

LOC	OBJECT CODE	ADDR1	ADDR2	STMT
-----	-------------	-------	-------	------

2				*****
3				*
4				* CLCLE Unaligned Buffers Test
5				*
6				* NOTE: This is a copy of the CLCL Unaligned Buffers Test
7				* modified to test the CLCLE instruction and CC=3
8				* with lengths > 4096.
9				* James Wekel August 2022
10				*****
11				*
12				* This program tests proper functioning of the CLCLE instruction's
13				* optimization logic (specifically, the "mem_cmp" function that the
14				* CLCLE instruction makes use of) to ensure the location of the in-
15				* equality is properly reported.
16				*
17				* Depending on the alignment of the two operands being compared, if
18				* the length of the compare is large enough that it would cause the
19				* comparison to cross a page boundary for both operands and the in-
20				* equality occurs at an offset past the distance each operand is
21				* from its respective page boundary added together, then the address
22				* of the inequality that CLCLE returns would be off by the shorter
23				* of the two distances.
24				*
25				* For example, if the operand addresses were X'123456' and X'456789'
26				* (and the page size was X'800') and the inequality was at (or past)
27				* X'123877', then CLCLE would incorrectly report the address of the
28				* inequality as being at address X'123877' - X'77' = X'123800':
29				*
30				* X'123456' is X'3AA' bytes from the end of its page boundary.
31				* X'456789' is X'77' bytes from the end of its page boundary.
32				* The true inequality is at X'123877' (X'123456' + X'77' + X'3AA').
33				*
34				* The optimization logic would perform three separate compares: the
35				* first starting at X'123456' for a length of X'77'. The second one
36				* at address X'1234CD' (X'123456' + X'77') for a length of X'3AA',
37				* and the third and final compare at address X'123877' (X'123456' +
38				* X'77' + X'3AA') for a length of at least one byte.
39				*
40				* Due to a bug in the original optimization logic however the length
41				* of the first compare would not be added to the calculated offset of
42				* where the inequality was located at. That is to say, the offset of
43				* the inequality would be calculated as operand-1 + X'3AA' instead of
44				* operand-1 + X'77' + X'3AA'. The X'77' offset would get erroneously
45				* lost, thereby causing the location of the inequality to be reported
46				* X'77' bytes BEFORE where the actual inequality was actually located
47				* at! (Oops!)
48				*
49				* The bug has since been fixed of course but to ensure such does not
50				* occur again, this standalone runtest test performs a series of CLCL
51				* comparisons whose parameters are such that they end up tripping the
52				* bug as described. Thank you to Dave Kreiss for reporting the bug.
53				*

LOC	OBJECT CODE	ADDR1	ADDR2	STMT
-----	-------------	-------	-------	------

54 *****

LOC	OBJECT CODE	ADDR1	ADDR2	STMT
-----	-------------	-------	-------	------

```

56 *****
57 *
58 *                               EXAMPLE RUNTEST TEST CASE
59 *
60 *           *Testcase CLCLE-02-unaligned-buffers Test
61 *
62 *           archlvl           390
63 *           mainsize          3
64 *           numcpu             1
65 *           sysclear
66 *
67 *           loadcore           "$ (testpath)/CLCLE-02-unaligned-buffers.core"
68 *
69 *           runtest            0.1
70 *           *Done
71 *
72 *****

```


LOC	OBJECT CODE	ADDR1	ADDR2	STMT
				3500 *****
				3501 * Initiate the CLCLE CSECT in the CODE region
				3502 * with the location counter at 0
				3503 *****
				3505 CLCLE ASALOAD REGION=CODE
		000000	081831	3506+CLCLE START 0, CODE
000000	000A0000	00000008		3508+ PSW 0,0,2,0,X'008' 64-bit Restart ISR Trap New PSW
000008		000008	000058	3509+ ORG CLCLE+X'058'
000058	000A0000	00000018		3511+ PSW 0,0,2,0,X'018' 64-bit External ISR Trap New PSW
000060	000A0000	00000020		3512+ PSW 0,0,2,0,X'020' 64-bit Supervisor Call ISR Trap New PSW
000068	000A0000	00000028		3513+ PSW 0,0,2,0,X'028' 64-bit Program ISR Trap New PSW
000070	000A0000	00000030		3514+ PSW 0,0,2,0,X'030' 64-bit Machine Check Trap New PSW
000078	000A0000	00000038		3515+ PSW 0,0,2,0,X'038' 64-bit Input/Output Trap New PSW
000080		000080	000200	3516+ ORG CLCLE+512
				3518 *****
				3519 * Create IPL (restart) PSW
				3520 *****
				3522 ASAIPL IA=BEGIN
		000000	081831	3523+CLCLE CSECT
000200		000200	000000	3524+ ORG CLCLE
000000	00080000	00000200		3525+ PSW 0,0,0,0,BEGIN,24
000008		000008	000200	3526+ ORG CLCLE+512 Reset CSECT to end of assigned storage area
		000000	081831	3527+CLCLE CSECT
				3529 *****
				3530 * The actual "CLCLE" program itself...
				3531 *****
				3532 *
				3533 * Architecture Mode: ESA/390
				3534 *
				3535 * Addressing Mode: 31-bit
				3536 *
				3537 * Register Usage: R12 - R13 Base registers
				3538 * R0 - R1 CLCLE Operand-1
				3539 * R14 - R15 CLCLE Operand-2
				3540 * R2 - R11 Work registers
				3541 *
				3542 *****
000200		000200		3544 USING BEGIN,R12 FIRST Base Register
000200		001200		3545 USING BEGIN+4096,R13 SECOND Base Register
000200	05C0			3547 BEGIN BALR R12,0 Initialize FIRST base register
000202	06C0			3548 BCTR R12,0 Initialize FIRST base register
000204	06C0			3549 BCTR R12,0 Initialize FIRST base register
000206	41D0 C800		000800	3551 LA R13,2048(,R12) Initialize SECOND base register
00020A	41D0 D800		000800	3552 LA R13,2048(,R13) Initialize SECOND base register

LOC	OBJECT CODE	ADDR1	ADDR2	STMT
				3554 *****
				3555 * Compare DATA1 and DATA2 one BUFFSIZE at a time...
				3556 *****
				3558 * R4 R5 R6 R7 R8 R9
00020E	9849 C090		000290	3559 LM R4,R9,=A(BUFFER1,DATA1,BUFFER2,DATA2,BUFFSIZE,DATASIZE)
000212	1598			3561 CLR R9,R8 DATASIZE greater than BUFFSIZE?
000214	47B0 C01A		00021A	3562 BNL CHNKLOOP Yes, get started...
000218	1889			3563 LR R8,R9 No, only compare however much we have!
				3565 * Fill buffers with next chunk of data...
00021A	1804			3567 CHNKLOOP LR R0,R4 R0 --> BUFFER1
00021C	1825			3568 LR R2,R5 R2 --> DATA1
00021E	1818			3569 LR R1,R8 R1 <= BUFFSIZE
000220	1838			3570 LR R3,R8 R3 <= BUFFSIZE
000222	0E02			3571 MVCL R0,R2 Copy into BUFFER1 <= next DATA1 chunk
000224	1806			3573 LR R0,R6 R0 --> BUFFER2
000226	1827			3574 LR R2,R7 R2 --> DATA2
000228	1818			3575 LR R1,R8 R1 <= BUFFSIZE
00022A	1838			3576 LR R3,R8 R3 <= BUFFSIZE
00022C	0E02			3577 MVCL R0,R2 Copy into BUFFER2 <= next DATA2 chunk
				3579 * Prepare for CLCLE...
00022E	1804			3581 LR R0,R4 R0 --> BUFFER1
000230	18E6			3582 LR R14,R6 R14 --> BUFFER2
000232	1818			3583 LR R1,R8 R1 <= BUFFSIZE
000234	18F8			3584 LR R15,R8 R15 <= BUFFSIZE
				3586 * Compare the two buffers...
				3588 * Compare BUFFER1 with BUFFER2..
000236	A90E 0000		000000	3589 CONTINUE CLCLE R0,R14,0 with padding x'00'
00023A	4710 C036		000236	3590 BC B'0001',CONTINUE CC=3, not finished
00023E	4780 C05C		00025C	3591 BE NXTCHUNK Equal: Buffer compare complete
				3593 * Inequality found: VERIFY ITS ACCURACY!
000242	18A0			3595 LR R10,R0 R10 --> Supposed unequal byte
000244	D500 A000 E000	000000	000000	3596 CLC 0(1,R10),0(R14) Valid inequality?
00024A	4780 C080		000280	3597 BE FAILURE Bogus inequality! CLCLE BUG! FAIL!
				3599 * CLCLE was correct. Get past inequality
				3600 * and finish comparing the buffer data if
				3601 * there is any data remaining in the buffer
				3602 * that we haven't compared yet...
00024E	4A00 C0A8		0002A8	3604 AH R0,=H'1' Get past unequal byte
000252	4AE0 C0A8		0002A8	3605 AH R14,=H'1' Get past unequal byte

LOC	OBJECT	CODE	ADDR1	ADDR2	STMT
					3635 *****
					3636 * Working Storage
					3637 *****
					3638 *
					3639 * The specific bug that was reported:
					3640 *
					3641 * 4DE 787FE B54 87F46 B54
					3642 * 4DF 787FF B53 87F47 B53
					3643 *
					3644 * F32 79252 100 8899A 100 (BOGUS!)
					3645 *
					3646 * FEA 7930A 048 88A52 048
					3647 * FEB 7930B 047 88A53 047
					3648 *
					3649 *****
000290					3651 LTORG , Literals pool
000290	00020320	00060000			3652 =A(BUFFER1,DATA1,BUFFER2,DATA2,BUFFSIZE,DATASIZE)
0002A8	0001				3653 =H'1'
			002000	000001	3655 BUFFSIZE EQU (8*1024)
			001832	000001	3656 DATASIZE EQU X'1832'
			000320	000001	3658 BUFF10FF EQU X'320'
			000A68	000001	3659 BUFF20FF EQU X'A68'
0002AA			0002AA	020320	3661 ORG CLCLE+(1*(128*1024))+BUFF10FF
020320	00000000	00000000			3662 BUFFER1 DC (BUFFSIZE/8)XL8'00'
022320			022320	040A68	3664 ORG CLCLE+(2*(128*1024))+BUFF20FF
040A68	00000000	00000000			3665 BUFFER2 DC (BUFFSIZE/8)XL8'00'
042A68			042A68	060000	3667 ORG CLCLE+(3*(128*1024)) X'60000'
060000	00000000	00000000			3668 DATA1 DC (DATASIZE)X'00' X'60000'
061832			061832	080000	3670 ORG CLCLE+(4*(128*1024)) X'80000'
080000	00000000	00000000			3671 DATA2 DC (DATASIZE)X'00' X'80000'
081832			081832	08104E	3673 ORG DATA2+X'104E'
08104E	FF				3674 DC X'FF'
08104F			08104F	08104F	3676 ORG DATA2+X'104F'
08104F	FF				3677 DC X'FF'
081050			081050	0816EA	3679 ORG DATA2+X'16EA'
0816EA	FF				3680 DC X'FF'
0816EB			0816EB	0816EB	3682 ORG DATA2+X'16EB'
0816EB	FF				3683 DC X'FF'
0816EC			0816EC	081832	3685 ORG DATA2+DATASIZE

SYMBOL	TYPE	VALUE	LENGTH	DEFN	REFERENCES									
BEGIN	I	000200	2	3547	3525	3544	3545							
BUFF1OFF	U	000320	1	3658	3661									
BUFF2OFF	U	000A68	1	3659	3664									
BUFFER1	X	020320	8	3662	3559									
BUFFER2	X	040A68	8	3665	3559									
BUFFSIZE	U	002000	1	3655	3662	3665	3559							
CHNKLOOP	I	00021A	2	3567	3562	3616	3618							
CLCLE	J	000000	530482	3506	3509	3516	3524	3526	3661	3664	3667	3670		
CODE	2	000000	530482	3506										
CONTINUE	I	000236	4	3589	3590	3607								
DATA1	X	060000	1	3668	3559									
DATA2	X	080000	1	3671	3673	3676	3679	3682	3685	3559				
DATASIZE	U	001832	1	3656	3668	3671	3685	3559						
DWAT0008	3	000278	8	3628	3627									
DWAT0009	3	000288	8	3633	3632									
FAILURE	H	000280	2	3631	3597									
IMAGE	1	000000	530482	0										
NXTCHUNK	I	00025C	2	3611	3591									
R0	U	000000	1	3691	3567	3571	3573	3577	3581	3589	3595	3604		
R1	U	000001	1	3692	3569	3575	3583	3606						
R10	U	00000A	1	3701	3595	3596								
R11	U	00000B	1	3702										
R12	U	00000C	1	3703	3544	3547	3548	3549	3551					
R13	U	00000D	1	3704	3545	3551	3552							
R14	U	00000E	1	3705	3582	3589	3596	3605						
R15	U	00000F	1	3706	3584	3607								
R2	U	000002	1	3693	3568	3571	3574	3577						
R3	U	000003	1	3694	3570	3576								
R4	U	000004	1	3695	3559	3567	3581							
R5	U	000005	1	3696	3568	3611								
R6	U	000006	1	3697	3573	3582								
R7	U	000007	1	3698	3574	3612								
R8	U	000008	1	3699	3561	3563	3569	3570	3575	3576	3583	3584	3611	3612
R9	U	000009	1	3700	3559	3561	3563	3614	3617					
SUCCESS	H	000270	2	3626	3615									
=A(BUFFER1,DATA1,BUFFER2,DATA2,BUFFSIZE,DATASIZE)														
	A	000290	4	3652	3559									
=H'1'	H	0002A8	2	3653	3604	3605								

DESC	SYMBOL	SIZE	POS	ADDR
------	--------	------	-----	------

Entry: 0

Image	IMAGE	530482	00000-81831	00000-81831
Region	CODE	530482	00000-81831	00000-81831
CSECT	CLCLE	530482	00000-81831	00000-81831

STMT

FILE NAME

```
1 /devstor/dev/satk/samples/tests/./CLCLE-02-unaligned-buffers.asm
2 /home/tn529/dev/satk/srcasm/satk.mac
```

**** NO ERRORS FOUND ****