encouragement of ERA, for developing an ERA 1103 translation program on WWI, the nature of which is described in the enclosed memorandum. It is hoped that this initial system will lead to a more elaborate system to operate on an 1103 and ultimately to a better understanding of the problems of a universal language.

On January 18, 1955, Byron Smith and Al Roberts of ERA expect to visit MIT. A meeting will be held at 9:00 am, in the Digital Computer Laboratory, 211 Massachusetts Avenue, Cambridge, Massachusetts to discuss the following items:

ERA 1103 assembly programs now being developed by ERA and ERA 1103 users

Existing MIT comprehensive systems on WWI

Details of Input-Output facilities existing and planned for the ERA 1103

The proposed WWI-1103 system

Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge 39, Massachusetts

To: Scientific and Engineering Group Staff Members and S&EC Problem #126 Programmers

From: John M. Frankovich

Date: January 4, 1955

SUBJECT: AN ERA 1103 TRANSLATION PROGRAM

Introduction

Project DIC 7138 in the MIT Servomechanisms Laboratory is writing experimental data reduction programs for Air Force fire-control tests. This work, which is supported by the Air Force Armament Laboratory under Contract No. AF33(616)2038, is presently being executed on the Whirlwind I (WWI) computer of the Digital Computer Laboratory. Anticipating eventual use of these programs on ERA 1103 computers, the Servomechanisms Laboratory offered to sponsor the development of a WWI program which will translate mnemonically coded 1103 programs punched in Flexowriter code to the standard hexa-bi-octal form used for 1103 input and which will include some of the facilities offered by the WWI Comprehensive System. The Digital Computer Laboratory has accepted this offer and will develop the translation program in conjunction with its advanced coding research which is supported by the Office of Naval Research. The vocabulary accepted by this translation program will include symbolic addresses, relative addresses, preset parameters, and integer numbers with variable base. This program will, perhaps, be followed by a more comprehensive system which will operate on the 1103 itself. A complete description of the proposed interim system follows.

WWI Operation of the Proposed Translation Program

The proposed translation program would be stored on a magnetic tape unit, MT#2, and would be brought onto and operate from the Buffer Drum in a manner similar to that used in the WWI simulated computers CS, SAC and TAC. This implies of course that 1103 program tapes would have a title structure similar to that used in these systems.

The entire Auxiliary Drum would be used to store a table of symbolic addresses (flads). This allows a maximum at least of 5000 (five thousand) flads ranging over a1, ..., a200, b1, ..., b200, c1, ..., z200. A table of approximately 50 (fifty) preset parameters would also be stored there.

The principal drawback in the CS scheme, i.e., the impossibility of assigning a flad (because of an "indefinite current address indicator", when the value of the flad is theoretically computable) would be eliminated by the following procedure: the number of possible flad tags will also be the number of simultaneously available flads. i.e., if at most 5000 flads can be used at once, then that is the number of available flad tags also. (This eliminates the need for a flad entry table during translation.) A table of, say, 5000 registers will be put aside for flad values. Each register will contain a minus zero if the flad is unassigned, or the value of the flad if the flad is assigned and the absolute value of the flad is known. If the flad is assigned and the value is defined in terms of other flads as well as some integer (preset parameters will be eliminated as later demonstrated), then the register will have a negative sign digit but will have the remaining digits specify the initial (drum) address of a sequence of registers, specifying the flad value. The sequence of registers will have a structure similar to a CS "logical" word: the first register contains a positive integer giving the number of registers in the remainder of the sequence (the first register of the sequence could be eliminated by using the sign bit to specify the initial register of the sequence). All of the remaining registers but the last will be tags specifying the other flads in terms of which this flad is defined. The last register contains the known part (i.e., the integer "stem" value) of the flad.

The flad table starts off with all minus zeros in it. Each time a flad is assigned, the value of the current address indicator is investigated: if it is only an integer, then its value is placed in the flad value table; if it is defined in terms of several falds, then the sequence thus defined is placed in the next available location on the drum and the address of the initial register of the sequence placed in the flad value table in the proper manner as described above. (This implies that the current address indicator itself can be kept in the form of the sequences stored on the drum.) Note that the flad value table thus serves as an entry table to the table of sequences for flads which are defined in terms of other flads.

The flad table will be filled in the above manner on the "1st pass" (defined as that step in the translation process during which the Flexo tape is read and rewritten on MT#1). The "2nd pass" will become the process during which all the "indefinite" flads are deciphered and given their integer values. Note that the evaluation and use of flad values cannot take place during the 1st pass; otherwise duplicate flads would not be handled properly. During the "3rd pass", when conversion of the program is completed, the values of the flads will be available when requested in words. Presumably a partially converted form of the program will be stored on a magnetic tape unit, MT#1, during the 1st pass, and the unit will either be rewound during the 2nd pass or read backwards during the 3rd pass (see the treatment of preset parameters below).

Preset parameters will be completely disposed of on the 1st pass, thereby eliminating any problems which might arise owing to reassignment of the same preset parameter. This is done in the following manner: Whenever a preset parameter assignment occurs during the first pass, its value is computed in terms of flads and an integer stem. This will immediately be the case if the assignment is in terms of flads and integers; if the assignment involves other preset parameters, then we assume their values are available in terms of flads and integers. This value is then stored in exactly the same manner (with one exception, noted later) as flad values on the part of the drum reserved for this purpose. By an induction argument we see that for every preset parameter request we can substitute a set of flads and an integer, thereby eliminating preset parameters in words altogether.

The one exception mentioned above is that whenever a preset parameter is reassigned and the new value involves more flads than the previous assignment, then some inefficiency in storing the new sequence will be introduced, either because the old sequence of registers must be abandoned or because broken sequences must be used. This points out that not only is the number of allowed flads and preset parameters restricted, but also that the "integral of the indefiniteness" is limited.

Note that this treatment of the preset parameters removes the only possibly fatal objection to reading the partially converted program on MT#1 backwards during the 3rd pass.

Vocabulary of the Proposed Translation Program

A. Characters

The keyboard characters and code of the Flexowriters used on the 1103's are apparently the same as those here at WWI, except that the two upper case characters "&" and "?" appear where we have "("and ")", respectively. Since these upper case characters are not used in the proposed scheme, they do not create any problems.

In the Comprehensive System the characters "o" and "0", and "l" and "1" are deliberately made ambiguous in order to reduce typing errors. However, the desired mnemonic vocabulary for the 1103 contains the operations"lq" and "la" and the single letter addresses "a" and "q". In order to distinguish these it appears necessary to distinguish between "l" and "1". The ambiguity of "o" and "0" can be retained without causing difficulties.

Among the remaining characters the tab and carriage return will be synonymous. The space, the color shift, the nullify, and the stop characters will be ignored, as will be combinations of the shift to upper case and shift to lower case characters which do not affect the appearance of the print of the program being translated. The back space character and all code combinations not corresponding to keyboard characters will be considered illegal and will produce a post-mortem.

The words in the vocabulary of the translation program will, for the most part, be composed of combinations of lower case letters and digits suitably punctuated and terminated.

B. Words

Only two types of words are translated so as to occupy registers in storage of the 1103: instructions, which may have no , one, two, three, or possibly more address sections; and (integer) numbers. This class of words we will call polysyllabic storage words. Each word in this class has, when translated, a 36-bit value.

Another class of words, called polysyllabic control words, consists of current address assignments, floating address assignments, preset parameter assignments, and starting address assignments. None of the words in this class actually occupy storage locations in the translated program, but they influence the form, location, and operation of the translated program. The value of these words may have either 15 bits or 36 bits.

The words in both of these classes are combinations of these syllables: operations, floating address tags, preset parameter tags, integers, the single letters "q" and "a", and the relative address tag "r". The words are distinguished only by their internal and terminal punctuation (except that the starting address word always has the initial syllable "START AT"). The punctuating characters are "+", "-", "=", ",", ".", "|", and the tab and carriage return.

A third class of words, called special words, are used for various control and identifying functions. They are the title, number base indicator, and, possibly, the ditto words. Only the number base indicator will differ from the corresponding CS word; it will be of the form "BASE k", where k is some integer like 8 or 10 (always to base 10). All the words in this class will be terminated by a tab or carriage return.

C. Syllables

1. Operations. These are mnemonic, lower case, two-letter pairs of characters corresponding to the operation code listed in the 1103 code books. The binary value of the translated code will occupy only the left six bits of a storage word. Two-letter pairs not corresponding to defined operations will produce a post-mortem.

2. Floating Address Tags. These are single-letter-and-number combinations. All of the allowed combinations are listed here: a1,....,a200,b1,....,b200, c1,....z200. (Note that "o" is excluded as an initial letter.) This gives 5000 possible flad tags, all of which may be used in a given program or group of programs translated at once. The

value of a flad tag will normally be a fifteen-bit address, and a post-mortem will result if such a range is exceeded. A flad tag can have only one value throughout a program; assignments which yield more than one value for a flad tag will produce a post-mortem, but the translation will be completed using the last assigned value.

3. Single Letter Address Tags. The letter "a" will have the value 20000(octal) and the letter "q" will have the value 10000(octal) as expected. The letter "r" will designate the relative address (in routines and subroutines written relative to zero), and may be assigned and reassigned whenever desired.

4. Preset Parameter Tags. These are two-letter-and-number tags. All of the allowed combinations are listed here: zz1,...,zz50. Preset parameters are given 36-bit values, whether they are used in fifteen-bit or smaller addresses, or in 36-bit words. The value of a preset parameter tag always refers back to the last assignment of a value to that tag, and a tag may be reassigned, within rather large limits, as often as desired.

5. Integers. No use of a radix point is made in integers. Negative integers are indicated by prefixing a minus sign. Ordinarily, integers used as (part of) an address will range from 0 to $2^{15}-1$, and integers used as numbers will range from 0 to $2^{35}-1$. However, a greater range (as yet undetermined) will be allowed to permit greater freedom in forming polysyllabic word values. Integer syllables are always converted to binary from the base specified by the last "Base k" special word appearing in the program. If no such word appears, the base is assumed to be decimal.

D. Polysyllabic Storage Words

1. Instructions. The structure of a typical instruction word in the 1103 is a six-bit operation followed by two fifteen-bit addresses. However, some instructions may have no, one, or three addresses (in the last case there are one three-bit, one twelve-bit, and one fifteen-bit addresses).

Instructions when written by a programmer will always have the following form: the first syllable will always be the mnemonic code for the operation. This will be followed, with no intervening non-ignored characters, by the several addresses, each separated by commas and the last followed by a tab or carriage return. Each address will be written as the sum of floating address tags, single letter address tags, preset parameter tags, and integers; the sum being taken in the obvious algebraic sense indicated by prefixing each syllable of the address by a plus or minus sign. This prefix must be indicated, except that the plus sign may be omitted when no ambiguity results. If the value of an address is zero, then the address may be omitted as long as the necessary punctuation is retained.

Examples are as follows (here "↲" denotes a carriage return or tab):

A uv instruction:	tu 6a97+zz3-4-h7,b4+r↲
A -v instruction:	ef 0,4b6↲ or simply ef ,4b6↲
A j-v instruction:	mj 2,0,zz3↲ or simply mj 2,,zz3↲
A -- instruction:	fs↲
A jnv instruction:	rp 1,zz34-5,q85↲
A jn- instruction:	am 0,15,0↲ or simply am ,15↲

Note that only the appearances of the commas determine the positions of the addresses in the translated word. Also the binary value of the translated address must fit into the number of bits available in the 1103 word.

2. <u>Numbers</u>. A number has the form of a single address of an instruction word, i.e., is the algebraic sum of floating address tags, single letter address tags, preset parameter tags, and integers. No periods are used and the last syllable of the word is followed by a tab or carriage return. The word is translated so as to have an integer value in the normal 1103 form. No provision is made for positioning the number to appear as a j or u address; this can only be done by making a fictitious instruction with the operation and other addresses having a zero binary value.

Examples of numbers follow:

3456↲

6q7-zz18+94h5-a68-2r↲

q↲

Note that the binary value of the translated number must be less than 2^{35} in magnitude.

E. Polysyllabic Control Word.

1. Current Address Assignment. This word has the form of a single address of an instruction word except that here the terminating character is a vertical bar: "|". This word causes the next polysyllabic storage word to occupy the register in 1103 storage, electrostatic or drum, whose address is the value of this address assignment. Successive storage words will go into successive registers until another current address assignment occurs. The values of "q" and "a" are not allowed as the value of a current address assignment.

Examples are:

2067|
4g7+h6-b39+zz8+r|

Note that the value of a current address assignment must be less than 2^{15}.

During the translation process an account is kept of the location to be occupied by the next storage word in the program being translated by means of the current address indicator. It is clear that the current address indicator is reset by each current address assignment and is indexed whenever a storage word occurs.

2. Floating Address Assignment. This word again has the form of a single address of an instruction word except that here the only syllables allowed are integers and a single flad tag (that of the flad being assigned), and that here the terminating character is a comma: ",". The value assigned to the flad is the value of the current address indicator minus the sum of the integers in the assignment.

Examples are:

f6, assigns f6 to have the value of the current address indicator

10+7f6-4, assigns f6 to have the value of the current address indicator minus 13.

Note that the assignment of a flad cannot be circular, i.e., we cannot assign f6 when the current address indicator is already defined explicitly or implicitly in terms of f6.

3. Preset Parameter Assignments. This word has the form of a flad assignment except that here only integers and a single preset parameter tag are allowed (that of the preset parameter being assigned), and that the terminating character is an equals sign: "=". The value assigned to the preset parameter is that of the storage word which immediately follows the "=". (No characters of any sort except spaces can intervene.) This storage word does not occupy a register of storage, and this is the only case in which this is true. This value is always considered as a 36-bit word, but if used in a 12-bit n address, for example, then something less than 36 bits will probably be used in the determination of the value of the address.

Examples are:

zz1=5
1760-47zz34=mp h7,4q3

Note that it is impossible to have a circular definition of a preset parameter, for if a preset parameter is defined in terms of the same preset parameter, then actually the previous value of the preset parameter (zero if not previously assigned) will be used in the definition of the new value.

4. Starting Address Assignment. A starting address assignment word occurs at the end of each program tape. Its value is that of the address at which operation of the translated program is to be begun when it is eventually read into an 1103. In form the word is written exactly like a number except that the first syllable must always be "START AT".

Examples are:

START AT q4
START AT 3h6+zz5-h8
START AT 100

Note that the value of this word must correspond to an actual storage address.

5. Relative Address Assignment. The relative address is denoted by the single letter "r". However, this letter need appear only where the value of the relative address is requested in a word. A value is assigned to the relative address whenever a floating address assignment occurs, and the value given to the relative address is the same as that given to the floating address. A value will also be assigned to the relative address whenever a word consisting of an integer terminated by a comma occurs; in this case as above the value of the relative address then becomes the value of the current address indicator minus this integer.

Examples are:

6f6,	assigns r as well as f6 to have the value of the current address indicator minus 6
10,	assigns r to have the value of the current address indicator minus 10

F. Special Words

1. Title. The form of a title will be similar to those used in the present WWI input scheme and will probably appear, for example, as

f2r 126-78-999 JONES,

The title will be used to call the translation program in from MT#2 onto the buffer drum and to produce some logging indication. The tape number will eventually be punched in visual form on the binary tape.

2. Number Base Indicator. The number base indicator will be written as "BASE k" where k has one of the values 2 to 10. The base will be decimal unless otherwise indicated by a number base indicator. All integers, whether in instructions, numbers, current address assignments, flad assignments, preset parameter assignments, or starting address assignments, will be converted in the base last indicated.

3. Ditto Word. The special word "DITTO", as defined in CS, might be included in the vocabulary.

G. Flad Table

At the end of each program translation a copy of the values assigned to all the flads in the program will be recorded. If the table is short enough, it will be recorded on the delayed printer, otherwise on the scope. However, if this is too complicated, the table will always be recorded on the scope. Flad values will be printed in the base indicated by the last base indicator.

H. Error Detection

1. Unassigned Flads. A table of the locations of requests for all unassigned flad tags will be printed at the end of each translation.

2. Duplicate Flads. A table of all flad tags which have multiple and differing assignment values will be printed at the end of the translation. Note that the last value assigned to a duplicated flad will be the one actually used in the program. The locations of the flad assignments will be printed with each flad.

3. If a number or address is too big for the number of bits allocated to it, the translation program will stop and print the location of the error.

4. If an illegal character occurs on the Flexo tape, the program will stop at that point on the tape.

5. If a flad is defined in terms of itself, the translation program will stop and print this.

6. If illegal flad or preset parameter tags are used the translation program will print them with their locations.

7. Illegal two-letter combinations will be detected and printed.

All locations of these errors will be printed in terms of the nearest flad. An attempt will be made to print as many errors as possible during a single translation.

I. Output of Translation Program

The translated program will be punched via the delayed punch in the standard hexa-bi-octal form used in the 1103. Otherwise the word sequence structure will be similar to that of WWI 556 tape, unless there already exists a better technique for input on the 1103. If necessary an input program for the 1103 can be written.

J. Separate and Simultaneous Translations

The preceding discussion applied to the translation of a single program. As in CS program tapes can be translated either separately or together. In the first case the translation is considered to be complete within itself. In the second, as many tapes as desired can be translated at once with cross reference of floating address assignments and requests. This facility allows a problem coded in separate pieces to be translated all at once.

K. Further Remarks

Some of the decisions embodied in the above discussion are at the moment rather arbitrary. For example, the difficulty in distinguishing the "lq" and "la" operations from floating address assignments of similar appearance could have been otherwise resolved by making the rules regarding the use of the plus sign more stringent, or simply by using periods instead of commas to punctuate instruction words. More upper case characters could have been used also.

The language as proposed contains relatively few varieties of words. It is hoped that what it does contain can be made to serve as a basis for the language of a translation program which will operate on an 1103. Such a language would contain more types of words, like decimal data numbers, octal numbers, as well as more facilities for coding and trouble-shooting programs. The translation program might also accept programs punched on cards. If such an enlarged system is to be acceptable to more than one group of 1103 users, it would be wise to postpone decisions about these additional facilities to a later date when more information is available about the equipment and needs of each 1103 installation.