1. Write an asm statement to reverse the order of the bytes within a 32-bit C variable named "x32".

```
asm   (
      "REV  %[reg1],%[reg2]"
      : [reg1] "=r" (x32)         // Output Operand
      : [reg2] "r"  (x32)         // Input Operand
      ) ;
```

3. Write an asm statement that returns a count of the number of leading zeroes in the 32-bit C variable named "x32".

```
asm   (
      "CLZ  %[reg1],%[reg2]"
      : [reg1] "=r" (count)       // Output Operand
      : [reg2] "r"  (x32)         // Input Operand
      ) ;
```

4. Write an inline stub function in C to perform a Bit Field Clear with an asm statement that uses the BFC instruction. Require the starting bit position and field width to be specified as integers so that the function call may be replaced by a single BFC instruction. The function prototype is:

```
static inline uint32_t BFC(uint32_t word, int lsb, int len) ;
```

```
static inline uint32_t BFC(uint32_t word, int lsb, int len)
   {
   asm   (
         "BFC %[reg1],%[const1],%[const2]"
         : [reg1] "+r" (word)     // Output Operand
         : [const1] "i" (lsb),    // Input Operand
           [const2] "i" (len)     // Input Operand
         ) ;

   return word ;
   }
```

5. Write an inline stub function in C to perform a Bit Field Insert with an asm statement that uses the BFI instruction. Require the starting bit position and field width to be specified as integers so that the function call may be replaced by a single BFI instruction. The function prototype is:

```
static inline uint32_t BFI(uint32_t dst, uint32_t src, int lsb, int len) ;
```

```
static inline uint32_t BFI(uint32_t dst, uint32_t src, int lsb, int len)
  {
  asm   (
        "BFI %[reg1],%[reg2],%[const1],%[const2]"
        : [reg1] "+r" (dst)      // Output Operand
        : [reg2] "r" (src),      // Input Operand
          [const1] "i" (lsb),    // Input Operand
          [const2] "i" (len)     // Input Operand
        ) ;

  return dst ;
  }
```

6. Write an inline stub function in C to perform a Signed Bit Field Extract with an asm statement that uses the SBFX instruction. Require the starting bit position and field width to be specified as integers so that the function call may be replaced by a single SBFX instruction. The function prototype is:

```
static inline int32_t SBFX(uint32_t word, int lsb, int len) ;
```

```
static inline int32_t SBFX(uint32_t word, int lsb, int len)
  {
  int32_t result ;

  asm   (
        "SBFX %[reg1],%[reg2],%[const1],%[const2]"
        : [reg1] "=r" (result)   // Output Operand
        : [reg2] "r" (word),     // Input Operand
          [const1] "i" (lsb),    // Input Operand
          [const2] "i" (len)     // Input Operand
        ) ;

  return result ;
  }
```

7. Write an inline stub function in C to perform an Unsigned Bit Field Extract with an asm statement that uses the UBFX instruction. Require the starting bit position and field width to be specified as integers so that the function call may be replaced by a single UBFX instruction. The function prototype is:

```
static inline uint32_t UBFX(uint32_t word, int lsb, int len) ;
```

```
static inline uint32_t UBFX(uint32_t word, int lsb, int len)
   {
   uint32_t result ;

   asm   (
         "UBFX %[reg1],%[reg2],%[const1],%[const2]"
         : [reg1] "=r" (result)   // Output Operand
         : [reg2] "r" (word),     // Input Operand
           [const1] "i" (lsb),    // Input Operand
           [const2] "i" (len)     // Input Operand
         ) ;

   return result ;
   }
```