José María Gómez García

Práctica 2

Estructura de Computadores 2018/2019

Diario de Trabajo

El domingo 7 de octubre estuve trabajando en casa antes de la sesión de prácticas para ver si tenía dificultades a la hora de enfrentarme a los ejercicios; la verdad es que pude seguir bastante bien el guión, solo me hizo falta buscar el orden de los registros para los atributos de una llamado a función como printf. Avancé bastante, completé la suma de números sin signo y números con signo y me dejé el último punto para trabajarlo en la sesión de prácticas dónde lo terminé.

Códigos fuente

1. **Media1.s**

```
.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 "Definit TEST entre 1..8"

        #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"
                .ascii "\t\t = 0x %08x %08x\n"

.section .text
.global _start

main: .global  main

        call trabajar  # subrutina de usuario
        call imprim    # printf()  de libC
        call acabar    # exit()    de libC
        ret

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

suma:
        mov  $0, %eax  #Lresultado
        mov  $0, %rsi  #Contador
        mov  $0, %edx  #Hresultado

bucle:
        add  (%rbx,%rsi,4), %eax     # %eax = 4 * %rsi + %rbx + %eax
        adc   $0, %edx               # %edx = 0 + %edx + CF
        inc   %rsi                   # i++
        cmp   %rsi,%rcx                      # Si %rsi == al tamaño de la lista
        jne    bucle                 # jump if not equal a bucle
```

```
        ret

imprim:                     # requiere libC
        mov   $formato, %rdi  # registros del estándar de llamadacon argumentos
        mov   resultado,%rsi
        mov   resultado,%rdx
        mov   resultado+4,%rcx
        mov   resultado,%r8
        mov          $0,%eax  # varargin sin xmm
        call  printf          # == printf(formato, res, res, Hres, Lres);
        ret

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

2. Media2.s

```
.section .data

#ifndef TEST
#define TEST 15
#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

    #else
        .error "Definit TEST entre 1..8"

    #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"
        .ascii "\t\t = 0x %08x %08x\n"

.section .text
.global _start

main: .global  main

    call trabajar    # subrutina de usuario
    call imprim    # printf()  de libC
    call acabar    # exit()  de libC
    ret

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

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

suma:
    mov  $0, %eax    # Lresultado
    mov  $0, %rsi    # Contador
    mov  $0, %edx    # Hresultado
bucle:
    mov (%rbx,%rsi,4),%ebp # ebp = Llista[%rsi]
    test %ebp,%ebp
    jns positivo
    add %ebp,%eax
    adc $0xffffffff,%edx
    jmp continua

positivo:
    add  %ebp, %eax
    adc  $0,%edx       # %edx = 0 + %edx + CF

continua:
    inc   %rsi
    cmp   %rsi,%rcx
    jne bucle                  # jump if not equal a bucle
    ret

imprim:                        # requiere libC
    mov   $formato, %rdi    # registros del estándar de llamadacon argumentos
    mov   resultado,%rsi
    mov   resultado,%rdx
    mov   resultado+4,%rcx
    mov   resultado,%r8
    mov        $0,%eax    # varargin sin xmm
    call  printf       # == printf(formato, res, res, Hres, Lres);
    ret

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

3. Media3

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

.section .text
.global _start

main: .global  main

    call trabajar    # subrutina de usuario
    call imprim    # printf()  de libC
    call acabar    # exit()  de libC
    ret

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

suma:
    mov  $0, %eax    # Lmedia
    mov  $0, %rsi    # Contador del bucle
    mov  $0, %edx    # Hmedia

bucle:
    mov (%rbx,%rsi,4),%ebp    # %ebp = Llista[%rsi]
    test %ebp,%ebp     # SF = signo de %ebp
    jns positivo       # Comprobar si es positivo
    add %ebp,%eax
```

```
    adc $0xffffffff,%edx    # %edx = 0xffffffff + %edx + CF
    jmp continua

positivo:
    add  %ebp, %eax
    adc  $0,%edx       # %edx = 0 + %edx + CF

continua:
    inc   %rsi         # i++
    cmp   %rsi,%rcx           # Si %rsi == al tamaño de la lista
    jne bucle          # jump if not equal a bucle
    idiv  %esi         # %edx:%eax/%esi --> Resultado=%eax Resto=%edx
    ret

imprim:                       # requiere libC
    mov   $formato, %rdi    # %rdi primer argumento
    mov   media,%rsi    # %rsi segundo argumento
    mov   resto,%rdx    # %rdx tercer argumento
    mov   media,%rcx    # %rcx cuarto argumento
    mov   resto,%r8           # %r8 quinto argumento
    mov   $0,%eax      # varargin sin xmm
    call  printf       # == printf(formato, media, resto, HexMedia, HexResto);
    ret

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

4. media_5_5.sh

```
.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 "Definit 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\t = 0x %08x \t\t = 0x %08x \n"
formatoq:     .ascii "media \t = %11d \t resto \t = %11d\n"
        .asciz "\t\t = 0x %16x \t\t = 0x %16x \n"
.section .text
.global _start

main: .global  main

    call trabajar    # subrutina de usuario
    call imprim    # printf()  de libC
    call trabajarq
    call imprimq
    call acabar    # exit()  de libC
    ret

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

suma:
    mov  $0, %eax    # Lmedia
    mov  $0, %rsi    # Contador del bucle
    mov  $0, %edx    # Hmedia

bucle:
    mov (%rbx,%rsi,4),%ebp    # %ebp = Llista[%rsi]
    test %ebp,%ebp     # SF = signo de %ebp
    jns positivo       # Comprobar si es positivo
    add %ebp,%eax
    adc $0xffffffff,%edx    # %edx = 0xffffffff + %edx + CF
    jmp continua

positivo:
    add  %ebp, %eax
    adc  $0,%edx       # %edx = 0 + %edx + CF

continua:
    inc   %rsi         # i++
    cmp   %rsi,%rcx           # Si %rsi == al tamaño de la lista
```

```
    jne bucle           # jump if not equal a bucle
    idiv  %esi          # %edx:%eax/%esi --> Resultado=%eax Resto=%edx
    ret

trabajarq:
    mov    $lista, %rbx
    mov  longlista, %ecx
    call sumaq
    mov  %rax, media
    mov  %rdx, resto
    ret

sumaq:
    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
    ret

imprim:                       # requiere libC
    mov   $formato, %rdi    # %rdi primer argumento
    mov   media,%rsi    # %rsi segundo argumento
    mov   resto,%rdx    # %rdx tercer argumento
    mov   media,%rcx    # %rcx cuarto argumento
    mov   resto,%r8           # %r8 quinto argumento
    mov   $0,%eax      # varargin sin xmm
    call  printf       # == printf(formato, media, resto, HexMedia, HexResto);
    ret

imprimq:                      # requiere libC
    mov   $formato, %rdi    # %rdi primer argumento
    mov   media,%rsi    # %rsi segundo argumento
    mov   resto,%rdx    # %rdx tercer argumento
    mov   media,%rcx    # %rcx cuarto argumento
    mov   resto,%r8           # %r8 quinto argumento
    mov   $0,%eax      # varargin sin xmm
```

```
    call  printf       # == printf(formato, media, resto, HexMedia, HexResto);
    ret

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

5. **script.sh**

```
#!/bin/bash

echo "Media1"
for i in $(seq 1 8); do
rm media1
gcc -x assembler-with-cpp -D TEST=$i -no-pie media1.s -o media1
echo -n "T#$i "; ./media1
done

echo "Media2"
for i in $(seq 1 14); do
rm media2
gcc -x assembler-with-cpp -D TEST=$i -no-pie media2.s -o media2
echo -n "T#$i "; ./media2
done

echo "Media3"
for i in $(seq 1 19); do
rm media3
gcc -x assembler-with-cpp -D TEST=$i -no-pie media3.s -o media3
echo -n "T#$i "; ./media3
done

echo "Media_5_5"
for i in $(seq 1 19); do
rm media_5_5
gcc -x assembler-with-cpp -D TEST=$i -no-pie media_5_5.s -o media_5_5
echo -n "T#$i "; ./media_5_5
done
```

Pruebas de Ejecución

Media1.s

T#1 resultado = 16 (sgn)

= 0x 10 (hex)

= 0x 00000000 00000010

T#2 resultado = 4294967280 (sgn)

= 0x fffffff0 (hex)

= 0x 00000000 fffffff0

T#3 resultado = 4294967296 (sgn)

= 0x 100000000 (hex)

= 0x 00000001 00000000

T#4 resultado = 68719476720 (sgn)

= 0x fffffff0 (hex)

= 0x 0000000f fffffff0

T#5 resultado = 68719476720 (sgn)

= 0x fffffff0 (hex)

= 0x 0000000f fffffff0

T#6 resultado = 3200000000 (sgn)

= 0x bebc2000 (hex)

= 0x 00000000 bebc2000

T#7 resultado = 4800000000 (sgn)

= 0x 11e1a3000 (hex)

= 0x 00000001 1e1a3000

T#8 resultado = 11280523264 (sgn)

= 0x 2a05f2000 (hex)

= 0x 00000002 a05f2000

Media2.s

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

Media3.s

T#1 media = -1 resto = 0

= 0x ffffffff = 0x 00000000

T#2 media = 67108864 resto = 0

= 0x 04000000 = 0x 00000000

T#3 media = 134217728 resto = 0

= 0x 08000000 = 0x 00000000

T#4 media = 268435456 resto = 0

= 0x 10000000 = 0x 00000000

T#5 media = 2147483647 resto = 0

= 0x 7fffffff = 0x 00000000

T#6 media = -2147483648 resto = 0

= 0x 80000000 = 0x 00000000

T#7 media = -268435456 resto = 0

= 0x f0000000 = 0x 00000000

T#8 media = -134217728 resto = 0

= 0x f8000000 = 0x 00000000

T#9 media = -134217729 resto = 0

= 0x f7ffffff = 0x 00000000

T#10 media = 100000000 resto = 0

= 0x 05f5e100 = 0x 00000000

T#11 media = 200000000 resto = 0

= 0x 0bebc200 = 0x 00000000

T#12 media = 300000000 resto = 0

= 0x 11e1a300 = 0x 00000000

T#13 media = 2000000000 resto = 0

= 0x 77359400 = 0x 00000000

T#14 media = -1294967296 resto = 0

= 0x b2d05e00 = 0x 00000000

T#15 media = -100000000 resto = 0

= 0x fa0a1f00 = 0x 00000000

T#16 media = -200000000 resto = 0

= 0x f4143e00 = 0x 00000000

T#17 media = -300000000 resto = 0

= 0x ee1e5d00 = 0x 00000000

T#18 media = -2000000000 resto = 0

= 0x 88ca6c00 = 0x 00000000

T#19 media = 1294967296 resto = 0

= 0x 4d2fa200 = 0x 00000000

Media_5_5.s

T#1 media = -1 resto = 0

= 0x ffffffff = 0x 00000000

media = -1 resto = 0

= 0x ffffffff = 0x 00000000

T#2 media = 67108864 resto = 0

= 0x 04000000 = 0x 00000000

media = 67108864 resto = 0

= 0x 04000000 = 0x 00000000

T#3 media = 134217728 resto = 0

= 0x 08000000 = 0x 00000000

media = 134217728 resto = 0

= 0x 08000000 = 0x 00000000

T#4 media = 268435456 resto = 0

= 0x 10000000 = 0x 00000000

media = 268435456 resto = 0

= 0x 10000000 = 0x 00000000

T#5 media = 2147483647 resto = 0

= 0x 7fffffff = 0x 00000000

media = 2147483647 resto = 0

= 0x 7fffffff = 0x 00000000

T#6 media = -2147483648 resto = 0

= 0x 80000000 = 0x 00000000

media = -2147483648 resto = 0

= 0x 80000000 = 0x 00000000

T#7 media = -268435456 resto = 0

= 0x f0000000 = 0x 00000000

media = -268435456 resto = 0

= 0x f0000000 = 0x 00000000

T#8 media = -134217728 resto = 0

= 0x f8000000 = 0x 00000000

media = -134217728 resto = 0

= 0x f8000000 = 0x 00000000

T#9 media = -134217729 resto = 0

= 0x f7ffffff = 0x 00000000

media = -134217729 resto = 0

= 0x f7ffffff = 0x 00000000

T#10 media = 100000000 resto = 0

= 0x 05f5e100 = 0x 00000000

media = 100000000 resto = 0

= 0x 05f5e100 = 0x 00000000

T#11 media = 200000000 resto = 0

= 0x 0bebc200 = 0x 00000000

media = 200000000 resto = 0

= 0x 0bebc200 = 0x 00000000

T#12 media = 300000000 resto = 0

= 0x 11e1a300 = 0x 00000000

media = 300000000 resto = 0

= 0x 11e1a300 = 0x 00000000

T#13 media = 2000000000 resto = 0

= 0x 77359400 = 0x 00000000

media = 2000000000 resto = 0

= 0x 77359400 = 0x 00000000

T#14 media = -1294967296 resto = 0

= 0x b2d05e00 = 0x 00000000

media = -1294967296 resto = 0

= 0x b2d05e00 = 0x 00000000

T#15 media = -100000000 resto = 0

= 0x fa0a1f00 = 0x 00000000

media = -100000000 resto = 0

= 0x fa0a1f00 = 0x 00000000

T#16 media = -200000000 resto = 0

= 0x f4143e00 = 0x 00000000

media = -200000000 resto = 0

= 0x f4143e00 = 0x 00000000

T#17 media = -300000000 resto = 0

= 0x ee1e5d00 = 0x 00000000

media = -300000000 resto = 0

= 0x ee1e5d00 = 0x 00000000

T#18 media = -2000000000 resto = 0

= 0x 88ca6c00 = 0x 00000000

media = -2000000000 resto = 0

= 0x 88ca6c00 = 0x 00000000

media_5_5.s: Mensajes del ensamblador:

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

media_5_5.s:74: Aviso: el valor 0xffffffff4d2fa200 se truncó a 0x4d2fa200

T#19 media = 1294967296 resto = 0

= 0x 4d2fa200 = 0x 00000000

media = 1294967296 resto = 0

= 0x 4d2fa200 = 0x 00000000