```
**********************************************************
    4341 Fri Nov 14 18:05:27 2014
new/usr/src/uts/intel/amd64/ml/amd64.il
5291 x86 {high,low}bit rely on undefined behavior
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/
/ In-line functions for amd64 kernels.
/

/
/ return current thread pointer
/
/ NOTE: the "0x18" should be replaced by the computed value of the
/       offset of "cpu_thread" from the beginning of the struct cpu.
/       Including "assym.h" does not work, however, since that stuff
/       is PSM-specific and is only visible to the 'unix' build anyway.
/       Same with current cpu pointer, where "0xc" should be replaced
/       by the computed value of the offset of "cpu_self".
/       Ugh -- what a disaster.
/
        .inline threadp,0
        movq    %gs:0x18, %rax
        .end

/
/ return current cpu pointer
/
        .inline curcpup,0
        movq    %gs:0x10, %rax
        .end

/
/ return caller
/
        .inline caller,0
        movq    8(%rbp), %rax
        .end

/
/ convert ipl to spl.  This is the identity function for i86
/
```



```
        .inline ipltospl,0
        movq    %rdi, %rax
        .end

/
/ find the low order bit in a word
/
        .inline lowbit,4
        movq    $-1, %rax
        bsfq    %rdi, %rax
        incq    %rax
        .end

/
/ Networking byte order functions (too bad, Intel has the wrong byte order)
/

        .inline htonll,4
        movq    %rdi, %rax
        bswapq  %rax
        .end

        .inline ntohll,4
        movq    %rdi, %rax
        bswapq  %rax
        .end

        .inline htonl,4
        movl    %edi, %eax
        bswap   %eax
        .end

        .inline ntohl,4
        movl    %edi, %eax
        bswap   %eax
        .end

        .inline htons,4
        movl    %edi, %eax
        bswap   %eax
        shrl    $16, %eax
        .end

        .inline ntohs,4
        movl    %edi, %eax
        bswap   %eax
        shrl    $16, %eax
        .end

/*
 * multiply two long numbers and yield a u_lonlong_t result
 * Provided to manipulate hrtime_t values.
 */
        /* XX64 These don't work correctly with SOS9 build 13.0 yet
        .inline mul32, 8
        xorl    %edx, %edx
        movl    %edi, %eax
        mull    %esi
        shlq    $32, %rdx
        orq     %rdx, %rax
        ret
        .end
        */
/*
 * Unlock hres_lock and increment the count value. (See clock.h)
 */
```

```
119         .inline unlock_hres_lock, 0
120         lock
121         incl    hres_lock
122         .end

124         .inline atomic_orb,8
125         movl    %esi, %eax
126         lock
127         orb     %al,(%rdi)
128         .end

130         .inline atomic_andb,8
131         movl    %esi, %eax
132         lock
133         andb    %al,(%rdi)
134         .end

136 /*
137  * atomic inc/dec operations.
138  *      void atomic_inc16(uint16_t *addr) { ++*addr; }
139  *      void atomic_dec16(uint16_t *addr) { --*addr; }
140  */
141         .inline atomic_inc16,4
142         lock
143         incw    (%rdi)
144         .end

146         .inline atomic_dec16,4
147         lock
148         decw    (%rdi)
149         .end

151 /*
152  * atomic bit clear
153  */
154         .inline atomic_btr32,8
155         lock
156         btrl %esi, (%rdi)
157         setc %al
158         .end

160 /*
161  * Call the pause instruction.  To the Pentium 4 Xeon processor, it acts as
162  * a hint that the code sequence is a busy spin-wait loop.  Without a pause
163  * instruction in these loops, the P4 Xeon processor may suffer a severe
164  * penalty when exiting the loop because the processor detects a possible
165  * memory violation.  Inserting the pause instruction significantly reduces
166  * the likelihood of a memory order violation, improving performance.
167  * The pause instruction is a NOP on all other IA-32 processors.
168  */
169         .inline ht_pause, 0
170         pause
171         .end

173 /*
174  * inlines for update_sregs().
175  */
176         .inline __set_ds, 0
177         movw    %di, %ds
178         .end

180         .inline __set_es, 0
181         movw    %di, %es
182         .end

184         .inline __set_fs, 0
```



```
185         movw    %di, %fs
186         .end

188         .inline __set_gs, 0
189         movw    %di, %gs
190         .end

192         /*
193          * OPTERON_ERRATUM_88 requires mfence
194          */
195         .inline __swapgs, 0
196         mfence
197         swapgs
198         .end

200 /*
201  * prefetch 64 bytes
202  */

204         .inline prefetch_read_many,8
205         prefetcht0      (%rdi)
206         prefetcht0      32(%rdi)
207         .end

209         .inline prefetch_read_once,8
210         prefetchnta     (%rdi)
211         prefetchnta     32(%rdi)
212         .end

214         .inline prefetch_write_many,8
215         prefetcht0      (%rdi)
216         prefetcht0      32(%rdi)
217         .end

219         .inline prefetch_write_once,8
220         prefetcht0      (%rdi)
221         prefetcht0      32(%rdi)
222         .end
```

```
**********************************************************
    2106 Fri Nov 14 18:05:27 2014
new/usr/src/uts/intel/asm/bitmap.h
5291 x86 {high,low}bit rely on undefined behavior
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2014 Nexenta Systems, Inc.  All rights reserved.
 */

#ifndef _ASM_BITMAP_H
#define _ASM_BITMAP_H

#include <sys/ccompile.h>
#include <sys/types.h>

#ifdef  __cplusplus
extern "C" {
#endif

#if !defined(__lint) && defined(__GNUC__)

#if defined(__amd64)
#define __SUF   "q"
#elif defined(__i386)
#define __SUF   "l"
#else
#error "port me"
#endif

extern __GNU_INLINE int
highbit(ulong_t i)
{
        long value;
        uint8_t zf;
        long value = -1l;

        __asm__(
            "bsr" __SUF " %2,%0;"
            "setz %1"
            : "=r" (value), "=q" (zf)
            : "mr" (i)
            "bsr" __SUF " %1,%0"
            : "+r" (value)
            : "r" (i)
            : "cc");

        return (zf ? 0 : (value + 1));
        return ((int)(value + 1));
}

extern __GNU_INLINE int
lowbit(ulong_t i)
{
        long value;
        uint8_t zf;
        long value = -1l;

        __asm__(
            "bsf" __SUF " %2,%0;"
            "setz %1"
            : "=r" (value), "=q" (zf)
            : "mr" (i)
            "bsf" __SUF " %1,%0"
            : "+r" (value)
            : "r" (i)
            : "cc");

        return (zf ? 0 : (value + 1));
        return ((int)(value + 1));
}
```



```
_____unchanged_portion_omitted_
```

```
**********************************************************
   82054 Fri Nov 14 18:05:27 2014
new/usr/src/uts/intel/ia32/ml/i86_subr.s
patch keith-feedback
5291 x86 {high,low}bit rely on undefined behavior
**********************************************************
______unchanged_portion_omitted_

2787 #endif  /* __i386 */
2788 #endif  /* __lint */


2791 #if defined(__lint)

2793 /*ARGSUSED */
2794 int
2795 lowbit(ulong_t i)
2796 { return (0); }

2798 #else   /* __lint */

2800 #if defined(__amd64)

2802         ENTRY(lowbit)
2803         movl    $-1, %eax
2804         bsfq    %rdi, %rdi
2805         cmovnz  %edi, %eax
2804         bsfq    %rdi, %rax
2806         incl    %eax
2807         ret
2808         SET_SIZE(lowbit)

2810 #elif defined(__i386)

2812         ENTRY(lowbit)
2812         movl    $-1, %eax
2813         bsfl    4(%esp), %eax
2814         jz      0f
2815 #endif /* ! codereview */
2816         incl    %eax
2817         ret
2818 0:
2819         xorl    %eax, %eax
2820         ret
2821 #endif /* ! codereview */
2822         SET_SIZE(lowbit)

2824 #endif  /* __i386 */
2825 #endif  /* __lint */

2827 #if defined(__lint)

2829 /*ARGSUSED*/
2830 int
2831 highbit(ulong_t i)
2832 { return (0); }

2834 /*ARGSUSED*/
2835 int
2836 highbit64(uint64_t i)
2837 { return (0); }

2839 #endif /* ! codereview */
2840 #else   /* __lint */

2842 #if defined(__amd64)
```



```
2844         ENTRY(highbit)
2845         ALTENTRY(highbit64)
2846 #endif /* ! codereview */
2847         movl    $-1, %eax
2848         bsrq    %rdi, %rdi
2849         cmovnz  %edi, %eax
2814         bsrq    %rdi, %rax
2850         incl    %eax
2851         ret
2852         SET_SIZE(highbit64)
2853 #endif /* ! codereview */
2854         SET_SIZE(highbit)

2856 #elif defined(__i386)

2858         ENTRY(highbit)
2817         movl    $-1, %eax
2859         bsrl    4(%esp), %eax
2860         jz      0f
2861 #endif /* ! codereview */
2862         incl    %eax
2863         ret
2864 0:
2865         xorl    %eax, %eax
2866         ret
2867 #endif /* ! codereview */
2868         SET_SIZE(highbit)

2819 #endif  /* __i386 */
2820 #endif  /* __lint */

2822 #if defined(__lint)

2824 /*ARGSUSED*/
2825 int
2826 highbit64(uint64_t i)
2827 { return (0); }

2829 #else   /* __lint */

2831 #if defined(__amd64)

2833         ENTRY(highbit64)
2834         movl    $-1, %eax
2835         bsrq    %rdi, %rax
2836         incl    %eax
2837         ret
2838         SET_SIZE(highbit64)

2840 #elif defined(__i386)

2870         ENTRY(highbit64)
2871         bsrl    8(%esp), %eax
2872         jz      highbit
2873         addl    $33, %eax
2844         jz      .lowbit
2845         addl    $32, %eax
2846         jmp     .done

2848 .lowbit:
2849         movl    $-1, %eax
2850         bsrl    4(%esp), %eax
2851 .done:
2852         incl    %eax
2874         ret
```

```
2875         SET_SIZE(highbit64)
```
_____***unchanged_portion_omitted_***

```
**********************************************************
    4246 Fri Nov 14 18:05:27 2014
new/usr/src/uts/intel/ia32/ml/ia32.il
5291 x86 {high,low}bit rely on undefined behavior
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/
/ Inline functions for i386 kernels.
/       Shared between all x86 platform variants.
/

/
/ return current thread pointer
/
/ NOTE: the "0x10" should be replaced by the computed value of the
/       offset of "cpu_thread" from the beginning of the struct cpu.
/       Including "assym.h" does not work, however, since that stuff
/       is PSM-specific and is only visible to the 'unix' build anyway.
/       Same with current cpu pointer, where "0xc" should be replaced
/       by the computed value of the offset of "cpu_self".
/       Ugh -- what a disaster.
/
        .inline threadp,0
        movl    %gs:0x10, %eax
        .end

/
/ return current cpu pointer
/
        .inline curcpup,0
        movl    %gs:0xc, %eax
        .end

/
/ return caller
/
        .inline caller,0
        movl    4(%ebp), %eax
        .end

/
```



```
/ convert ipl to spl.  This is the identity function for i86
/
        .inline ipltospl,0
        movl    (%esp), %eax
        .end

/
/ find the low order bit in a word
/
        .inline lowbit,4
        movl    $-1, %eax
        bsfl    (%esp), %eax
        incl    %eax
        .end

/
/ find the high order bit in a word
/
        .inline highbit,4
        movl    $-1, %eax
        bsrl    (%esp), %eax
        incl    %eax
        .end

/
/ Networking byte order functions (too bad, Intel has the wrong byte order)
/
        .inline htonll,4
        movl    (%esp), %edx
        movl    4(%esp), %eax
        bswap   %edx
        bswap   %eax
        .end

        .inline ntohll,4
        movl    (%esp), %edx
        movl    4(%esp), %eax
        bswap   %edx
        bswap   %eax
        .end

        .inline htonl,4
        movl    (%esp), %eax
        bswap   %eax
        .end

        .inline ntohl,4
        movl    (%esp), %eax
        bswap   %eax
        .end

        .inline htons,4
        movl    (%esp), %eax
        bswap   %eax
        shrl    $16, %eax
        .end

        .inline ntohs,4
        movl    (%esp), %eax
        bswap   %eax
        shrl    $16, %eax
        .end

/*
 * multiply two long numbers and yield a u_longlong_t result
 * Provided to manipulate hrtime_t values.
```

```
 110  */
 111         .inline mul32, 8
 112         movl    4(%esp), %eax
 113         movl    (%esp), %ecx
 114         mull    %ecx
 115         .end

 117 /*
 118  * Unlock hres_lock and increment the count value. (See clock.h)
 119  */
 120         .inline unlock_hres_lock, 0
 121         lock
 122         incl    hres_lock
 123         .end

 125         .inline atomic_orb,8
 126         movl    (%esp), %eax
 127         movl    4(%esp), %edx
 128         lock
 129         orb     %dl,(%eax)
 130         .end

 132         .inline atomic_andb,8
 133         movl    (%esp), %eax
 134         movl    4(%esp), %edx
 135         lock
 136         andb    %dl,(%eax)
 137         .end

 139 /*
 140  * atomic inc/dec operations.
 141  *      void atomic_inc16(uint16_t *addr) { ++*addr; }
 142  *      void atomic_dec16(uint16_t *addr) { --*addr; }
 143  */
 144         .inline atomic_inc16,4
 145         movl    (%esp), %eax
 146         lock
 147         incw    (%eax)
 148         .end

 150         .inline atomic_dec16,4
 151         movl    (%esp), %eax
 152         lock
 153         decw    (%eax)
 154         .end

 156 /*
 157  * Call the pause instruction.  To the Pentium 4 Xeon processor, it acts as
 158  * a hint that the code sequence is a busy spin-wait loop.  Without a pause
 159  * instruction in these loops, the P4 Xeon processor may suffer a severe
 160  * penalty when exiting the loop because the processor detects a possible
 161  * memory violation.  Inserting the pause instruction significantly reduces
 162  * the likelihood of a memory order violation, improving performance.
 163  * The pause instruction is a NOP on all other IA-32 processors.
 164  */
 165         .inline ht_pause, 0
 166         rep                     / our compiler doesn't support "pause" yet,
 167         nop                     / so we're using "F3 90" opcode directly
 168         .end

 170 /*
 171  * prefetch 64 bytes
 172  *
 173  * prefetch is an SSE extension which is not supported on older 32-bit processor
 174  * so define this as a no-op for now
 175  */
```



```
 177         .inline prefetch_read_many,4
 178 /       movl            (%esp), %eax
 179 /       prefetcht0      (%eax)
 180 /       prefetcht0      32(%eax)
 181         .end

 183         .inline prefetch_read_once,4
 184 /       movl            (%esp), %eax
 185 /       prefetchnta     (%eax)
 186 /       prefetchnta     32(%eax)
 187         .end

 189         .inline prefetch_write_many,4
 190 /       movl            (%esp), %eax
 191 /       prefetcht0      (%eax)
 192 /       prefetcht0      32(%eax)
 193         .end

 195         .inline prefetch_write_once,4
 196 /       movl            (%esp), %eax
 197 /       prefetcht0      (%eax)
 198 /       prefetcht0      32(%eax)
 199         .end
```