

THE TRACE METHOD FOR DEBUGGING  
PROGRAMS ON THE IBM 1401\*

The technique known as "tracing" has been a standard method for securing information needed to correct errors in a program ever since stored program computers were first invented. The idea involved is simple--the capabilities of the computer itself are used to provide information about what is happening as each instruction of an undebugged program is being executed. This is generally achieved by having the computer print a line as each instruction is executed, this line providing such information as the address of the instruction, the instruction itself, the contents of the fields addressed by the instruction before and after its execution, and the contents of the index registers at the time of execution. This printed record provides the programmer with the kind of information he needs to correct a program--knowledge of the path the computer actually took through the program, what the instructions actually were at the time of execution, what data were in the fields addressed by each instruction, and what the results of each instruction were. As every programmer knows, errors in writing a program can result in any of these items being quite different from what the programmer intended them to be.

On some computers, the ability to produce a trace is built into the electronic mechanism of the machine and is triggered by flipping a switch. On others, a trace is obtained through use of a program written for that purpose. The trace program is loaded into the machine at the same time as the program that is being tested. The trace program then in effect considers the instructions of the program being tested as the "data"

---

\*Contributed by Emanuel Melichar, Economist, Division of Research and Statistics, Board of Governors.

on which it operates. Its job is to take each of these instructions in the order in which they would have been executed during the running of that program, cause them to be executed, and print the desired information about them.

The IBM 650 and 1410 are examples of computers for which the manufacturer or users provided trace programs. Programmers for these machines have found the trace to be a powerful aid in testing programs.

For the 1401, however, IBM has not provided a trace program. As the majority of 1401 programmers have been trained solely through courses using IBM 1401 manuals rather than general textbooks on programming, most of them are not even aware of the existence of the trace method for debugging programs. However, I think it is safe to say that most 1401 programmers have spent hours working their way through programs trying to localize errors that would have been immediately obvious in the output of a trace that could have been printed at the rate of nearly 600 instructions per minute.

I believe that every 1401 programmer in the System should take steps to obtain his own trace program deck suited for his size of machine and programming method (SPS or Autocoder). Such trace programs have been available through the Research Program Library since July, 1962 (Programs 3.04.01.0 and 3.04.02.0). Because I believe each programmer should have his own program deck and copy of operating instruction, I stand ready to supply such materials in response to either individual or group requests. Furthermore, I urge that as new programmers are hired or return from programming school, they should be given a trace program deck and operating instructions and shown how to use the method. It is clearly a debugging method through which new programmers can learn the many quirks of the 1401 machine.

To appreciate and learn how to make efficient use of a trace program, however, it is necessary to see exactly what it does and what one has to do to make it work. The remainder of this article therefore presents a demonstration of the use of the trace program in debugging another program written especially for the demonstration.

The special program to be debugged in this demonstration has been made just complicated enough to illustrate different aspects of the trace. It employs a couple of instruction loops, a little bit of indexing, and a few chained instructions.

Here is what the special program does. It reads data cards that have alphabetic information in columns 1-60, a number field in columns 61-70, and another number field in columns 70-80. As each card is read the contents are printed and the number fields are each added to one of two accumulator fields in core. When a card with the word TOTAL in columns 1-5 is read, the contents of the two accumulator fields are printed. Because the two fields are debits and credits, respectively, their contents should be equal at this time. One of the accumulator fields is therefore subtracted from the other and the result is compared with a field of zeros. If the result is not zero, the word ERROR and difference between the two fields is also printed to the right of the totals.

The object listing of this program is given on the next two pages. The source instructions are shown in Autocoder because I think that an SPS programmer can read Autocoder language if he tries, but that an Autocoder programmer would have difficulty with SPS. Because the program was originally written in SPS, however, an SPS listing is available on request.

PROGRAM TO ILLUSTRATE 1401 TRACE			ADD	PAGE	1	
LABEL	OP	OPERANDS	SFX CT	LOCN	INSTRUCTION TYPE	CARD
000	JOB	PROGRAM TO ILLUSTRATE 1401 TRACE				
	CTL	631				
	ORG	00333			0333	
BEGIN	SW	0001,0061	7	0333	, 001 061	4
	SW	0071,0087	7	0340	, 071 087	4
READ	BLC	LAST	5	0347	B 533 A	4
	R		1	0352	1	4
	SS	1	2	0353	K 1	4
	MLC	0060,0260	7	0355	M 060 260	4
	C	0005,TOTAL	7	0362	C 005 586	4
	BE	SUM	5	0369	B 437 S	5
	A	0080,ACCUM2	7	0374	A 080 565	5
	A		1	0381	A	5
LOAD	LCA	EDIT,0280&X1	7	0382	L 581 2Y0	5
	MCE	0070&X2,0280&X1	7	0389	E 0P0 2Y0	5
	BCE	DONE,0093,1	8	0396	B 422 093 1	5
	SBR	0089,0020&X1	7	0404	H 089 0S0	6
	SBR	0094,0010&X2	7	0411	H 094 0J0	6
	B	LOAD	4	0418	B 382	6
DONE	S	0095	4	0422	S 095	6
	BCV	NUPAGE	5	0426	B 503 @	6
PRINT	CC	S	2	0431	F S	6
	W	READ	4	0433	2 347	6
SUM	LCA	EDIT,0280	7	0437	L 581 280	7
	MCE	ACCUM1,0280	7	0444	E 553 280	7
	LCA	EDIT,0300	7	0451	L 581 300	7
	MCE	ACCUM2,0300	7	0458	E 565 300	7
	S	ACCUM2,ACCUM1	7	0465	S 565 553	7

LABEL	OP	OPERANDS	SFX	CT	LOCN	INSTRUCTION	TYPE	CARD
	C	ACCUM1,ZERO12	7		0472	C 553 603		8
	BU	ERROR	5		0479	B 508 /		8
PRSUM	CC	J	2		0484	F J		8
	CC	A	2		0486	F A		8
	W		1		0488	2		8
	CS	0332	4		0489	/ 332		8
	CS		1		0493	/		8
	S	ACCUM2	4		0494	S 565		9
	S		1		0498	S		9
	B	READ	4		0499	B 347		9
NUPAGE	CCB	PRINT,1	5		0503	F 431 1		9
ERROR	MLC	ERR,0310	7		0508	M 591 310		9
	LCA	EDIT,0330	7		0515	L 581 330		9
	MCE	ACCUM1,0330	7		0522	E 553 330		9
	B	PRSUM	4		0529	B 484		10
LAST	NOP	0000	4		0533	N 000		10
	H		1		0537	.		10
	B	READ	4		0538	B 347		10
ACCUM1	DCW	@ @	12		0553			10
ACCUM2	DCW	@ @	12		0565			10
EDIT	DCW	@ , , , 0-@	16		0581			11
TOTAL	DCW	@TOTAL@	5		0586			11
ERR	DCW	@ERROR@	5		0591			11
ZERO12	DCW	@000000000000@	12		0603			11
	END	BEGIN				/ 333 080		12

Test data prepared for this program consisted of two data cards and a TOTAL card. The numbers in the data cards were rigged so that the totals would be equal. When the program was run, the following output was obtained:

ITEM 1	48,456,277	526,784	
ITEM 2	526,784	48,456,277	
TOTAL	48,983,061	48,983,061	ERROR

It is obvious that the difference between the two totals is zero and so the word ERROR should not have been printed. We will now use the trace program to find the cause of this error.

To obtain a trace, we must have a trace program deck assembled by ~~our~~ Autocoder, because ~~our~~ undebugged program deck is in Autocoder format. (If we were testing an uncondensed SPS program deck, we would use an uncondensed SPS trace program deck; if testing a condensed SPS program deck, use a condensed SPS trace deck. All versions are available on request.) As the first instruction of our undebugged program is located in 333, the trace deck is ready for use without modification. We remove the last card (END card) from the undebugged program and put the trace program behind the rest of it. The data cards then go behind the trace program. We put the combined deck into the reader hopper, press Load, and a few seconds later have obtained the output shown on the next three pages.

A line-by-line commentary on this trace output is presented shortly. The following list, however, provides a summary of the

333	, 001 061	*		* ‡	
340	, 071 087	*		* ‡	
347	B 533 A	*N		‡	
352	1			‡	
353	K 1			‡	
355	M 060 260	*	*	‡	
362	C 005 586		*ITEM	*TOTAL ‡	
369	B 437 S		*L	‡	
374C	A 080 565		*0000526784	*000000526784 ‡	
382	L 581 2Y0	*	, , , 0-*	, , , 0- ‡	, ,
389	E OP0 2Y0		*0048456277*	48,456,277 ‡	48,45
396	B 422 093 1		*S	* ‡	48,45
404	H 089 0S0		*020*	‡	48,45
411	H 094 0J0 020		*020 010*	‡	48,45
418	B 382 020 010		*L	‡	48,45
382	L 581 2Y0 020 010	*	, , , 0-*	, , , 0- ‡	48,45
389	E OP0 2Y0 020 010		*0000526784*	526,784 ‡	48,45
396	B 422 093 1 020 010		*S	*020 01 ‡	48,45
422	S 095 020 010		*000000000&	‡	48,45
426	B 503 @ 000 000		*F	‡	48,45
431	F S 000 000			‡	48,45
ITEM 1					48,45
433	2 347 000 000		*B	‡	48,45
347	B 533 A 000 000		*N	‡	48,45
352	1 000 000			‡	48,45

353	K 1	000 000			‡	48,45
355	M 060 260	000 000	*	*	‡	48,45
362	C 005 586	000 000		*ITEM *TOTAL	‡	48,45
369	B 437 S	000 000		*L	‡	48,45
374C	A 080 565	000 000		*0048456277 *000048983061	‡	48,45
382	L 581 2Y0	000 000	* , , , 0-	* , , , 0-	‡	
389	E 0P0 2Y0	000 000		*0000526784*	‡	526,784
396	B 422 093 1	000 000		*S *0000000	‡	52
404	H 089 0S0	000 000		*020*	‡	52
411	H 094 0J0	020 000		*02000010*	‡	52
418	B 382	020 010		*L	‡	52
382	L 581 2Y0	020 010	* , , , 0-	* , , , 0-	‡	52
389	E 0P0 2Y0	020 010		*0048456277*	‡	48,456,277
396	B 422 093 1	020 010		*S *0200001	‡	52
422	S 095	020 010		*000000000	‡	52
426	B 503 a	000 000		*F	‡	52
431	F S	000 000			‡	52
ITEM 2						52
433	2 347	000 000		*B	‡	52
347	B 533 A	000 000		*N	‡	52
352	1	000 000			‡	52
353	K 1	000 000			‡	52
355	M 060 260	000 000	*	*	‡	52
362	C 005 586	000 000		*TOTAL *TOTAL	‡	52
369	B 437 S	000 000		*L	‡	52



437	L	581 280	000 000	* , , , 0-* , , , 0- ‡		
444	E	553 280	000 000	*000048983061*	48,983,061 ‡	48,983,061
451	L	581 300	000 000	* , , , 0-* , , , 0- ‡		48,983,061
458	E	565 300	000 000	*000048983061*	48,983,061 ‡	48,983,061
465	S	565 553	000 000	*000048983061	*000000000000 ‡	48,983,061
472	C	553 603	000 000	*000000000000 ‡	*000000000000 ‡	48,983,061
479	B	508 /	000 000	*M	‡	48,983,061
508	M	591 310	000 000	*ERROR*061	ERROR ‡	48,983,061
515	L	581 330	000 000	* , , , 0-* , , , 0- ‡		48,983,061
522	E	553 330	000 000	*000000000000 ‡	‡	48,983,061
529	B	484	000 000	*F	‡	48,983,061
484	F	J	000 000		‡	48,983,061
486	F	A	000 000		‡	48,983,061
TOTAL						48,983,061
488	2		000 000		‡	48,983,061
489C	/	332	000 000	*	‡	
494C	S	565	000 000	*000000000000 ‡	‡	
499	B	347	000 000	*B	‡	
347	B	533 A	000 000	*N	‡	
533	N	000	000 000		‡	

information that has been printed on each line:

<u>Print positions</u>	<u>Information printed</u>
1-3	Address of the instruction being traced.
4	The letter "C" if one or more fully-chained instructions follow the instruction being traced.
6	OP-code of the instruction being traced.
8-10	A-operand of the instruction being traced.
12-14	B-operand of the instruction being traced.
16	d-character of the instruction being traced.
18-20	Contents of Index 1.
22-24	Contents of Index 2.
26-28	Contents of Index 3.
30-43	Up to 14 digits of the contents, after execution, of the field addressed by the A-operand of the instruction being traced. An asterisk indicates the length of the field.
45-58	Up to 14 digits of the contents, after execution, of the field addressed by the B-operand of the instruction being traced. An asterisk indicates the length of the field.
59	Always a blank.
60	Always a record-mark.

In actual use of the trace program, we would proceed directly to examine the suspected instructions, in this case the compare instruction in location 472. (Look down the left hand column of the output and find this instruction. It has been circled.) The printout of the fields being compared shows that we got an unequal indication because we compared a signed zero field with an unsigned zero field. Looking at the line above, we see that the sign was introduced by the subtract instruction used to obtain the difference between the two fields. The unequal indication caused the word ERROR to be printed. The mistake in programming has been easily located.

As we are not yet familiar with the trace output, however, let us start at the beginning of the output and follow the path taken by the computer through our undebugged program, as recorded in the trace output.

333 , 001 061

\* \* †

340 , 071 087

\* \* †

The program starts in 333. Note that the address of the instructions executed and the instructions themselves are printed at the left side of the page.

The first two instructions executed set word marks in the read area and in index register 1. The position of the asterisks, each of which is followed by a blank, shows that the A and B fields were one-digit blank fields after the instructions were executed. The record marks are printed down the page to form a "fence" separating the trace information from anything else that may be in the rest of the print area.

347 B 533 A

\*N †

For this branch instruction testing the last card switch, the contents of the A-field printed by the trace is, of course, the op code of the instruction located there. This item of information is not very interesting and is simply ignored by the programmer.

352 I

†

353 K 1

†

This select stacker instruction following the read instruction was executed, but was not effective during the trace run because the time limit was exceeded. During the trace run, all data cards fell into the normal read pocket.

355 M 060 260

\* \* †

This instruction moved the contents of 1 - 60 into 201 - 260. The trace has printed only the last 14 digits of each of these fields. The trace output shows that these digits were blanks.

362 C 005 586

\*ITEM \*TOTAL †

The trace of this compare instruction shows the contents of the two 5-digit fields that were compared. This is the first interesting information that has turned up in the A and B fields of the instructions traced so far. Note how the positions of the asterisks show that there was a word mark over the first letter of each of the words being compared.

369 B 437 S

\*L

‡

The fields were unequal and so the branch to 437 was not taken. Again, the contents of location 437 were printed, but are simply ignored by us.

374C A 080 565

\*0000526784 \*000000526784 ‡

The contents of both fields addressed by this add instruction are printed after the instruction is executed. Note the "C" printed right after the address of the instruction. This indicates that a chained instruction followed the instruction shown. The chained instruction was executed during the trace run, but the trace did not print a line for it.

382 L 581 2Y0

\* , , , 0-\* , , , 0- ‡

389 E 0P0 2Y0

\*0048456277\* 48,456,277 ‡ 48,456

Note that these instructions moved an edit word and then a number into the part of the print area not used by the trace program. We simply ignore anything to the right of the "fence" formed by the record marks.

396 B 422 093 1

\*S \* ‡ 48,456

This branch was not taken because, as the trace shows, the contents of 093 were a blank rather than a 1.

404 H 089 0S0

\*020\* ‡ 48,456

411 H 094 0J0 020

\*020 010\* ‡ 48,456

Note how the Store B Register instructions were used here to add to the index registers, using the technique described in the May 1, 1963 Newsletter.

418 B 382 020 010

\*L ‡ 48,456

Note where the contents of the index registers are printed. All three registers were printed out on each line from the beginning of the trace, but until now they were all blank. Note that on each line the contents of the registers were printed as they were before the instruction on that line was executed, because these are the values by which that instruction would be indexed if it is an indexed instruction.

This unconditional branch was obviously taken. The instruction in location 382 was executed for a second time during the trace run. Note how the trace printout is revealing the path taken during the execution of the program.

382	L	581	2Y0	020	010	*	,	,	,	0-	*	,	,	,	0-	‡	48,450
389	E	0P0	2Y0	020	010	*0000526784*		526,784		‡						48,450	
396	B	422	093	1	020	010	*S		*020		01		‡				48,450

This branch is taken this time because, as the trace shows, the contents of location 093 are a 1.

422	S	095		020	010	*00000000E				‡						48,450
-----	---	-----	--	-----	-----	------------	--	--	--	---	--	--	--	--	--	--------

Note how this single 4-digit subtract instruction has cleared 2 index registers, employing the technique described in the May 1, 1963 Newsletter.

426	B	503	@	000	000	*F				‡						48,450
431	F	S		000	000					‡						48,450
ITEM	1															48,450
433	2	347		000	000	*B				‡						48,450

The print instruction has caused the contents of the print area to be written just above the trace information for the print instruction. The skip after print instruction which preceded the print instruction was not effective during the test run because it was nullified by the trace program's own skip after print instruction given during the printing of the trace information for the skip instruction.

347	B	533	A	000	000	*N				‡						48,450	
352	1			000	000					‡						48,450	
353	K	1		000	000					‡						48,450	
355	M	060	260	000	000	*	*				‡						48,450
362	C	005	586	000	000	*ITEM		*TOTAL		‡						48,450	
369	B	437	S	000	000	*L				‡						48,450	

374C	A	080 565	000 000	*0048456277	*000048983061	‡	48,450
382	L	581 2Y0	000 000	, , , 0-	, , , 0-	‡	
389	E	0P0 2Y0	000 000	*0000526784*	526,784	‡	520
396	B	422 093 1	000 000	*S	*0000000	‡	520
404	H	089 050	000 000	*020*		‡	520
411	H	094 0J0	020 000	*020000I0*		‡	520
418	B	382	020 010	*L		‡	520
382	L	581 2Y0	020 010	, , , 0-	, , , 0-	‡	520
389	E	0P0 2Y0	020 010	*0048456277*	48,456,277	‡	520
396	B	422 093 1	020 010	*S	*0200001	‡	520
422	S	095	020 010	*00000000&		‡	520
426	B	503 a	000 000	*F		‡	520
431	F	S	000 000			‡	520
ITEM 2							520
433	2	347	000 000	*B		‡	520

A second card was read and was processed by the same routines as the first card.

347	B	533 A	000 000	*N		‡	520
352	1		000 000			‡	520
353	K	1	000 000			‡	520
355	M	060 260	000 000	*	*	‡	520
362	C	005 586	000 000	*TOTAL	*TOTAL	‡	520
369	B	437 S	000 000	*L		‡	520

A third card was read and the trace output shows that it had the word TOTAL in columns 1 - 5, which caused a branch to the routine that prints and compares the totals that have been accumulated.

437	L	581 280	000 000	* , , , 0-* , , , 0- ‡		
444	E	553 280	000 000	*000048983061*	48,983,061 ‡	48,983
451	L	581 300	000 000	* , , , 0-* , , , 0- ‡		48,983
458	E	565 300	000 000	*000048983061*	48,983,061 ‡	48,983

The contents of the accumulator fields were edited and moved to the print area.

465	S	565 553	000 000	*000048983061	*000000000000& ‡	48,983
472	C	553 603	000 000	*000000000000&	*000000000000 ‡	48,983
479	B	508 /	000 000	*M	‡	48,983
508	M	591 310	000 000	*ERROR*061	ERROR ‡	48,983

One of the accumulators was subtracted from the other and the result compared with a field of zeros. The programmer's error is revealed by the trace output. The result of the subtraction was a signed field of zeros and so the comparison with an unsigned field of zeros resulted in the setting of the unequal indicator.

515	L	581 330	000 000	* , , , 0-* , , , 0- ‡		48,983
522	E	553 330	000 000	*000000000000&*	‡	48,983
529	B	484	000 000	*F	‡	48,983
484	F	J	000 000		‡	48,983
486	F	A	000 000		‡	48,983
TOTAL						48,983
488	2		000 000		‡	48,983
489C	/	332	000 000	*	‡	
494C	S	565	000 000	*000000000000&	‡	

A chained 4-digit subtract instruction provided an efficient way of clearing both accumulator fields.



499	B	347	000 000	*B	‡
347	B	533 A	000 000	*N	‡
533	N	000	000 000		‡

The last instruction printed by the trace is the instruction that came just before the one in which a programmed or process halt occurred. This is because the trace information for each instruction is printed after the execution of that instruction.

The I-address of the halt shown on the computer panel after the trace run will be in the area of core used by the trace program and therefore has no value to the programmer.