PRÁCTICA 2. ESTRUCTURA DE COMPUTADORES
Noelia Escalera Mejías. Grupo A3

Ejercicio 1:
Lo primero que se nos pide es copiar el código de suma.s en un archivo diferente con el nombre de media.s. Luego, cambiar la lista por una en la que haya dieciséis unos y verificar que el resultado es 16.

```
.section .data
lista:    .int 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
longlista:  .int   (.-lista)/4
resultado:  .int   0
  formato:  .asciz  "suma = %u = 0x%x hex\n"
```

```
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ as media.s -o media.o
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ ld media.o -o media
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ ./media
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ echo $?
16
```

Luego, habrá que hacer lo mismo con el primer valor que repetido 16 veces produce acarreo: se trata de 1000000. Lo ejecutamos y vemos que el resultado es 0.

```
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ as media.s -o media.o
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ ld media.o -o media
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ ./media
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ echo $?
0
```

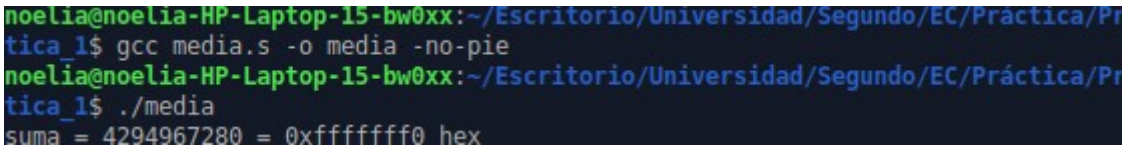
Si ponemos 16 veces el número 0x0fffffff:

```
.section .data
lista:    .int 0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff,
0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff
longlista:  .int   (.-lista)/4
resultado:  .int   0
  formato:  .asciz  "suma = %u = 0x%x hex\n"
```

```
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ as media.s -o media.o
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ ld media.o -o media
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ ./media
noelia@noelia-HP-Laptop-15-bw0xx:~/Escritorio/Universidad/Segundo/EC/Práctica/Pr
tica_1$ echo $?
240
```

El resultado es 240, no se produce acarreo.

No obstante, si usamos la función printf de libC, no obtendremos estos resultados:



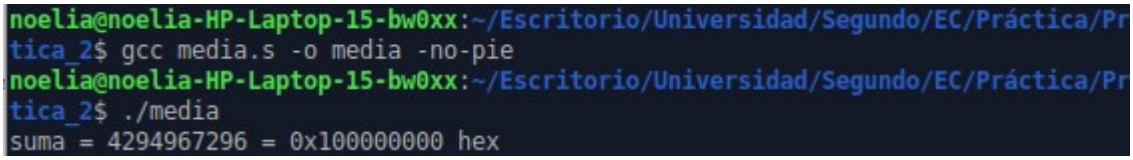
La primera modificación que se nos pide hacer al programa es usar otro registro de 32 bits para guardar el acarreo, teniendo así una suma sin signo de 32 bits sobre dos registros de 32 bits. La solución que ha quedado es la siguiente:

```
.section .data
lista: .int
0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,
0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF
longlista:    .int    (.-lista)/4
resultado:    .quad   0
  formato:    .asciz "suma = %lu = 0x%lx hex\n"

.section .text
 main: .global  main

       call trabajar # subrutina de usuario
       call imprim_C # printf()  de libC
       call acabar_C # exit()    de libC
       ret

trabajar:
       mov     $lista, %ebx
       mov  longlista, %ecx
       call suma           # == suma(&lista, longlista);
       mov  %edx, resultado+4 # Registro para el acarreo
       mov  %eax, resultado   # Registro para la parte que no lleva acarreo
       ret

suma:
       push     %rsi
       mov  $0, %eax
       mov  $0, %rsi
       mov  $0, %edx
bucle:
       add  (%rbx,%rdx,4), %eax
       jnc   siguesuma    # Si no hay acarreo, continua la suma
       inc   %edx         # Incrementa el acarreo
siguesuma:
       inc   %rsi
       cmp   %rsi,%rcx
       jne    bucle

       pop   %rsi
       ret
```

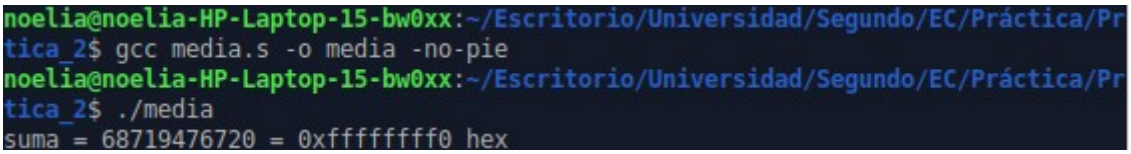
Noelia Escalera Mejías
Grupo A3

```
imprim_C:                   # requiere libC
       mov  $formato,%edi
       mov   resultado,%rsi
       mov   resultado,%rdx
       mov           $0,%eax# varargin sin xmm
       call  printf         # == printf(formato, res, res);
       ret

acabar_C:                   # requiere libC
       mov  resultado, %rdi
       call _exit           # ==  exit(resultado)
       ret
```

Si ejecutamos el programa con dieciséis 0x10000000, nos da el siguiente resultado:



Y si lo ejecutamos con dieciséis 0xFFFFFFFF, nos queda lo siguiente:



Luego obtenemos los resultados deseados. A partir de aquí, se compilará todo con

gcc media.s -o media -no-pie

Ejercicio 2:

La siguiente modificación del programa media.s consiste en suprimir el salto jnc, la etiqueta y el inc, para realizar directamente una suma con acarreo (instrucción adc). Luego el código modificado quedaría de la siguiente manera:

```
.section .data
#ifndef TEST
#define TEST 9
#endif
               .macro linea
#if TEST==1
       .int 1,1,1,1
#elif TEST==2
       .int 0x0fffffff,0x0fffffff,0x0fffffff,0x0fffffff
#elif TEST==3
       .int 0x10000000,0x10000000,0x10000000,0x10000000
#elif TEST==4
       .int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
#elif TEST==5
       .int -1,-1,-1,-1
#elif TEST==6
       .int 200000000,200000000,200000000,200000000
#elif TEST==7
       .int 300000000,300000000,300000000,300000000
#elif TEST==8
       .int 5000000000,5000000000,5000000000,5000000000
#else
       .error "Definir TEST entre 1..8"
#endif
       .endm
lista: .irpc i,1234
                                                          linea
       .endr
longlista:     .int   (.-lista)/4
resultado:     .quad   0
  formato:     .ascii  "resultado \t =               %18lu (uns)\n"
               .ascii           "\t\t = 0x%18lx (hex)\n"
               .asciz  "\t\t = 0x %08x %08x\n"
.section .text
 main: .global  main

       call trabajar  # subrutina de usuario
       call imprim_C  # printf()  de libC
       call acabar_C  # exit()    de libC
       ret

trabajar:
       mov     $lista, %ebx
       mov  longlista, %ecx
       call suma             # == suma(&lista, longlista);
       mov  %edx, resultado+4
       mov  %eax, resultado
       ret

suma:
       push     %rsi
       mov  $0, %eax
       mov  $0, %rsi
       mov  $0, %edx
```

Noelia Escalera Mejías
Grupo A3

```
bucle:
        add  (%rbx,%rdx,4), %eax
        adc   $0,%edx
        inc   %rsi
        cmp   %rsi,%rcx
        jne    bucle

        pop   %rsi
        ret

imprim_C:                    # requiere libC
        mov  $formato,%edi
        mov   resultado,%rsi
        mov   resultado,%rdx
        mov   resultado+4,%ecx
        mov   resultado,%r8d
        mov          $0,%eax  # varargin sin xmm
        call  printf          # == printf(formato, res, res);
        ret

acabar_C:                    # requiere libC
        mov  resultado, %rdi
        call _exit            # ==  exit(resultado)
        ret
```

Para automatizar el proceso mostrar los resultados se ha diseñado el siguiente script:

```
#!/bin/bash

for i in $(seq 1 8); do
rm media
gcc -x assembler-with-cpp -D TEST=$i -no-pie media.s -o media
echo -n "T#$i "; ./media
done
```

Tras su ejecución, obtenemos lo siguiente:

```
T#1 resultado    =                        16 (uns)
                 = 0x                10 (hex)
                 = 0x 00000000 00000010
T#2 resultado    =                4294967280 (uns)
                 = 0x          fffffff0 (hex)
                 = 0x 00000000 fffffff0
T#3 resultado    =                4294967296 (uns)
                 = 0x         100000000 (hex)
                 = 0x 00000001 00000000
T#4 resultado    =               68719476720 (uns)
                 = 0x         ffffffff0 (hex)
                 = 0x 0000000f fffffff0
T#5 resultado    =               68719476720 (uns)
                 = 0x         ffffffff0 (hex)
                 = 0x 0000000f fffffff0
T#6 resultado    =                3200000000 (uns)
                 = 0x          bebc2000 (hex)
                 = 0x 00000000 bebc2000
T#7 resultado    =                4800000000 (uns)
                 = 0x         11e1a3000 (hex)
                 = 0x 00000001 1e1a3000
media.s: Mensajes del ensamblador:
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
media.s:28: Aviso: valora 0x12a05f200 truncado a 0x2a05f200
T#8 resultado    =               11280523264 (uns)
                 = 0x         2a05f2000 (hex)
```

Tabla para mejor visualización de los resultados:

Número de test	Lista	Resultado en decimal	Resultado en hexadecimal	Comentario
TEST 1	1,1……...1	16	0x10	Todo unos, ejemplo fácil para ver que funciona
TEST 2	0x0FFFFFFF……0x0FFFFFFF	4294967280	0xFFFFFFF0	Máximo para el cuál no se produce acarreo
TEST 3	0x10000000……0x10000000	4294967296	100000000	Mínimo para el cuál se produce acarreo (la suma tiene un bit más)
TEST 4	0xFFFFFFFF……0xFFFFFFFF	68719476720	0xFFFFFFFF0	Número máximo que se puede escribir con 8 bits
TEST 5	-1,-1……..-1	68719476720	0xFFFFFFFF0	Incorrecto: Aún no podemos sumar números negativos
TEST 6	200000000……200000000	3200000000	0x773594000	
TEST 7	300000000……300000000	4800000000	0xB2D05E000	
TEST 8	5000000000……5000000000	11280523264	0x2A05F2000	Incorrecto: Cada sumando usa más de 8 bits, no cabe en el registro

Noelia Escalera Mejías
Grupo A3

Ejercicio 3:
Ahora sí que vamos a trabajar con números con signo, para ello, extendemos el signo de cada elemento de la lista de eax a edx con la orden cdq y posteriormente sumando la parte menos significativa y la más significativa respectivamente. También tendremos que modificar el script para hacer 19 tests. Nos queda el siguiente código en media.s:

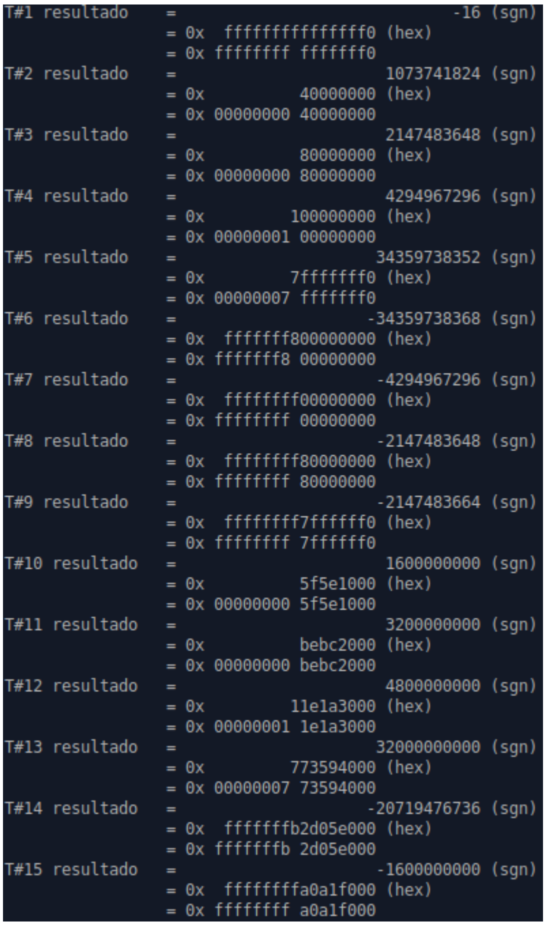
```
.section .data
#ifndef TEST
#define TEST 20
#endif
                    .macro linea
       #if TEST==1
                                    .int -1,-1,-1,-1
#elif TEST==2
                                    .int 0x04000000,0x04000000,0x04000000,0x04000000
#elif TEST==3
                                    .int 0x08000000,0x08000000,0x08000000,0x08000000
#elif TEST==4
                                    .int 0x10000000,0x10000000,0x10000000,0x10000000
#elif TEST==5
                                    .int 0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF
#elif TEST==6
                                    .int 0x80000000,0x80000000,0x80000000,0x80000000
#elif TEST==7
                                    .int 0xF0000000,0xF0000000,0xF0000000,0xF0000000
#elif TEST==8
                                    .int 0xF8000000,0xF8000000,0xF8000000,0xF8000000
#elif TEST==9
                                    .int 0xF7FFFFFF,0xF7FFFFFF,0xF7FFFFFF,0xF7FFFFFF
#elif TEST==10
                                    .int 100000000,100000000,100000000,100000000
#elif TEST==11
                                    .int 200000000,200000000,200000000,200000000
#elif TEST==12
                                    .int 300000000,300000000,300000000,300000000
#elif TEST==13
                                    .int 2000000000,2000000000,2000000000,2000000000
#elif TEST==14
                                    .int 3000000000,3000000000,3000000000,3000000000
#elif TEST==15
                                    .int -100000000,-100000000,-100000000,-100000000
#elif TEST==16
                                    .int -200000000,-200000000,-200000000,-200000000
#elif TEST==17
                                    .int -300000000,-300000000,-300000000,-300000000
#elif TEST==18
                                    .int -2000000000,-2000000000,-2000000000,-2000000000
#elif TEST==19
                                    .int -3000000000,-3000000000,-3000000000,-3000000000
#else
               .error "Definir TEST entre 1..19"
#endif
               .endm
lista:  .irpc i,1234
                                                    linea
                                .endr
longlista:      .int    (.-lista)/4
resultado:      .quad    0
  formato:      .ascii  "resultado \t =               %18ld (sgn)\n"
                                    .ascii           "\t\t = 0x%18lx (hex)\n"
```

Noelia Escalera Mejías
Grupo A3

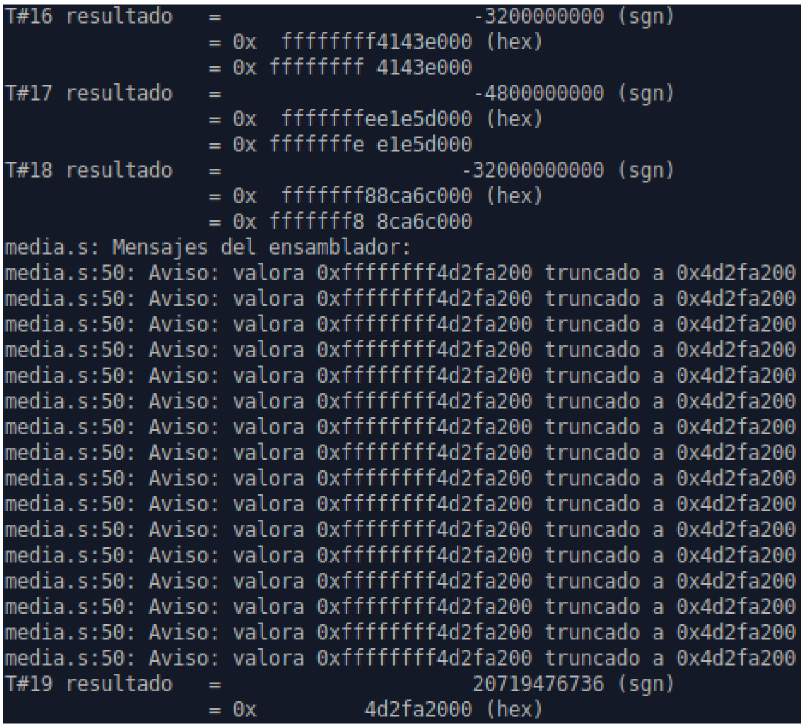
```
                                    .asciz  "\t\t = 0x %08x %08x\n"


.section .text
 main: .global  main

       call trabajar  # subrutina de usuario
       call imprim_C  # printf()  de libC
       call acabar_C  # exit()    de libC
       ret

trabajar:
       mov     $lista, %ebx
       mov  longlista, %ecx
       call suma              # == suma(&lista, longlista);
       mov  %edx, resultado+4
       mov  %eax, resultado
       ret

suma:
       push     %rsi
       mov  $0, %eax
       mov  $0, %rsi
       mov  $0, %edx
       mov  $0, %ebp
       mov  $0, %edi
bucle:
       mov  (%rbx,%rsi,4),%eax
       cdq
       add   %eax,%ebp
       adc   %edx,%edi
       mov   %ebp,%eax
       mov   %edi,%edx
       inc   %rsi
       cmp   %rsi,%rcx
       jne    bucle

       pop   %rsi
       ret

imprim_C:                     # requiere libC
       mov  $formato,%edi
       mov   resultado,%rsi
       mov   resultado,%rdx
       mov   resultado+4,%ecx
       mov   resultado,%r8d
       mov          $0,%eax   # varargin sin xmm
       call  printf           # == printf(formato, res, res);
       ret

acabar_C:                     # requiere libC
       mov  resultado, %rdi
       call _exit             # ==  exit(resultado)
       ret
```

Al ejecutar el script, nos da los siguientes resultados:

Noelia Escalera Mejías
Grupo A3



```
T#1 resultado    =                          -16 (sgn)
                 = 0x  fffffffffffffff0 (hex)
                 = 0x ffffffff fffffff0
T#2 resultado    =                   1073741824 (sgn)
                 = 0x          40000000 (hex)
                 = 0x 00000000 40000000
T#3 resultado    =                   2147483648 (sgn)
                 = 0x          80000000 (hex)
                 = 0x 00000000 80000000
T#4 resultado    =                   4294967296 (sgn)
                 = 0x         100000000 (hex)
                 = 0x 00000001 00000000
T#5 resultado    =                  34359738352 (sgn)
                 = 0x         7fffffff0 (hex)
                 = 0x 00000007 fffffff0
T#6 resultado    =                 -34359738368 (sgn)
                 = 0x  fffffff800000000 (hex)
                 = 0x fffffff8 00000000
T#7 resultado    =                  -4294967296 (sgn)
                 = 0x  ffffffff00000000 (hex)
                 = 0x ffffffff 00000000
T#8 resultado    =                  -2147483648 (sgn)
                 = 0x  ffffffff80000000 (hex)
                 = 0x ffffffff 80000000
T#9 resultado    =                  -2147483664 (sgn)
                 = 0x  ffffffff7ffffff0 (hex)
                 = 0x ffffffff 7ffffff0
T#10 resultado   =                   1600000000 (sgn)
                 = 0x          5f5e1000 (hex)
                 = 0x 00000000 5f5e1000
T#11 resultado   =                   3200000000 (sgn)
                 = 0x          bebc2000 (hex)
                 = 0x 00000000 bebc2000
T#12 resultado   =                   4800000000 (sgn)
                 = 0x         11e1a3000 (hex)
                 = 0x 00000001 1e1a3000
T#13 resultado   =                  32000000000 (sgn)
                 = 0x         773594000 (hex)
                 = 0x 00000007 73594000
T#14 resultado   =                 -20719476736 (sgn)
                 = 0x  fffffffb2d05e000 (hex)
                 = 0x fffffffb 2d05e000
T#15 resultado   =                  -1600000000 (sgn)
                 = 0x  ffffffffa0a1f000 (hex)
                 = 0x ffffffff a0a1f000
```



```
T#16 resultado   =                  -3200000000 (sgn)
                 = 0x  ffffffff4143e000 (hex)
                 = 0x ffffffff 4143e000
T#17 resultado   =                  -4800000000 (sgn)
                 = 0x  fffffffee1e5d000 (hex)
                 = 0x fffffffe e1e5d000
T#18 resultado   =                 -32000000000 (sgn)
                 = 0x  fffffff88ca6c000 (hex)
                 = 0x fffffff8 8ca6c000
media.s: Mensajes del ensamblador:
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:50: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
T#19 resultado   =                  20719476736 (sgn)
                 = 0x         4d2fa2000 (hex)
```

Tabla para mejor visualización de los resultados:

Número de test	**Lista**	**Resultado en decimal**	**Resultado en hexadecimal**	**Comentario**
TEST 1	-1………………..	-16	0xFFFFFFFFFFFFFFF0	Ejemplo sencillo
TEST 2	0x4000000………	1073741824	0x40000000	
TEST 3	0x0800000………	2147483648	0x80000000	
TEST 4	0x1000000………	4294967296	0x100000000	
TEST 5	0x7FFFFFFF……	34359738352	0x7FFFFFFF0	
TEST 6	0x80000000……..	-34359738368	0xFFFFFFF800000000	
TEST 7	0xF0000000……..	-4294967296	0xFFFFFFFF00000000	
TEST 8	0xF8000000……..	-2147483648	0xFFFFFFFF80000000	
TEST 9	0xF7FFFFFF……	-2147483664	0xFFFFFFFF7FFFFFF0	
TEST 10	100000000………	1600000000	0x5F5E1000	
TEST 11	200000000………	3200000000	0xBEBC2000	

TEST 12	300000000……	4800000000	0x11E1A3000	
TEST 13	2000000000	32000000000	773594000	
TEST 14	3000000000	-20719476736	-20719476736	Incorrecto
TEST 15	-100000000	-1600000000	-1600000000	
TEST 16	-200000000	-3200000000	-3200000000	
TEST 17	-300000000	-4800000000	-4800000000	
TEST 18	-2000000000	-32000000000	-32000000000	
TEST 19	-3000000000	20719476736	4D2FA2000	Incorrecto

Ejercicio 4:

A continuación, se nos pide calcular la media de los datos de la lista. Para ello, usaremos la instrucción idiv, que almacena los resultados de cociente y resto en los registros eax y edx respectivamente. El código nos ha quedado así:

```
.section .data
#ifndef TEST
#define TEST 20
#endif

                        .macro linea
       #if TEST==1
                                        .int 1,2,1,2
#elif TEST==2
                                        .int -1,-2,-1,-2
#elif TEST==3
                                        .int 0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF
#elif TEST==4
                                        .int 0x80000000,0x80000000,0x80000000,0x80000000
#elif TEST==5
                                        .int 0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF
#elif TEST==6
                                        .int 2000000000,2000000000,2000000000,2000000000
#elif TEST==7
                                        .int 3000000000,3000000000,3000000000,3000000000
#elif TEST==8
                                        .int -2000000000,-2000000000,-2000000000,-2000000000
#elif TEST==9
                                        .int -3000000000,-3000000000,-3000000000,-3000000000
#elif TEST>=10 && TEST<=14
                                        .int 1,1,1,1
#elif TEST>=15 && TEST<=19
                                        .int -1,-1,-1,-1
#else
               .error "Definir TEST entre 1...19"
#endif
               .endm

               .macro linea0
       #if TEST>=1 && TEST<=9
                                        linea
```

Noelia Escalera Mejías
Grupo A3

```
#elif TEST==10
                                        .int 0,2,1,1
#elif TEST==11
                                        .int 1,2,1,1
#elif TEST==12
                                        .int 8,2,1,1
#elif TEST==13
                                        .int 15,2,1,1
#elif TEST==14
                                        .int 16,2,1,1
#elif TEST==15
                                        .int 0,-2,-1,-1
#elif TEST==16
                                        .int -1,-2,-1,-1
#elif TEST==17
                                        .int -8,-2,-1,-1
#elif TEST==18
                                        .int -15,-2,-1,-1
#elif TEST==19
                                        .int -16,-2,-1,-1
#else
                .error "Definir TEST entre 1..19"
#endif
                .endm
lista:    linea0
                                .irpc i,123
                                                        linea
                .endr
longlista:      .int   (.-lista)/4
media:          .int   0
resto:          .int   0
formato:          .ascii "media \t = %11d \t resto \t = %11d\n"
                                        .asciz       "\t\t = 0x %08x \t\t = 0x %08x \n"
.section .text
 main: .global  main

        call trabajar  # subrutina de usuario
        call imprim_C  # printf()  de libC
        call acabar_C  # exit()    de libC
        ret

trabajar:
        mov     $lista, %ebx
        mov  longlista, %ecx
        call suma              # == suma(&lista, longlista);
        mov  %eax, media
        mov  %edx, resto
        ret

suma:
        push     %rsi
        mov  $0, %eax  # Ponemos a cero los registros que usaremos
        mov  $0, %rsi
        mov  $0, %edx
        mov  $0, %ebp
        mov  $0, %edi
```

Noelia Escalera Mejías
Grupo A3

```
bucle:
        mov   (%rbx,%rsi,4),%eax # Se lee el elemento de la lista
        cdq                      # Extensión de signo
        add    %eax,%ebp         # Suma sin acarreo
        adc    %edx,%edi         # Suma con acarreo
        mov    %ebp,%eax         # Movemos el cociente y el resto a los registros convenientes para idiv
        mov    %edi,%edx
        inc    %rsi              # Incrementamos el contador
        cmp    %rsi,%rcx         # Comprobamos si ha sumado todos los elementos
        jne     bucle            # Si no lo ha hecho, repite el bucle

        idiv   %ecx              # Divide lo que hay en los registros edx:eax entre el número de elementos
                                # de la lista (que está en ecx)

        pop    %rsi
        ret

imprim_C:                       # requiere libC
        mov   $formato,%edi      # 'Avisa' del formato que va a usar
        mov    media,%rsi        # Paso de parámetros para la primera representación
        mov    resto,%rdx
        mov    media,%ecx        # Paso de parámetros para la segunda representación
        mov    resto,%r8d
        mov           $0,%eax   # varargin sin xmm
        call   printf           # == printf(formato, res, res);
        ret

acabar_C:                       # requiere libC
        mov   media, %rdi
        call _exit              # ==  exit(resultado)
        ret
```

Al ejecutar el script obtenemos los siguientes resultados:

```
T#1 media       =           1   resto   =           8
                = 0x 00000001           = 0x 00000008
T#2 media       =          -1   resto   =          -8
                = 0x ffffffff           = 0x fffffff8
T#3 media       =  2147483647   resto   =           0
                = 0x 7fffffff           = 0x 00000000
T#4 media       = -2147483648   resto   =           0
                = 0x 80000000           = 0x 00000000
T#5 media       =          -1   resto   =           0
                = 0x ffffffff           = 0x 00000000
T#6 media       =  2000000000   resto   =           0
                = 0x 77359400           = 0x 00000000
T#7 media       = -1294967296   resto   =           0
                = 0x b2d05e00           = 0x 00000000
T#8 media       = -2000000000   resto   =           0
                = 0x 88ca6c00           = 0x 00000000
```

```
media.s: Mensajes del ensamblador:
media.s:61: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:61: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:61: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:61: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
T#9 media       =  1294967296   resto   =           0
                = 0x 4d2fa200           = 0x 00000000
T#10 media      =           1   resto   =           0
                = 0x 00000001           = 0x 00000000
T#11 media      =           1   resto   =           1
                = 0x 00000001           = 0x 00000001
T#12 media      =           1   resto   =           8
                = 0x 00000001           = 0x 00000008
T#13 media      =           1   resto   =          15
                = 0x 00000001           = 0x 0000000f
T#14 media      =           2   resto   =           0
                = 0x 00000002           = 0x 00000000
T#15 media      =          -1   resto   =           0
                = 0x ffffffff           = 0x 00000000
T#16 media      =          -1   resto   =          -1
                = 0x ffffffff           = 0x ffffffff
T#17 media      =          -1   resto   =          -8
                = 0x ffffffff           = 0x fffffff8
T#18 media      =          -1   resto   =         -15
                = 0x ffffffff           = 0x fffffff1
T#19 media      =          -2   resto   =           0
```

Tabla para mejor visualización de los resultados:

Número de test	Lista	Resultado en decimal	Resultado en hexadecimal
TEST 1	1,2,1,2,……..	Media: 1 Resto: 8	Media: 0x00000001 Resto: 0x00000008
TEST 2	-1,-2,-1,-2,...……..	Media: -1 Resto: -8	Media: 0xffffffff Resto: 0xfffffff8
TEST 3	0x7FFFFFFF,……..…..	Media: 2147483647 Resto: 0	Media: 0x7fffffff Resto: 0x00000000
TEST 4	0x80000000,....…..…..	Media: -2147483648 Resto: 0	Media: 0x80000000 Resto: 0x00000000
TEST 5	0xFFFFFFFF,……..……	Media: -1 Resto: 0	Media: 0xffffffff Resto: 0x00000000
TEST 6	2000000000,…….	Media: 2000000000 Resto: 0	Media: 0x77359400 Resto: 0x00000000
TEST 7 (Incorrecto)	3000000000,…….	Media: -1294967296 Resto: 0	Media: 0xb2d05e00 Resto: 0x00000000
TEST 8	-2000000000,…….	Media: -2000000000 Resto: 0	Media: 0x88ca6c00 Resto: 0x00000000
TEST 9 (Incorrecto)	-3000000000,…….	Media: 1294967296 Resto: 0	Media: 0x4d2fa200 Resto: 0x00000000
TEST 10	0,2,1,1,1,1,…….	Media: 1 Resto: 0	Media: 0x00000001 Resto: 0x00000000
TEST 11	1,2,1,1,1,1,…….	Media: 1 Resto: 1	Media: 0x00000001 Resto: 0x00000001
TEST 12	8,2,1,1,1,1,…….	Media: 1 Resto: 8	Media: 0x00000001 Resto: 0x00000008
TEST 13	15,2,1,1,1,1,...……..	Media: 1 Resto: 15	Media: 0x00000001 Resto: 0x0000000f
TEST 14	16,2,1,1,1,1,……..	Media: 2 Resto: 0	Media: 0x00000002 Resto: 0x00000000
TEST 15	0,-2,-1,-1,-1,-1,…….	Media: -1 Resto 0	Media: 0xffffffff Resto: 0x00000000
TEST 16	-1,-2,-1,-1,-1,-1,…….	Media: -1 Resto: -1	Media: 0xffffffff Resto: 0xffffffff
TEST 17	-8,-2,-1,-1, -1,-1,……..	Media: -1 Resto: -8	Media: 0xffffffff Resto: 0xfffffff8
TEST 18	-15,-2,-1,-1, -1,-1,…….	Media: -1 Resto: -15	Media: 0xffffffff Resto: 0xfffffff1

Noelia Escalera Mejías
Grupo A3

TEST 19	-16,-2,-1,-1, -1,-1,…….	Media: -2 Resto: 0	Media: 0xfffffffe Resto: 0x00000000

Ejercicio 5:

El último ejercicio consiste en hacer la media pero usando un registro de 64 bits, en vez de dos registros de 32 bits. El programa imprimirá para cada test el resultado calculándolo como se ha visto en el apartado 4 y posteriormente como se pide para este apartado. El código nos queda así:

```
.section .data
#ifndef TEST
#define TEST 20
#endif

                        .macro linea
       #if TEST==1
                                        .int 1,2,1,2
#elif TEST==2
                                        .int -1,-2,-1,-2
#elif TEST==3
                                        .int 0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF
#elif TEST==4
                                        .int 0x80000000,0x80000000,0x80000000,0x80000000
#elif TEST==5
                                        .int 0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF
#elif TEST==6
                                        .int 2000000000,2000000000,2000000000,2000000000
#elif TEST==7
                                        .int 3000000000,3000000000,3000000000,3000000000
#elif TEST==8
                                        .int -2000000000,-2000000000,-2000000000,-2000000000
#elif TEST==9
                                        .int -3000000000,-3000000000,-3000000000,-3000000000
#elif TEST>=10 && TEST<=14
                                        .int 1,1,1,1
#elif TEST>=15 && TEST<=19
                                        .int -1,-1,-1,-1
#else
               .error "Definir TEST entre 1...19"
#endif
               .endm

               .macro linea0
       #if TEST>=1 && TEST<=9
                                        linea
#elif TEST==10
                                        .int 0,2,1,1
#elif TEST==11
                                        .int 1,2,1,1
#elif TEST==12
                                        .int 8,2,1,1
#elif TEST==13
                                        .int 15,2,1,1
#elif TEST==14
                                        .int 16,2,1,1
#elif TEST==15
                                        .int 0,-2,-1,-1
#elif TEST==16
                                        .int -1,-2,-1,-1
#elif TEST==17
                                        .int -8,-2,-1,-1
```

```
#elif TEST==18
                                    .int -15,-2,-1,-1
#elif TEST==19
                                    .int -16,-2,-1,-1
#else
               .error "Definir TEST entre 1..19"
#endif
               .endm
lista:    linea0
                              .irpc i,123
                                                    linea
               .endr
longlista:     .quad   (.-lista)/4
media:         .int   0
resto:         .int   0
formato:         .ascii "media \t = %11d \t resto \t = %11d\n"
                                    .asciz       "\t\t = 0x %08x \t\t = 0x %08x \n"
formatq:         .ascii "media \t = %11d \t resto \t = %11d\n"
                                    .asciz       "\t\t = 0x %08x \t\t = 0x %08x \n"

.section .text
 main: .global  main

       call trabajar  # subrutina de usuario
       call imprim_C  # printf()  de libC
       call trabajarq
       call imprim_Cq
       call acabar_C  # exit()    de libC
       ret

trabajar:
       mov     $lista, %ebx
       mov  longlista, %ecx
       call suma              # == suma(&lista, longlista);
       mov  %eax, media
       mov  %edx, resto
       ret

suma:
       push     %rsi
       mov  $0, %eax  # Ponemos a cero los registros que usaremos
       mov  $0, %rsi
       mov  $0, %edx
       mov  $0, %ebp
       mov  $0, %edi
bucle:
       mov  (%rbx,%rsi,4),%eax # Se lee el elemento de la lista
       cdq                     # Extensión de signo
       add   %eax,%ebp         # Suma sin acarreo
       adc   %edx,%edi         # Suma con acarreo
       mov   %ebp,%eax         # Movemos el cociente y el resto a los registros convenientes para idiv
       mov   %edi,%edx
       inc   %rsi              # Incrementamos el contador
       cmp   %rsi,%rcx         # Comprobamos si ha sumado todos los elementos
       jne    bucle            # Si no lo ha hecho, repite el bucle

       idiv  %ecx              # Divide lo que hay en los registros edx:eax entre el número de
elementos de la lista (que está en ecx)

       pop   %rsi
       ret
```

Noelia Escalera Mejías
Grupo A3

```
trabajarq:
       mov     $lista, %rbx
       mov  longlista, %ecx
       call sumaq
       mov  %eax, media
       mov  %edx, resto
       ret

sumaq:
       push           %rsi
       mov  $0,%rsi # Pone a 0 los registros que usaremos
       mov  $0,%rdi
       mov  $0,%rax
       mov  $0,%rdx
bucleq:
       mov (%rbx,%rsi,4),%eax # Se lee el elemento de la lista
       cdqe                   # Extensión de signo
       add  %rax,%rdi         # Suma
       mov  %rdi,%rax         # Movemos el resultado al registro conveniente para idiv
       inc  %rsi              # Incrementamos el contador
       cmp  %rsi,%rcx         # Miramos si ha sumado todo
       jne  bucleq            # Si no ha terminado, sigue con el bucle

       cqo                    # Extensión de signo
       idiv %rcx                   # Divide

       pop %rsi

       ret

imprim_C:                     # requiere libC
       mov  $formato,%rdi      # 'Avisa' del formato que va a usar
       mov   media,%rsi        # Paso de parámetros para la primera representación
       mov   resto,%rdx
       mov   media,%ecx        # Paso de parámetros para la segunda representación
       mov   resto,%r8d
       mov          $0,%eax   # varargin sin xmm
       call  printf           # == printf(formato, res, res);
       ret

imprim_Cq:                    # requiere libC
       mov  $formatq,%rdi      # 'Avisa' del formato que va a usar
       mov   media,%esi        # Paso de parámetros para la primera representación
       mov   resto,%edx
       mov   media,%ecx        # Paso de parámetros para la segunda representación
       mov   resto,%r8d
       mov          $0,%eax   # varargin sin xmm
       call  printf           # == printf(formato, res, res);
       ret


acabar_C:                     # requiere libC
       mov  media, %edi
       call _exit             # ==  exit(resultado)
       ret
```

Noelia Escalera Mejías
Grupo A3

Al ejecutar el script obtenemos los siguientes resultados:

```
T#1 media        =           1    resto    =           8
                 = 0x 00000001             = 0x 00000008
media    =          1    resto    =        8
                 = 0x 00000001             = 0x 00000008
T#2 media        =          -1    resto    =          -8
                 = 0x ffffffff             = 0x fffffff8
media    =         -1    resto    =       -8
                 = 0x ffffffff             = 0x fffffff8
T#3 media        =  2147483647    resto    =           0
                 = 0x 7fffffff             = 0x 00000000
media    =  2147483647   resto    =        0
                 = 0x 7fffffff             = 0x 00000000
T#4 media        = -2147483648    resto    =           0
                 = 0x 80000000             = 0x 00000000
media    = -2147483648   resto    =        0
                 = 0x 80000000             = 0x 00000000
T#5 media        =          -1    resto    =           0
                 = 0x ffffffff             = 0x 00000000
media    =         -1    resto    =        0
                 = 0x ffffffff             = 0x 00000000
T#6 media        =  2000000000    resto    =           0
                 = 0x 77359400             = 0x 00000000
media    =  2000000000    resto    =       0
                 = 0x 77359400             = 0x 00000000
T#7 media        = -1294967296    resto    =           0
                 = 0x b2d05e00             = 0x 00000000
media    = -1294967296    resto    =       0
                 = 0x b2d05e00             = 0x 00000000
T#8 media        = -2000000000    resto    =           0
                 = 0x 88ca6c00             = 0x 00000000
media    = -2000000000    resto    =       0
                 = 0x 88ca6c00             = 0x 00000000
```

```
media.s: Mensajes del ensamblador:
media.s:61: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:61: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:61: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:61: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
media.s:64: Aviso: valora 0xffffffff4d2fa200 truncado a 0x4d2fa200
T#9 media        =  1294967296    resto    =           0
                 = 0x 4d2fa200             = 0x 00000000
media    =  1294967296    resto    =       0
                 = 0x 4d2fa200             = 0x 00000000
T#10 media       =           1    resto    =           0
                 = 0x 00000001             = 0x 00000000
media    =          1    resto    =        0
                 = 0x 00000001             = 0x 00000000
T#11 media       =           1    resto    =           1
                 = 0x 00000001             = 0x 00000001
media    =          1    resto    =        1
                 = 0x 00000001             = 0x 00000001
T#12 media       =           1    resto    =           8
                 = 0x 00000001             = 0x 00000008
media    =          1    resto    =        8
                 = 0x 00000001             = 0x 00000008
T#13 media       =           1    resto    =          15
                 = 0x 00000001             = 0x 0000000f
media    =          1    resto    =       15
                 = 0x 00000001             = 0x 0000000f
T#14 media       =           2    resto    =           0
                 = 0x 00000002             = 0x 00000000
media    =          2    resto    =        0
                 = 0x 00000002             = 0x 00000000
```

```
T#15 media       =          -1    resto    =           0
                 = 0x ffffffff             = 0x 00000000
media    =         -1    resto    =        0
                 = 0x ffffffff             = 0x 00000000
T#16 media       =          -1    resto    =          -1
                 = 0x ffffffff             = 0x ffffffff
media    =         -1    resto    =       -1
                 = 0x ffffffff             = 0x ffffffff
T#17 media       =          -1    resto    =          -8
                 = 0x ffffffff             = 0x fffffff8
media    =         -1    resto    =       -8
                 = 0x ffffffff             = 0x fffffff8
T#18 media       =          -1    resto    =         -15
                 = 0x ffffffff             = 0x fffffff1
media    =         -1    resto    =      -15
                 = 0x ffffffff             = 0x fffffff1
T#19 media       =          -2    resto    =           0
                 = 0x fffffffe             = 0x 00000000
media    =         -2    resto    =        0
```