```
**********************************************************
    9015 Wed Aug 13 09:57:57 2014
new/usr/src/uts/intel/asm/atomic.h
patch atomic-8-bit
**********************************************************
______unchanged_portion_omitted_

  88 __ATOMIC_OPXX(atomic_inc_8,      uint8_t,  "inc" SUF_8)
  89 __ATOMIC_OPXX(atomic_inc_16,     uint16_t, "inc" SUF_16)
  90 __ATOMIC_OPXX(atomic_inc_32,     uint32_t, "inc" SUF_32)
  91 __ATOMIC_OP64(atomic_inc_64,     uint64_t, "inc" SUF_64)
  92 __ATOMIC_OPXX(atomic_inc_uchar,  uchar_t,  "inc" SUF_8)
  93 __ATOMIC_OPXX(atomic_inc_ushort, ushort_t, "inc" SUF_16)
  94 __ATOMIC_OPXX(atomic_inc_uint,   uint_t,   "inc" SUF_32)
  95 __ATOMIC_OPXX(atomic_inc_ulong,  ulong_t,  "inc" SUF_LONG)

  97 __ATOMIC_OPXX(atomic_dec_8,      uint8_t,  "dec" SUF_8)
  98 __ATOMIC_OPXX(atomic_dec_16,     uint16_t, "dec" SUF_16)
  99 __ATOMIC_OPXX(atomic_dec_32,     uint32_t, "dec" SUF_32)
 100 __ATOMIC_OP64(atomic_dec_64,     uint64_t, "dec" SUF_64)
 101 __ATOMIC_OPXX(atomic_dec_uchar,  uchar_t,  "dec" SUF_8)
 102 __ATOMIC_OPXX(atomic_dec_ushort, ushort_t, "dec" SUF_16)
 103 __ATOMIC_OPXX(atomic_dec_uint,   uint_t,   "dec" SUF_32)
 104 __ATOMIC_OPXX(atomic_dec_ulong,  ulong_t,  "dec" SUF_LONG)

 106 #undef __ATOMIC_OPXX

 108 #define __ATOMIC_OPXX(fxn, type1, type2, op, reg)                       \
 108 #define __ATOMIC_OPXX(fxn, type1, type2, op)                            \
 109 extern __GNU_INLINE void                                               \
 110 fxn(volatile type1 *target, type2 delta)                               \
 111 {                                                                      \
 112         __asm__ __volatile__(                                          \
 113             "lock; " op " %1,%0"                                       \
 114             : "+m" (*target)                                           \
 115             : "i" reg (delta));                                        \
 115             : "ir" (delta));                                           \
 116 }

 118 __ATOMIC_OPXX(atomic_add_8,      uint8_t,  int8_t,      "add" SUF_8,    "q")
 119 __ATOMIC_OPXX(atomic_add_16,     uint16_t, int16_t,     "add" SUF_16,   "r")
 120 __ATOMIC_OPXX(atomic_add_32,     uint32_t, int32_t,     "add" SUF_32,   "r")
 121 __ATOMIC_OP64(atomic_add_64,     uint64_t, int64_t,     "add" SUF_64,   "r")
 122 __ATOMIC_OPXX(atomic_add_char,   uchar_t,  signed char, "add" SUF_8,    "q")
 123 __ATOMIC_OPXX(atomic_add_short,  ushort_t, short,       "add" SUF_16,   "r")
 124 __ATOMIC_OPXX(atomic_add_int,    uint_t,   int,         "add" SUF_32,   "r")
 125 __ATOMIC_OPXX(atomic_add_long,   ulong_t,  long,        "add" SUF_LONG, "r")
 118 __ATOMIC_OPXX(atomic_add_8,      uint8_t,  int8_t,      "add" SUF_8)
 119 __ATOMIC_OPXX(atomic_add_16,     uint16_t, int16_t,     "add" SUF_16)
 120 __ATOMIC_OPXX(atomic_add_32,     uint32_t, int32_t,     "add" SUF_32)
 121 __ATOMIC_OP64(atomic_add_64,     uint64_t, int64_t,     "add" SUF_64)
 122 __ATOMIC_OPXX(atomic_add_char,   uchar_t,  signed char, "add" SUF_8)
 123 __ATOMIC_OPXX(atomic_add_short,  ushort_t, short,       "add" SUF_16)
 124 __ATOMIC_OPXX(atomic_add_int,    uint_t,   int,         "add" SUF_32)
 125 __ATOMIC_OPXX(atomic_add_long,   ulong_t,  long,        "add" SUF_LONG)

 127 /*
 128  * We don't use the above macro here because atomic_add_ptr has an
 129  * inconsistent type.  The first argument should really be a 'volatile void
 130  * **'.
 131  */
 132 extern __GNU_INLINE void
 133 atomic_add_ptr(volatile void *target, ssize_t delta)
 134 {
 135         volatile void **tmp = (volatile void **)target;
```



```
 137         __asm__ __volatile__(
 138             "lock; add" SUF_PTR " %1,%0"
 139             : "+m" (*tmp)
 140             : "ir" (delta));
 141 }

 143 __ATOMIC_OPXX(atomic_or_8,       uint8_t,  uint8_t,  "or" SUF_8,    "q")
 144 __ATOMIC_OPXX(atomic_or_16,      uint16_t, uint16_t, "or" SUF_16,   "r")
 145 __ATOMIC_OPXX(atomic_or_32,      uint32_t, uint32_t, "or" SUF_32,   "r")
 146 __ATOMIC_OP64(atomic_or_64,      uint64_t, uint64_t, "or" SUF_64,   "r")
 147 __ATOMIC_OPXX(atomic_or_uchar,   uchar_t,  uchar_t,  "or" SUF_8,    "q")
 148 __ATOMIC_OPXX(atomic_or_ushort,  ushort_t, ushort_t, "or" SUF_16,   "r")
 149 __ATOMIC_OPXX(atomic_or_uint,    uint_t,   uint_t,   "or" SUF_32,   "r")
 150 __ATOMIC_OPXX(atomic_or_ulong,   ulong_t,  ulong_t,  "or" SUF_LONG, "r")
 143 __ATOMIC_OPXX(atomic_or_8,       uint8_t,  uint8_t,  "or" SUF_8)
 144 __ATOMIC_OPXX(atomic_or_16,      uint16_t, uint16_t, "or" SUF_16)
 145 __ATOMIC_OPXX(atomic_or_32,      uint32_t, uint32_t, "or" SUF_32)
 146 __ATOMIC_OP64(atomic_or_64,      uint64_t, uint64_t, "or" SUF_64)
 147 __ATOMIC_OPXX(atomic_or_uchar,   uchar_t,  uchar_t,  "or" SUF_8)
 148 __ATOMIC_OPXX(atomic_or_ushort,  ushort_t, ushort_t, "or" SUF_16)
 149 __ATOMIC_OPXX(atomic_or_uint,    uint_t,   uint_t,   "or" SUF_32)
 150 __ATOMIC_OPXX(atomic_or_ulong,   ulong_t,  ulong_t,  "or" SUF_LONG)

 152 __ATOMIC_OPXX(atomic_and_8,      uint8_t,  uint8_t,  "and" SUF_8,    "q")
 153 __ATOMIC_OPXX(atomic_and_16,     uint16_t, uint16_t, "and" SUF_16,   "r")
 154 __ATOMIC_OPXX(atomic_and_32,     uint32_t, uint32_t, "and" SUF_32,   "r")
 155 __ATOMIC_OP64(atomic_and_64,     uint64_t, uint64_t, "and" SUF_64,   "r")
 156 __ATOMIC_OPXX(atomic_and_uchar,  uchar_t,  uchar_t,  "and" SUF_8,    "q")
 157 __ATOMIC_OPXX(atomic_and_ushort, ushort_t, ushort_t, "and" SUF_16,   "r")
 158 __ATOMIC_OPXX(atomic_and_uint,   uint_t,   uint_t,   "and" SUF_32,   "r")
 159 __ATOMIC_OPXX(atomic_and_ulong,  ulong_t,  ulong_t,  "and" SUF_LONG, "r")
 152 __ATOMIC_OPXX(atomic_and_8,      uint8_t,  uint8_t,  "and" SUF_8)
 153 __ATOMIC_OPXX(atomic_and_16,     uint16_t, uint16_t, "and" SUF_16)
 154 __ATOMIC_OPXX(atomic_and_32,     uint32_t, uint32_t, "and" SUF_32)
 155 __ATOMIC_OP64(atomic_and_64,     uint64_t, uint64_t, "and" SUF_64)
 156 __ATOMIC_OPXX(atomic_and_uchar,  uchar_t,  uchar_t,  "and" SUF_8)
 157 __ATOMIC_OPXX(atomic_and_ushort, ushort_t, ushort_t, "and" SUF_16)
 158 __ATOMIC_OPXX(atomic_and_uint,   uint_t,   uint_t,   "and" SUF_32)
 159 __ATOMIC_OPXX(atomic_and_ulong,  ulong_t,  ulong_t,  "and" SUF_LONG)

 161 #undef __ATOMIC_OPXX

 163 #define __ATOMIC_OPXX(fxn, type, op, reg)                               \
 164 extern __GNU_INLINE type                                               \
 165 fxn(volatile type *target, type cmp, type new)                         \
 166 {                                                                      \
 167         type ret;                                                      \
 168         __asm__ __volatile__(                                          \
 169             "lock; " op " %2,%0"                                       \
 170             : "+m" (*target), "=a" (ret)                               \
 171             : reg (new), "1" (cmp)                                     \
 172             : "cc");                                                   \
 173         return (ret);                                                  \
 174 }
______unchanged_portion_omitted_
```