Universidad De Granada

Grado ingeniería informática (2017 – 2018)

Estructura De Los Computadores

Practica 2: Suma sin signo, con y media.

Trabajo realizado por Antonio Miguel Morillo Chica

---

1. Memoria de la practica.

- [12/10/2017] : He hecho la parte de depuración de saludo y todo lo concerniente a p1 y p2.c.
- [19/10/2017] : He acabado el suma sin signo y casi he acabado la suma con con signo. Por la tarde he creado el odt para los ejercicios y las baterías de pruebas de suma con/sin signo. Además hemirado por encima como hacer la media (div).
- [20/10/2017] : He comentado el código de la suma con signo y sin signo. Además he hecho 2 ejecicios de la media.
- [22/10/2017] : He terminado la media y he hecho la batería de pruebas, los ejercicios de media y he acabado los ejercicios de saludo.

2. Códigos de los programas.

2.1. Suma sin signo.

```
.section .data

# Batería de pruebas
.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 0x10000000,0x20000000,0x40000000,0x80000000
                        .int 0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF
.endm

lista:                  .irpc i, 12345678
                        linea
.endr

longlista:              .int (.-lista)/4      # Tamañao de la lista

resultado:              .quad 0x0123456789ABCDEF # Lugar donde vamos a guardar
resultado

formato: .ascii "suma = %llu = %llx hex\n\0" # formato para 64bits
```

```
.section .text
main: .global main

                          mov    $lista, %ebx        # Ajuste del marco de pila
                          mov longlista, %ecx        #
                          call suma                  # Llamo a suma

                          mov %eax, resultado        # Guardo eax
                          mov %edx, resultado+4      # Guardo arriba edx

                          push resultado+4
                          push resultado
                          push resultado+4
                          push resultado
                          push $formato
                          call printf
                          add $20, %esp              # Muevo el puntero

                          mov $1, %eax               # Como retornamos la
finalización
                          mov $0, %ebx               # del programa
                          int $0x80                  #

suma:
                          mov $0, %eax                # Pone a 0 la suma
                          mov $0, %edx                # Pone a 0 indice
                          mov $0, %esi                # Pone a 0 el acumul (carry)
                                 #ecx                  # tam del vector

bucle:
                          add (%ebx,%esi,4), %eax   # sumo
                          jnc no_carry              # miro el flag y salto si no
hay accareo
                          inc %edx                  # si hay cary incremento en
edx

no_carry:
        inc %esi                    # Incremento el contador
        cmp %esi   ,%ecx           # Comparo indice con el tam del vector
        jne bucle                  # en caso afir sigon (otra vuelta)
ret
```

2.2. Suma con signo.

```
.section .data

# Batería de pruebas
.macro linea
    #       .int -1,-1,-1,-1
  #    .int  1,-2,1,-2
  #    .int  1,2,-3,-4
       .int 0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF
  #    .int 0x7fffffff
  #    .int 0x80000000,0x80000000,0x80000000,0x80000000
  #    .int 0x04000000,0x04000000,0x04000000,0x04000000
  #    .int 0x08000000,0x08000000,0x08000000,0x08000000
  #    .int 0xfc000000,0xfc000000,0xfc000000,0xfc000000
  #    .int 0xf8000000,0xf8000000,0xf8000000,0xf8000000
  #  .int 0xf0000000,0xe0000000,0xe0000000,0xd0000000
.endm

lista: .irpc i, 12345678
       linea
.endr

longlista:    .int (.-lista)/4    # Tamañao de la lista

resultado:    .quad 0x0123456789ABCDEF # Lugar donde vamos a guardar resultado

formato: .ascii "suma = %llu = %llx hex\n\0" # formato para 64bits

.section .text
main: .global main

  mov    $lista, %ebx       #
  mov longlista, %ecx       #
  call suma                 # Llamo a suma

  mov %eax, resultado       # Guardo eax
  mov %edx, resultado+4     # Guardo arriba edx

  push resultado+4
  push resultado
  push resultado+4
  push resultado
  push $formato
  call printf
  add $20, %esp             # Muevo el puntero

  mov $1, %eax              # Como retornamos la finalización
  mov $0, %ebx              # del programa
  int $0x80                 #
```

```
suma:
  mov $0, %edi                  # pone a 0 la suma.
  mov $0, %esi                  # pone a 0 segunda palabra edi:esi
  mov $0, %ebp                  # pone a 0 el indice
  #       %ecx                  # tam del vector

bucle:
  mov (%ebx,%ebp,4), %eax   # leo no sumo
  cltd                      # Esta instrucción extiende un número con signo de
                            # 32 bits a uno de 64 bits duplicando el signo
  add %eax, %esi            # Sumo la parte menos significativa
  adc %edx, %edi            # Sumo la parte más significativa

  inc %ebp                  # Incremento cnt
  cmp %ebp, %ecx            # Comparo la desigualdad del bucle indice != tam
  jne bucle                 # Vuelvo a ejecutaar el bucle

  mov %edi, %edx            # En caso contrario he terminado salvaguardo los
  mov %esi, %eax            # resultados en este tipo de registro: EDX:EAX

ret
```

2.3 Media.

```
.section .data
.macro linea
#       .int 1,-2,1,-2
#       .int 1,2,-3,-4
#       .int 0x7fffffff, 0x7fffffff, 0x7fffffff, 0x7fffffff
        .int 0x80000000, 0x80000000, 0x80000000, 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 -2,-1,-1,-1
#       .int 2,-2,-1,-1
#       .int 3,-2,-1,-1
#       .int 32,-2,-1,-1
#       .int 33,-2,-1,-1
#       .int 34,-2,-1,-1
#       .int 35,-2,-1,-1
#       .int 63,-2,-1,-1
#       .int 64,-2,-1,-1
#       .int 95,-2,-1,-1
#       .int -31,-2,-1,-1
        .endm

lista:
        .irpc i,12345678
```

```
                    linea
       .endr

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

formato:
       .ascii "media = %d = %x hex\n"
       .ascii "resto = %d = %x hex\n\0"

.section .text
main:  .global main

       mov    $lista, %ebx
       mov longlista, %ecx
       call suma
       mov %eax, media
       mov %edx, resto

       push resto #parte mas significativa del primer
       push resto
       push media
       push media
       push $formato
       call printf
       add $20, %esp             # Muevo el puntero

       mov $1, %eax              # Como retornamos la finalización
       mov $0, %ebx              # del programa
       int $0x80                 #

suma:
       mov $0, %edi             # pone a 0 la suma.
       mov $0, %esi             # pone a 0 segunda palabra edi:esi
       mov $0, %ebp             # pone a 0 el indice
       #       %ecx              # tam del vector

bucle:
       mov (%ebx,%ebp,4), %eax   # leo no sumo
       cltd                      # Esta instrucción extiende un número con signo
de

                                 # 32 bits a uno de 64 bits duplicando el signo
       add %eax, %esi            # Sumo la parte menos significativa
       adc %edx, %edi            # Sumo la parte más significativa

       inc %ebp                  # Incremento cnt
       cmp %ebp, %ecx            # Comparo la desigualdad del bucle indice != tam
       jne bucle                 # Vuelvo a ejecutaar el bucle

       mov %edi, %edx            # En caso contrario he terminado salvaguardo los
```

```
      mov %esi, %eax            # resultados en este tipo de registro: EDX:EAX

      idiv %ecx

ret
```