

Kolsky

MEETING OR CONTACT REPORT

Date of Report: February 14, 1958

Organization & Location: IBM - POUGHKEEPSIE	Date: February 11, 1958
	Reported By: E. F. Codd
Project: Stretch Programming Committee Meeting No. 4	Department: 749
	Follow-up Date: March 4, 1958

PERSONNEL PARTICIPATING:
 (Place asterisk next to those on distribution list. Other distribution show at end of report)

Messrs: L. Briner *
 F. P. Brooks*
 J. E. Hamlin*
 W. P. Heising *
 P. S. Herwitz *
 I. Ziller*
 E. F. Codd *

1. Discussion of Machine

Information was presented on recent changes in indicators, branch instructions, arithmetic and logical operations, and assignment of addresses. Provisions for Swift and Tractor tapes were discussed. It was agreed that use of locations 32-54 and indicators 43-49 would be avoided in all programming systems intended for Basic and Sigma, so that these programs would be usable on the Harvest system.

On the subject of a low cost drum or disk, Dr. Brooks reported that there was no provision in the Stretch program for any drum or disk other than the high speed disk for Los Alamos. Mr. Heising and the Product Planning representative agreed to investigate together the disk and drum development projects being carried on outside the Stretch program to determine if a unit suitable for Stretch programming systems is or will be available.

2. Programming Language

Messrs Heising and Ziller described a novel symbolism for use in programming Stretch. The committee felt this scheme looked sufficiently promising to warrant postponing decisions on mnemonic code and crude assembly programs. The authors of the scheme agreed to distribute descriptive material (even if rough) before the next meeting. There was general agreement 1) that the programming language must be capable of representing in a single statement any machine instruction whatsoever; and 2) that the language should not contain macro-operations as single statements initially, but should be so constructed that these may readily be added in a consistent manner at a later time.

3. Elementary Supervisory Program

A proposal was put before the committee (see attached copy). The suggestion was made that an additional function be added to the system; the ability to call in the next job from tape as soon as the current one is completed.

4. Simulator

A firm commitment of adequate manpower was not forthcoming. It is hoped that it will be forthcoming by the time the next meeting is held.

5. Future Action

Committee members agreed to continue the activities as outlined in the report of the last meeting.

The next meeting was scheduled for Tuesday, March 4 at the South Road Laboratory, Poughkeepsie.


E. F. Codd

EFC/jcv

cc: Mr. J. T. Ahlin
Mr. J. W. Backus
Dr. W. Buchholz
Dr. C. R. DeCarlo
Mr. S. W. Dunwell
Mr. W. J. Lawless
Mr. J. W. Luke
Mr. J. C. McPherson
Mr. D. W. Pendery
Mr. D. W. Sweeney

ELEMENTARY SUPERVISORY PROGRAM REQUIREMENTS

1. Objectives

- 1.1 To be used as a tool for program development and testing activity during the early stages of operation of the laboratory model of the Basic computer.
- 1.2 To act as a pilot model for more advanced supervisory programs, and particularly for multiprogramming techniques.
- 1.3 Not intended for distribution to customers.

2. Comments on Objectives

Objectives 1.1 and 1.2 should be met in a two-stage manner, the end result being two supervisory programs rather than one. The essential distinction between these programs is that one (UniSupervisor 1) is capable of supervising only a single problem program at a time, while the other (MultiSupervisor 1) is capable of supervising many at a time. Much of the programming of the UniSupervisor would become a part of the MultiSupervisor. Only the UniSupervisor is discussed in any detail in this report.

3. UniSupervisor 1

- 3.1 This program is intended to provide a minimum operating facility consistent with the crude assembly program CB. It is primarily concerned with:
 - 1) making the inquiry station an effective unit for manual supervision, particularly when debugging
 - 2) providing a simple means of catching programs which get stuck for one reason or another
 - 3) handling all those interrupts for which the problem program fails to provide programming steps
 - 4) protecting itself from being clobbered by the problem program or by the operator.
- 3.2 Supervision by the operator of the machine's activities requires that the operator and the machine be able to communicate with each other in a language understandable by both. This language should allow all the frequently used supervisory functions to be expressed in a simple and concise manner. Thus, mnemonic coding is used.

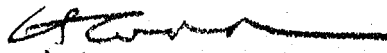
- 3.3 The instrument normally used by the operator for manual supervision will be an inquiry station. From the keyboard of the IQS typewriter the operator will be able to send any of the signals defined for operator-to-supervisory-program communication. Signals in the reverse direction will be automatically typed on the IQS typewriter by the supervisory program.
- 3.4 The on-line card reader and printer are useful adjuncts to the inquiry station and, in an emergency, are capable of assuming all the functions provided by the inquiry station. As far as possible, signals entered via control cards will have precisely the same form as those entered via the IQS keyboard. A similar correspondence will hold between messages displayed on the IQS typewriter and messages printed on the line printer.
- 3.5 Some consideration will be given to possible uses for the optional console. It appears capable of handling a fairly limited subset of the signals which the IQS can handle. Undoubtedly, it will serve better as an adjunct to the IQS rather than as a substitute for it.
- 3.6 The I/O interrupt facilities in Stretch make an interpretive approach to manual supervision both possible and practical. With this approach it becomes possible to exercise manual supervision without bringing the computer to a standstill. The degree of supervision possible is much greater than sense switches provided on earlier machines, since changes in the state or mode of operation of a program can now be made on a completely unpreplanned basis. An operator will be said to have obtained established control if, during the execution of a problem program, he depresses the unit signal key on the IQS and issues a RELEASE PROGRAM or STOP PROGRAM signal (see below). He remains in established control over this problem program until such time as he issues a RESUME signal. If it appears unnecessary or undesirable to stop the CPU activity upon the problem program, an operator may obtain fleeting control by depressing the unit signal key on the IQS and issuing any signal other than RELEASE PROGRAM or STOP PROGRAM.
- 3.7 It appears unnecessary to provide the operator with such extensive facilities for keyboard programming and card programming that he can direct in detail highly intricate procedures, such as file assembly for example. Instead, such operations will normally be planned in advance and internal programming will be used. The main emphasis, therefore, in defining the signals between the operator and the supervisory program is on unpreplanned activities.

- 3.8 Several of the routines required to interpret signals between the operator and the supervisory program are of general utility: for example, the code translation and conversion routines. Such routines will be available for use by problem programs on a subroutine basis.
- 3.9 As a general rule, after each entry is made by the operator on the IQS keyboard, the supervisory program will respond on the IQS typewriter in such a way that the operator will have some confidence that his manual operation was correctly performed or that the machine has completed a requested activity. This automatic response may possibly be omitted when signals are derived from control cards, the assumption being that these cards have been verified.
- 3.10 The numeric part of all addresses, relative or absolute, is assumed to be expressed in octal. Operation codes and modifiers are invariably expressed in symbolic form.
- 3.11 The entire supervisory program (including all service routines) is stored on a supervisory tape with a small self-loading program at the leading end. The load button may therefore be used at any time to restore the supervisory program.
- 3.12 In case trouble should arise with the supervisory program and it is desired to track it down, a diagnostic program will be available which will compare word by word UniSupervisor 1 as it stands in memory with the version on file and print out in octal all discrepancies with their location.

4. Further Information on UniSupervisor 1

The following memos are in preparation:

- 1) UniSupervisor 1 Signals
- 2) The WHENEVER Mode - A Debugging Aid
- 3) Code Translation and Conversion
- 4) UniSupervisor 1 Interrupt Handling

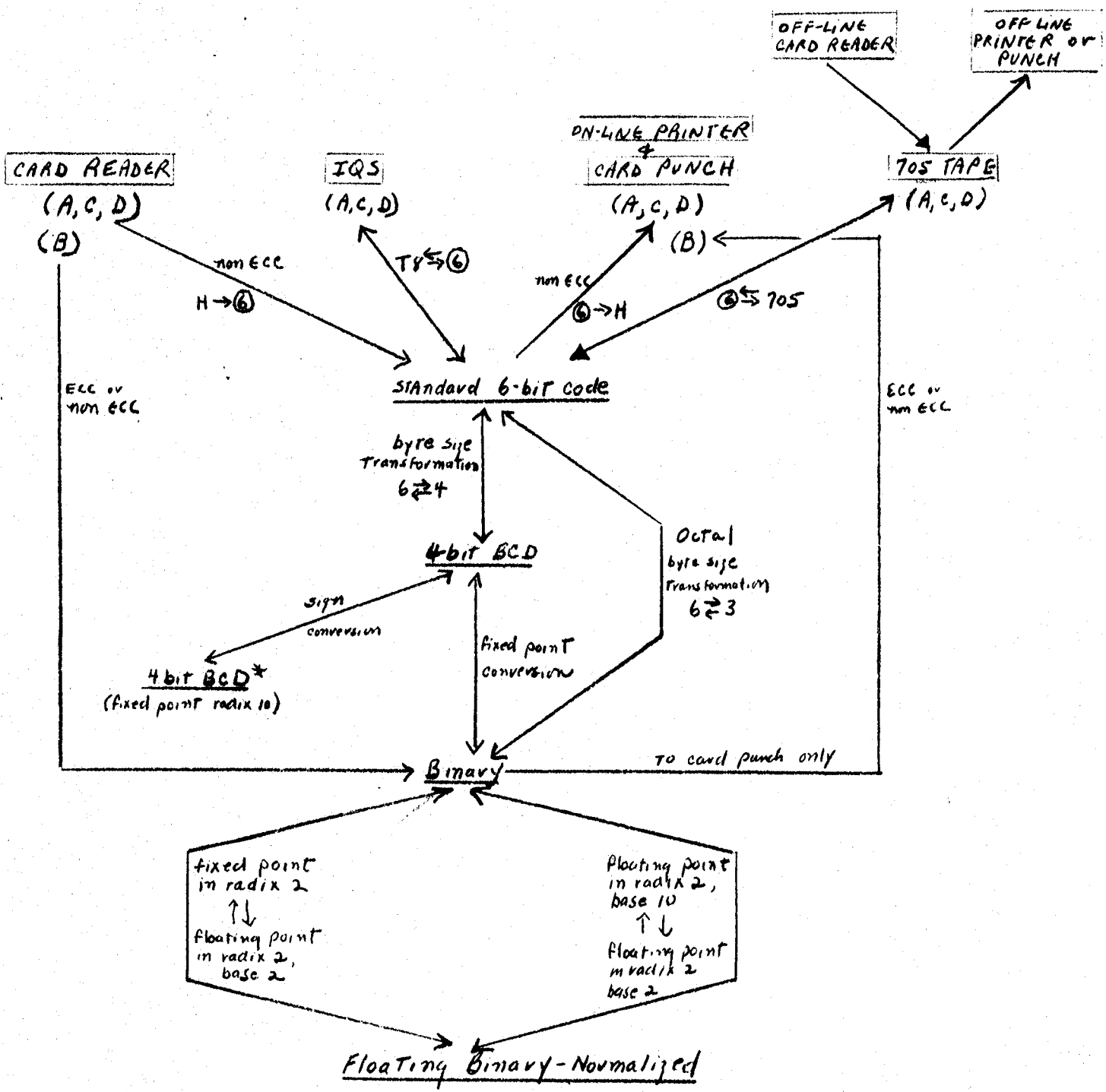


E. F. Codd

EFC/jcv

February 10, 1958

CODE TRANSLATION AND CONVERSION FOR I-O



still require to determine

1. necessary degree of compatibility with 704
2. nature of standard 6-bit rule

KEY

- A alpha-decimal
- B binary
- C octal
- D decimal
- T8 typewriter 8-bit code
- Ⓢ 6-bit code (standard)
- H Hollerith (12-bit card column image)

UNISUPERVISOR 1 SIGNALS

The signals described below allow the operator to communicate with the supervisory program using the typewriter keyboard of an inquiry station or, alternatively, control cards placed in a card reader. The main purposes of these signals are:

- 1) To stop and start execution of a problem program;
- 2) To enter and display items of information manually;
- 3) To control the loading and dumping of programs;
- 4) To debug a problem program and track down machine errors;
- 5) To change the source of operator-to-supervisory-program signals and the destination of supervisory-program-to-operator signals.

In what follows the abbreviation PP is used for problem program.

1. Stop and Start Signals

RESUME PP

If an address is not specified, execution of the problem program is resumed at the point of interruption (this address has been remembered by the supervisory program). If an address *m* is specified, execution is resumed at *m*. Hence, this signal may be used to get execution started after a program has been loaded.

RELEASE PP

1. Stops CPU activity on PP.
2. Issues release instructions for all channels used by PP.
3. Prints out location of last PP instruction executed.

STOP PP

1. Stops CPU activity on PP.
2. Allows I/O activities to continue and compiles an internal log of I/O status interrupts so that these may be simulated if CPU activity is resumed.
3. Prints out location of last PP instruction executed.

2. Enter and Display Signals

An ENTER signal indicates the type of information to be entered together with the memory location for the first item. Immediately following the signal are the items themselves. Any number of items may be entered with a single ENTER signal providing they have a common specification. The END OF MESSAGE key is used to inform the supervisory program that the last item of a set has been entered. The END OF ITEM key permits items of variable length to be entered without tedious zero-padding by the operator.

The five distinct ENTER signals are listed below, together with the type of information for which each may be used:

<u>Signal</u>	<u>Type of Information</u>
ENTER INSTRUCTIONS	Instructions with format corresponding to the assembly listing
ENTER FLOATING DATA	Floating data in octal or decimal
ENTER FIXED DATA	Fixed data in octal or decimal
ENTER DATA BYTES	Data bytes (for which radix conversion is not applicable)
ENTER CONTROL WORDS	Control words and index words

The ENTER FIXED DATA signal is accompanied by a specification of internal field length, internal number of binary places, and external number of octal or decimal places.

There are five distinct DISPLAY signals having a one-to-one correspondence with the ENTER signals. Information is normally displayed on the inquiry station typewriter. Hence, this signal, like the ENTER signal, is used only for small volumes of information. The option is provided of displaying the memory location of each item along with the item itself.

The code translations and byte size transformations permitted by the ENTER DATA BYTES and DISPLAY DATA BYTES signals are indicated on the attached chart.

3. Load and Dump Signals

These signals are intended primarily for unpreplanned, high volume I/O operations such as emergency dumping or restoring of blocks of instructions, data, etc. Unlike the ENTER and DISPLAY signals, the I/O device must

be specified - it is not implied. The following devices are permitted: on-line card reader, printer and punch; also, tapes to be used either as an auxiliary storage or for communication with off-line equipment. While the signal to load or dump will normally originate from an inquiry station, such a station may not be used for handling the information which is loaded or dumped.

The five types of information described for ENTER and DISPLAY are handled by these signals. An additional (sixth) type of information, binary with or without ECC, is also handled, except when dumping on the printer.

With any LOAD or DUMP signal it is necessary to state whether control information is involved. For example, when using binary cards each card may contain as a control item the number of words punched in the card.

4. Debugging Signals

CLEAR PP AREA

If no addresses are specified, the entire area assigned to the program is cleared. If one address is specified, this is taken as the starting address. If two addresses are specified, the first is interpreted as the starting address, the second as the terminating address.

EXECUTE

This signal informs the supervisory program that the immediately following instructions are to be loaded into the supervisory area and, after receipt of the END OF MESSAGE signal, are to be executed.

If the instructions introduced by an EXECUTE signal result in interrupts of any kind whatsoever, the supervisory program unconditionally logs these interrupts on the inquiry station typewriter. The EXECUTE signal provides on an optional basis an additional internal log of the I/O status interrupts resulting from the manually - introduced instructions.

SINGLE STEP PP

If an address is specified, the corresponding PP instruction is executed interpretively. A simulated single step instruction counter is advanced as required by this instruction. If an address is not specified, the PP instruction which is selected for interpretive execution is that corresponding to the current setting of the single step instruction counter. After its execution the effective PP instruction is displayed together with its location.

SEARCH PP AREA

This signal may be used to search part or all of the PP area for a specified operation code, address part, index address part, control (index) word, or item. Whenever the specified information is found, the supervisory program displays it with its location. The manner in which the information is displayed corresponds to that defined for the DISPLAY signals.

The general item search requires that a starting location, field length and field interval be specified. For the other searches a starting address only, a starting address and a terminating address or neither may be specified. An omitted address implies that the corresponding PP boundary address be used.

WHENEVER

See attached memo entitled: "The WHENEVER Mode - A Debugging Aid."

LEAVE WHENEVER MODE

The WHENEVER mode of operation is dropped and the supervisory program prints out the location of the last PP instruction executed in the mode. If the operator issued the LEAVE WHENEVER MODE while in established control, the supervisory program awaits a further signal from the operator. Otherwise, normal execution of the PP is resumed.

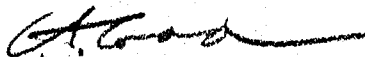
5. Source and Destination Change Signals

CHANGE SOURCE

The supervisory program prepares to accept control signals from the operator using the specified I/O unit (an inquiry station or card reader or control console).

CHANGE DESTINATION

The supervisory program prepares to issue signals to the operator using the specified I/O device (an inquiry station or on-line printer).



E. F. Codd

EFC:EM:jcv
February 12, 1958


E. McDonough

When debugging a program or tracking down machine trouble it is often found convenient to alter the manner in which the program is executed so that operations are introduced which were not planned for at the time the program was written. The WHENEVER mode provides the operator with a means of introducing in simple language (and, if necessary, on the spur of the moment) a wide range of operations whose execution is conditional upon the occurrence of a simple event or a complex combination of events.

The WHENEVER Statement

A simple WHENEVER statement has the form:

WHENEVER {EARLY} X {IS} Y } Z
 {LATE} {IN} {NOT Y}

X names a set of states or events. Y specifies one or more elements of this set. Z specifies the action to be carried out when the required state (s) or event (s) occur.

Example: whenever an instruction is executed from location 1066, the program is to be stopped as soon as the execution of this instruction is completed.

For X substitute LOC, meaning location of instruction.

For Y substitute 1066.

For Z substitute STOP.

Select the option LATE, since the instruction is to be completed.

Select the option IS, since a range of locations is not relevant.

The resulting statement would be typed into the machine from an inquiry station in the following coded form:

WL LOC IS 1066, STOP

where W stands for WHENEVER, and L for LATE.

IS and IN

A WHENEVER statement contains the word IS or the word IN. IS is followed either by a single state or event Y as in the example above or by a list of states or events Y₁, Y₂, Y₃, etc. as in the example below:

Print the contents of location 1837 (a floating point word) in decimal whenever the instruction whose execution has just been completed was picked up from location 1066, 1215, or 1588. An appropriate statement is:

WL LOC IS 1066, 1215, 1588, PRINT FLO DEC 1837

The word IN implies that a range is to be specified in the form $Y_1 - Y_2$ (where $Y_1 < Y_2$).

Print the contents of location 1837 (a floating point word) in decimal whenever the instruction whose execution has just been completed was picked up from any of the locations 1066, through 1215. An appropriate statement is:

WL LOC IN 1066-1215, PRINT FLO DEC 1837

NOT

A single example should suffice to explain the use of this word.

Print the contents of location 1837 (a floating point word) in decimal whenever the instruction whose execution has just been completed was picked up from any location assigned to this problem program other than 1066 through 1215. An appropriate statement is:

WL LOC IN NOT 1066-1215, PRINT FLO DEC 1837

The X Substatement

The options available for X are listed hereunder with examples of their use.

INT

An interrupt occurs. The Y substatement specifies a single indicator address or a list or range of such addresses.

Example: WL INT IS 5, PRINT "WHAT NOW?"

OP

The instruction to be executed (if EARLY) or just executed (if LATE) has an operation code. The Y substatement specifies a particular operation code or a list or range of such codes.

Example: WE OP IS RD, STOP

ADDR

The instruction to be executed (if EARLY) or just executed (if LATE) has an address part. Since certain instructions have two address parts, the left or right must be specified by a suffix L or R. Further, the address may be either the specified one or the effective one, and the choice here is made clear by another suffix S or E. The Y substatement specifies a value or values for the address part.

Example: WE ADDR LE IS 1066, STOP

IA

The instruction to be executed or just executed has an index address part. The suffix Lor R applies here again. A further suffix is required to distinguish between the I and J fields.

Example: WE IA LJ IS X15, STOP

Cwwwww. bb. ff

The meaning of C is "content of", while wwwwww indicates a word address, bb a bit address, and ff the field length. For this substatement a zero word address specifies the Execution register and therefore provides a general way of referring to any field within an (effective) instruction.

Example: WL C1763.5503 IS 1, 2, 7, STOP

The Z Substatement

Z may specify almost any signal or pair of signals defined in the memo entitled "UniSupervisor I Signals".

Example: Whenever a successful branch occurs (interrupt no. 25), search the area from 1215 through 1815 for all references to 1837, printing out the location of each such reference.

WL INT IS 25, SEARCH 1215-1815 ADDR 1837

Note that Z may itself be a WHENEVER.

Example: As soon as the count in location 1763 reaches 22, start the activity described in the above example.

WL C1763 IS 22, WL INT IS 25, SEARCH 1215-1815 ADDR 1837

A statement such as this in which the Z substatement is itself a WHENEVER will be referred to as a WHENEVER chain.

In general, a complex WHENEVER statement is not equivalent to a chain of simple WHENEVER statements.

Complex WHENEVER Statements

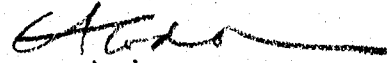
In place of X IS or IN Y or NOT Y there may be several such clauses joined by AND or OR. The expression is assumed to be in alternational normal form.

Example: Whenever the instruction to be executed is a READ or WRITE with channel address 7, stop the program.

WE OP IS RD, WT AND ADDR RE IS 7, STOP

Note on Implementation of WHENEVER Mode

Except for certain special cases (such as successful branch), the mode is put into effect by executing the PP instructions interpretively under control of the supervisory program. This approach is practical on Stretch because of its unusual efficiency in interpreting.



E. F. Codd

EFC/jcv

February 17, 1958