

```

*****
13181 Tue Nov 10 18:29:54 2015
new/usr/src/uts/intel/asm/atomic.h
6137 implement static inlines for atomic_{add,inc,dec,or,and}_*_nv on intel
*****
_____unchanged_portion_omitted_____

250 #define __ATOMIC_OPXX(fxn, type1, type2, suf, reg) \
251 extern __GNU_INLINE type1 \
252 fxn(volatile type1 *target, type2 delta) \
253 { \
254     type1 orig; \
255     __asm__ volatile ( \
256         "lock; xadd" suf " %1, %0" \
257         : "+m" (*target), "=r" reg (orig) \
258         : "1" (delta) \
259         : "cc"); \
260     return (orig + delta); \
261 }

263 __ATOMIC_OPXX(atomic_add_8_nv,      uint8_t,      int8_t,      SUF_8,      "q")
264 __ATOMIC_OPXX(atomic_add_16_nv,     uint16_t,     int16_t,     SUF_16,     "r")
265 __ATOMIC_OPXX(atomic_add_32_nv,     uint32_t,     int32_t,     SUF_32,     "r")
266 __ATOMIC_OP64(atomic_add_64_nv,     uint64_t,     int64_t,     SUF_64,     "r")
267 __ATOMIC_OPXX(atomic_add_char_nv,   unsigned char, signed char, SUF_8,      "q")
268 __ATOMIC_OPXX(atomic_add_short_nv,  ushort_t,    short,      SUF_16,     "r")
269 __ATOMIC_OPXX(atomic_add_int_nv,    uint_t,      int,        SUF_32,     "r")
270 __ATOMIC_OPXX(atomic_add_long_nv,   ulong_t,     long,       SUF_LONG,   "r")

272 #undef __ATOMIC_OPXX

274 /*
275  * We don't use the above macro here because atomic_add_ptr_nv has an
276  * inconsistent type. The first argument should really be a 'volatile void
277  * ***'.
278  */
279 extern __GNU_INLINE void *
280 atomic_add_ptr_nv(volatile void *target, ssize_t delta)
281 {
282     return ((void *)atomic_add_long_nv((volatile ulong_t *)target, delta));
283 }

285 #define __ATOMIC_OPXX(fxn, implfxn, type, c) \
286 extern __GNU_INLINE type \
287 fxn(volatile type *target) \
288 { \
289     return (implfxn(target, c)); \
290 }

292 __ATOMIC_OPXX(atomic_inc_8_nv,      atomic_add_8_nv,      uint8_t,  1)
293 __ATOMIC_OPXX(atomic_inc_16_nv,     atomic_add_16_nv,     uint16_t, 1)
294 __ATOMIC_OPXX(atomic_inc_32_nv,     atomic_add_32_nv,     uint32_t, 1)
295 __ATOMIC_OP64(atomic_inc_64_nv,     atomic_add_64_nv,     uint64_t, 1)
296 __ATOMIC_OPXX(atomic_inc_uchar_nv,  atomic_add_char_nv,  uchar_t,  1)
297 __ATOMIC_OPXX(atomic_inc_ushort_nv, atomic_add_short_nv, ushort_t,  1)
298 __ATOMIC_OPXX(atomic_inc_uint_nv,   atomic_add_int_nv,   uint_t,   1)
299 __ATOMIC_OPXX(atomic_inc_ulong_nv,  atomic_add_long_nv,  ulong_t,  1)

301 __ATOMIC_OPXX(atomic_dec_8_nv,      atomic_add_8_nv,      uint8_t, -1)
302 __ATOMIC_OPXX(atomic_dec_16_nv,     atomic_add_16_nv,     uint16_t, -1)
303 __ATOMIC_OPXX(atomic_dec_32_nv,     atomic_add_32_nv,     uint32_t, -1)
304 __ATOMIC_OP64(atomic_dec_64_nv,     atomic_add_64_nv,     uint64_t, -1)
305 __ATOMIC_OPXX(atomic_dec_uchar_nv,  atomic_add_char_nv,  uchar_t,  -1)
306 __ATOMIC_OPXX(atomic_dec_ushort_nv, atomic_add_short_nv, ushort_t,  -1)
307 __ATOMIC_OPXX(atomic_dec_uint_nv,   atomic_add_int_nv,   uint_t,   -1)
308 __ATOMIC_OPXX(atomic_dec_ulong_nv,  atomic_add_long_nv,  ulong_t,  -1)

```

```

310 #undef __ATOMIC_OPXX

312 #define __ATOMIC_OPXX(fxn, cas, op, type) \
313 extern __GNU_INLINE type \
314 fxn(volatile type *target, type delta) \
315 { \
316     type old; \
317     do { \
318         old = *target; \
319     } while (cas(target, old, old op delta) != old); \
320     return (old op delta); \
321 }

323 __ATOMIC_OPXX(atomic_or_8_nv,      atomic_cas_8,      , uint8_t)
324 __ATOMIC_OPXX(atomic_or_16_nv,     atomic_cas_16,     , uint16_t)
325 __ATOMIC_OPXX(atomic_or_32_nv,     atomic_cas_32,     , uint32_t)
326 __ATOMIC_OP64(atomic_or_64_nv,     atomic_cas_64,     , uint64_t)
327 __ATOMIC_OPXX(atomic_or_uchar_nv,  atomic_cas_uchar, , uchar_t)
328 __ATOMIC_OPXX(atomic_or_ushort_nv, atomic_cas_ushort, , ushort_t)
329 __ATOMIC_OPXX(atomic_or_uint_nv,   atomic_cas_uint,  , uint_t)
330 __ATOMIC_OPXX(atomic_or_ulong_nv,  atomic_cas_ulong, , ulong_t)

332 __ATOMIC_OPXX(atomic_and_8_nv,     atomic_cas_8,     &, uint8_t)
333 __ATOMIC_OPXX(atomic_and_16_nv,    atomic_cas_16,    &, uint16_t)
334 __ATOMIC_OPXX(atomic_and_32_nv,    atomic_cas_32,    &, uint32_t)
335 __ATOMIC_OP64(atomic_and_64_nv,    atomic_cas_64,    &, uint64_t)
336 __ATOMIC_OPXX(atomic_and_uchar_nv, atomic_cas_uchar, &, uchar_t)
337 __ATOMIC_OPXX(atomic_and_ushort_nv, atomic_cas_ushort, &, ushort_t)
338 __ATOMIC_OPXX(atomic_and_uint_nv,  atomic_cas_uint,  &, uint_t)
339 __ATOMIC_OPXX(atomic_and_ulong_nv, atomic_cas_ulong, &, ulong_t)

341 #undef __ATOMIC_OPXX

343 #endif /* ! codereview */
344 #else
345 #error "port me"
346 #endif

348 #undef SUF_8
349 #undef SUF_16
350 #undef SUF_32
351 #undef SUF_64
352 #undef SUF_LONG
353 #undef SUF_PTR

355 #undef __ATOMIC_OP64

357 /* END CSTYLED */

359 #endif /* !_lint && __GNUC__ */

361 #ifdef __cplusplus
362 }
363 #endif

365 #endif /* _ASM_ATOMIC_H */

```