J. E. Griffith

January 4, 1956

Attached are two memos which relate to Stretch, and if they had been written more recently, they would have been Stretch memos. They are, therefore, being distributed to the people who receive Stretch memos.

Nathaniel Rochester

IR-00038
November 21, 1955

Suggested Maintenance Techniques for Stretch

N. Rochester

A modern large automatic calculator cost so much that it is not economically sensible to allow it to be out of service while someone as slow as a man tries to figure out what has gone wrong with it. Only another large calculator is fast enough to locate the trouble soon enough. Stretch contains three automatic calculators, the arithmetic calculator, the control calculator, and the input-output calculator. It would be very nice if any one of these could service any one of the others.

Most calculators have an indicator attached to each trigger so that the customer engineer can determine visually what state it is in while he is servicing the machine. To provide an equivalent feature for the machine these triggers of the serviced calculator should be arranged in registers which can be read by the servicing calculator. Furthermore, the servicing calculator should be able to advance the serviced calculator step by step at whatever rate of speed is convenient. This would allow the servicing calculator to study the action of very small parts of the serviced calculator.

It might also be desirable for the servicing calculator to store any numbers it chose in the trigger registers of the serviced calculator. In this way it could set up better tests of the action of some parts than it could get by merely observing these parts in action.

The servicing calculator should be able to control marginal testing of the serviced calculator while its own voltages, frequencies, etc. should stay normal. This would allow fully automatic efficient marginal testing.

If the machine can be made to be fully automatic so as to allow unattended operation at night the marginal testing could be done at undesirable times of day. Then at some more reasonable time of day the potential faulty parts could be replaced and the machine could quick check to see that they were satisfactory.

This same maintenance technique could be used on a smaller machine than Stretch. If input-output control was carried out by a general purpose automatic calculator it would probably cost no more than special purpose controls. Such input-output control would have the added advantage of allowing the type of maintenance described above.

IR-00034
November 14, 1955

A Stored Program Calculator with A Simplified Control

N. Rochester

The programming tricks used to operate a calculator often suggest improvements which should be incorporated in subsequent machines. A feature of the PACT system of programming the 701 is that one ordinarily does not have to bother to write the steps needed to save the results of a calculation. For example, a simplified version is illustrated by the following program:

```
1   R ADD    u
2   ADD      v
3   R ADD    w
4   DIVIDE   R (2)
```

This program would calculate

$$\frac{C\,(w)}{C\,(u) + C(v)}$$

The address R (2) means "the result of step 2". The equivalent program in machine language would be

```
11  R ADD    u
12  ADD      v
13  STORE    x
14  R ADD    w
15  DIVIDE   x
```

The extra step, 13, is implied by step 4 so is actually unnecessary in specifying what is to be done. This paper describes an example of a calculator in which this programming trick is, to some extent, built into the hardware.

The first step is done by an assembly program and that is to mark the add instruction to indicate that the result of that instruction should be stored. This step is actually a little more complicated than this because sometimes the automatic storage feature won't work so the assembly program, realizing this, would put a store instruction into the program. However, discussion of this will be deferred until later.

Another facet of this proposal is that it is necessary for the calculator under control of the assembly program to determine in some way that certain memory addresses are vacant and that it can safely store the result in it. Probably this can be done by the assembly program, but if not, it can be done by the programmer. The symptom of this is that in the machine language program,

certain addresses are marked with an indication that the number from that address will not be needed again.

Another thing that the assembly program will need to do is to calculate the address that the machine will choose for the result and insert this address later in the program wherever that result is needed again. How it does this will be evident when the machine behavior has been described.

There are several registers associated with the accumulator, and these receive any numbers calculated under the control of instructions which are marked for storage. These registers hold the results until an opportunity arises for storing the results.

Such an opportunity arises whenever the address part of an instruction is marked with an indication that the number from that address will not be needed again. Seizing this opportunity, the calculator slips the result into memory on the second half of the memory cycle that was used to secure the number that will not be needed again.

What has been accomplished is that the result has been stored without requiring either a memory cycle to get the store instruction or a memory cycle to do the storing. This would be of great importance in iteration loops which constitute a major portion of some problems. Furthermore, the space in memory needed by certain store instructions has been saved. This could be of crucial importance if an iteration loops was stored in the transistor registers of Stretch. It could also be important if the assembly program could manage this problem well enough by itself to save a significant fraction of the space needed for the program.

Whether the vacant addresses have been marked by the programmer or by the assembly program, the assembly program will have to decide whether to use a store instruction or to mark the preceding instruction to indicate that the result is to be stored. It will decide this on the basis of whether or not there will be enough room in the registers for the result. There will always be enough room if this preceding instruction is also marked to indicate that the contents of the indicated address are no longer needed. There will always be enough room if one of the registers is empty. If neither of these conditions holds, a store instruction is needed.

Some policy must be established as to which result to store first. Regarding the registers as a temporary storage, one possible policy would be: first in, first out. Another possible policy is: last in, first out. It might even be worthwhile to have the order dependent on program control. The choice between these will have to be made on the basis of a fairly complex example because in any simple example the three registers attached to the accumulator could retain any such intermediate results.

In Stretch, the control computer could probably manage this principle with little or no extra hardware. The transistor registers could serve to retain numbers to be stored until a suitable occasion arose. It could be argued that Stretch cannot benefit from such a scheme because a fundamental principle of Stretch is that the control computer must be able to furnish the high speed arithmetic unit with its operands and dispose of the results well enough so that the high speed arithmetic unit is never idle. However, this objective has been achieved only for problems with enough floating point work, especially multiplication, and even then, at a cost of a very expensive 2 usec memory. This new scheme will reduce the memory access requirement and hence either speed up the machine or to reduce the cost of memory.

Nathaniel Rochester