



# UNIVERSIDAD DE GRANADA

ESTRUCTURA DE COMPUTADORES  
GRADO EN INGENIERÍA INFORMÁTICA  
CURSO 2018-2019



---

## Memoria Práctica 2. Programación ensamblador x86-64 Linux.

---

Félix Ramírez García  
felixramirezgarcia@correo.ugr.es

14 de octubre de 2018

## Índice

<b>1</b>	<b>Diario de trabajo</b>	<b>3</b>
<b>2</b>	<b>Código de los ejercicios</b>	<b>3</b>
2.1	5.1 Sumar N enteros <u>s</u> insigno de 32 bits sobre registros de 32 bits usando uno de ellos como acumulador de acarreos. (N=16) . . . . .	3
2.2	5.2 Sumar N enteros <u>s</u> insigno de 32 bits sobre registros de 32 bits mediante extensión con ceros. (N=16) . . . . .	4
2.3	5.3 Sumar N enteros <u>c</u> onsigno de 32 bits sobre registros de 32 bits (mediante extensión de signo). (N=16) . . . . .	5
2.4	5.4 Media y resto de N enteros <u>c</u> onsigno de 32 bits calculada usando registros de 32 bits. (N=16) . . . . .	6
<b>3</b>	<b>Pruebas de ejecución</b>	<b>7</b>

## Índice de figuras

## Índice de tablas

1.1	Diario de trabajo de la práctica. . . . .	3
3.1	Resultado batería ejercicio 5.1 . . . . .	8
3.2	Resultado batería ejercicio 5.2 . . . . .	8
3.3	Resultado batería ejercicio 5.3 . . . . .	8
3.4	Resultado batería ejercicio 5.4 . . . . .	9

## 1. Diario de trabajo.

Fecha	Tarea desarrollada
03-10-2018	Repaso de la parte A del guion de la práctica
06-10-2018	Terminado el ejercicio 5.1 y empezado el 5.2
09-10-2018	Terminados los ejercicios 5.2 , 5.3 y 5.4
12-10-2018	Pasar la batería de test a los ejercicios
14-10-2018	Redacción de este documento en látex y últimos retoques a los ejercicios

Tabla 1.1: Diario de trabajo de la práctica.

## 2. Código de los ejercicios

### 2.1. 5.1 Sumar N enteros sin signo de 32 bits sobre registros de 32 bits usando uno de ellos como acumulador de acarreo. (N=16)

```
.section .data
    .macro linea
        #.int 1,1,1,1
        #.int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
        .int 0x10000000,0x10000000,0x10000000,0x10000000
    .endm
lista: .irpc i,1234
    linea
.endr
longlista: .int    (.-lista)/4
resultado: .quad 0x0123456701234567
formato:   .asciz  "suma = %lu = 0x%lx hex\n"

.section .text
main: .global main
    mov     $lista, %rbx
    mov     longlista, %ecx
    call    suma          # == suma(&lista, longlista);
    mov     %eax, resultado
    mov     %edx, resultado+4 # 4 bytes mas significativos

    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:
    push    %rsi
    mov     $0, %eax
    mov     $0, %edx
    mov     $0, %rsi
bucle:
    add     (%rbx,%rsi,4), %eax # realiza la adición de los registros
    jc      moreindex          # salto condicional si hay acarreo
otrobucle:
    inc     %rsi                # incremento el índice
    cmp     %rsi,%rcx           # comparo con rcx
    jne     bucle               # salto a bucle si no son iguales
    pop     %rsi
    ret
moreindex:
    inc     %edx                # incremento el registro
    cmp     %edx,%edx           # comparo para forzar el salto a
        otrobucle
    je      otrobucle          # salto a otrobucle
    ret

```

## 2.2. 5.2 Sumar N enteros sin signo de 32 bits sobre registros de 32 bits mediante extensión con ceros. (N=16)

```

.section .data
    .macro linea
        #.int 1,1,1,1
        #.int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
        #.int 0x10000000,0x10000000,0x10000000,0x10000000
        #.int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
        #.int -1,-1,-1,-1
        #.int 200000000,200000000,200000000,200000000
        #.int 300000000,300000000,300000000,300000000
        .int 500000000,500000000,500000000,500000000
    .endm
lista: .irpc i,1234
linea
.endr
longlista: .int    (.-lista)/4
resultado: .quad 0x0123456701234567
formato:   .asciz  "suma = %lu = 0x%lx hex\n"

.section .text
main: .global main
    mov     $lista, %rbx
    mov     longlista, %ecx
    call    suma          # == suma(&lista, longlista);
    mov     %eax, resultado
    mov     %edx, resultado+4 # 4 bytes mas significativos

```

```

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:
push   %rsi          # indice
mov    $0, %eax
mov    $0, %edx
mov    $0, %rsi

bucle:
add    (%rbx,%rsi,4), %eax # realiza la adición de los registros
adc    $0, %edx           # incrementa si bit CF de acarreo activo
inc    %rsi              # incrementa el índice
cmp    %rsi,%rcx          # realiza la comparativa
jne    bucle             # salta a bucle si no es igual
pop    %rsi
ret

```

### 2.3. 5.3 Sumar N enteros con signo de 32 bits sobre registros de 32 bits (mediante extensión de signo). (N=16)

```

.section .data
    .macro linea
        #.int -1,-1,-1,-1
        #.int 0x04000000,0x04000000,0x04000000,0x04000000
        #.int 0x08000000,0x08000000,0x08000000,0x08000000
        #.int 0x10000000,0x10000000,0x10000000,0x10000000
        #.int 0x7fffffff,0x7fffffff,0x7fffffff,0x7fffffff
        #.int 0x80000000,0x80000000,0x80000000,0x80000000
        #.int 0xf0000000,0xf0000000,0xf0000000,0xf0000000
        #.int 0xf8000000,0xf8000000,0xf8000000,0xf8000000
        #.int 0xf7fffffff,0xf7fffffff,0xf7fffffff,0xf7fffffff
        #.int 100000000,100000000,100000000,100000000
        #.int 200000000,200000000,200000000,200000000
        #.int 300000000,300000000,300000000,300000000
        #.int 2000000000,2000000000,2000000000,2000000000
        #.int 3000000000,3000000000,3000000000,3000000000
        #.int -100000000,-100000000,-100000000,-100000000
        #.int -200000000,-200000000,-200000000,-200000000
        #.int -300000000,-300000000,-300000000,-300000000
        #.int -2000000000,-2000000000,-2000000000,-2000000000
        #.int -3000000000,-3000000000,-3000000000,-3000000000
    .endm
lista: .irpc i,1234
linea
.endr

```

```

longlista:      .int    (.-lista)/4
resultado:      .quad   0x0123456701234567
formato:        .asciz  "suma = %ld = 0x%lx hex\n"

.section .text
main: .global  main
        mov     $lista, %rbx
        mov     longlista, %ecx
        call    suma          # == suma(&lista, longlista);
        mov     %eax, resultado
        mov     %edx, resultado+4 # 4 bytes mas significativos

        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      # acumulador
        mov     $0, %ebp      # acumulador
        mov     $0, %esi      # indice

bucle:
        mov     (%ebx,%esi,4), %eax
        cltd                # signed long to signed double long
        add     %eax,%edi      # realizar la adición de los registros
        adc     %edx,%ebp      # realiza la adición si CF activo
        inc     %esi          # incrementa el indice
        cmp     %esi,%ecx      # comparamos
        jne     bucle         # saltamos a bucle si no son iguales
        mov     %edi,%eax
        mov     %ebp,%edx
        ret

```

## 2.4. 5.4 Media y resto de N enteros con signo de 32 bits calculada usando registros de 32 bits. (N=16)

```

.section .data
        .macro linea
                .int 1,2,1,2
                #.int -1,-2,-1,-2
                #.int 0x7fffffff,0x7fffffff,0x7fffffff,0x7fffffff
                #.int 0x80000000,0x80000000,0x80000000,0x80000000
                #.int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
                #.int 2000000000,2000000000,2000000000,2000000000
                #.int 3000000000,3000000000,3000000000,3000000000
                #.int -2000000000,-2000000000,-2000000000,-2000000000
                #.int -3000000000,-3000000000,-3000000000,-3000000000
        .endm

```

```

        .endm
lista:  .irpc i,1234
        linea
    .endr
longlista:  .int    (.-lista)/4
media:     .int    0x89ABCDEF
resto:     .int    0x01234567
formato:   .asciz  "media = %8d  resto = %8d \n"

.section .text
main:  .global  main
        mov     $lista, %rbx
        mov     longlista, %ecx
        call    suma                # == suma(&lista, longlista);
        mov     %eax, media
        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            # acumulador
        mov     $0, %ebp            # acumulador
        mov     $0, %esi            # indice

bucle:
        mov     (%ebx,%esi,4), %eax
        cltd                        # signed long to signed double long
        add     %eax,%edi            # adicion de registros
        adc     %edx,%ebp            # adicion si bit CF activo
        inc     %esi                # incrementar indice
        cmp     %esi,%ecx            # comparar
        jne     bucle               # saltar a bucle si no son iguales
        mov     %edi,%eax
        mov     %ebp,%edx
        idiv    %ecx                 # realizar la division
        ret

```

### 3. Pruebas de ejecución

Input list (x16)	Output
1	suma = 16 = 0x10 hex
0x0ffffff	suma = 4294967280 = 0xffffffff0 hex
0x10000000	suma = 4294967296 = 0x100000000 hex

Tabla 3.1: Resultado bateria ejercicio 5.1

Input list (x16)	Output
1	suma = 16 = 0x10 hex
0x0ffffff	suma = 4294967280 = 0xffffffff0 hex
0x10000000	suma = 4294967296 = 0x100000000 hex
0xffffffff	suma = 68719476720 = 0xfffffffff0 hex
-1	suma = 68719476720 = 0xfffffffff0 hex
200000000	suma = 3200000000 = 0xbebc2000 hex
300000000	suma = 4800000000 = 0x11e1a3000 hex
5000000000	suma = 11280523264 = 0x2a05f2000 hex

Tabla 3.2: Resultado bateria ejercicio 5.2

Input list (x16)	Output
-1	suma = -16 = 0xfffffffffff0 hex
0x04000000	suma = 1073741824 = 0x40000000 hex
- 0x08000000	suma = 2147483648 = 0x80000000 hex
0x10000000	suma = 4294967296 = 0x100000000 hex
0x7ffffff	suma = 34359738352 = 0x7ffffff0 hex
0x80000000	suma = -34359738368 = 0xfffff800000000 hex
0xf0000000	suma = -4294967296 = 0xfffff000000000 hex
0xf8000000	suma = -2147483648 = 0xfffff800000000 hex
0xf7000000	suma = -2147483664 = 0xfffff7fffff0 hex
100000000	suma = 1600000000 = 0x5f5e1000 hex
200000000	suma = 3200000000 = 0xbebc2000 hex
300000000	suma = 4800000000 = 0x11e1a3000 hex
2000000000	suma = 32000000000 = 0x773594000 hex
3000000000	suma = -20719476736 = 0xfffffb2d05e000 hex
-100000000	suma = -1600000000 = 0xfffffa0a1f000 hex
-200000000	suma = -3200000000 = 0xfffff4143e000 hex
-300000000	suma = -4800000000 = 0xfffffee1e5d000 hex
-2000000000	suma = -32000000000 = 0xfffff88ca6c000 hex
-3000000000	suma = 20719476736 = 0x4d2fa2000 hex

Tabla 3.3: Resultado bateria ejercicio 5.3



Input list (x8)	Output
1,2	media = 1 resto = 8
-1,-2	media = -1 resto = -8
0x7fffffff,0x7fffffff	media = 2147483647 resto = 0
0x80000000,0x80000000	media = -2147483648 resto = 0
0xffffffff,0xffffffff	media = -1 resto = 0
2000000000,2000000000	media = 2000000000 resto = 0
3000000000,3000000000	media = -1294967296 resto = 0
-2000000000,-2000000000	media = -2000000000 resto = 0
-3000000000,-3000000000	media = -2000000000 resto = 0

Tabla 3.4: Resultado bateria ejercicio 5.4