UNIVERSITY OF ILLINOIS

DIGITAL COMPUTER

LIBRARY ROUTINE D 1 - 95

TITLE	Check Point Code II
TYPE	Special
NUMBER OF WORDS	$57 + 3\ell + s + j$ (see text)

DESCRIPTION:

This code is designed to print out intermediate information about the operation of some other code. Its purpose is to help the programmer locate mistakes in a code which is not working properly, or else to verify that a code suspected of working properly is actually doing so. The principle of operation of the check point code is to place "blocking orders" at particular points of your program (these points to be specified by you on a specification tape to be discussed later). Each blocking order results in a transfer of control to Code 95. When control is transferred, Code 95 prints out the location (fixed) of the order at which the block has been placed, and then prints out other information which you think will help you in locating the mistake in your program. The nature of the other information to be printed out at the checkpoint is also specified on the specification tape. This other information is printed out before the blocked order is finally obeyed. The blocking order itself is placed in the left hand side of the blocked placed in the memory.

For each checkpoint, the programmer can specify how many times no printing is to occur (e.g., do not print the first 14 times) and how often to print thereafter (e.g., print 18 times and then print no more). In this particular case, the following things would happen: When Code 95 is read in, it places a blocking order at the checkpoint. The first 14 times the blocking order is reached, control is transferred to Code 95. Code 95 prints nothings, executes the blocked order, and transfers control back to your program. The 15th time the blocking order is reached, control is transferred to Code 95. Code 95 prints two line feed characters, the location of the blocking order, and prints such other information as you have asked it to print (on the specification tape). It then executes the blocked order, and transfers control back to your program. The 16th time the blocking order is reached, printing occurs, and so on. The 32nd time (32 = 14 + 18) the blocking

order is encountered, printing occurs, the blocked order is obeyed, and the blocked order is restored to its rightful place in the memory the 33rd time. The 34th time the previously blocked order is encountered, it is simply obeyed just as if it had never been blocked, and the same occurs forever thereafter, until your program terminates with an OFF order, or fails someplace else.

Certain precautions must be obeyed in deciding on the location of blocking orders:

(1) The blocked order must not be overwritten by your program during the time it is blocked (however, it may be overwritten after code 95 has relinquished control).

(2) The blocked order must not be used as a number by your program during the time it is blocked (however, it may be used as a number after Code 95 has relinquished control).

(3) It is not necessary to avoid locating your checkpoint at a place which contains a control transfer order. The printing occurs before any transfer of control (to another part of your program) is obeyed.

(4) It is not necessary to make special provision for identifying each checkpoint (assuming that you have decided to insert several blocking orders), since Code 95 always prints the location of the checkpoint before any other information.

(5) Blocking orders can be overwritten by Code 95, i.e., it is possible to print information at some particular checkpoint the 47th time it is reached, the 58th time it is reached, but at no other times. The specification tape necessary for this will be discussed later.

NATURE OF INFORMATION WHICH CAN BE PRINTED AT EACH CHECKPOINT

At each checkpoint, we may take a number from any place in the memory, or from R_1, or from R_2, and print it in a number of ways:

(1) As an order pair

(2) As a right hand address

(3) As a left hand address

(4) As a ten character sexadecimal word

(5) As a signed integer

(6) As a signed 12 figure decimal fraction, not rounded

(7) As a signed 5 figure fraction, not rounded

The number of items to be printed at each checkpoint is arbitrary and so is the number of different checkpoints. Both are limited only by the memory space available for the operation of Code 95.

Code 95 will print two line feed and carriage return characters before each checkpoint print, but it will not print such characters during each checkpoint print unless the programmer so specifies. If the amount of printing you want to do at a given checkpoint exceeds the capacity of a line on the teletype machines (about 60 characters), you can and should specify additional carriage returns and line feeds by means of the specification tape.

If the number to be printed consists of a sign digit only (i.e., the sexadecimal word is 8000000000) the following things will be printed:

Order pair: 80 0 00 0
R. H. address: 0
L. H. address: 0
Sexadecimal: 8000000000
Signed integer: -0
Signed 12 figure fraction: = -000000000000
Signed 5 figure fraction: = -00000

Zero is always preced by a + sign.

Two spaces are left between any two printed items.

Non-significant zeros are omitted in the printing of integers or addresses. Zero is printed as a single zero. The integer 1235×2^{-39} is printed as + 1235, not as +000000001235.

THE SPECIFICATION TAPE

For each checkpoint, we must specify the location of the checkpoint, the nature of the information to be printed at the checkpoint, how often the printing is to be suppressed at first, and how often printing should occur thereafter until the order is unblocked. This is done as follows:

(1) Let "c" be the location of the checkpoint. Put cL on the specification tape.

(2) Let "f" be the number of times printing is to be suppressed (f = 14 in the example on page 1 of this write-up). Put fF on the specification tape. f must lie between 0 and 511. If f = 0, i.e., if you wish to print the very first time your program reaches the blocking order at c, you do not need to specify f at all.

(3) Let "k" be the number of times printing is to occur before the order is unblocked (k = 18 in the example on page 1 of this write-up). Put kK on the specification tape. If k = 1 (i.e., if you wish to print once and once only at this check point), you need not specify k at all.

(4) Let a, b, c,..., be the addresses of words you wish to have printed at this checkpoint. Let "G" be any one of the following seven sexadecimal characters: G = 0, 2, 5, 6, S, N, F. The character G specifies the way in which the number is to be printed:

(i) As an order pair: G = 0

(ii) As a right hand address: G = 2

(iii) As a left hand address: G = 6

(iv) As a ten-character sexadecimal word: G = S

(v) As a signed integer: G = N

(vi) As a signed twelve figure fraction: G = F

(vii) As a signed five figure decimal fraction: G = 5

Put on the specification tape: aSG_a, bSG_b ,..., where G_a specifies the way N(a) is to be printed, G_b specifies how N(b) is to be printed, and so on.

If you wish to print the number in R_1, specify the location of word 0 of Code 95,(this is 900 if you use the ILLIAC copy).

If you wish to print the number in R_2, specify the location of word 1 of Code 95,(this is 901 if you use the ILLIAC copy).

If you wish to print a line feed and carriage return character, specify the symbol "J" on the specification tape.

(5) After all the checkpoints and the printing at each checkpoint have been specified, we conclude the specification tape with nNB where "n" is the location of the first order in your program which you wish to have obeyed, and "B" is a sexadecimal character which specifies the nature of the control transfer (in practice, B will be one of 0, 2, 4, 6, in most cases). The control transfer is accomplished by means of a 2Bn order.

(6) If we wish to print, say, the order pair at location 12 during the 10th and 20th times control passes the check point at c, this may be done by having 2 check points at the same address:

e.g. cL 19F 12S0

cL 9F 12S0

MEMORY SPACE NEEDED FOR CODE 95

Code 95 itself occupies 57 memory positions. However, more memory positions must be allowed after the last word of Code 95 to contain the blocked orders, locations of numbers to be printed, and so on. Let "ℓ" be the number of check-points (= the number of L's on your specification tape). Allow 3ℓ memory positions. Furthermore, allow one memory position for each piece of information (word) you wish to have printed out, and one memory position for each line feed character (each J on the specification tape) over and above the automatic line feeds. Thus the total number of memory positions needed is

Memory positions necessary = $57 + 3\ell + s + j$

where "ℓ" is the number of L's on the specification tape, "s" is the number of S's (not counting the second S in SS, however), and "j" is the number of J's.

READING IN OF CODE 95

Code 95 is in two parts, both on the same tape. Read in the program to be checked. Then read in Part I. When Part I is in, the ILLIAC stops and the Specification Tape is read in. The ILLIAC stops again and Part II is read in. The program then starts.

ILLIAC COPY OF CODE 95

The operator's copy uses a bootstrap input which places the code at positions beginning with 900 and overwrites the Decimal Order Input. If you have no program beyond 899, then you can use this and will have 67 locations for the list.

RT: 9/22/55
DATE November 30, 1953
WRITTEN BY D.J. Wheeler
APPROVED BY J.P. Nash

DJW:mge
Sept. 22, 1955

LOCATION	ORDER	NOTES	PAGE 1
0	L5 1L		
	46 53L		
1	L5 F	Used for S0	
	26 47L		
2	00 F		
	00 10F	10×2^{-39}	
3	L5 F		
	26 51L	Used for S2	
4	00 1F		
	00 1F		
5	40 0L	Used for entry from checkpoint	
	S5 1F		
6	L5 F		
	26 28L	Used for S5	
7	L5 F		
	22 50L	Used for S6	
8	L5 (m+1)F*	Used for "closed"	
	22 5L	entry in list	
9	L5 (m)F*		
	26 (m)F		
10	L5 (m+2)F"	Set suppression count	
	S4 3L		
11	40 (m+2)F		F
	27 16L		
12	L5 F		
	26 18L	Used for SS	
13	L5 F		
	22 30L	Used for SN	
14	92 129F		
	22 18L	Used for J	
15	L5 F		
	26 27L	Used for SF	
16	40 F		
	81 4F		
17	50 F	Convert to decimal address a	
	L0 2L		

LOCATION	ORDER		NOTES PAGE 2
18	36 21L		
	L4 2L		
19	74 2L		
	S5 23L		
20	26 16L		
	00 1L		
21	L4 19L		
	42 22L		
22	L5 F		Add address switch to
	26 ( )F		K S N J F or L
23	42 (m')F		Set count for print
	27 16L		
24	81 4F		
	26 43L		
25	L5 9L		
	26 46L		
26	L5 14L		
	26 45L		
27	00 30F		
	26 10L		Shift address
28	42 32L	n	
	00 20F		Set up address of check point
29	46 32L		
	46 30L		
30	L5 (n)F		Replace L. H. order at n by
	10 20F		26 (m)F
31	L5 9L		
	00 20F		
32	50 (n)F		Save order in Q
	40 (n)F		
33	S5 F		
	10 40F		Place 00 nF as L.H. order and
34	L5 F		L.H. address of n.
	00 20F		
35	40 (m+2)F*		As R. H. order in (m+2)
	L5 8L		Place entry in list

LOCATION	ORDER	NOTES PAGE 3 D1
36	40 (m+1)F*	
	L5 5L	
37	40 (m)F*	
	L5 37L	Save address for use of K
38	46 23L	
	L5 35L	
39	46 10L	Save address for use of F
	46 11L	
40	L5 4L	
	L4 4L	Advance m by 1 or 3.
41	L4 4L	
	L4 9L	
42	40 9L	
	22 51L	
43	L4 20L	Construct list entry
	42 44L	
44	00 55F	L5 aF 26 pF from a x 2^{-39} S
	L4 ( )F	2
45	40 (m)F	and L5 F 2^{6}_{2} pF
	27 41L	
46	10 16F	Construct and store L5 (m)F 2B aF N
	80 4F	
47	50 4L	L5 (m)F 2B aF in 1L
	00 12F	
48	L4 F	
	24 49L	Wait for new tape
49	40 1L	
	81 40F	Store $2^{36}/10^{11}$ at end of list
50	40 (m)F	
	L5 10L	To D. P. I. read in program to 3L onwards
51	22 1014F	
	46 37L	
52	46 45L	
	46 50L	Set all addresses dependent on m.
53	L4 4L	
	46 8L	

LOCATION	ORDER		
54	46 36L		
	L4 4L		
55	46 35L		
	27 16L		
	24 1N		
	SPACES		
	57 L5L		
	L85F6		$2^{36}/10^{11}$

KEY: X means --
m = 56L originally
m = current entry in list
n = location of current check point
a = address of number to be printed

3	80 F		
	00 (1)F		
4	00 F		
	00 F	y	
5	22 (n)F		
	L0 10L		
6	46 14L	m	
	46 15L	m	
7	L4 9L		
	46 21L	m+2	
8	46 11L	m+2	
	46 25L	m+2	
9	S5 2F		
	40 1L	q	Save Q
10	09 1F		
	10 9F	-2^{-9}	Store f-1, n, (L.H. order of n)
11	L4 (m+2)F		
	46 5L	n	
12	40 4L		
	36 25L		Bypass print

LOCATION	ORDER		NOTES
13	46 25L	n	
	46 23L	n	
14	L5 (m-1)F		Have we finished printing?
	L0 3L	-2^{-39}	
15	42 (m-1)F		
	00 27F		
16	36 23L	No	→ Yes
	92 133F	↓	Carriage returns (2)
17	L5 23L		
	22 50L		Print n
18	82 40F		
	92 965F		Two spaces
19	19 18F		
	L4 21L		Increase m
20	36 46L		Print R. H. order
	46 21L		
21	L5 (m+2)F		
	32 25L		End Print
22	40 55L		
	26 55L		
23	L5 (n)F		
	10 20F		
24	L5 4L		Restore program
	00 20F		
25	40 (n or m+2)F		
	50 1L		Return to program
26	L5 L		
	22 4L		
27	40 55L		
	22 29L		
28	40 55L		
	L5 49L		Set to count 5
29	46 54L		
	43 3L		

LOCATION	ORDER	NOTES
30	L5 55L	
	40 55L	Print Sign
31	32 32L	
	92 704F	-
32	26 33L	
	92 642F	+
33	L7 55L	
	50 2L	Modulus to Q
34	10 39F	
	L7 3L	
35	32 52L	
	75 2L	10
36	L0 31	
	32 39L	Test for 0
37	L4 3L	
	00 36F	Print decimal digit
38	82 4F	
	10 40F	
39	43 3L	
	L5 54L	
40	L0 50L	Digit Count
	46 54L	
41	32 35L	
	L5 52L	Restore digit count
42	46 54L	
	L1 3L	
43	00 2F	
	32 44L	Reset zero suppression
44	92 02F	
	19 38F	
45	42 3L	
	22 18L	
46	L5 55L	Move R. H. order to L. H. position
	00 20F	
47	40 55L	
	L1 21L	Switch

LOCATION	ORDER		NOTES
48	40 21L		
	L5 55L		Print function
49	82 328F		
	92 961F		Space
50	10 71F		
	10 20F		Convert address to an integer inQ
51	50 2L		
	10 12F		
52	11 795F	(27)	
	L5 50L		
53	74 (u)F		$2^{36}/10^{22} + 47 \times 2^{-39}$
	10 9F		Convert
54	10 (795)F	27	Integer
	26 36L		into a
55	00 F		fraction
	26 L	x	
	26 1N		

RT: 9/22/55
DJW/mge