Subject: MCR-10030, interpret_ptr_ Return Data Repairs

Author: Gary Dixon

Date: February 16, 2017

Introduction

The interpret_ptr_ subroutine fills in a caller-provided structure with information about an input pointer. While the structure has elements covering a wide range of pointer attributes, the subroutine sometimes returns incorrect information, or fails to provide meaningful attributes. Also, pointers to hardcore segments are not handled by the subroutine.

• URL of Multics Change Ticket: http://multics-trac.swenson.org/ticket/44

interpret_ptr_ is used as follows:

```
   interpret_ptr_:
        procedure (a_ptr, a_framep, strp);

   dcl a_ptr ptr,        /* ptr to be interpreted (input)                  */
       a_framep ptr,     /* ptr to associated stack frame, or null (input) */
       strp ptr;         /* ptr to return structure (input)                */
```

The strp parameter points to storage for a structure provided by the caller; interpret_ptr_ populates this structure with return data.

```
/* BEGIN INCLUDE FILE ... interpret_ptr_struc.incl.pl1 */

dcl 1 struc based (strp) aligned,   /* This useful structure is in trace_stack pgms */
    2 comment char (64),            /* name of boundseg, or thing pointed to, or .. */
    2 segment char (32),            /* name of segment, or segment number with #    */
    2 entryn char (33),             /* entry name, preceded by $, or blank          */
    2 offset char (6),              /* offset in segment, or blank                  */
    2 opname char (32),             /* may be pl1 operator name                     */
    2 ring char (1) unal,           /* ring number of ptr                           */
    2 std_flag bit (1) unal,        /* TRUE if std object.                          */
    2 xpad bit (26) unal,
    2 bitoff char (4),              /* bit offset                                   */
    2 modifier char (8),            /* ptr modifier, or special code like "noaccess"*/
    2 symbolp ptr,                  /* ptr to symbol section                        */
    2 instruction_counter fixed bin,/* binary instruction counter                   */
    2 sourcemap fixed bin,          /* source map offset in symbol section          */
    2 compiler char (8),            /* name of compiler                             */
    2 text_ptr ptr,                 /* ptr to segment text                          */
    2 stack_ptr ptr;                /* ptr to stack frame                           */

dcl 1 strbuf aligned like struc;    /* If referenced, will make storage for struc   */

/* END INCLUDE FILE ... interpret_ptr_struc.incl.pl1 */
```

The subroutine has several flaws in returning data:

1. The subroutine fails to fully initialize its return structure. Some values are never set. Some are set only later in the program; whereas earlier exits from the program return garbage in fields set later.
2. The program declares its own version of an ITS pointer structure, which is missing some fields. Fields are declared solely as bit strings, forcing the program to do conversions from bit to numeric values in many cases.
3. struc.ring returns a value 1-less than the ring number actually in the input pointer value.
4. struc.bitoff should return (<bit-offset>) if a nonzero bit offset appears in the input pointer value. However, right now, the code does not initialize or place a correct value in that return field.
5. When asked to interpret a location not containing a pointer, the program faults as it tries to load the possible pointer value into a pointer register.
6. The subroutine fails to obtain information about unsnapped links.
7. The program identifies the lowest stack number (2nd argument of hcs_$high_low_seg_count) as being within ring 0. The documentation for the subroutine states this value is the first segment number outside ring 0.

Proposed Changes

Change interpret_ptr_ as follows:

1. Fully initialize the return structure when entering the program.
2. Use the its and its_unsigned structure declarations in its.incl.pl1 as declarations of internal pointer structure. The its_unsigned structure declares most fields as fixed bin(N) unsigned unaligned elements, allowing them to be used directly as numbers. Use the ITS_MODIFIER constant in this include file to identify an ITS tag.
3. Converting ring number from pointer to a character was done via a substr operation. While possible ring numbers are 0-based, substr positions begin with position 1 (first character of string). This led to a one-off conversion from the ring number bits to a character describing the ring number. Instead, use a direct conversion from an unsigned ring number to the corresponding character.
4. Existing code totally ignores any nonzero bit offset in the input pointer. Using the its_unsigned.bit_offset number, a nonzero bit offset may be returned. Thus, an offset referencing bit 27 would be returned in struc.bitoff as: “(27)”. If the offset is zero, then struc.bitoff would be set to: “”.
5. Reorder initial program code to verify that the pointer has an ITS modifier, or a Fault Tag 2 (FT2, or unsnapped link) modifier before attempting to treat the location as a pointer. Return immediately with struc.comment = “(argument is not a pointer)” unless one of these two modifiers is found.

6. For an unsnapped link, the program was calling the interpret_link_ subroutine with an incorrect argument. Change to pass a pointer to the storage containing the original pointer parameter, rather than a copy of that original parameter. interpret_link_ needs to know where the unsnapped link is located in order to determine its attributes. Include the object_link_dcls.incl.pl1 to use its FAULT_TAG_2 constant to identify the unsnapped link modifier field value.
7. Change the test for a ring 0 segment number to be < the value returned in 2nd argument of hcs_$high_low_seg_count.

Bound Segment: >sss>bound_trace_stack_

Changed source: interpret_ptr_.pl1

Documentation

Though an external entrypoint in bound_trace_stack_ that is called from several other bound segments, the interpret_ptr_ subroutine is not documented in any manual or info segment.

Version History

Date	Revision	Author	Comment
2017-02-16	0.1	Gary Dixon	Initial version of MCR.
2017-02-17	0.2	Gary Dixon	Add more changes for problems found while testing the fixes mentioned in the ticket.