Primera práctica

Estructura de computadores

Escuela Técnica Superior de Ingeniería Informática de Granada (ETSIIT)

Juan Pablo Sáez Gutiérrez
2º Ingeniería Informática (1º Cuatrimestre)
Granada - 14 de octubre de 2018

Índice general

Capítulo 1

Suma de N enteros de 32 bits unsigned

1.1. Ejercicio 5.1 Suma de enteros unsigned sobre dos registros usando uno de ellos como acumulador de acarreos

```
.section .text

_start:

        .global _start

        mov     $lista, %rbx
        mov  longlista, %r10
        call suma
        #Little endian------------------------------------------------
        mov  %eax, resultado   #Posiciones de memoria anteriores contienen
        mov         %r9d, resultado+4 #bytes menos significativos

        #Salida por stdout mediante printf--------------------------------
         mov   $formato, %rdi           #formato para printf
         mov   resultado,%rsi           #Parametro para %lld
         mov   resultado,%rdx           #Parametro para %llx
         mov          resultado+4,%ecx  #Parametro para %08x
         mov   resultado,%r8d           #Parametro para %08x
```

```
        mov          $0,%eax
        call  printf

        mov  resultado, %edi
        call _exit
        ret

        #Inicializacion variables para suma--------------------------------
suma:
        mov  $0, %eax
        mov         $0, %r9d
        mov  $0, %rcx

bucle:
        add  (%rbx,%rcx,4), %eax
        jnc         salto           #Acarreo en r9d implementado
        add          $1,%r9d        #mediante jnc. Si no CF, se
salto:                              #evita incrementar %r9d.
        inc   %rcx                  #Implementacion "artesanal"
        cmp   %rcx,%r10             #de instrucción adc.
        jne    bucle

        ret
```

1.2. Ejercicio 5.2 Suma de enteros unsigned sobre dos registros sobre dos registros mediante extensión de ceros

```
.section .text

_start:

        .global _start

        mov     $lista, %rbx
        mov  longlista, %r10
        call suma
        #Little endian-----------------------------------------------------
        mov  %eax, resultado #Posiciones de memoria anteriores contienen
        mov         %r9d, resultado+4 #bytes menos significativos

        #Salida por stdout mediante printf---------------------------------
         mov   $formato, %rdi           #formato para printf
         mov   resultado,%rsi           #Parametro para %lld
         mov   resultado,%rdx           #Parametro para %llx
         mov          resultado+4,%ecx  #Parametro para %08x
         mov   resultado,%r8d           #Parametro para %08x

         mov          $0,%eax
         call  printf

         mov  resultado, %edi
         call _exit
         ret

         #Inicializacion variables para suma--------------------------------
suma:
        mov  $0, %eax
        mov         $0, %r9d
        mov  $0, %rcx

bucle:
        add  (%rbx,%rcx,4), %eax
```

```
adc    $0,%r9d          #Si CF==1, %r9d=0+CF
inc    %rcx               #esta instruccion implementa
cmp    %rcx,%r10          #el acarreo.
jne     bucle

ret
```

1.3. Batería de test suma de enteros unsigned (ambas implementaciones)

Test	Valor	resultado u 64b	hex 64 bits	hex 2*32bits
1	1, ...	16	0x10	0x 00000000 00000010
2	0x0FFF FFFF, ...	4294967280	0xfffffff0	0x 00000000 fffffff0
3	0x1000 0000, ...	0x100000000	0x100000000	0x 00000001 00000000
4	0xFFFF FFFF, ...	68719476720	0xfffffff0	0x 0000000f fffffff0
5	-1, ...	68719476720	0xfffffff0	0x 0000000f fffffff0
6	200 000 000, ...	3200000000	0xbebc2000	0x 00000000 bebc2000
7	300 000 000, ...	4800000000	0x11e1a3000	0x 00000001 1e1a3000
8	5 000 000 000, ...	11280523264	0x2a05f2000	0x 00000002 a05f2000
9	T8+T5+16	80000000000	0x12a05f2000	0x 00000012 a05f2000

Cuadro 1.1: Valores suma sin signo usando registro como acumulador de acarreos y mediante extensión de ceros

Capítulo 2

Suma de N enteros de 32 bits signed

2.1. Ejercicio 5.3 Suma de enteros signed de 32 bits, sobre dos registros de 32 bits mediante extensión de signo

```
.section .text

_start:

        .global _start

        mov     $lista, %rbx
        mov  longlista, %r10
        call suma
        #Little endian-----------------------------------------------------
        mov  %esi, resultado  #Posiciones de memoria anteriores contienen
        mov         %edi, resultado+4#bytes menos significativos

        #Salida por stdout mediante printf---------------------------------
         mov   $formato, %rdi           #formato para printf
         mov   resultado,%rsi           #Parametro para %lld
         mov   resultado,%rdx           #Parametro para %llx
         mov          resultado+4,%ecx  #Parametro para %08x
         mov   resultado,%r8d           #Parametro para %08x
```

```
        mov          $0,%eax
        call  printf

        mov  resultado, %edi
        call _exit
        ret
       #Inicializacion variables para suma--------------------------------
suma:
       mov  $0, %eax
       mov  $0, %edx
       mov  $0, %rcx
       mov  $0, %esi
       mov  $0, %edi

bucle:
       mov (%rbx,%rcx,4), %eax #Sustituido add por mov. Se acumula en %esi
       cdq          #Sign extension de dw a quad. 32-->64, eax--> edx:eax
                    #Es la instrucción relevante del ejercicio.

       add           %eax,%esi
       adc           %edx,%edi #%edi acumula el acarreo signed.

       inc   %rcx
       cmp   %rcx,%r10
       jne    bucle

       ret
```

2.2. Batería de test suma de enteros signed

Test	Valor	resultado u 64b	hex 64 bits	hex 2*32bits
1	-1, ...	-16	0xfffffffffffffff0	0x ffffffff fffffff0
2	0x0400 0000, ...	1073741824	0x40000000	0x 00000000 40000000
3	0x0800 0000, ...	2147483648	0x8000000	0x 00000000 80000000
4	0x1000 0000, ...	4294967296	0x100000000	0x 00000001 00000000
5	0x7FFF FFFF, ...	34359738352	0x7fffffff	0x 00000007 fffffff0
6	0x8000 0000, ...	-34359738368	0xfffffff800000000	0x fffffff8 00000000
7	0xF000 0000, ...	-4294967296	0xffffffff00000000	0x ffffffff 00000000
8	0xF800 0000, ...	-2147483648	0xffffffff80000000	0x ffffffff 80000000
9	0xF7FF FFFF, ...	-2147483664	0xfffffff7fffffff0	0x fffffff7 fffffff0
10	100 000 000, ...	1600000000	0x5f5e1000	0x 00000000 5f5e1000
11	200 000 000, ...	3200000000	0xbebc2000	0x 00000000 bebc2000
12	300 000 000, ...	4800000000	0x11e1a3000	0x 00000001 1e1a3000
13	2 000 000 000, ...	32000000000	0x773594000	0x 00000007 73594000
14	3 000 000 000, ...	-20719476736	0xfffffffb2d05e000	0x fffffffb 2d05e000
15	-100 000 000, ...	-1600000000	0xfffffffa0a1f000	0x fffffff a0a1f000
16	-200 000 000, ...	-3200000000	0xffffffff4143e000	0x ffffffff 4143e000
17	-300 000 000, ...	-4800000000	0xfffffffee1e5d000	0x fffffffe e1e5d000
18	-2 000 000 000, ...	-32000000000	0xffffff88ca6c000	0x ffffff8 8ca6c000
19	-3 000 000 000, ...	20719476736	0x4d2fa2000	0x 00000004 d2fa2000
20	T19*2T6	-48000000000	0xffffff4d2fa2000	0x ffffff4 d2fa2000

Cuadro 2.1: Valores suma con signo

Capítulo 3

Media de N enteros de 32 bits signed

3.1. Ejercicio 5.4 Media de N enteros signed de 32 bits usando registros de 32 bits.

```
_start:
        .global _start

        mov     $lista, %rbx
        mov  longlista, %r10
        call suma

        mov  %eax, media        #media y resto separados en dos variables
        mov         %edx, resto #de 32 bits segun formato recomendado

        #Salida por stdout mediante printf--------------------------------
         mov   $formato, %rdi
         mov   media,%esi
         mov   resto,%edx
         mov          media,%ecx
         mov   resto,%r8d
         mov          $0,%eax
         call  printf

         call _exit
         ret
```

```
        #Inicializacion variables para suma--------------------------------
suma:
        mov  $0, %eax
        mov  $0, %edx
        mov  $0, %rcx
        mov  $0, %esi
        mov  $0, %edi

bucle:
        mov (%rbx,%rcx,4), %eax
        cdq  #Sign extension de dw a quad. 32-->64, EAX--> EDX:EAX

        add          %eax,%esi
        adc          %edx,%edi #%edi acumula el acarreo signed.

        inc   %rcx
        cmp   %rcx,%r10
        jne    bucle

        mov %esi,%eax
        mov %edi,%edx

        idiv %r10d      #division de EDX:EAX/%r10d.
                        #conciente--->EAX, resto-->EDX

        ret
```

3.2. Batería de test media

Test	Valor	Media d	Media hex	Resto d	Resto hex
1	1, 2, 1, 2, ...	1	0x 00000001	8	0x 00000008
2	-1, -2, -1, -2, ...	-1	0x ffffffff	-8	0x fffffff8
3	0x7FFF FFFF, ...	2147483647	0x 7fffffff	0	0x 00000000
4	0x8000 0000, ...	-2147483648	0x 80000000	0	0x 00000000
5	0x7FFF FFFF, ...	-1	0x ffffffff	0	0x 00000000
6	2 000 000 000, ...	2000000000	0x 77359400	0	0x 00000000
7	3 000 000 000, ...	-1294967296	0x b2d05e00	0	0x 00000000
8	-2 000 000 000, ...	-2000000000	0x 88ca6c00	0	0x 00000000
9	-3 000 000 000, ...	1294967296	0x 4d2fa200	0	0x 00000000
10	0,2,1,1, 1,1, ...	1	0x 00000001	0	0x 00000000
11	1,2,1,1, 1,1, ...	1	0x 00000001	1	0x 00000001
12	8,2,1,1, 1,1, ...	1	0x 00000001	8	0x 00000008
13	15,2,1,1, 1,1, ...	1	0x 00000001	15	0x 0000000f
14	16,2,1,1, 1,1, ...	2	0x 00000002	0	0x 00000000
15	0,-2,-1,-1, -1,-1, ...	-1	0x ffffffff	0	0x 00000000
16	-1,-2,-1,-1, -1,-1, ...	-1	0x ffffffff	-1	0x ffffffff
17	-8,-2,-1,-1, -1,-1, ...	-1	0x ffffffff	-8	0x fffffff8
18	-15,-2,-1,-1, -1,-1, ...	-1	0x ffffffff	-15	0x fffffff1
19	-16,-2,-1,-1, -1,-1, ...	-2	0x fffffffe	0	0x 00000000

Cuadro 3.1: Valores de media y resto.