April 9, 1958

INTERNAL NOTE #29

Subject: Preliminary Specifications of a Basic Computer Coding System

The coding system we propose for the Basic computer is based on the philosophy that the most applicable coding system for any computer is one directly related to the machine it is meant to serve. Therefore, the makeup of instructions for our proposed scheme will have a pattern similar to that drawn up for instruction words by the Basic machine planners.

Basic computer instruction words come in two sizes, a half-word and a full-word. A half-word instruction is composed essentially of an operation, an address, and address modifiers. The full-word instruction is composed of the previously described half-word plus an additional half-word for specifying essentially a second address and address modifiers. An easily recognizable difference then exists between these two possible types of instructions. For both types there will be a primary address, but in addition, the full-word instruction will have a secondary address. Therefore, half-word instructions will be grouped as single address instructions and full-word instructions will be grouped as dual address instructions.

Single address and dual address instructions can be further subdivided into six instruction categories each. (These categories have already been defined by the Basic machine planners.) Among the instruction categories we find that the registers affected by instruction operations are either specified, implied, or non-existant. (These registers may be the accumulator, accumulator sign byte, mask, indicator, or index registers.) If the registers are non-existant or implied, there is no need for coding. If the registers are specified, there is need for coding. Therefore, a classification with respect to implied and specified registers should exist for these instruction categories.

Our proposed coding system for the Basic machine now takes shape. There will be two groups of instructions - single address and dual address. Each of these groups will be subdivided into an implied register class and a specified register class. Under each class will fall two or more of the six instruction categories of each instruction group. The outlined organization is as follows:

I. Single Address Group

  A. Implied Register Class

    1) Floating Point Instructions
    2) Miscellaneous Instructions

  B. Specified Register Class

    1) Direct Index Instructions
    2) Immediate Index Instructions
    3) Count and Branch Instructions
    4) Indicator Branch Instructions

II. Dual Address Group

  A. Implied Register Class

    1) Transmit Instructions
    2) Input-Output Instructions
    3) Store I. C. and Branch Instructions
    4) Branch on Bit Instructions

  B. Specified Register Class

    1) Integer Instructions
    2) Connection Instructions

A discrepancy may be apparent in placing integer instructions under the specified register class but as we shall see later, integer instructions are more suited to this classification. Now within this framework we can establish a system for expressing instructions.

The tenets for expressing instructions in our proposed coding system are these. Instructions may be expressed with numeric or mnemonic symbols or actual binary notation. Symbolic notations will be unrestricted as to length whether these notations represent operations, addresses, index registers, or special modifiers. The coding format will be free of pre-established instruction subdivisions and reliance will be placed on punctuation to denote the intent and extent of instruction notations. For those coders who insist on precisely defined coding fields for expressing instructions, they have only to follow the same instruction expression order as we are proposing and if they follow this order, they have but to define their own special field restrictions for our general coding system to achieve complete compatibility. In fact an important tenet of our system should be that it be amenable to minor coding variations, such as the above special case, merely by notifying the assembly program of the variation.

How the proposed tenets will be effected is fairly obvious except for denoting instruction notation intent and extent by punctuation. Now, if we recall the instruction organization as already set forth, we remember that there is a single address group and a dual address group. The single address will stand alone and needs no punctuation as long as the register is implied. However, if there is a specified register, it will follow and will be separated from the address by a comma. Whenever there is an address modifier, it will be placed immediately after the address and enclosed in parenthesis. In the case of dual addresses, the second address will follow and will be separated from the first address by a comma. If there is a specified register, it will follow and will be separated from the second address by a comma. Again, whenever there is an address modifier, it will be placed following the address but inside the following comma and enclosed in parenthesis. Operations need no separation as they will appear first in the entry and are easily recognizable. This should be a simple and comprehensive advanced system for coding instructions.

The general organization of the proposed coding system is as follows:

I. Single Address Group

A. Implied Register Class

1) Floating Point (n/u) $\boxed{S}$ Address (Ix)
2) Miscellaneous Address (Ix)

B. Specified Register Class

1) Direct Index Address (Ix), Jx
2) Immediate Index Address (Ix), Jx
3) Count and Branch Address (M), Jx
4) Indicator Branch Address (M), Indicator

II. Dual Address Group

A. Implied Register Class

1) Transmit (K(x)) Address (Ix), Address (Ix)
2) Input-Output Address (Ix), Address (Ix)
3) Branch on Bit Address (Ix), Address (M)
4) Store I. C. and Branch Address (Ix), Address

B. Specified Register Class

1) Integer (b/d) $\boxed{s}$ Address (Ix, P), Fld (Ix), Reg (off)
2) Connection (1111) Address (Ix, P), Fld (Ix), Reg (off)

Herein we note that the integer instruction category is placed under the specified register class whereas a register can actually be implied. However, since the register address can be modified by an offset, we adopt the same scheme as used in the connection instruction category. But in practical use the accumulator register need not be specified for integer instructions; therefore, the register coding can be omitted and the offset specified in the normal way (enclosed in parenthesis).

Further elaboration of the coding system is offered here for a more thorough understanding of it. We shall discuss the system by proceeding through the coding system as previously outlined.

Floating point operations may be performed with normalized fractions or unormalized fractions, thus, at the coders option either (n) or (u) can be selected. The addressed data which the operation will affect can have one of several signs $\boxed{s}$. The sign may be specified by the coder or it may be the same as assigned to the data when it was last stored. The address may be modified by any index register (Ix) from 0 through 15. Miscellaneous instructions have only an address modifier (Ix) which is used in the same way as for floating point instructions.

For the first two instruction categories under the specified register class of the single address group, the address modifiers can be any index register from 0 through 15. For the last two, the address modifiers can only be index registers 0 or 1 plus several special modifiers whose definition will be found in the Basic manual. The specified registers (Jx) are index registers 0 through 15. Here J is used instead of I to denote the difference between an index register that is operated upon and one that modifies an address. The specified Indicator will refer to one of the 64 possible conditions in the Basic machine which can be interrogated for.

For the implied register class of the dual address group, the symbols should be self explanatory except for K(x). K(x) furnishes the number of half words to be transmitted and can be either an constant number (K) or an index register address from 0 through 15 (Kx) which supplies the constant number.

Under the specified register class of the dual address group we find the most complex instruction expressions - integer and connections. The integer operations may be performed in binary or decimal arithmetic; thus, at the coders option (b) or (d) can be selected. There are 16 connections which can be selected and the truth table binary output that would result from any particular connection is the best means for specifying that particular connection. Integer addresses have a sign modifier which is the same as that explained for floating point instructions. The primary addresses for both integer and connection instructions can be modified by index registers as previously described and in addition the index registers can be changed in accordance with the P modifier. If the primary address is an immediate value, a number symbol (#) following the value is sufficient to indicate that it is actual data. For both integer and connection instructions a specific size field must be stated and this field may or may not be indexed. For both integer and connection instructions an operation register is either implied or stated. Unless otherwise specified the register operation begins in the normal position and need not be stated. But for those cases where it is offset, the offset will be symbolically noted in parenthesis.

It should be obvious that for this coding system any part of the instruction enclosed in parenthesis may be omitted entirely and sufficient information is still available for the machine to execute the instruction without dire consequences. Thus, when coding addresses that do not have modifiers, a statement of omission is unnecessary.

Additional information pertinent to coding is:

1. $\boxed{s}$ - sign to be used in treating data at location indicated by either actual or symbolic address

   + = unsigned positive or absolute positive

   – = unsigned negative or absolute negative

   – = opposite sign under appropriate conditions

   nothing = data sign.

2. Ix - index registers 0 through 15, may be actual or mnemonic representation.

3. P - increment (I), increment-count (IC), increment-count-refill (ICR), or decrement (D), etc, whichever is appropriate.

4. Fld - either decimal or mnemonic representation of field to be processed. If actual numeric, byte size and number of characters must be provided.

5. Off - offset from starting point of accumulator or register.

6. Reg - register to be affected by operation. Accumulator register AB implied for integer instructions.

7. $K_{(x)}$ - either constant (K) indicating number of half words to be transmitted or index register number (Kx) containing such a number.

8. Jx - index register operated on by an operation.

9. M - branch modifier codes indicating advance, condition on which to branch, and final indicator conditions.

[illegible]

LaMar L. Briner

LLB/bb