```
**********************************************************
   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_

#define __ATOMIC_OPXX(fxn, type1, type2, suf, reg)                      \
extern __GNU_INLINE type1                                               \
fxn(volatile type1 *target, type2 delta)                                \
{                                                                       \
        type1 orig;                                                     \
        __asm__ __volatile__(                                           \
            "lock; xadd" suf " %1, %0"                                  \
            : "+m" (*target), "=" reg (orig)                            \
            : "1" (delta)                                               \
            : "cc");                                                    \
        return (orig + delta);                                          \
}

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

#undef __ATOMIC_OPXX

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

#define __ATOMIC_OPXX(fxn, implfxn, type, c)                            \
extern __GNU_INLINE type                                                \
fxn(volatile type *target)                                              \
{                                                                       \
        return (implfxn(target, c));                                    \
}

__ATOMIC_OPXX(atomic_inc_8_nv,      atomic_add_8_nv,     uint8_t,  1)
__ATOMIC_OPXX(atomic_inc_16_nv,     atomic_add_16_nv,    uint16_t, 1)
__ATOMIC_OPXX(atomic_inc_32_nv,     atomic_add_32_nv,    uint32_t, 1)
__ATOMIC_OP64(atomic_inc_64_nv,     atomic_add_64_nv,    uint64_t, 1)
__ATOMIC_OPXX(atomic_inc_uchar_nv,  atomic_add_char_nv,  uchar_t,  1)
__ATOMIC_OPXX(atomic_inc_ushort_nv, atomic_add_short_nv, ushort_t, 1)
__ATOMIC_OPXX(atomic_inc_uint_nv,   atomic_add_int_nv,   uint_t,   1)
__ATOMIC_OPXX(atomic_inc_ulong_nv,  atomic_add_long_nv,  ulong_t,  1)

__ATOMIC_OPXX(atomic_dec_8_nv,      atomic_add_8_nv,     uint8_t,  -1)
__ATOMIC_OPXX(atomic_dec_16_nv,     atomic_add_16_nv,    uint16_t, -1)
__ATOMIC_OPXX(atomic_dec_32_nv,     atomic_add_32_nv,    uint32_t, -1)
__ATOMIC_OP64(atomic_dec_64_nv,     atomic_add_64_nv,    uint64_t, -1)
__ATOMIC_OPXX(atomic_dec_uchar_nv,  atomic_add_char_nv,  uchar_t,  -1)
__ATOMIC_OPXX(atomic_dec_ushort_nv, atomic_add_short_nv, ushort_t, -1)
__ATOMIC_OPXX(atomic_dec_uint_nv,   atomic_add_int_nv,   uint_t,   -1)
__ATOMIC_OPXX(atomic_dec_ulong_nv,  atomic_add_long_nv,  ulong_t,  -1)
```



```
#undef __ATOMIC_OPXX

#define __ATOMIC_OPXX(fxn, cas, op, type)                               \
extern __GNU_INLINE type                                                \
fxn(volatile type *target, type delta)                                  \
{                                                                       \
        type old;                                                       \
        do {                                                            \
                old = *target;                                          \
        } while (cas(target, old, old op delta) != old);                \
        return (old op delta);                                          \
}

__ATOMIC_OPXX(atomic_or_8_nv,      atomic_cas_8,      |, uint8_t)
__ATOMIC_OPXX(atomic_or_16_nv,     atomic_cas_16,     |, uint16_t)
__ATOMIC_OPXX(atomic_or_32_nv,     atomic_cas_32,     |, uint32_t)
__ATOMIC_OP64(atomic_or_64_nv,     atomic_cas_64,     |, uint64_t)
__ATOMIC_OPXX(atomic_or_uchar_nv,  atomic_cas_uchar,  |, uchar_t)
__ATOMIC_OPXX(atomic_or_ushort_nv, atomic_cas_ushort, |, ushort_t)
__ATOMIC_OPXX(atomic_or_uint_nv,   atomic_cas_uint,   |, uint_t)
__ATOMIC_OPXX(atomic_or_ulong_nv,  atomic_cas_ulong,  |, ulong_t)

__ATOMIC_OPXX(atomic_and_8_nv,      atomic_cas_8,      &, uint8_t)
__ATOMIC_OPXX(atomic_and_16_nv,     atomic_cas_16,     &, uint16_t)
__ATOMIC_OPXX(atomic_and_32_nv,     atomic_cas_32,     &, uint32_t)
__ATOMIC_OP64(atomic_and_64_nv,     atomic_cas_64,     &, uint64_t)
__ATOMIC_OPXX(atomic_and_uchar_nv,  atomic_cas_uchar,  &, uchar_t)
__ATOMIC_OPXX(atomic_and_ushort_nv, atomic_cas_ushort, &, ushort_t)
__ATOMIC_OPXX(atomic_and_uint_nv,   atomic_cas_uint,   &, uint_t)
__ATOMIC_OPXX(atomic_and_ulong_nv,  atomic_cas_ulong,  &, ulong_t)

#undef __ATOMIC_OPXX

#endif /* ! codereview */
#else
#error  "port me"
#endif

#undef SUF_8
#undef SUF_16
#undef SUF_32
#undef SUF_64
#undef SUF_LONG
#undef SUF_PTR

#undef __ATOMIC_OP64

/* END CSTYLED */

#endif /* !__lint && __GNUC__ */

#ifdef __cplusplus
}
#endif

#endif  /* _ASM_ATOMIC_H */
```