```
# BitCounter returns the number of bits that are equal 1 in an N-bit long bit vector.
#
# Assumes a big endian machine.
# For example, a 40-bit vector with bits 0, 9, 18, 27, 36 equal 1 is stored in memory:
#
# 1000 0000  0100 0000  0010 0000  0001 0000  0000 1000  xxxx xxxx  xxxx xxxx xxxx xxxx
#
# Input Parameters:
#   $a0: Memory address of the first position of the vector.
#   $a1: unsigned integer that contains the number of bits in the bit vector (could be 0).
#
# Return Value:
#   $v0: Number of bits that are one in the vector
#
# Algorithm:
#   Pointer = MemoryAddress
#   OneCount = 0
#   BitCount = NumberOfBits
#   while(BitCount > 0)
#       word = *Pointer;
#       for( mask = 0x80000000 ; mask != 0 && BitCount > 0 ; mask = mask >> 1){
#           temp = word & mask
#           if(temp != 0)
#             OneCount = OneCount + 1;
#           BitCount = BitCount -1;
#         }
#       Pointer++;
#   return OneCount
#
# Register Usage:
#   $a0: Pointer
#   $a1: BitCount
#   $t1: word
#   $t2: mask
#   $t5: temp
#   $v0: OneCount
```

Figure 1: File Header for `BitCounter`

Question 3 (30 points): A common requirement in the implementation of static analysis for a compiler is the manipulation of bit vectors with arbitrary length. In this question you will write the assembly MIPS code for a subroutine, `BitCounter`, that receives two parameters:

`$a0`: contains the memory address of the first byte that contains the bit vector

`$a1`: contains an unsigned integer that specifies the length, expressed in bits, of the bit vector

`BitCounter` returns in `$v0` the number of bits in the bit vector that are equal 1.

In the interest of time, you are provided with the header for your assembly file, containing an algorithm specification and register usage, is provided in Figure 2.