Practica 2 EC

por: Arturo Cortés Sánchez

Diario de trabajo

Dia 27/9/2018: Finalización de los ejercicios de la primera sesión
Dia 1/10/2018: Comienzo con los primeros ejercicios de la sesión 2
Dia 8/10/2018: Finalización de los ejercicios 5.1 y 5.2
Dia 12/10/2018 Finalización del ejercicio 5.3
Dia 14/10/2018 Finalización de los ejercicios 5.4 y 5.5

Ejercicio 5.1

```
.section .data

#ifndef TEST
#define TEST 5
#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 0x0fffffff, 0x0fffffff, 0x0fffffff, 0x0fffffff
#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:     .int   0,0
  formato:     .asciz    "suma = %lu = 0x%lx hex\n"

.section .text
#_start: .global _start
main: .global  main

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

trabajar:
     mov     $lista, %rbx
     mov  longlista, %ecx
     call suma      # == suma(&lista, longlista);

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

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

     bucle:
          add  (%rbx,%rsi,4), %eax
          jnc  etiqueta
          inc   %edx
          etiqueta:
          inc   %rsi
          cmp   %rsi,%rcx
          jne    bucle
          pop   %rsi

     ret
```

```
imprim_C:           # requiere libC
     mov   $formato, %rdi
     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, %edi
     call _exit          # ==  exit(resultado)
     ret
```

Salida del ejercicio 5.1

```
__TEST01__-----------------------------------
suma = 16 = 0x10 hex
__TEST02__-----------------------------------
suma = 4294967280 = 0xfffffff0 hex
__TEST03__-----------------------------------
suma = 4294967296 = 0x100000000 hex
__TEST04__-----------------------------------
suma = 4294967280 = 0xfffffff0 hex
__TEST05__-----------------------------------
suma = 68719476720 = 0xfffffffff0 hex
__TEST06__-----------------------------------
suma = 3200000000 = 0xbebc2000 hex
__TEST07__-----------------------------------
suma = 4800000000 = 0x11e1a3000 hex
__TEST08__-----------------------------------
suma = 11280523264 = 0x2a05f2000 hex
```

Ejercicio 5.2

```
.section .data

#ifndef TEST
#define TEST 5
#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 0x0fffffff, 0x0fffffff, 0x0fffffff, 0x0fffffff
#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:     .int   0,0
  formato:     .asciz    "suma = %lu = 0x%lx hex\n"

.section .text
#_start: .global _start
main: .global  main
```

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

trabajar:
     mov     $lista, %rbx
     mov  longlista, %ecx
     call suma      # == suma(&lista, longlista);

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

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

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

     ret


imprim_C:          # requiere libC
     mov   $formato, %rdi
     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, %edi
     call _exit          # ==  exit(resultado)
     ret
```

Salida del ejercicio 5.2

```
__TEST01__-----------------------------------
suma = 16 = 0x10 hex
__TEST02__-----------------------------------
suma = 4294967280 = 0xfffffff0 hex
__TEST03__-----------------------------------
suma = 4294967296 = 0x100000000 hex
__TEST04__-----------------------------------
suma = 4294967280 = 0xfffffff0 hex
__TEST05__-----------------------------------
suma = 68719476720 = 0xfffffff0 hex
__TEST06__-----------------------------------
suma = 3200000000 = 0xbebc2000 hex
__TEST07__-----------------------------------
suma = 4800000000 = 0x11e1a3000 hex
__TEST08__-----------------------------------
suma = 11280523264 = 0x2a05f2000 hex
```

Ejercicio 5.3

```
.section .data

#ifndef TEST
#define TEST 3
#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 "Definit TEST entre 1..19"
#endif
.endm

lista:    .irpc i,1234
     linea
     .endr

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

.section .text
#_start: .global _start
main: .global  main
```

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

trabajar:
     mov     $lista, %rbx
     mov  longlista, %ecx
     call suma      # == suma(&lista, longlista);

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

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

     bucle:
          mov  (%rbx,%rsi,4), %ebp
          test %ebp, %ebp
          jns positivo
          add %ebp, %eax
          adc $0xffffffff, %edx
          jmp continua
          positivo:
          add %ebp, %eax
          adc   $0, %edx #
          continua:
          inc   %rsi
          cmp   %rsi,%rcx
          jne    bucle
          pop   %rsi

     ret


imprim_C:           # requiere libC
     mov   $formato, %rdi
```

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

acabar_L:
     mov        $60, %rax
     mov  resultado, %edi
     syscall             # == _exit(resultado)
     ret

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

Salida del ejercicio 5.3

```
__TEST01__-----------------------------------
suma = -16 = 0xfffffffffffffff0 hex
__TEST02__-----------------------------------
suma = 1073741824 = 0x40000000 hex
__TEST03__-----------------------------------
suma = 2147483648 = 0x80000000 hex
__TEST04__-----------------------------------
suma = 4294967296 = 0x100000000 hex
__TEST05__-----------------------------------
suma = 34359738352 = 0x7ffffff0 hex
__TEST06__-----------------------------------
suma = -34359738368 = 0xfffffff800000000 hex
__TEST07__-----------------------------------
suma = -4294967296 = 0xffffffff00000000 hex
__TEST08__-----------------------------------
suma = -2147483648 = 0xffffffff80000000 hex
__TEST09__-----------------------------------
suma = -2147483664 = 0xffffffff7ffffff0 hex
__TEST10__-----------------------------------
suma = 1600000000 = 0x5f5e1000 hex
__TEST11__-----------------------------------
suma = 3200000000 = 0xbebc2000 hex
__TEST12__-----------------------------------
```

```
suma = 4800000000 = 0x11e1a3000 hex
__TEST13__----------------------------------
suma = 32000000000 = 0x773594000 hex
__TEST14__----------------------------------
suma = -20719476736 = 0xfffffffb2d05e000 hex
__TEST15__----------------------------------
suma = -1600000000 = 0xffffffffa0a1f000 hex
__TEST16__----------------------------------
suma = -3200000000 = 0xffffffff4143e000 hex
__TEST17__----------------------------------
suma = -4800000000 = 0xfffffffee1e5d000 hex
__TEST18__----------------------------------
suma = -32000000000 = 0xfffffff88ca6c000 hex
__TEST19__----------------------------------
suma = 20719476736 = 0x4d2fa2000 hex
```

Ejercicio 5.4

```
.section .data

#ifndef TEST
#define TEST 3
#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 0x7fffffff,0x7fffffff,0x7fffffff,0x7fffffff
#elif TEST==6
.int 200000000,200000000,200000000,200000000
#elif TEST==7
.int 300000000,300000000,300000000,300000000
#elif TEST==8
```

```
.int -200000000,-200000000,-200000000,-200000000
#elif TEST==9
.int -300000000,-300000000,-300000000,-300000000
#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 "Definit TEST entre 1..19"
#endif
.endm

lista:    .irpc i,1234
     linea
     .endr
longlista:     .int   (.-lista)/4
resultado:     .int 0
resto:         .int 0
formato:  .ascii "resto=%i \t resultado=%i  \n"

.section .text

main: .global  main

     call trabajar  # subrutina de usuario
     call imprim_C  # printf()  de libC
```

```
    call acabar_C  # exit()    de libC


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

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

    bucle:
        mov  (%rbx,%rsi,4), %ebp
        test %ebp, %ebp
        jns positivo
        add %ebp, %eax
        adc $0xffffffff, %edx
        jmp continua
        positivo:
        add %ebp, %eax
        adc   $0, %edx
        continua:
        inc   %rsi
        cmp   %rsi,%rcx
        jne    bucle
        idiv  %esi
        pop   %rsi

    ret


imprim_C:           # requiere libC
    mov   $formato, %rdi
    mov   resto,%rdx
    mov   resultado,%rsi
```

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

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

Salida del ejercicio 5.4

```
resto=8   resultado=1
__TEST02__-----------------------------------
resto=-8  resultado=-1
__TEST03__-----------------------------------
resto=0   resultado=2147483647
__TEST04__-----------------------------------
resto=0   resultado=-2147483648
__TEST05__-----------------------------------
resto=0   resultado=2147483647
__TEST06__-----------------------------------
resto=0   resultado=200000000
__TEST07__-----------------------------------
resto=0   resultado=300000000
__TEST08__-----------------------------------
resto=0   resultado=-200000000
__TEST09__-----------------------------------
resto=0   resultado=-300000000
__TEST10__-----------------------------------
resto=0   resultado=1
__TEST11__-----------------------------------
resto=4   resultado=1
__TEST12__-----------------------------------
resto=0   resultado=3
__TEST13__-----------------------------------
resto=12    resultado=4
__TEST14__-----------------------------------
resto=0   resultado=5
__TEST15__-----------------------------------
resto=0   resultado=-1
__TEST16__-----------------------------------
resto=-4    resultado=-1
```

```
__TEST17__-----------------------------------
resto=0   resultado=-3
__TEST18__-----------------------------------
resto=-12   resultado=-4
__TEST19__--------------------------
```

Ejercicio 5.5

```
.section .data

#ifndef TEST
#define TEST 3
#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 0x7fffffff,0x7fffffff,0x7fffffff,0x7fffffff
#elif TEST==6
.int 200000000,200000000,200000000,200000000
#elif TEST==7
.int 300000000,300000000,300000000,300000000
#elif TEST==8
.int -200000000,-200000000,-200000000,-200000000
#elif TEST==9
.int -300000000,-300000000,-300000000,-300000000
#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 "Definit TEST entre 1..19"
#endif
.endm

formatq:  .ascii "64-bit: resultado=%i \t resto=%i  \n"
lista:    .irpc i,1234
     linea
     .endr
longlista:     .int   (.-lista)/4
resultado:     .int 0,0
resto:         .int 0,0
formato:  .ascii "resultado=%i \t resto=%i  \n"

.section .text

main: .global  main

     call trabajar  # subrutina de usuario
     mov $formato, %r8
     call imprim_C  # printf()  de libC
     mov  $lista, %rbx
    mov  longlista, %ecx
    call sumaq
    mov  %rax, resultado
    mov  %rdx, resto
     mov $formatq, %r8
     call imprim_C
     call acabar_C  # exit()    de libC
```

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

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

     bucle:
          mov  (%rbx,%rsi,4), %ebp
          test %ebp, %ebp
          jns positivo
          add %ebp, %eax
          adc $0xffffffff, %edx
          jmp continua
          positivo:
          add %ebp, %eax
          adc   $0, %edx
          continua:
          inc   %rsi
          cmp   %rsi,%rcx
          jne    bucle
          idiv  %esi
          pop   %rsi

     ret


imprim_C:           # requiere libC
     mov   %r8, %rdi
     mov   resto,%rdx
     mov   resultado,%rsi
     mov          $0,%eax     # varargin sin xmm
     call  printf        # == printf(formato, res, res);
```

```
     ret

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


sumaq:
     push     %rsi
    mov   $0,%rax
    mov   $0,%rsi

     bucleq:
     movslq  (%rbx,%rsi,4),%rdi
     add     %rdi,%rax
     inc     %rsi
     cmp     %rsi,%rcx
     jne     bucleq
     test    %rax,%rax
     jns     positivoq
     mov     $-1,%rdx
     jmp     continuaq
     positivoq:
     mov     $0,%rdx
     continuaq:
     idiv    %rsi
     pop   %rsi
ret
```

Salida del ejercicio 5.5

```
__TEST01__----------------------------------
resultado=1  resto=8
64-bit: resultado=1  resto=8
__TEST02__----------------------------------
resultado=-1  resto=-8
64-bit: resultado=-1  resto=-8
__TEST03__----------------------------------
resultado=2147483647  resto=0
64-bit: resultado=2147483647  resto=0
```

```
__TEST04__-----------------------------------
resultado=-2147483648   resto=0
64-bit: resultado=-2147483648   resto=0
__TEST05__-----------------------------------
resultado=2147483647  resto=0
64-bit: resultado=2147483647  resto=0
__TEST06__-----------------------------------
resultado=200000000  resto=0
64-bit: resultado=200000000  resto=0
__TEST07__-----------------------------------
resultado=300000000  resto=0
64-bit: resultado=300000000  resto=0
__TEST08__-----------------------------------
resultado=-200000000  resto=0
64-bit: resultado=-200000000  resto=0
__TEST09__-----------------------------------
resultado=-300000000  resto=0
64-bit: resultado=-300000000  resto=0
__TEST10__-----------------------------------
resultado=1  resto=0
64-bit: resultado=1  resto=0
__TEST11__-----------------------------------
resultado=1  resto=4
64-bit: resultado=1  resto=4
__TEST12__-----------------------------------
resultado=3  resto=0
64-bit: resultado=3  resto=0
__TEST13__-----------------------------------
resultado=4  resto=12
64-bit: resultado=4  resto=12
__TEST14__-----------------------------------
resultado=5  resto=0
64-bit: resultado=5  resto=0
__TEST15__-----------------------------------
resultado=-1  resto=0
64-bit: resultado=-1  resto=0
__TEST16__-----------------------------------
resultado=-1  resto=-4
64-bit: resultado=-1  resto=-4
__TEST17__-----------------------------------
resultado=-3  resto=0
64-bit: resultado=-3  resto=0
```

```
__TEST18__-----------------------------------
resultado=-4  resto=-12
64-bit: resultado=-4  resto=-12
__TEST19__-----------------------------------
resultado=-5  resto=0
64-bit: resultado=-5  resto=0
```