Practica 2

Sección de ejercicios:

5.1 Sumar N enteros SIN signo de 32 bits en una plataforma de 32 bits sin perder precisión

En clase se nos planteaban dos soluciones la primera sin usar adc:

```
.section .data
lista:          .int 1,2,10,  1,2,0b10,  1,2,0x10
longlista:      .int (.-lista)/4
resultado:      .int -1
formato:        .ascii "suma = %u = %x hex\n\0"

.section .text
#_start:        .global _start
main: .global main
        mov $lista, %ebx        #mueve la dirección de memoria de lista al registro ebx
        mov longlista, %ecx     #mueve la longitud (el valor) al registro ecx
        call suma               #llama a suma
        mov %eax, resultado

        push resultado
        push resultado
        push $formato
        call printf
        add $12,%esp
        mov $1, %eax            #eax = 1
        mov $0, %ebx            #ebx = 0
        int $0x80


suma:
        push %edx#guarda el valor del registro edx
        mov $0, %eax            #pone a 0 el registro eax
        mov $0, %edx            #pone a 0 el regristo edx
bucle:
        add (%ebx,%edx,4), %eax #edx*4 + ebx --> eax += lista
        jnc finalbucle          #FORMA DE COMPILADOR saltar si no hay acarreo al final del
bucle
        add $1,%esi             #FORMA DE COMPILADOR

finalbucle:
        inc %edx                #edx += 1
```

```
        cmp %edx,%ecx           #COMPARA edx == ecx y pone 0 o 1 en el flags de comp
        jne bucle               #JUMP NOT EQUAL (es decir , salta si el flags no es 0) a la direccion
                                #de memoria de blucle:

        pop %edx                #recupera el valor de edx
        ret                     #return resultado
```

Y el segundo haciendo uso de él:

```
.section .data
lista:          .int 1,2,10,  1,2,0b10,  1,2,0x10
longlista:      .int (.-lista)/4
resultado:      .int -1
formato:        .ascii "suma = %u = %x hex\n\0"

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

        mov $lista, %ebx        #mueve la dirección de memoria de lista al registro ebx
        mov longlista, %ecx     #mueve la longitud (el valor) al registro ecx
        call suma               #llama a suma
        mov %eax, resultado
        push resultado
        push resultado
        push $formato
        call printf
        add $12,%esp
        mov $1, %eax            #eax = 1
        mov $0, %ebx            #ebx = 0
        int $0x80


suma:
        push %edx#guarda el valor del registro edx
        mov $0, %eax            #pone a 0 el registro eax
        mov $0, %edx            #pone a 0 el regristo edx
bucle:
        add (%ebx,%edx,4), %eax #edx*4 + ebx --> eax += lista
        adc $0,%esi             #suma operando1 + acarreo , pero realmente hace op1+op2+acarreo

finalbucle:
        inc %edx                #edx += 1
        cmp %edx,%ecx           #COMPARA edx == ecx y pone 0 o 1 en el flags de comp
        jne bucle               #JUMP NOT EQUAL (es decir , salta si el flags no es 0) a la direccion
                                #de memoria de blucle:

        pop %edx                #recupera el valor de edx
        ret                     #return resultado
```

BATERIA DE PRUEBAS 5.1:

En ambos casos se muestra el resultado idéntico, procedemos a mostrar los resultados de la bateria de datos que se nos indican utilizando:

```
.macro linea
		#.int 1,1,1,1
		#.int 2,2,2,2
		#.int 1,2,3,4
		#.int -1,-1,-1,-1
		#.int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
		#.int 0x08000000,0x08000000,0x08000000,0x08000000
		#.int 0x10000000,0x20000000,0x30000000,0x40000000
	.endm
lista: .irpc i,12345678
		linea
	.endr
```

·Todos 1: suma = 32 = 20 hex Suma normal
·Todos 2: suma = 64 = 40 hex Suma normal
·4 veces 1,2,3,4: suma = 80 = 50 hex Suma normal
·Todos -1: suma = 4294967264 = fffffe0 hex Por supuesto no entiende de números sin signo
·Todos 0x08000000 suma = 0 = 0 hex No hay acarreo por tanto la suma no se muestra
·Todos 0x10000000,0x20000000,0x30000000,0x40000000: suma = 0 = 0 hex Mismo de antes

5.2 Sumar N enteros con signo de 32 bits en una plataforma de 32 bits sin perder precisión

En este caso se nos comunica utilizar cltd para extender el signo, de tal manera que:

```
.section .data
lista:		.int 1,1,1,1,1,1,1,2
#.int 1,2,10,  1,2,0b10,  1,2,0x10
longlista:	.int (.-lista)/4
resultado:	.quad -1
formato:	.ascii "suma = %lld = %llx hex\n\0"


.section .text
#_start:	.global _start
main: .global main
	mov $lista, %ebx	#mueve la dirección de memoria de lista al registro ebx
	mov longlista, %ecx	#mueve la longitud (el valor) al registro ecx
	call suma		#llama a suma
	mov %edi, resultado
	mov %ebp, resultado+4
	push resultado+4		#metemos el resultado 2 veces
	push resultado
	push resultado+4
	push resultado
```

```
        push $formato       #metemos el formato
        call printf         #pintamos
        mov $20 ,%esp               #dejamos el puntero a pila donde estaba

        mov $1, %eax        #eax = 1
        mov $0, %ebx        #ebx = 0
        int $0x80

        #eax -> resultado final(menos significativos)
        #ebx -> lista
        #ecx -> longitudlista
        #edx -> resultado final(más significativos)
        #edi -> resultado parcia(menos significativos)
        #ebp -> resultado parcial(mas significativo)
        #esi -> iterador

suma:
        push %edx           #guarda el valor del registro edx
        mov $0, %eax        #pone a 0 el registro eax
        mov $0, %edx        #pone a 0 el regristo edx
        mov $0, %edi        #pone a 0 el registro edi
        mov $0, %esi
bucle:
        mov (%ebx,%esi,4),%eax  #cargamos en eax el valor de la lista
        cltd                #primero extendemos y luego sumamos
        add %eax,%edi               #edx*4 + ebx --> edi += lista #sumamos
        adc %edx,%ebp               # suma con el acarreo justo de antes
        jmp finalbucle

finalbucle:
        inc %esi            #esi += 1
        cmp %esi,%ecx               #COMPARA edi == ecx y pone 0 o 1 en el flags de comp
        jne bucle           #JUMP NOT EQUAL (es decir , salta si el flags no es 0) a la direccion
                            #de memoria de blucle:

        pop %edx            #recupera el valor de edx
        ret                 #return resultado
```

BATERIA DE PRUEBAS 5.2:

Utilizando la misma macro observamos que:

```
.macro linea
		#.int 1,1,1,1
		#.int 2,2,2,2
		#.int 1,2,3,4
		#.int -1,-1,-1,-1
		#.int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
		#.int 0x08000000,0x08000000,0x08000000,0x08000000
		#.int 0x10000000,0x20000000,0x30000000,0x40000000

		#.int 1,-2,1,-2
		#.int 1,2,-3,-4
		#.int 0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF
		#.int 0x80000000,0x80000000,0x80000000,0x80000000
		#.int 0x04000000,0x04000000,0x04000000,0x04000000
		#.int 0xFC000000,0xFC000000,0xFC000000,0xFC000000
		#.int 0xF8000000,0xF8000000,0xF8000000,0xF8000000
		#.int 0xF0000000,0xE0000000,0xD0000000,0xF0000000
	.endm
lista: .irpc i,12345678
		linea
	.endr
```

·Todos 1: suma = 32 = 20 hex Suma normal
·Todos 2: suma = 64 = 40 hex Suma normal
·4 veces 1,2,3,4: suma = 80 = 50 hex Suma normal
·Todos -1: suma = -32 = ffffffffffffffe0 hex Esta vez podemos observar que si realiza la suma
·Todos 0x08000000 suma = 4294967296 = 100000000 hex Suma bien realizada
·Todos 0x10000000,0x20000000,0x30000000,0x40000000: suma = 21474836480 = 500000000 hex
Suma bien realizada
·Todos 1,-2,1,-2: suma = -16 = fffffffffffffff0 hex
·Todos 1,2,-3,-4: suma = -32 = ffffffffffffffe0 hex
·Todos 0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF,0x7FFFFFFF: suma = 68719476704 = fffffffe0 hex
·Todos 0x80000000,0x80000000,0x80000000,0x80000000: suma = -68719476736 =
fffffff000000000 hex
·Todos 0x04000000,0x04000000,0x04000000,0x04000000: suma = 2147483648 = 80000000 hex
·Todos 0xFC000000,0xFC000000,0xFC000000,0xFC000000: suma = -2147483648 =
ffffffff80000000 hex
·Todos 0xF8000000,0xF8000000,0xF8000000,0xF8000000: suma = -4294967296 =
ffffffff00000000 hex
·Todos 0xF0000000,0xE0000000,0xD0000000,0xF0000000: suma = -15032385536 =
fffffffc80000000 hex

5.3 Media de N enteros con signo de 32 bits en una plataforma de 32 bits sin perder precisión

En este ejercicio calcularemos la media utilizando idiv puesto que son números con signo tal que:

```
.section .data
```

```
        .macro linea
                #.int 1,1,1,1
              #.int 2,2,2,2
                #.int 1,2,3,4
                .int -1,-1,-1,-1
                #.int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
                #.int 0x08000000,0x08000000,0x08000000,0x08000000
                #.int 0x10000000,0x20000000,0x30000000,0x40000000
        .endm
lista: .irpc i,12345678
                linea
        .endr
longlista:      .int (.-lista)/4
cociente:       .int -1
resto:          .int -1
formato:        .ascii "media:\n\t cociente = %8d = 0x%08x = hex\n\t resto = %8d = 0x%08x =
hex\n\0"


.section .text
#_start:        .global _start
main: .global main
        mov $lista, %ebx        #mueve la dirección de memoria de lista al registro ebx
        mov longlista, %ecx     #mueve la longitud (el valor) al registro ecx
        call suma               #llama a suma



        push resto
        push resto
        push cociente
        push cociente
        push $formato           #metemos el formato
        call printf             #pintamos
        mov $20 ,%esp                   #dejamos el puntero a pila donde estaba

        mov $1, %eax            #eax = 1
        mov $0, %ebx            #ebx = 0
        int $0x80

        #eax -> resultado final(menos significativos)
        #ebx -> lista
        #ecx -> longitudlista
        #edx -> resultado final(más significativos)
        #edi -> resultado parcia(menos significativos)
        #ebp -> resultado parcial(mas significativo)
        #esi -> iterador

suma:
        push %edx               #guarda el valor del registro edx
        mov $0, %eax            #pone a 0 el registro eax
        mov $0, %edx            #pone a 0 el regristo edx
        mov $0, %edi            #pone a 0 el registro edi
```

```
        mov $0, %esi            #pone a 0 el registro esi
bucle:
        mov (%ebx,%esi,4),%eax  #cargamos en eax el valor de la lista
        cltd                    #primero extendemos y luego sumamos
        add %eax,%edi                   #edx*4 + ebx --> edi += lista #sumamos
        adc %edx,%ebp                   # suma con el acarreo justo de antes
        jmp finalbucle

finalbucle:
        inc %esi                #esi += 1
        cmp %esi,%ecx                   #COMPARA edi == ecx y pone 0 o 1 en el flags de comp
        jne bucle               #JUMP NOT EQUAL (es decir , salta si el flags no es 0) a la direccion
                                #de memoria de blucle:
        mov %edi,%eax           #guardamos el resultado en %eax puesto que idiv así lo indica
        idiv %ecx
        mov %eax, cociente
        mov %edx, resto
        pop %edx
        ret                     #return resultado
```

BATERIA DE PRUEBAS 5.3:

Utilizando la misma macro observamos que:

```
.macro linea
                #.int 1,1,1,1
                #.int 2,2,2,2
                #.int 1,2,3,4
                #.int 0x10000000,0x20000000,0x30000000,0x40000000

                #.int 1,-2,1,-2
                #.int 1,2,-3,-4
                #.int 0x7FFFFFFF
                #.int 0x80000000
                #.int 0xF0000000,0xE0000000,0xE0000000,0xD0000000
                #.int -1,-1,-1,-1
                #.int 0,-1,-1,-1
                #.int 0,-2,-1,-1
                #.int 1,-2,-1,-1
                #.int 63,-2,-1,-1


        .endm
lista: .irpc i,12345678
                linea
        .endr
```

·Todos 1: media:

cociente = 1 = 0x00000001 = hex

resto = 0 = 0x00000000 = hex

·Todos 2: media:

cociente = 2 = 0x00000002 = hex

resto = 0 = 0x00000000 = hex

·4 veces 1,2,3,4: media:

cociente = 2 = 0x00000002 = hex

resto = 16 = 0x00000010 = hex

·Todos 0x10000000,0x20000000,0x30000000,0x40000000:

media:

cociente = 0 = 0x00000000 = hex

resto = 0 = 0x00000000 = hex

·Todos 1,-2,1,-2 media:

cociente = 0 = 0x00000000 = hex

resto = -16 = 0xfffffff0 = hex

·Todos 1,2,-3,-4 media:

cociente = -1 = 0xffffffff = hex

resto = 0 = 0x00000000 = hex

·Todos 0x7FFFFFFF media:

cociente = 536870911 = 0x1fffffff = hex

resto = 0 = 0x00000000 = hex

·Todos 0x80000000 media:

cociente = -536870912 = 0xe0000000 = hex

resto = 0 = 0x00000000 = hex

·Todos 0xF0000000,0xE0000000,0xE0000000,0xD0000000 media:

cociente = -134217728 = 0xf8000000 = hex

resto = 0 = 0x00000000 = hex

·Todos -1,-1,-1,-1 media:

cociente = -1 = 0xffffffff = hex

resto = 0 = 0x00000000 = hex

·Todos 0,-1,-1,-1 media:

cociente = 0 = 0x00000000 = hex

resto = -24 = 0xffffffe8 = hex

·Todos 0,-2,-1,-1 media:

cociente = -1 = 0xffffffff = hex

resto = 0 = 0x00000000 = hex

·Todos 1,-2,-1,-1 media:

cociente = 0 = 0x00000000 = hex

resto = -24 = 0xffffffe8 = hex

·Todos 63,-2,-1,-1 media:

cociente = -134217713 = 0xf800000f = hex

resto = -8 = 0xfffffff8 = hex