Raúl Ruano Narváez **Grupo: 2ºA2**

5.1 Sumar N enteros sin signo de 32 bits sobre dos registros de 32 bits usando uno de ellos como acumulador de acarreos (N≈16)

```
.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
        mov    $lista, %ebx          # Guardamos posición de la lista
        mov  longlista, %ecx         # Guardamos el tamaño de la lista
        call suma                    # Llamada a la función suma

        mov  %eax, resultado
        mov  %edx, resultado+4

        mov   $formato, %rdi
        mov   resultado,%rsi
```

```
        mov   resultado,%rdx
        mov        $0,%eax   # varargin sin xmm
        call  printf         # == printf(formato, res, res);

        mov  resultado, %edi
        call _exit                    # ==  exit(resultado)
        ret

suma:
        mov  $0, %eax
        mov  $0, %edx
        mov  $0, %esi

bucle:
        add   (%ebx,%esi,4), %eax
        jnc   no_acarreo
        inc   %edx

no_acarreo:
        inc     %esi
        cmp   %esi,%ecx
        jne   bucle
        ret
```

Datos obtenidos:

```
__TEST01__-----------------------------------
resultado       =              16 (uns)
                = 0x           10 (hex)
                = 0x 00000010 e2f6cd80
__TEST02__-----------------------------------
resultado       =      4294967280 (uns)
                = 0x       fffffff0 (hex)
                = 0x 00000010 03456d80
__TEST03__-----------------------------------
resultado       =      4294967296 (uns)
                = 0x       100000000 (hex)
                = 0x 00000010 07196d80
__TEST04__-----------------------------------
resultado       =      68719476720 (uns)
                = 0x       fffffff0 (hex)
                = 0x 00000010 f78f2d80
__TEST05__-----------------------------------
resultado       =      68719476720 (uns)
                = 0x       fffffff0 (hex)
                = 0x 00000010 d4a16d80
```

```
__TEST06__----------------------------------
resultado       =       3200000000 (uns)
                = 0x        bebc2000 (hex)
                = 0x 00000010 ec6b1d80
__TEST07__----------------------------------
resultado       =       4800000000 (uns)
                = 0x       11e1a3000 (hex)
                = 0x 00000010 ecbe0d80
__TEST08__----------------------------------
resultado       =      11280523264 (uns)
                = 0x       2a05f2000 (hex)
                = 0x 00000010 b28f9d80
```

5.2 Sumar N enteros sin signo de 32 bits sobre dos registros de 32 bits mediante extensión con ceros (N≈16)

```
.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
        mov     $lista, %ebx            # Guardamos posición de la lista
        mov  longlista, %ecx            # Guardamos el tamaño de la lista
        call suma                                       # Llamada a la función
suma

        mov  %eax, resultado            # Metemos los resultado de suma en resultado para luego
mostrarlo
        mov  %esi, resultado+4

        mov   $formato, %rdi
        mov   resultado,%rsi
        mov   resultado,%rdx
        mov         $0,%eax     # varargin sin xmm
        call  printf            # == printf(formato, res, res);

        mov  resultado, %edi
        call _exit                      # ==  exit(resultado)
        ret

suma:
        mov  $0, %eax
        mov  $0, %edx
        mov  $0, %esi

bucle:
        add      (%ebx, %edx,4), %eax
        adc      $0, %esi                       # acumular i-ésimo elemento
        inc  %edx                       # incrementar índice
        cmp  %edx,%ecx                  # comparar con longitud
        jne      bucle
        ret
```

Datos obtenidos:

```
__TEST01__----------------------------------
resultado       =              16 (uns)
                = 0x            10 (hex)
                = 0x 00000010 3ba3dd80
__TEST02__----------------------------------
resultado       =      4294967280 (uns)
                = 0x       fffffff0 (hex)
                = 0x 00000010 744c1d80
__TEST03__----------------------------------
resultado       =      4294967296 (uns)
                = 0x       100000000 (hex)
                = 0x 00000010 baa87d80
__TEST04__----------------------------------
resultado       =      68719476720 (uns)
                = 0x       fffffff0 (hex)
                = 0x 00000010 01d37d80
__TEST05__----------------------------------
resultado       =      68719476720 (uns)
                = 0x       fffffff0 (hex)
                = 0x 00000010 d1d05d80
__TEST06__----------------------------------
resultado       =      3200000000 (uns)
                = 0x       bebc2000 (hex)
                = 0x 00000010 098b3d80
__TEST07__----------------------------------
resultado       =      4800000000 (uns)
                = 0x       11e1a3000 (hex)
                = 0x 00000010 0186dd80
__TEST08__----------------------------------
resultado       =      11280523264 (uns)
                = 0x       2a05f2000 (hex)
                = 0x 00000010 2f572d80
```

5.3 Sumar N enteros con signo de 32 bits sobre dos registros de 32 bits (mediante extensión de signo, naturalmente) (N≈16)

```
.section .data
#ifndef TEST
#define TEST 19
#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 (uns)\n"
                        .ascii "\t\t = 0x%18lx (hex)\n"
                        .asciz "\t\t = 0x %08x %08x \n"

.section .text
main: .global  main
        mov     $lista, %ebx          # Guardamos posición de la lista
        mov  longlista, %ecx          # Guardamos el tamaño de la lista
        call suma                                          # Llamada a la función
suma

        mov  %eax, resultado              # Metemos los resultado de suma en resultado para
luego mostrarlo
        mov  %edx, resultado+4

        mov   $formato, %rdi
        mov   resultado,%rsi
        mov   resultado,%rdx
        mov        $0,%eax   # varargin sin xmm
        call  printf         # == printf(formato, res, res);

        mov  resultado, %edi
        call _exit                    # ==  exit(resultado)
        ret

suma:
        mov  $0, %edi
        mov  $0, %esi
        mov  $0, %ebp

bucle:
        mov     (%ebx, %ebp,4), %eax
 cdq

 add  %eax, %esi
        adc      %edx, %edi

        inc  %ebp
        cmp  %ebp,%ecx
        jne      bucle
```

```
mov  %edi, %edx
mov  %esi, %eax

ret
```

Datos obtenidos:

```
__TEST01__----------------------------------
resultado       =              -16 (uns)
                = 0x  fffffffffffffff0 (hex)
                = 0x 00000010 09b37d80
__TEST02__----------------------------------
resultado       =       1073741824 (uns)

                = 0x         40000000 (hex)
                = 0x 00000010 db868d80
__TEST03__----------------------------------
resultado       =       2147483648 (uns)
                = 0x         80000000 (hex)
                = 0x 00000010 07882d80
__TEST04__----------------------------------
resultado       =       4294967296 (uns)
                = 0x        100000000 (hex)
                = 0x 00000010 f4aa4d80
__TEST05__----------------------------------
resultado       =       34359738352 (uns)
                = 0x        7fffffff0 (hex)
                = 0x 00000010 84aadd80
__TEST06__----------------------------------
resultado       =      -34359738368 (uns)
                = 0x  ffffffe800000000 (hex)
                = 0x 00000010 1e2a1d80
__TEST07__----------------------------------
resultado       =       -4294967296 (uns)
                = 0x  ffffffff00000000 (hex)
                = 0x 00000010 42bcbd80
__TEST08__----------------------------------
resultado       =       -2147483648 (uns)
                = 0x  ffffffff80000000 (hex)
                = 0x 00000010 1eb70d80
__TEST09__----------------------------------
resultado       =       -2147483664 (uns)
                = 0x  ffffffff7ffffff0 (hex)
                = 0x 00000010 f668bd80
__TEST10__----------------------------------
resultado       =       1600000000 (uns)
                = 0x         5f5e1000 (hex)
                = 0x 00000010 b4211d80
```

```
__TEST11__-----------------------------------
resultado       =         3200000000 (uns)
                = 0x          bebc2000 (hex)

                = 0x 00000010 ee2aed80
__TEST12__-----------------------------------
resultado       =         4800000000 (uns)
                = 0x         11e1a3000 (hex)
                = 0x 00000010 471dad80
__TEST13__-----------------------------------
resultado       =        32000000000 (uns)
                = 0x         773594000 (hex)
                = 0x 00000010 c5998d80
__TEST14__-----------------------------------
resultado       =       -20719476736 (uns)
                = 0x  fffffffb2d05e000 (hex)
                = 0x 00000010 5a106d80
__TEST15__-----------------------------------
resultado       =        -1600000000 (uns)
                = 0x  fffffffa0a1f000 (hex)
                = 0x 00000010 a9d9bd80
__TEST16__-----------------------------------
resultado       =        -3200000000 (uns)
                = 0x  fffffff4143e000 (hex)
                = 0x 00000010 5ef4fd80
__TEST17__-----------------------------------
resultado       =        -4800000000 (uns)
                = 0x  fffffffee1e5d000 (hex)
                = 0x 00000010 dbc0dd80
__TEST18__-----------------------------------
resultado       =       -32000000000 (uns)
                = 0x  ffffff88ca6c000 (hex)
                = 0x 00000010 23c90d80
__TEST19__-----------------------------------
resultado       =        20719476736 (uns)
                = 0x         4d2fa2000 (hex)
                = 0x 00000010 b9eded80
```

5.4 Media y resto de N enteros con signo de 32 bits calculada usando registros de 32 bits (N≈16)

```
.section .data
#ifndef TEST
#define TEST 19
#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
                    .int 0,2,1,1,1,1
  #elif TEST==11
                    .int 1,2,1,1,1,1
      #elif TEST==12
                    .int 8,2,1,1,1,1
  #elif TEST==13
                    .int 15,2,1,1,1,1
  #elif TEST==14
                    .int 16,2,1,1,1,1
  #elif TEST==15
                    .int 0,-2,-1,-1,-1,-1
  #elif TEST==16
                    .int -1,-2,-1,-1,-1,-1
      #elif TEST==17
                    .int -8,-2,-1,-1,-1,-1
  #elif TEST==18
                    .int -15,-2,-1,-1,-1,-1
  #elif TEST==19
                    .int -16,-2,-1,-1,-1,-1
                #else
                    .error "Definir TEST entre 1..19"
```

```
                #endif
                        .endm
lista: .irpc i,1234
                         linea
                        .endr

longlista: .int (.-lista)/4
media: .int 0
resto: .int 0

formato: .ascii "media \t = %11d \t resto \t = %11d \n"
                                .asciz "\t = 0x %08x \t \t = 0x %08x\n"

.section .text
main: .global  main
        mov    $lista, %ebx          # Guardamos posición de la lista
        mov  longlista, %ecx         # Guardamos el tamaño de la lista
        call suma                                                  # Llamada a la función
suma

        mov  %eax, media             # Metemos los resultado de suma en resultado para luego
mostrarlo
        mov  %edx, resto

        mov   $formato, %rdi
        mov   media,%rsi
        mov   resto,%rdx
        mov        $0,%eax   # varargin sin xmm
        call  printf             # == printf(formato, res, res);

        mov  media, %edi
        call _exit                       # ==  exit(resultado)
        ret

suma:
        mov  $0, %edi
        mov  $0, %esi
        mov  $0, %ebp

bucle:
        mov     (%ebx, %ebp,4), %eax
 cdq

 add  %eax, %esi
        adc      %edx, %edi

        inc  %ebp
        cmp  %ebp,%ecx
```

```
        jne     bucle

 mov  %edi, %edx
 mov  %esi, %eax

        idiv %ecx

 ret
```

Datos obtenidos:

```
__TEST01__----------------------------------
media   =         1   resto   =         8
        = 0x 00000010         = 0x 30fa2d80
__TEST02__----------------------------------
media   =        -1   resto   =        -8
        = 0x 00000010         = 0x 26f86d80
__TEST03__----------------------------------
media   =  2147483647   resto   =         0
        = 0x 00000010         = 0x 23ca7d80
__TEST04__----------------------------------
media   = -2147483648   resto   =         0
        = 0x 00000010         = 0x 67c77d80
__TEST05__----------------------------------
media   =        -1   resto   =         0
        = 0x 00000010         = 0x 27a67d80
__TEST06__----------------------------------
media   =  2000000000   resto   =         0
        = 0x 00000010         = 0x fdabbd80
__TEST07__----------------------------------
media   = -1294967296   resto   =         0
        = 0x 00000010         = 0x fdbfbd80
__TEST08__----------------------------------
media   = -2000000000   resto   =         0
        = 0x 00000010         = 0x 35b00d80
__TEST09__----------------------------------

media   =  1294967296   resto   =         0
        = 0x 00000010         = 0x e9eaed80
__TEST10__----------------------------------
media   =         1   resto   =         0
        = 0x 00000018         = 0x 0a481d80
__TEST11__----------------------------------
media   =         1   resto   =         4
        = 0x 00000018         = 0x 15a60d80
__TEST12__----------------------------------
media   =         2   resto   =         8
        = 0x 00000018         = 0x 17e1fd80
```

```
__TEST13__----------------------------------
media   =         3   resto   =         12
        = 0x 00000018          = 0x 32094d80
__TEST14__----------------------------------
media   =         3   resto   =         16
        = 0x 00000018          = 0x 79616d80
__TEST15__----------------------------------
media   =        -1   resto   =          0
        = 0x 00000018          = 0x f2f6ed80
__TEST16__----------------------------------
media   =        -1   resto   =         -4
        = 0x 00000018          = 0x b08ddd80
__TEST17__----------------------------------
media   =        -2   resto   =         -8
        = 0x 00000018          = 0x b919dd80
__TEST18__----------------------------------
media   =        -3   resto   =        -12
        = 0x 00000018          = 0x ce823d80
__TEST19__----------------------------------
media   =        -3   resto   =        -16
        = 0x 00000018          = 0x ec38dd80
```

5.5 Media y resto de N enteros calculada en 32 y en 64 bits (N≈16)

En este ejercicio no he conseguido que funcione correctamente ya que los resultados no llegan a ser los correctos pero no he encontrado el fallo, igualmente adjunto lo que he llegado a realizar.

```
.section .data
#ifndef TEST
#define TEST 19
#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
                    .int 0,2,1,1,1,1
  #elif TEST==11
                    .int 1,2,1,1,1,1
      #elif TEST==12
                    .int 8,2,1,1,1,1
  #elif TEST==13
                    .int 15,2,1,1,1,1
  #elif TEST==14
                    .int 16,2,1,1,1,1
  #elif TEST==15
                    .int 0,-2,-1,-1,-1,-1
  #elif TEST==16
                    .int -1,-2,-1,-1,-1,-1
      #elif TEST==17
                    .int -8,-2,-1,-1,-1,-1
  #elif TEST==18
                    .int -15,-2,-1,-1,-1,-1
  #elif TEST==19
                    .int -16,-2,-1,-1,-1,-1
            #else
                    .error "Definir TEST entre 1..19"
            #endif
                    .endm
lista: .irpc i,1234
                     linea
                    .endr

longlista: .int (.-lista)/4
media: .quad 0
resto: .quad 0

formato: .ascii "media \t = %11d \t resto \t = %11d \n"
                                .asciz "\t = 0x %08x \t \t = 0x %08x\n"

.section .text
main: .global  main
        mov    $lista, %ebx          # Guardamos posición de la lista
        mov  longlista, %ecx         # Guardamos el tamaño de la lista
        call suma                                          # Llamada a la función
suma

        mov  %rax, media             # Metemos los resultado de suma en resultado para luego
mostrarlo
        mov  %rdx, resto

        mov   $formato, %rdi
```

```
        mov   media,%rsi
        mov   resto,%rdx
        mov         $0,%eax   # varargin sin xmm
        call  printf          # == printf(formato, res, res);

        mov  media, %rdi
        call _exit                    # ==  exit(resultado)
        ret

suma:
        mov $0, %rdi
        mov $0, %ebp
        mov $0, %rsi

bucle:
        mov (%ebx, %ebp,4), %rax
        cqo

        add %rax, %rdi
        adc %rdx, %rsi

        inc %ebp
        cmp %ebp, %ecx
        jne bucle

        mov %rdi, %rax
        mov %rsi, %rdx

        idiv %rcx

        ret
```

Datos obtenidos:

```
__TEST01__---------------------------------
media   =  1879048193   resto   =         8
        = 0x 00000010          = 0x baaded80
__TEST02__---------------------------------

media   = -1879048194   resto   =         8
        = 0x 00000010          = 0x 73d50d80
__TEST03__---------------------------------
media   = -1879048193   resto   =         0
        = 0x 00000010          = 0x bf483d80
__TEST04__---------------------------------
media   = -2147483648   resto   =         0
        = 0x 00000010          = 0x 7ca1dd80
```

```
__TEST05__----------------------------------
media = 268435455 resto = 0
      = 0x 00000010       = 0x 9f19cd80

__TEST06__----------------------------------
media = 2000000000 resto = 0
      = 0x 00000010       = 0x 8b6a7d80
__TEST07__----------------------------------
media = -1294967296 resto = 0
      = 0x 00000010       = 0x 75441d80
__TEST08__----------------------------------
media = -2000000000 resto = 0
      = 0x 00000010       = 0x 78d78d80
__TEST09__----------------------------------
media = 1294967296 resto = 0

      = 0x 00000010       = 0x 56ad0d80
__TEST10__----------------------------------
media = 1 resto = 0
      = 0x 00000018       = 0x e3132d80
__TEST11__----------------------------------
media = 536870913 resto = 4
      = 0x 00000018       = 0x 368efd80
__TEST12__----------------------------------
media = 2 resto = 8
      = 0x 00000018       = 0x 48f5fd80
__TEST13__----------------------------------
media = -536870909 resto = 12
      = 0x 00000018       = 0x 28281d80
__TEST14__----------------------------------
media = 3 resto = 16
      = 0x 00000018       = 0x e3415d80
__TEST15__----------------------------------
media = -715827884 resto = 8
      = 0x 00000018       = 0x a841cd80
__TEST16__----------------------------------
media = -536870914 resto = 20
      = 0x 00000018       = 0x a8e52d80
__TEST17__----------------------------------
media = -2 resto = -8
      = 0x 00000018       = 0x 85fe1d80
__TEST18__----------------------------------
media = 536870909 resto = -12
      = 0x 00000018       = 0x 0fc27d80
__TEST19__----------------------------------
media = -3 resto = -16
      = 0x 00000018       = 0x fd316d80
```

Raúl Ruano Narváez **Grupo: 2ºA2**

DIARIO DE TRABAJO:

El trabajo realizado en esta practica no ha sido documentado como un diario debido a que me he dado cuenta a días de la entrega que debía hacerlo. Lo que si he realizado fue una primera asimilación de los conceptos los primeros días de la practica, entendiendo lo que se pedía así como las diferentes llamadas que debía realizar en el código. Una vez asimilé dichos conceptos me puse a trabajar en suma5.1, la cual, tras muchas pruebas llegue a conseguir entender perfectamente y a realizarla en principio sin problemas ya que los resultados son los correctos. El segundo ejercicio fue cambiar lo que nos decía en el ejercicio, luego no tuve mayor problema. En el tercer ejercicio si tuve más problemas debido a que no llegaba a comprender correctamente la extensión de signo pedida en esta, al final llegué a comprenderla. Por ultimo los ejercicios de media eran pequeñas modificaciones del ejercicio de suma con signo así que no tuve gran problema, fueron pequeños fallos que en días lo solucioné. El ultimo ejercicio no he comprendido correctamente que es lo que se pedía, según lo que entendí he realizado una forma, pero no esta correcta debido a que el resultado que produce no es el bueno y por falta de tiempo ya no he podido seguir.