Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: AUTOMATIC ASSEMBLY OF PROGRAMS

To: Air Defense and Applications Groups

From: John W. Carr III

Date: April 23, 1952

Abstract: A study of methods of programming with the present and proposed conversion programs shows that most programs, with present methods, can be written in such a fashion that they can be assembled automatically, with no calculation of storage positions on the part of the programmer.

General Discussion

The philosophy of this memorandum is diametrically opposed to that of the programmer who uses octal notation, likes it, and is convinced that this is the most satisfactory way to write a program. Nevertheless, under certain conditions, the method proposed here may prove to be a much more speedy, more easily corrected way to give complete machine instructions. To the programmer who is already using decimal notation and preset parameters, the proposed methods are merely a logical extension of certain devices he has already been using.

This method of assembly, used with the Whirlwind conversion program as of this date (or most probably with any future conversion program) will parallel the general method of coding first proposed by von Neumann and Goldstine (1)*, but which up to now has been impossible to use.

It is very similar to the "free-" or "floating-address" device first proposed by Wilkes at the informal programming conference at the Philadelphia A.I.E.E.-I.R.E.-A.C.M. meeting. (2). With a few changes in the notation of the present control combinations of the conversion program, and the addition of a few extra control combinations, all possible programs should be handled satisfactorily. With the present conversion program, any program not using multi-register number storage can be assembled completely automatically. Programs using multi-register number storage will still require a small amount of program layout on the part of the programmer, at least until one or two more control combinations are added to the conversion program.

* Numbers in parentheses refer to the Bibliography at the end of the paper.

The proposed method of assembly is most useful for programs that use subroutines from a given subroutine library. However, with a conversion program that provides sufficient preset parameters, it can be very useful in programming any program, whether or not it uses subroutines.

Method of Assembly

In the original octal (and later decimal) programming methods, each storage register in the machine was known by a preassigned number, which was the direct equivalent of the binary number appearing in the machine program counter. Under the new proposed method, storage locations are to be known in the outside programmer's language by "names," which will have no direct connection with the corresponding address in the machine, except through the translation or conversion program itself.

Thus, for example, at present the first address of a print-subroutine is known on the outside as x5. This actually implies that the address of the subroutine is stored in the x5 parameter, and upon read-in any address printed on tape as x5 will be converted to yield the machine address stored in preset parameter x5.

However, up until now, programs converted from paper tape have not been assembled automatically, even though the preset parameter technique has been used to some extent for addresses. What has been needed is the use of a "two-pass" system of paper tape read-in. Such a system was originally planned for the next conversion program, using magnetic tape, but lack of a suitable line-by-line photoelectric reader made postponement necessary. What was not immediately understood, however, was that a two-pass system was possible with the present paper-tape direct conversion program. Now that an experimental program has been written and operated using the proposed automatic assembly method, a "two-pass" paper tape method may become useful at least until a "two-pass" or "multi-pass" magnetic tape scheme is perfected.

The basis of the method is as follows: main program, subroutines, constants, and temporary storage registers are stored indiscriminately (with the exceptions noted below) one after another in storage. The addresses of any given register (such as first address of a subroutine, or any register referred to by another instruction) may be given a "name" corresponding to one or the preset parameters. It is then referred to, by that preset parameter, in the normal fashion. This address is thereafter referred to by this "name." Instructions, constants, subroutines, etc., are stored consecutively in the machine, so that there are no gaps in the used registers in the machine.

However, since some programs of necessity refer to addresses which have not yet been assigned to the corresponding preset parameters, two "passes" along the tape must be made. The first read-in automatically determines the position of each register in storage. When any register

has a "name" given by the corresponding parameter, this value is automatically assigned. On the first read-in, some of the addresses corresponding to their "named" registers will be incorrect, because they refer to preset parameters that have not yet been assigned addresses. On the second read-in, however, after the correct addresses have been assigned to each preset parameter in use, the correct addresses will be inserted into the registers calling for addition of the required preset parameters.

Single Length Assembly

Because assembly of single-length programs does not involve calculation of the separation constant K and use of the present special multi-length input programs, we will consider such an assembly first. The following rules will apply:

1. Temporary registers will be assigned last, so that the question of how many temporary registers are needed will not enter (except in the question of whether storage has been exceeded).

2. All other storage may be placed in any sequence.

3. Any address which is referred to by any instruction may be assigned a preset-parameter "name."

4. All other addresses are considered to be "nameless," although they may be referred to if needed.

5. Assignment of registers is made automatically by control combinations as follows. The "current storage address" is placed in the relative address parameter storage (r storage) by the control combination "0/." The current address is then transferred to the particular preset parameter storage by another control combination. For example, if a particular address is to be "named" by parameter v1, then the current address would be stored by means of the combination

   0/v1/p0r.

   At the end of this sequence on the tape the current address would be stored in the v1 parameter, since v1/p0r transfers the contents of the relative address storage into v1.

An example of a single length program coded by this automatic assembly method, and using the "floating-address" scheme, is the tape T-1070-1.

```
ifkm  10,p1,p0,p7,p0,p31,p1    ,i32
```

ifkm: 5-5-6 form for sp 844 (control to direct conversion)
10,: Decimal addresses, erase storage
p1,p0,p7,p0,p31,p1,: T 1070-1
i32: Begin storing at register 32
} These control combination: print the tape title

```
f10
```

f10: Return control to 5-5-6 input program at reg. 10

```
ifkm  10/i32
```

ifkm: 5-5-6 form for sp 844 (control to direct conversion program)
10/: Decimal addresses, do not erase
i32: Begin storing at reg. 32
} Second input title for second pass with tape

```
0/vx1/p0r.
```

Put current address (32) in preset parameter vx1.

```
caax2
daax3
0/vx3/p0r.
```

Put current address (34) in preset parameter vx3.

```
ca0
spax4
aoax3
suax5
cpax3
rs0
0/vx4/p0r.
```

Put current address (40) in preset parameter vx4.

```
v1/tal9r....      1/ts25r....
```

{Subroutine fo printing integers.

```
48/p49
0/vx6/p0r.
```

Put current address (89) in preset parameter vx6.

```
p0
p1
p2
p3
p4
p5
p6
p7
0/vx5/p0r.
ca7ax6
0/vx2/p0r.
pax6
0/v/p0r.
```

Put current address (97) in preset parameter vx5.
Put current address (98) in preset parameter vx2.
Put current address (99) in preset parameter v.

```
fax1
```

Return control to the address stored in preset parameter x1.

This program is supposed, as a test, to print out 0,1,2,...7, in that order. It was coded successively from beginning to end, without the necessity of assigning any numerical addresses to storage. After the first "pass" of the tape through the photoelectric reader, the following will be stored in the machine:

```
32  ca 0
33  td 34
34  ca 0
35  sp 0
36  ao 34
37  su 0
38  cp 34
39  rs 0
40 - 88 (subroutine) .
89  p0
90  p1
91  p2
92  p3
93  p4
94  p5
95  p6
96  p7
97  ca 96
98  p89
```

The contents of the preset parameters storage will be as follows:

```
vx1/p32
vx2/p98
vx3/p34
vx4/p40
vx5/p97
vx6/p89
  v/p99
```

Thus at the end of the first "pass," all preset parameters will have the correct value.

For the second pass, the tape must be read in starting immediately after the f10, since at present the conversion program uses preset parameter storage for printing the tape number, which would write over whatever had been stored on the first pass. At the end of the second read-in, storage would be as follows:

```
32  ca 98
33  td 34
34  ca 0
35  sp 40
36  ao 34
37  su 97
38  cp 34
39  rs 0
40 - 88 (subroutine)
89  p0
90  p1
91  p2
92  p3
93  p4
94  p5
95  p6
96  p7
97  ca 96
98  p89
99...(on) temporary storage d,lt,...
```

Corrections

This formulation is very easy to correct, in case of an error, both during programming and on the tape. For example, if during programming it is found that several registers have been omitted, they can be inserted directly in the proper place in sequence, without having to renumber all the successive registers and their addresses. For example, in the program above, if the programmer wanted to stop printing at the first negative number to be called in, immediately after the caax2 he could insert cpax7, and before the rs0, 0/vx7/p0r. Aside from this, no renumbering would be necessary.

Double Precision and Floating Point Numbers

As of the time of writing, the situation as regards read-in and conversion of Double Precision and Floating Point numbers is not helpful in automatizing storage in those cases where those types of numbers and interpretive subroutines are used. At present, two conversion programs are necessary, one for single-length words, and a second for multi-length number storage. The second input program at the present time also unfortunately uses preset parameter storage v1,...v15 as temporary storage, and thus would change the contents of those registers arbitrarily on a second "pass." Finally, because the final portion of storage is used for the various conversion programs, and temporary registers must now be double- or multi-length, other difficulties are inherent.

For this reason, we shall present an automatic assembly plan making use of the new magnetic tape conversion program, now being written. This program has the important properties that preset parameter storage is not used to store characters to be typed, or as a temporary storage. It will also have more preset parameters available for use.

The new conversion program will contain a "multi-length current address" register, which is tentatively scheduled to be the vx1 parameter. Thus, to give a multi-register number a "name," the following control combination could be used to place the "multi-length current address" into the vx5 preset parameter, for example:

vx5/pax1.

The indexing of the "multi-length current address" is thus done automatically.

The problem of temporary storage, which was solved, or more truthfully avoided in the single-length case by placing the temporary registers last, is a stumbling block in the multi-precision case. The separation constant can be calculated only when the number of multi-length registers depends on the number of temporary registers. However, temporary registers, at present, may be either single- or multi-length, and there is no control combination or automatic section of the conversion program that can determine the maximum number of temporary registers. Until such a change is made in the input program, the following scheme is proposed:

1. Multi-length constants, of every sort, must be stored in one block, last in the automatic assembly section of storage.

2. Temporary registers must be placed <u>last</u> in the group of multi-length registers.

3. Any address which is referred to by any instruction may be assigned a preset parameter "name."

Single-length words will be assembled first, with the exception of temporary storage, which will be considered to consist of all multi-length storage. Following the last single-length word, a preset parameter "name" should be given to the first multi-length address for some preset parameter (such as vx4, for example) by the control combination

0/vx4/p0r vxl/pax4.

This then sets the beginning of double-length storage to the proper place. Any number referred to later on in the group of multi-length storage can be "named" by

vx5/paxl....,etc.

Following the last multi-length number, the separation parameter is calculated automatically by

vx2/paxlsx4,

where vx4 is the parameter that "names" the first multi-length register. Thus vx2 contains the difference between the contents of the vxl and vx4 parameters.

However, if temporary storage is to be included, the vx2 parameter should be calculated automatically by

vx2/pmax/sx4,

where m is the maximum number of temporary registers used.

We can thus give a short example of automatic assembly of a multi-length program as follows, assuming now that there will be no use of preset parameter storage for other purposes. Under the present multi-length separation scheme, m would have to equal "maximum number of temporary storage registers used plus total number of floating subroutine (vxl) storage used."

```
10, Title -----, i32
0/vx3/p0r
ca 0
td ax4
  :
  :
0/vx4/p0r.
ca 0
ts ax5
  :
  :
etc.
0/vx6/p0r.  vxl/pax6.
```

Single-length words

```
Subroutines with double
length constants
gaxl
p0
gaxlax2
n0
vl00/paxl.  vxl/plal00.
vx7/paxl.
+.1/+1
  :
  :
vx8/paxl.
+.5/+1
  :
  :
v/paxl.
pmaxlsx7.
fax3
```

Double-length numbers

Temporary storage

Cautions in Use

At present, the preset parameter scheme is not completely satisfactory for such a scheme, since there are not enough permanent (vx) parameters available for a complete assembly. Use can be made of the temporary (v) parameters, but it should be noted that many of these are changed during read-in of subroutines.

The programmer should also guard against other changing of the preset parameters, such as on read-in during the printing of the tape title, and on read-in of multi-length numbers with the present (special) multi-length conversion program.

A good rule to follow until some specific final policy is made concerning preset parameters, is that only vx parameters may be considered unchanged during the course of a program. However, other parameters may be used if the subroutine situation is investigated.

It also may be possible, in many cases, by judicious arrangement of the program's components, to make the assembly process automatic on only one pass. This requires that the maximum number of temporary registers be known, and that all preset parameters referred to in instructions have their addresses assigned previously.

New Control Combinations

Several new control combinations are obviously called for if this scheme is to be completely useful. One is a single combination to replace the group 0/vx4/p0r., for example. This might be "kx4," meaning "keep the current address in the x4 parameter."

Another might be incorporated in the f control combination. This would check to see if the current address were still less than or equal to a "final available register address." If it were not, the conversion program in some fashion could say,"not enough available storage."

A useful change would be to set up a third set of assembly parameters that would replace v, vx, vx1, vx2, vx3, etc., just as r is not now included among them. The "floating address parameters" would not then be cluttered up with parameters basically used for other purposes.

Finally, the question of temporary storage remains. One way to solve it would be for the conversion program to store the "name" of the largest temporary register in some "temporary storage length parameter" and to check each successive temporary register encountered to see if its name is larger. This value could then be inserted automatically in the proper place in place of the m mentioned above.

Another way would be to store the number of temporary storage registers used by a subroutine as one of the preset parameters on the subroutine tape. The conversion program could note the largest of these.

On multi-length programs, some decision must be made concerning the assignment to single- and multi-length temporary storage. One solution is, for all multi-length subroutines to use all temporary storage registers multi-length, numbering the single-length temporary registers d, dax2, lt, ltax2, etc., successively, so as to save storage space.

The use of the d temporary register, with its exceptional character, should perhaps be reconsidered, since it prevents use of an Ot multi-length temporary register, and is therefore often wasteful of storage.

Special Aids

Because this method of assembly makes use of a completely different nomenclature for registers outside of the machine, it is important that satisfactory retranslation schemes be made available. At present there is no post mortem program that can yield a suitable retranslation. However, it is apparent that with certain provisos a suitable retranslation post mortem might be easily written.

It might thus be useful to reclassify the preset parameters into further subdivisions, for example, "single-length constants," "subroutines," "multi-length constants," "main program." Then addresses which occurred within a certain block (between two particular preset parameters) could be reprinted exactly as they were put in. An alternative method is to include the form of each word with its storage, for example.

For the moment, two tapes entitled "Print Out Preset Parameters," T-1132 and T-1133, are available in 5-5-6 form. After read-in of the program, with the direct conversion program, the tape is read in over the conversion program and prints out the preset parameters in a standard form. The space is then available for the multi-length interpretive and print program, if necessary. Tape T-1032 is read in over registers 32 - 200, Tape T-1033 over registers 734-on (the conversion program).

Conclusions

At present, single-length automatic assembly is possible, with, however, no checks on overflow of storage. Multi-length completely automatic storage awaits the new conversion program and further additional control combinations, as well as decisions on temporary storage assignments.

All such automatic assembly methods depend on two passes of the paper tape, or judicious placing of components. Final automatic assembly in the most efficient fashion must await a "two-pass" magnetic tape or magnetic drum scheme.

Bibliography

(1) Goldstine, H. H., and von Neumann, John, "Planning and Coding of Problems for an Electronic Computing Instrument," The Institute of Advanced Study, Princeton, N. J., Parts I and II, Vol. III, 1947.

(2) "Review of Electronic Digital Computers," Joint A.I.E.E.-I.R.E. Computer Conference, (A.I.E.E., Feb. 1952), pp. 113-114.

Signed: John W. Carr III
John W. Carr III

Approved: C W A
C. W. Adams

JWC:mas