Digital Computer Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SUBJECT: SHORT GUIDE TO CODING AND WHIRLWIND I OPERATION CODE

To: Group 61 and Applications Group

From: Philip R. Bagley

Date: September 2, 1952; Revised November 28, 1952

Abstract: This note contains an up-to-date version of the Short Guide to Coding and the Whirlwind I Operation Code.

FOREWORD

The following definitions have been adopted and consistently adhered to:

sequence: the numberical or other arrangement of a set of words stored or performed
instruction: a 16-digit binary word used to control the computer
operation: the 5 digits of an instruction which go to the operation control switch
command: a control pulse from the control matrix
process: an automatic manipulation initiated by a command
modulo: (abbreviated "mod") A number p modulo q is defined as the numerator of the fractional remainder when p is divided by q

Ex. 1: 60 mod 32. $\frac{60}{32} = 1 + \frac{28}{32}$, hence 60 mod 32 = 28

Ex. 2: 1.37 mod 1. $\frac{1.37}{1} = 1 + \frac{.37}{1}$, hence 1.37 mod 1 = .37

SECTION 1. SHORT GUIDE TO CODING

COMPUTER PROGRAMS

Program. A program is a sequence of actions by which a computer handles a problem. The process of determining the sequence of actions is known as programming.

Flow diagrams. A flow diagram is a series of statements of what the computer has to do at various stages in a program. Lines of flow indicate how the computer passes from one stage of the program to another.

Coded program. Programs and flow diagrams are somewhat idependent of computer characteristics, but instructions for a computer must be expressed in terms of a code. A set of instructions that will enable a computer to execute a program is called a coded program, and the process of preparing a coded program is known as coding. Individual coded instructions call for specific operations such as multiply, add, shift, etc.

COMPUTER COMPONENTS

Registers and words. A register has 16 digit positions, each able to store a one or a zero. A word is a set of 16 digits that may be stored in a register. A word can represent an instruction or a number.

Arithmetic element. Arithmetic operations take place in the arithmetic element, whose main components are three flip-flop registers, the A-Register, the Accumulator, and the B-Register (AR, AC, and BR). The 16 digit positions of AR starting from the left are denoted by AR 0, AR 1 ... AR 15. The digit positions of AC and BR are denoted in a similar fashion. Words enter AC through AR; BR is an extension of AC to the right.

Storage. The term "register" by itself refers to the main electrostatic storage, which consists of 1024 registers, each of which is identified by an address. These addresses are 11-digit binary numbers from 32 to 1055. The computer identifies a register by its address. Electrostatic storage may at some future date be expanded to 2048 registers, numbered 0 through 2047.

Input-output. All information entering or leaving the computer is temporarily stored in the input-output register (IOR). The computer regulates the flow of information between the internal storage and IOR, and also calls for any necessary manipulation of external units.

Control element. The control element controls the sequence of computer operations and their execution. The control element takes the instructions one at a time from storage, where the instructions are stored as individual words.

Inter-connections. The four main elements of the computer (storage, control, arithmetic, and input-output) are connected by a parallel communications system, known as the bus.

REPRESENTATION OF INSTRUCTIONS

Operation section. When a word is used to represent an instruction the first (left-hand) 5 digits, or operation section, specify a particular operation in accordance with the operation code.

Address section. The remaining 11 digits, or address section, are interpreted as a number with the binary point at the right-hand end. For the majority of instructions this number is the address of the register whose contents will be used in the operation. In the instructions slh, slr, srh, srr, clc, and clh, the number specifies the extent of a shift, and also an

additional variant, such as roundoff; in rs, rd, and rc, the address section is not used.

Example. The instruction ca x has the effect of clearing AC (making all the digits zero) and then copying into AC the word that is in the register whose address is x. If q is a quantity in some register, the operation needed to copy q in AC is not ca q but ca x, where x is the address of the register that contains q.

REPRESENTATION OF NUMBERS

Single-word representations. When a word is used to represent a number the first digit indicates the sign and the remaining 15 are numerical digits. For a positive number the sign digit is zero, and the 15 numerical digits with a binary point at their left specify the magnitude of the number. The negative -y of a positive number y is represented by complementing all the digits, including the sign digit, that would represent y. (The complement is formed by replacing every zero by a one and every one by a zero.) In this way a word can represent any multiple of 2^{-15} from $-1 + 2^{-15}$ to $1 - 2^{-15}$. Neither +1 nor -1 can be represented by a single word. Zero has two representations, either 16 zeros or 16 ones, which are called +0 and -0 respectively.

Overflow--increase of range and precision. With single-word representation the range is limited to numbers between $-1 + 2^{-15}$ and $1 - 2^{-15}$. Programs must be so planned that arithmentic operations will not cause an overflow beyond this range. The range may be extended by using a scale factor, which must be a 30-digit number. Overflow will stop the computer in an arithmentic check alarm except where special provision has been made to accommodate the overflow (see sa operation).

COMPUTER PROCEDURE

Sequence of operations. After the execution of an instruction the program counter in the control element holds the address of the register from which the next instruction is to be taken. Control calls for this instruction and carries out the specified operation. If the operation is not sp or cp the address in the program counter then increases by one so that the next instruction is taken from the next consecutive register. The sp and cp instructions permit a change in this sequential procedure.

Tranfers. A transfer of a digit from one digit position to another affects only the latter digit position, whose previous content is lost.

Zero. All sums and differences resulting in zero are represented as negative zero (1.111 111 111 111 111) except in the two cases: (+0) + (+0) and (+0) - (-0). The sign of a zero resulting from multiplication, division, or shifting is in accordance with the usual sign convention.

Manipulation of instructions. Words representing instructions may be handled in the arithmetic element as numbers.

Procedure in the arithmetic element. The execution of an addition includes the process of adding in carries; this process treats all 16 digits as if they were numerical digits, a carry from AC 0 being added into AC 15. (This compensation is necessary because of the method of representing negative numbers.) A subtraction is executed by adding the complement. Multiplication, division, scale factoring, shifting (by not cycling) and roundoff are all executed with positive numbers, complementing being performed before and after the process when necessary. For roundoff the digit in BR 0 is effectively added into AC 15.

BR. The final binary value of digits which pass from AC to BR or vice versa as a result of operations which multiply, divide, scale factor, or shift (but not cycle) is determined by the sign digit assigned to AC at the end of the operation. If the sign is negative the digits were in effect complemented as they crossed the digit-boundary between AC and BR. If the sign is positive no complementing occurred. The net effect is that a number in BR is treated as a positive magnitude, the sign of the number being indicated by the sign digit of AC. Therefore, if a number is to be recalled from BR for further operations, it is necessary to compensate for any change in the sign digit of AC which may have occurred after the number was placed in BR.. No complementing of any sort occurs in the execution of the cycle instructions, during which AC and BR may be considered a closed ring of 32 digit positions.

NOTATION FOR CODING

Addresses. A coded program requires certain registers to be used for specified purposes. The addresses of these registers must be chosen before the program can be run on the computer, but for study purposes this final choice is unnecessary, and the addresses can be indicated by a system of symbols or index numbers.

Writing a coded program. Reigsters from which control obtains instructions may be called action registers, and should be listed separately from registers containing other information, which may be called data registers. A coded program is written out in two columns: the first contains the index number of each action or data registers, and the second column indicates the word that is initially stored in that register. In many cases part or all of a word may be immaterial because the contents of the register in question will be changed during the course of the program. This state of affairs is indicated by two dashes, for example, ca--.

Conventional notation. In order to make a program more readily understandable to others and more easily remembered by the author himself, it is desirable to write short descriptions of the functions served by certain key instructions and groups of instructions. It is also desirable to indicate breaks and confluences in the "flow" of the program and to indicate instructions which are altered or otherwise abnormally used during the program. Some generally accepted symbols for this purpose are exemplified and described below:

```
            120  td 124
start--->   121  ca 161     initial entry (i.e., start of program)
            122  td 132
```

139--→	123	ca 181	re-entry point, showing origin of re-entry
	124	su(182)	address altered by program, initial value shown
	125	sr 16	
	126	cp 128	conditional short break in consecutivity (note other form below)
	127	ad 140	
	128	ad 133	
	129	ts	address indicated by arrow (e.g. address = 130 in this case), used primarily at early stages of writing
	130	(ca217/cs217)	word altered by program, alternative values shown
	131	sp 78	no break in consecutivity, despite sp operation, where a closed subroutine is called in
(122,167)	132	ts (-)	address altered by program, initial value immaterial, locations of altering instructions shown, alternative values not shown
	133	ca 217	semi-pseudo instruction, serves both as instruction and number
	134	sp 95	short break in consecutivity, used especially where a closed subroutine with program parameters is called for
	135	p3	
	136	ex 114	
	137	cp 141	conditional break in consecutivity (note short form above)
	138	ts 114	
	139	sp 123	break in consecutivity (note short form above)
	140	rs 0	pseudo-instruction, serves only as a number, not as instruction
137, 171-→	141	ts 171	entry point, showing origins of entry

The abbreviations RC, CR. Abbreviations used in referring to the register that contains a certain word or the word in a certain register are

RC . . . = (Address of) register containing . . .

CR . . . = Contents of register (whose address is) . . .

The symbol si x. When an address forms part of an instruction it is represented by the last 11 digits of a word whose first 5 digits specify an operation. An address that is not part of an instruction is represented by the last 11 digits of a word whose first 5 digits are zero, which is equivalent to specifying the operation si. Thus the word for an unattached address x may be written si x. It may also be written as +x or as $+x \times 2^{-15}$.

SECTION 2. WHIRLWIND I OPERATION CODE

NOTES ON THE OPERATION CODE

Introduction. The Whirlwind I Operation Code has been rewritten to bring it up-to-date, and to incorporate all notes, wherever possible, with the specific operations to which they apply, regardless of the undue repetition. Included under each operation are the average time of execution, the function, the contents (if altered) of AC, BR, AR, IOR, SAM, and register x after the operation, and possible alarms.

Abbreviations. The abbreviations used are the following:

AC =	Accumulator	IOR =	In-Out Register
AR =	A-Register	ES =	Electrostatic Storage
BR =	B-Register	x =	address of a storage register
SAM =	Special-Add Memory	n =	a positive integer

Contents of various registers. The contents of AC, BR, AR, IOR, SAM, and the register whose address is x are undisturbed unless the contrary is stated.

Alarms. Arithmetic check, divide error, and check register alarms due to programming cannot be caused except as specifically noted. M-1623, "Programming for In-Out Units" discusses in-out alarms.

Execution times. The times given are average times for the execution of single instructions which are stored in ES and which refer to addresses in ES. Further details are given in M-1623 and in E-440.

In-Out Operations. Operations which call for the transmission of information to and from various units of terminal equipment termed "in-out operations," are described briefly in the Operation Code. Details of the actual application of these operations (si, bi, rd, bo, and rc) appear in M-1623.

Three-letter operations. The three-letter operations slh, slr, srh, srr, clc, and clh utilize part of the address section of the instruction (namely, digit 6) to specify the operation. If an address is inserted in one of these instructions by a ta or td operation, care must be taken to maintain the presence or absence of digit 6 in the address of the modified instruction. The two-letter designations, sl, sr, cl, are ambiguous and cannot be used in programs, but they may be used in general descriptions and comments.

Operation	Function	Number	Binary	Time	
si pqr	select in-out unit/stop	#0	00000	45 microsec	si

Stop any in-out unit that may be running. Select a particular in-out unit and start it operating in a specified mode, designated by the digits p q r; or, stop the computer. si 0 will stop the computer; si 1 will stop the computer only if the "Conditional Stop" switch is ON. An in-out alarm may subsequently occur if the computer is not ready to receive information transmitted to it from the selected in-out unit. A transfer check alarm may result from the use of an illegal si address. For further details, see M-1623, "Programming For In-Out Units."

rs x	reset	#1	00001	30 microsec	rs

Reset any flip-flop storage registers connected to the "reset on rs" circuit.

bi x	block transfer in	#2	00010	(see M-1623)	bi
	AVAILABLE ABOUT JAN. 1953				

Transfer a block of n words or characters from an in-out unit to ES, where register x is the initial address of the block in ES, and $\pm$ n times 2^{-15} is contained in AC. The computer is stopped while the transfer is taking place. After a block transfer, AC contains the address which is one greater than the ES address at which the last word was placed; AR contains the initial address of the block in ES. Fur further details, see M-1623, "Programming For In-Out Units."

rd x	read	#3	00011	30 microsec	rd

Transfer word from IOR to AC, then clear IOR. (Wait, if necessary, for information to arrive in IOR from an in-out unit.) Contents of AR is identical to contents of AC. The address section of the instruction has no significance. For further details, see M-1623.

bo x	block transfer out	#4	00100	(see M-1623)	bo
	AVAILABLE ABOUT JAN. 1953				

Transfer block of n words from ES to an in-out unit, where x is the initial address of the block in ES, and $\pm$ n times 2^{-15} is contained in AC. The computer is stopped while the transfer is taking place. After the block transfer, AC contains the address which is one greater than the ES address from which the last word was taken and stored; AR contains the initial address of the block in ES. For further details, see M-1623, "Programming For In-Out Units."

Operation	Function	Number	Binary	Time of operation	
rc x	record	#5	00101	30 microsec	rc

Transfer contents of AC via IOR to an in-out unit. IOR is cleared only after an rc used as a display instruction. The address section of the instruction has no significance. For further details, see M-1623, "Programming For In-Out Units."

ts x	transfer to storage	#8	01000	86 microsec	ts

Transfer contents of AC to register x. The original contents of x is destroyed.

td x	transfer digits	#9	01001	86 microsec	td

Transfer last 11 digits of AC to last 11 digit positions of register x. The original contents of the last 11 digit positions of register x is destroyed.

ta x	transfer address	#10	01010	86 microsec	ta

Transfer last 11 digits of AR to last 11 digit positions of register x. The original contents of the last 11 digit positions of register x is destroyed. The ta operation normally follows an sp or sf operation.

ck x	check	#11	01011	48 microsec	ck

Compare contents of AC with contents of register x. If contents of AC is identical to contents of register x, proceed to next instruction; otherwise stop the computer and give a "check-register alarm." (+0 is not identical to -0).

ex x	exchange	#13	01101	86 microsec	ex

Exchange contents of AC with contents of register x (original contents of AC in register x, original contents of register x in AC and AR). ex 0 will clear AC without clearing BR.

cp x	conditional program	#14	01110	30 microsec	cp

If number in AC is negative, proceed as in sp. If number in AC is positive, proceed to next instruction, but clear the AR.

sp x	subprogram	#15	01111	30 microsec	sp

Take next instruction from register x. If the sp instruction was at address y, store y + 1 in last 11 digit positions of AR. All of the original contents of AR is lost.

ca x	clear and add	#16	10000	48 microsec	ca

Clear AC and BR, then obtain contents of SAM (+1, 0, or -1) times 2^{-15} and add contents of register x, storing result in AC. The contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

Operation	Function	Number	Binary	Time	
cs x	clear and subtract	#17	10001	48 microsec	cs

Clear AC and BR, then obtain contents of SAM (+1, 0, or -1) times 2^{-15} and subtract contents of register x, storing result in AC. The contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

ad x	add	#18	10010	48 microsec	ad

Add the contents of register x to contents of AC, storing result in AC. The contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

su x	subtract	#19	10011	48 microsec	su

Subtract contents of register x from contents of AC, storing result in AC. The contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

cm x	clear and add magnitude	#20	10100	48 microsec	cm

Clear AC and BR, then obtain contents of SAM (+1, 0, -1) times 2^{-15} and add magnitude of contents of register x, storing result in AC. The magnitude of the contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

sa x	special add	#21	10101	48 microsec	sa

Add contents of register x to contents of AC, storing result in AC and retaining in SAM any overflow (including sign) for use with next ca, cs, or cm instruction. Between sa and the next ca, cs, or cm, for which the sa is a preparation, the use of any instruction which clears SAM will result in the loss of the overflow, with no other effect on the normal function of the intervening operation. (In addition to ca, cs, and cm, the following operations clear SAM: ad, su, sa, ao, dm, mr, mh, dv, sl, sr, and sf.) If the overflow resulting from the sa is to be disregarded, care must be taken to destroy it before the next ca, cs, or cm instruction. The contents of register x appears in AR. SAM is cleared before, but not after, the addition is performed.

ao x	add one	#22	10110	86 microsec	ao

Add the number 1 times 2^{-15} to contents of register x, storing the result in AC and in register x. The original contents of register x appears in AR. SAM is cleared. Overflow may occur, giving an arithmetic check alarm.

Operation	Function	Number	Binary	Time	
dm x	difference magnitudes	#23	10111	48 microsec	dm

Subtract the magnitude of contents of register x from the magnitude of contents of AC, leaving result in AC. The magnitude of contents of register x appears in AR. SAM is cleared.

mr x	multiply and roundoff	#24	11000	65 microsec	mr

Multiply contents of AC by contents of register x. Roundoff result to 15 significant binary digits and store it in AC. Clear BR. The magnitude of contents of register x appears in AR. SAM is cleared.

mh x	multiply and hold	#25	11001	65 microsec.	mh

Multiply contents of AC by contents of register x. Retain the full product in AC and the first 15 digit positions of BR, the last digit position of BR being cleared. The magnitude of contents of register x appears in AR. SAM is cleared.

dv x	divide	#26	11010	120 microsec	dv

Divide contents of AC by contents of register x, leaving 16 binary digits of the quotient in BR and $\pm$ 0 in AC according to the sign of the quotient. The instruction slr 15 following the dv operation will roundoff the quotient to 15 binary digits and store it in AC. Let u and v be the numbers in AC and register x respectively when the instruction dv x is performed. If $|u| < |v|$, the correct quotient is obtained and no overflow can arise. If $|u| > |v|$, the quotient exceeds unity and a divide-error alarm will result. If $u = v \neq 0$, the dv instruction leaves 16 ones in BR; roundoff in a subsequent slr 15 will cause overflow and give an arithmetic check alarm. If $u = v = 0$, a zero quotient of the appropriate sign is obtained. The magnitude of contents of register x appears in AR. SAM is cleared.

slr n	shift left and roundoff	#27	11011	41 microsec	sl

Shift contents of AC and BR (except sign digit) to the left n places. The integer n is treated modulo 32; digits shifted left out of AC 1 are lost. (Shifting left n places is equivalent to multiplying by 2^n, with the result reduced modulo 1.) Roundoff the result to 15 binary digits and store it in AC. Clear BR. Negative numbers are complemented before and after the shift, hence ones appear in the digit places made vacant by the shift of negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction slr n must be a zero to distinguish slr n from slh n described below. The instruction slr 0 simply causes roundoff and clears BR. SAM is cleared. Roundoff may cause overflow, with a consequent arithmetic check alarm.

Operation	Function	Number	Binary	Time	
slh n	shift left and hold	#27	11011	41 microsec	slh

Shift contents of AC and BR (except sign digit) to the left n places. The integer n is treated modulo 32; digits shifted left out of AC 1 are lost. (Shifting left n places is equivalent to multiplying by 2^n, with the result reduced modulo 1.) Do not roundoff nor clear BR. Negative numbers are complemented before and after the shift, hence ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction slh n must be a one to distinguish slh n from slr n described above. SAM is cleared.

srr n	shift right and roundoff	#28	11100	41 microsec	srr

Shift contents of AC and BR (except sign digit) to the right n places. The integer n is treated modulo 32; digits shifted right out of BR 15 are lost. (Shifting right n places is equivalent to multiplying by 2^{-n}.) Roundoff the result to 15 binary digits and store it in AC. Clear BR. Negative numbers are complemented before and after the shift, hence ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction srr n must be a zero to distinguish srr n from srh n described below. The instruction srr 0 simply causes roundoff and clears BR. SAM is cleared. Roundoff (in a srr 0) may cause overflow, with a consequent arithemetic check alarm.

srh n	shift right and hold	#28	11100	41 microsec	srh

Shift contents of AC and BR (except sign digit) to the right n places. The integer n is treated modulo 32; digits shifted right out of BR 15 are lost. (Shifting right n places is equivalent to multiplying by 2^{-n}.) Do not roundoff the result nor clear BR. Negative numbers are complemented before and after the shift, hence ones appear in the digit places made vacant by the shift of a negative number. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction srh n must be a one to distinguish srh n from srr n described above. SAM is cleared.

sf x	scale factor	#29	11101	97 microsec	sf

Multiply the contents of AC and BR by 2 sufficiently often to make the positive magnitude of the product equal to or greater than 1/2. Leave the final product in AC and BR. Store the number of multiplications in last 11 digit places of AR and register x, the first 5 digits being undisturbed. If all the digits in BR are zero and AC contains + 0, the instruction sf x leaves AC and BR undisturbed and stores the number 33 times 2^{-15} in the last 11 digit positions of AR and register x. Negative numbers are complemented before and after the multiplication (by shifting), hence ones appear in the digit places made vacant by the shift. SAM is cleared.

Operation	Function	Number	Binary	Time	
clc n	cycle left and clear (BR)	#30	11110	41 microsec	clc

Shift the full contents of AC and BR (including sign digit) to the left n places. The integer n is treated modulo 32; digits shifted left out of AC 0 are carried around into BR 15 so that no digits are lost. Clear BR. No roundoff. With the clc operation there is no complementing of AC either before or after the shift; the actual numerical digits in AC and BR are cycled to the left. The digit finally shifted into the sign digit position determines whether the result is to be considered a positive or negative quantity. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction clc n must be a zero to distinguish clc n from clh n described below. The instruction clc 0 simply clears BR without affecting AC.

clh n	cycle left and hold	#30	11110	41 microsec	clh

Shift the full contents of AC and BR (including sign digit) to the left n places. The integer n is treated modulo 32; digits shifted left out of AC 0 are carried around into BR 15 so that no digits are lost. With the clh operation there is no complementing of AC either before or after the shift; the actual numerical digits in AC and BR are cycled to the left. The digit finally shifted into the sign digit position determines whether the result is to be considered a positive or negative quantity. Digit 6 (the $2^9 = 512$ digit of the address) of the instruction clh n must be a one to distinguish clh n from clc n described above. The instruction clh 0 does nothing.

ALPHABETIC LIST OF OPERATIONS

This is an alphabetic list of Whirlwind operations, including operations and designations which have become obsolete since 1950.

Operation	Number	Remarks	Operation	Number	Remarks
ad	18		qm	0	now dm
ao	22		qp	31	obsolete
bi	2		qr	30	obsolete
bo	4		qs	12	obsolete
ca	16		rc	5	
ck	11		rd	3	
cl	2	now clc	ri	0	obsolete
cl*	2	now clh	rs	1	
clc	30		sa	21	
clh	30		sf	29	
cm	20		si	0	
cp	14		sl	27	now slr
cs	17		sl*	27	now slh
dm	23		slh	27	
dv	26		slr	27	
ex	13		sp	15	
mh	25		sr	28	now srr
mr	24		sr*	28	now srh
qd	7	obsolete	srh	28	
qe	13	now ex	srr	28	
qf	23	obsolete	su	19	
qh	6	obsolete	ta	10	
ql	2	now clc	td	9	
ql*	2	now clh	ts	8	

Signed P.R. Bagley
P.R. Bagley

Approved C.R. Wieser
C.R. Wieser

cc: C.W. Adams
PRB/sc/jmm/jmc/mrs