```
    1: #EJERCICIO 5.1
    2: #
    3: # Se han seguido los pasos tal y como se indican en el guiÃ³n.
    4: # 'lista' se ha dividido en cuatro lineas para una lectura mÃ¡s cÃ³moda del cÃ³digo
    5: # 'resultado' se ha cambiado por tipo de dato 'quad' para permitir imprimir datos de
 64 bits
    6: # 'formato' ha sido modificado para que la impresion mediante 'printf' fuera posible
 y mostrase adecuadamente los datos (unsigned long y hexadecimal long)
    7: # '_start' fue cambiado por 'main' para poder compilar mediante gcc y las funciones
fueron sustituidas por su cÃ³digo directamente
    8: # A muchas instrucciones se les ha aÃ±adido un subfijo para ir acostumbrandonos a su
 uso y significado
    9:
   10:
   11: .section .data
   12: lista:          .int 0x10000000,0x10000000,0x10000000,0x10000000
   13:                       .int 0x10000000,0x10000000,0x10000000,0x10000000
   14:                       .int 0x10000000,0x10000000,0x10000000,0x10000000
   15:                       .int 0x10000000,0x10000000,0x10000000,0x10000000
   16: longlista:      .int   (.-lista)/4
   17: resultado:      .quad   0
   18: formato:        .asciz  "suma = %lu = 0x%lx hex\n"
   19:
   20: .section .text
   21: main: .global  main
   22:
   23: #trabajar
   24:         movq     $lista, %rbx
   25:         movl  longlista, %ecx
   26:         call suma               # == suma(&lista, longlista);
   27:         movl  %eax, resultado
   28:         movl %edx, resultado+4
   29:
   30:         # Como 'resultado' es de 64 bits, es almacenado en pila y la arquitectura ut
ilizada almacena los datos en 'little endian'
   31:         # su parte mÃ¡s significativa (%edx) tiene que ser guarda antes que la menos
 significativa (%eax)
   32:         # por eso almacenamos %edx en resultado+4 y %eax en resultado
   33:
   34: #imprim_C
   35:         movq   $formato, %rdi
   36:         movq   resultado,%rsi
   37:         movq   resultado,%rdx
   38:         movl          $0,%eax   # varargin sin xmm
   39:         call  printf            # == printf(formato, res, res);
   40:
   41:         # SegÃºn el manual de 'prinf' formato debe ser especificado en %rdi,
   42:         # el primer resultado a mostrar (unsigned long) en %rsi y el segundo (hexade
cimal long) en %rdx
   43:
   44: #acabar_C
   45:         mov  resultado, %edi
   46:         call _exit              # ==  exit(resultado)
   47:         ret
   48:
   49: suma:
   50:         movq     $0, %rsi       # iterador de la lista
   51:         movl     $0, %eax       # acumulador de la suma. Representa la parte menos s
ignificativa
   52:         movl     $0, %edx       # acumulador de la suma. Representa la parte mÃ¡s si
gnificativa
   53: bucle:
   54:         addl    (%rbx,%rsi,4), %eax
   55:         jnc             no_inc          # si no hay acarreo no incrementamos %edx y
seguimos con las iteraciones del bucle
   56:         inc     %edx
   57: no_inc:
   58:         inc     %rsi
```

```
59:         cmpq    %rsi,%rcx
60:         jne     bucle
61:
62:         ret
63:
```