INTER-OFFICE
MEMORANDUM

Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

To: Group 61

From: Peter Cioffi

Date: August 6, 1952

SUBJECT: CODING NOTES, I

Introduction

This paper describes a number of techniques for coding various minor routines (sequence of orders to perform a certain function) that are sometimes used in main programs. The particular schemes included here are in no official way standard methods for performing these functions; however, they have been used in the past successfully so that it seems worthwhile to present them to beginning programmers. The collection here is hardly exhaustive, but it is intended that this presentation will in some measure assist beginners in getting accustomed to programming techniques.

1. Sensing to indicate that an event has or has not occurred

A "cue" register may be used to indicate by the sign of the number stored in it that an event has or has not occurred. The event may be the outcome of some previous routine, e.g. a "yes" or "no" decision. Two methods are described.

a. Some register (x) may be set aside permanently in a program and used as a "cue" register. This register is reset to one state or another (positive or negative) depending on the result of a routine (R), the assignment of the meaning of positive and negative states being arbitrary. To select the proper course of action depending on the result of the routine R, it is only necessary to sense the sign of register x. i.e.

```
ca x
cp y -
______
______
```

where the cp either continues the program in register y or is inactive according to the sign of the number in x.

b. If in a program a register (x) containing a positive constant (k) is used only as the address of cm or ca orders, this register may be used as a "cue" or sensing register. This register is reset to + or − k depending on the result of some routine R and a "cue" register sign convention.

When sensing, it is necessary only to

```
ca x
cp -
```

If it is desired to reset the "cue" register to + k immediately after sensing, the sequence of orders should be

```
cm x
ex x
cp -
```

Notice that for the second of the methods above it is necessary to avoid going through any of the ca x orders in the program when x contains − k rather than + k because of its use as a "cue" register. This uncertainty can be eliminated by changing all the ca x orders to cm x.

2. Economy of storage

Since a stored word can be regarded either as a command or number, frequently an order in a program may also serve as a constant. For example, ts 0 $\equiv \frac{1}{2}$, qh 0 $\equiv$ 3/8 and ca 0 $\equiv -1 + 2^{-15}$.

Sometimes a constant can be constructed from program orders whose address sections may be arbitrary or partially arbitrary, particularly those orders where addresses specify selective roundoff and shifting.

On occasion it may be desirable to use an order which does nothing such as in modifying an existing program so that an order is rendered ineffective. Any of the following orders may be used without affecting the input-output equipment or the contents of the accumulator, B register or storage registers:

td –, ts –, ta –, cp or sp to next register, cl*0, sr*0, sl*0, ad 0.

In addition, any of the following orders may be used if the contents of the accumulator, A register and B register may be disturbed:

cl – –, ca – –, cs – –, cm – –, sm 0, sa – –, mr – –, mh – –, sl – –, sr – –, sf –.

Some of the addresses of the orders in the above two groups are marked by either one dash or two dashes. The addresses marked by two dashes may be any address; those marked by one dash can be any of the addresses for the registers in test storage (decimally 0 to 31) but excluding the flip-flop storage registers, at present comprising registers 8,23,28,29,30. The test storage registers with the exceptions mentioned just above are at present set up electrically such that reading into any one of them through the transfer bus is impossible.

These registers are controlled by toggle switches in the computer room (not the control room) and, at the present, their contents cannot be altered by means other than manually changing toggle switches.

3. Complementing a number in the accumulator

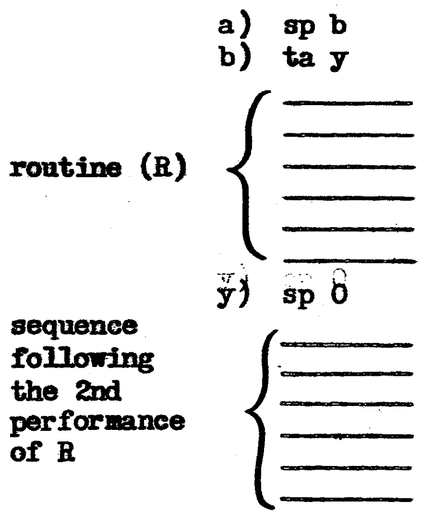
The complement (- n) of a number (n) in the accumulator may be obtained by mr RC ca0 where ca0 is equivalent to $-1 + 2^{-15}$. If $\lvert n\rvert \leq \frac{1}{2}$, the exact complement will be in the accumulator, otherwise it will be in error by 2^{-15}.

Arithmetically, the computer operates as follows: $\lvert n\rvert \times \lvert 1 - 2^{-15}\rvert + 2^{-16}$ is formed first, the sign being affixed later. (The computer multiplies positive numbers only.) The term 2^{-16} is an addition due to the round-off action of the mr order. Since n is in the range $-1 + 2^{-15}$ to $1 - 2^{-15}$, $\lvert n\rvert \times 2^{-15}$ will be a number in the B register, and $-\lvert n\rvert \times 2^{-15} + 2^{-16}$ will detract nothing from $\lvert n\rvert$ in the accumulator only if it is equal to or greater than zero or, what is the same thing, if $\lvert n\rvert \leq \frac{1}{2}$.

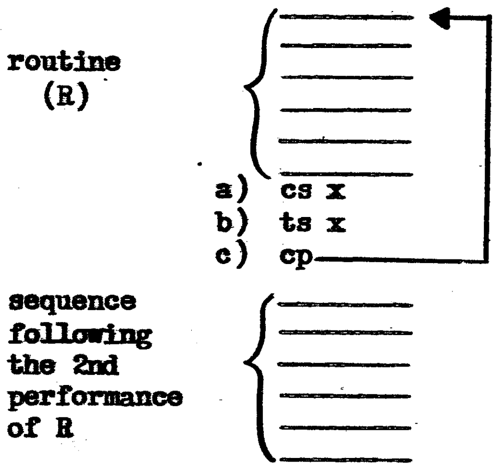
4. Counters of two

The following two methods provide that some routine R be performed twice.

a.



b.



where register x contains a positive number not called for in routine R except, possibly, by cm orders. This number in x, however, could be used elsewhere in the program since after the completion of the sequences involving R the original contents of register x is restored. The idea expressed in the method under a can be extended for constructing counters of n.

5. Printing with the Flexowriter

a. Printing numbers

To print any digit 0 through 9 with the Flexowriter, obtain the numerical value of the digit (in the desired base) in the accumulator and add to this the address of the first register of the stored table of Flexowriter codes for the digits $0 \longrightarrow n - 1$ (n is base) to form the address of the register containing the code for the digit to be printed.

The following example illustrates how the first three digits of the number $n \times 2^{-7}$ in the accumulator is printed in base ten.

```
 0)  si 224                           50)  10^-3 x 2^7
 1)  mr 50                            51)  10 x 2^-15
 2)  ts 63                            52)  ri 53
 3)  cp 65                            53)  Flexocode for 0
 4)  cm 63                            54)       "      " 1
 5)  mh 51                            55)       "      " 2
 6)  ad 52                             .
 7)  td 10                             .
 8)  sl 15                             .
 9)  ts 63                            62)       "      " 9
10)  ca 0                             63)  ri 0
11)  rc 0                             64)  n 2
12)  ao 64 }  negative counter to     65)  ca 68 }  this sequence
13)  cp 4  }  determine when three    66)  rc 0  }  prints a minus
              digits have been        67)  sp 4  }  sign for n negative
              printed                 68)  Flexocode for (-)
```

6. Comparison of two numbers

The following scheme can be used to determine if two numbers a & b are equal:

```
0)  cs x                      x)  a
1)  sa y                      y)  b
2)  ts z                      z)  ri 0
3)  ca z
4)  ts z
5)  cm z
6)  su 0
7)  cp ----------> (a = b)
     |
     v
  (a ≰ b)
```

The only arithmetic operations which result in a positive zero (0.000000000000000) in the accumulator are listed below; all others whose result is a zero leave it in its negative form (1.111111111111111) in the accumulator.

1. addition of plus 0 to plus 0

2. subtraction of minus 0 from plus 0

3. multiplication or division of zero by anything, where the two numbers are of the same sign.

7. Uses of Flip Flop registers

It was mentioned earlier that the registers in test storage, with certain exceptions, can not be read into. These exceptional registers have decimal addresses 8,23,28,29 and 30 (or in octal 10,27,34,35 and 36) and are called Flip Flops 3,0,4,1 and 2 respectively. These five flip flop registers behave like any of the ES storage registers electrically, that is, they can be read into and out of. They have the added features of being provided with sixteen toggle switches (one per register digit) each so that their contents may be changed manually, and with lights, again sixteen each, which display their contents. These flip flop registers may be reset also electrically during the operation of a program by programming and by setting certain reset switches associated with the flip flops. Such a provision as this allows a flip flop to be used for storing two numbers (at the same time).

The following example illustrates how this is accomplished using an rs reset. Say that FFO (address 23 decimal) is used as a register to store and display the result of a computation, during the operation of some real time problem. This same flip flop may still be used to introduce some other constant (k) to the program. If the toggle switches of FFO are set up to read the value k, then the following scheme will enable the FF to produce this constant k and then to recover the constant which was stored and displayed there previously. The rs reset switch for FFO and the "no switch to push button" (No SW to PB) switch are turned on. The former of these switches resets FFO to the toggle switches value every time the computer comes to an rs order; the latter inactivates the stop feature of the rs order (see WWI Order Code).

```
ca 23         puts original contents of FFO in ac
rs 0          resets FFO to value set in switches
ex 23         returns the original contents of FFO
              and puts k in the ac
```

The five FF registers may be reset also by Time pulse three (TP3), and Program Counter End Carry (PCEC) switches associated with the FF registers. The TP3 reset action occurs during the operation of every program order so that the effect of this reset is to hold the contents of the FF register fixed to the toggle switches value. The PCEC reset action occurs whenever the program counter register has a one in at least all the digit positions 9 through 16. This will occur whenever the program counter reads (377 + 400k) where k = 0,1, 2, ..., (octally) and max k being determined by the machine storage capacity.

POC/efh