April 15, 1958

INTERNAL NOTE # 30

Subject: Coding Examples for Basic SIS

The preliminary specifications of a Basic computer coding system as proposed in Internal Note #29 have furnished us with a general technique for expressing instructions. In this internal note we wish to illustrate the flexibility and potential of this proposed coding system through sample instructions and programs.

First, though, we would like to point out that flexibility and potential will be attained not only through the use of the proposed coding arrangements but also through the varity of terminology that can be employed. However, as this Basic coding system terminology and its syntax does not conform to any standard language, we shall call the language for this system the SYNTHETIC language. Then our system for preparing Basic machine instructions becomes the SYNTHETIC INSTRUCTION SYSTEM or SIS. Machine interpretation of SIS terminology and syntax will be performed by a SYNTHETIC TRANSLATOR, ASSEMBLER, and COMPILER, or SYNTAC. SIS coding will be illustrated here but SYNTAC will be explained in a subsequent internal note.

As our first sample, we choose the ADD operation. The variations in the way the ADD instruction can be coded are as follows:

1. Add (d) + 13.54 #, 6-4, (3)
   Six decimal numbers (001345) as indicated by the field length (6), the decimal modifier (d), and the number sign (#) will be added to the accumulator. The two leading zeros are implied by the field length of 6 and the decimal byte size is denoted by the 4 following the 6. The (3) indicates the offset and infers an offset of 3 four bit bytes or 3 decimal characters.

2. Add (b) + 13.54 #, (5)
   The binary value of 1354 as determined by SYNTAC will be added to the accumulator in the binary mode. No length is

specified for the binary field because SYNTAC will determine this length. The (5) will mean a 5 bit offset and since it is enclosed in parenthesis, it can not be confused with a possible field length specification.

3. Add (d) -256.25, 3-4
Decimal data in memory location 256.25 will be subtracted from the contents of the accumulator. The byte size will be 4 bits and the field length is 3 bytes or 3 decimal characters. No offset is specified.

4. Add (b) 256.25 (9, ICR), 20
Twenty binary bits of a memory location specified as 256.25 but indexed by register 9 will be added algebraically to the accumulator. The absence of a sign here means that the sign of the data will be used. After indexing the address, the index register's value field will be incremented, its count will be reduced by one, and if its count reaches zero, the register will be refilled with the contents of the word designated by its refill address.

5. Add (b) Current Wages (9, ICR), 20
Same instruction as 4 except that the symbolic memory location 256.25 is replaced by the tag "Current Wages".

6. Add (b) Current Wages (250-20-250.00, ICR), 20
Same instruction as 5 except that index register 9 has been replaced by the contents of an unspecified register to be assigned by SYNTAC. The contents of this psuedo register will be stored in the location specified by the refill address (250.00).

7. Add (b) Current Wages (Wage Record - # Records, ICR), 20
Same instruction as 6 except that "Wage Record" has replaced 250 as the record length in the value field and "# Records" has replaced 20 as the record count in the count field. The refill address is unspecified and therefore SYNTAC is responsible for locating and setting-up the refill word. If instead of ICR this had only been IC, a refill word would have been unnecessary.

8. Add (b) 256.25 (9 #, I), 20
Same instruction as 4 except that 9 # is the actual value of an index register to be assigned by SYNTAC. This is the only case where confusion between an index register designation and an index value will occur.

Of course the ADD instruction may be expressed with other variations but by now these should be apparent and the way in which the other operations can be expressed should be evident too.

As programing examples we have selected short routines from the Project 7000 Preliminary Manual of Operation. The first program, found on page 6.21, is an FICA routine. The parameters in the program are:

Current Gross Wage
Current FICA
FICA Balance (remaining FICA to be paid, max. = 94.50)

The FICA program is:

```
Load (d) Current Gross Wage, (2)
Multiply + .0225 #, 3-4
Add + 5#, 1-4, (5)
Store (d) Current FICA, (6)
Augment - FICA Balance, (6)
Add - Current FICA, (6)
Store (d) - Current FICA, (6)
Add Memory - FICA Balance, (6)
```

The second program, found on page 6.25, is a continued compare.

```
Load (d) + M, 12-5
Compare (d) + N, 12-5
Load (d) + M + 60, 12-5
C. Compare (d) + N + 60, 12-5
Load (d) + M + 120, 6-5
C. Compare (d) + N + 120, 6-5
```

These examples should present a fair idea of the flexibility and potential of SIS. Further detailed elaboration on the coding methods will be provided in the final specifications.

LaMar L. Briner

LLB/bb