```
    1: #EJERCICIO 5.5
    2: #
    3: # Se ha usado el uso de dos macros para poder cambiar la 'lista' de nÃºmeros que se
van a sumar segÃºn explica el guiÃ³n
    4: # A la hora de imprimir hemos usado los registros %rsi y %rdx para imprimir los resu
ltados en decimal de la media y resto respectivamente
    5: # y %rcx y %r8 para imprimirlos en hexadecimal
    6: # La media la hemos realizado con 'idiv'. Divide edx:eax entre el parametro pasado.
Almacena el cociente en eax y el resto en edx
    7: # La media la hemos realizado con 'idiv'. Divide rdx:rax entre el parametro pasado.
Almacena el cociente en rax y el resto en rdx
    8: # Por lo demÃ¡s el cÃ³digo es igual al del 5.4
    9:
   10: #COMANDO PARA LA EJECUCIÃ\223N:
   11: #for i in $(seq 1 20); do rm media; gcc -x assembler-with-cpp -D TEST=$i -no-pie med
ia.s -o media; printf "__TEST%02d__%35s\n" $i "" | tr " " "-" ; ./media; done
   12:
   13: .section .data
   14: #ifndef TEST
   15: #define TEST 20
   16: #endif
   17: .macro linea
   18: #if  TEST==1                    //1             8
   19:         .int 1,2,1,2
   20: #elif TEST==2                    //-1    -8
   21:         .int -1,-2,-1,-2
   22: #elif TEST==3                    //2147483647    0
   23:         .int 0x7fffffff, 0x7fffffff, 0x7fffffff, 0x7fffffff
   24: #elif TEST==4                    //-2147483648   0
   25:         .int 0x80000000, 0x80000000, 0x80000000, 0x80000000
   26: #elif TEST==5                    //-1    0
   27:         .int 0xffffffff, 0xffffffff, 0xffffffff, 0xffffffff
   28: #elif TEST==6                    //2000000000    0
   29:         .int 2000000000, 2000000000, 2000000000, 2000000000
   30: #elif TEST==7                    //desbordamiento--> -1294967296 0
   31:         .int 3000000000, 3000000000, 3000000000, 3000000000
   32: #elif TEST==8                    //-2000000000   0
   33:         .int -2000000000, -2000000000, -2000000000, -2000000000
   34: #elif TEST==9                    //desbordamiento--> 1294967296  0
   35:         .int -3000000000, -3000000000, -3000000000, -3000000000
   36: #elif TEST>=10 && TEST<=14
   37:         .int 1, 1, 1, 1
   38: #elif TEST>=15 && TEST<=19
   39:         .int -1, -1, -1, -1
   40: #else                   //
   41:         .error "Definir TEST ente 1..19"
   42: #endif                  //
   43:         .endm
   44:
   45: .macro linea0
   46: #if TEST>=1 && TEST<=9
   47:         linea
   48: #elif TEST==10                   //1     0
   49:         .int 0,2,1,1
   50: #elif TEST==11                   //1     1
   51:         .int 1,2,1,1
   52: #elif TEST==12                   //1     8
   53:         .int 8,2,1,1
   54: #elif TEST==13                   //1     15
   55:         .int 15,2,1,1
   56: #elif TEST==14                   //2     0
   57:         .int 16,2,1,1
   58: #elif TEST==15                   //-1    0
   59:         .int 0,-2,-1,-1
   60: #elif TEST==16                   //-1    -1
   61:         .int -1,-2,-1,-1
   62: #elif TEST==17                   //-1    -8
   63:         .int -8,-2,-1,-1
```

```
 64: #elif TEST==18                  //-1    -15
 65:         .int -15,-2,-1,-1
 66: #elif TEST==19                  //-2    0
 67:         .int -16,-2,-1,-1
 68: #else
 69:         .error "Definir TEST ente 1..19"
 70: #endif
 71:         .endm
 72: lista:          linea0
 73:         .irpc i,123
 74:                 linea
 75:         .endr
 76:
 77: longlista:      .int   (.-lista)/4
 78: resultado:      .quad   0
 79: media:          .quad   0
 80: resto:          .quad   0
 81: formato: .ascii "media \t = %11d \t resto \t = %11d\n"
 82:                  .asciz "media \t = 0x %08x \t resto \t = 0x %08x\n"
 83:
 84: formatoq: .ascii "media_x64 \t = %11d \t resto_x64 \t = %11d\n"
 85:                  .asciz "media_x64 \t = 0x %08x \t resto_x64 \t = 0x %08x\n"
 86:
 87: .section .text
 88: main: .global  main
 89:
 90: #trabajar
 91:         movq     $lista, %rbx
 92:         movl  longlista, %ecx
 93:         call suma               # == suma(&lista, longlista);
 94:         mov     %esi, %eax
 95:         mov     %edi, %edx
 96:         idivl %ecx
 97:         movl  %eax, media
 98:         movl %edx, resto
 99:
100: #imprim_C
101:         movq   $formato, %rdi
102:         movl   media,%esi
103:         movl   resto,%edx
104:         movl   media, %ecx
105:         movl   resto, %r8d
106:         movl          $0,%eax   # varargin sin xmm
107:         call  printf            # == printf(formato, res, res);
108:
109: #trabajar_q
110:         movq     $lista, %rbx
111:         movq  longlista, %rcx
112:         call sumaq              # == suma(&lista, longlista);
113:         mov     %rdi, %rax
114:         cqto
115:         idivq %rcx
116:         movq  %rax, media
117:         movq %rdx, resto
118:
119: #imprim_C_q
120:         movq   $formatoq, %rdi
121:         movq   media,%rsi
122:         movq   resto,%rdx
123:         movq   media, %rcx
124:         movq   resto, %r8
125:         movq          $0,%rax   # varargin sin xmm
126:         call  printf            # == printf(formato, res, res);
127:
128: #acabar_C
129:         mov  resultado, %rdi
130:         call _exit              # ==  exit(resultado)
131:         ret
```

```
  132:
  133: suma:
  134:          movq     $0, %r8         # iterador de la lista
  135:          movl     $0, %eax        # En un principio se usarÃ¡ para extender el signo a
 %edx. Representa la parte menos significativa
  136:          movl     $0, %esi        # Acumulador de la suma. Representa la parte menos s
ignificativa
  137:          movl     $0, %edi        # Acumulador de la suma. Representa la parte mÃ¡s si
gnificativa
  138: bucle:
  139:          movl    (%rbx,%r8,4), %eax
  140:          cltd
  141:          add     %eax, %esi
  142:          adc     %edx, %edi
  143:          inc     %r8
  144:          cmpq    %r8,%rcx
  145:          jne     bucle
  146:
  147:          ret
  148:
  149: sumaq:
  150:          movq     $0, %r8         # iterador de la lista
  151:          movq     $0, %rax        # En un principio se usarÃ¡ para extender el signo a
 %edx. Representa la parte menos significativa
  152:          movq     $0, %rdi        # Acumulador de la suma
  153: bucleq:
  154:          movl    (%rbx,%r8,4), %eax
  155:          cdqe
  156:          add     %rax, %rdi
  157:          inc     %r8
  158:          cmpq    %r8,%rcx
  159:          jne     bucleq
  160:
  161:          ret
  162:
```