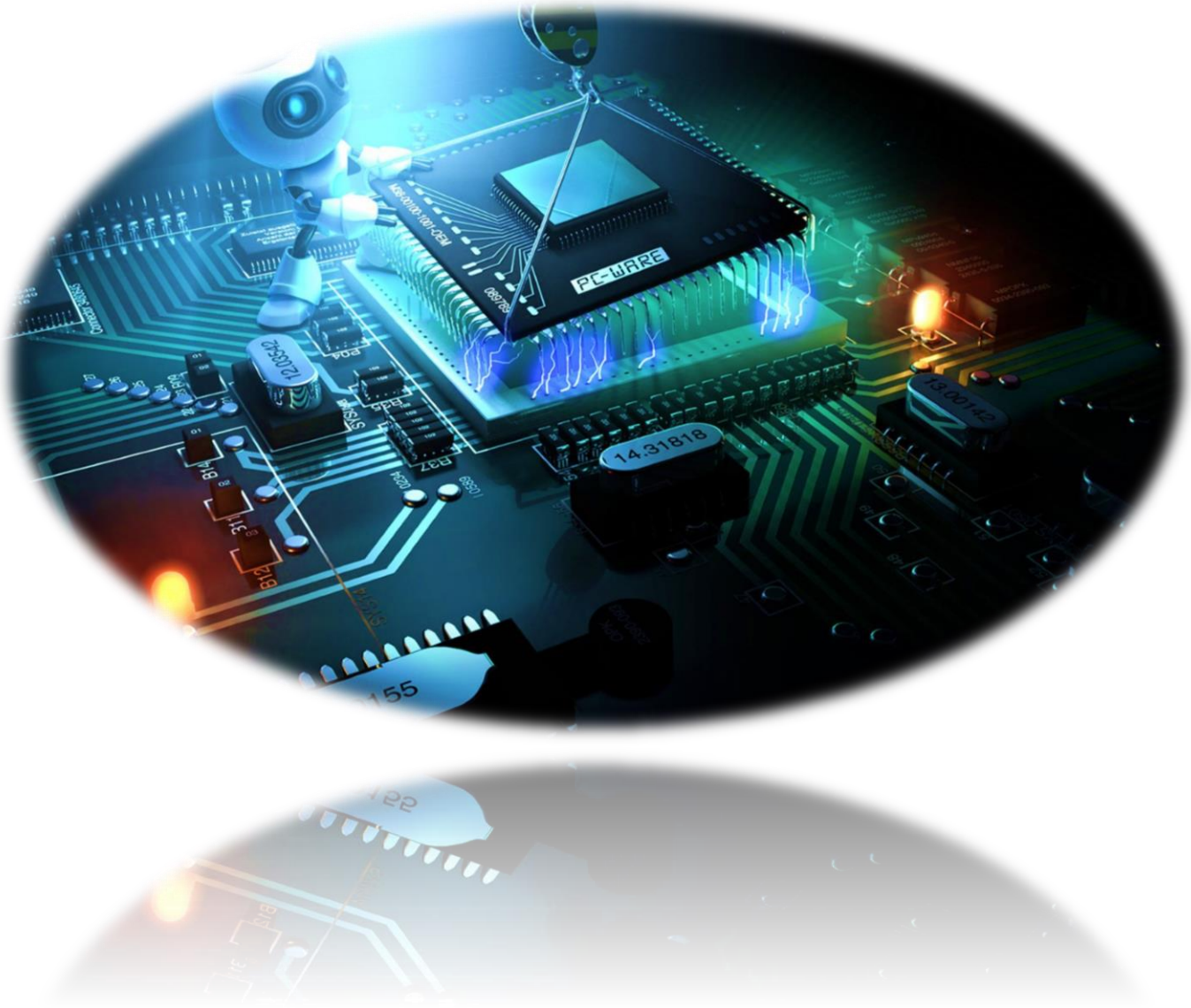
PRÁCTICA 2

ESTRUCTURA DE COMPUTADORES



BRYAN MORENO PICAMÁN

Contenido

Diario de trabajo

A continuación se detalla un diario de trabajo con los días dedicados a la práctica y las partes que se han desarrollado:

- 9 octubre.- Primera lectura del tutorial de prácticas, y realización de pruebas que se indican, también primera lectura del guion de prácticas.
- 11 octubre.- Segunda lectura del guion, primeras pruebas y comienzo de los códigos de la práctica en clase.
- 16 octubre.- Búsqueda de información de comandos necesarios para la práctica, familiarización con el funcionamiento y su utilidad.
- 18 octubre.- Realización de los códigos de la práctica, pruebas iniciales de comprobación de funcionamiento, preguntar de dudas en clases de prácticas.
- 21 octubre.- Finalización de los códigos y terminar de comentar las parte que me faltaban
- 25 octubre.- Realización de preguntas del cuestionario.
- 26 octubre.- Realización de memoria de la práctica y entrega

5.1.- Suma N enteros sin signo de 32 bits en una plataforma de 32 bits sin perder precisión

Código:

```
#SECCION DE DATOS (.data, variables globales inicializadas)
.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 0x08000000,0x08000000,0x08000000,0x08000000
                #       .int 0x10000000,0x20000000,0x40000000,0x80000000
        .endm
lista:.irpc i,12345678
                linea
        .endr
        longlista:   .int (.-lista)/4
        resultado: .quad -1

#Sección de formato para la salida por pantalla
formato:
        .ascii "suma=%llu hex=%llx  \n \0"

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

        mov  $lista, %ebx               #Ubicamos la lista
        mov  longlista, %ecx            #Guardamos el tamaño de la lista
        call suma                       #Llamada a la función suma
        mov  %esi, resultado+4                  #Movemos los resultados para mostrarlo
        mov  %eax, resultado

        #Metemos el resultado en orden inverso y el formato
        push resultado+4
        push resultado
        push resultado+4
        push resultado
        push $formato
        call printf                     #Llamada a print para mostrar
        add  $20, %esp                          #Regresamos la pila a su posición inicial
        mov  $1, %eax                           #Datos para la salida
        mov  $0, %ebx
        int  $0x80                      #Llamada a exit

suma:
        push %edx                       #Guardamos lo anterior en la pila
        mov $0, %eax                            #Limpieza de registros
        mov $0, %edx
        mov $0, %esi
bucle:
        add (%ebx,%edx,4), %eax         #Sumamos
        adc $0,%esi                             #Suma de acarreo
        inc %edx                        #Incrementar índice
        cmp %edx,%ecx                   #¿Fin del bucle?
        jne bucle

        pop %edx                        #Regresamos al estado anterior de la pila.
        ret
```

Salidas:

5.1 Ejemplos

Ejemplo	Resultado	Comentario
[1,...]	32	Todos al mismo valor
[2,...]	64	Todos al mismo valor
[1,2,3,4,...]	80	Cíclico de los valores
[-1,...]	137438953440	Todos al mismo valor
[0x08000000,...]	4294967296	Todos al mismo valor
[0x10000000,0x20..,0x40..,0x80..,0x10..,...]	32212254720	Cíclico de los valores

5.2.- Suma N enteros con signo de 32 bits en una plataforma de 32 bits sin

Código:

```
#SECCION DE DATOS (.data, variables globales inicializadas)
.section .data
        .macro linea
                        .int -1,-1,-1,-1
                #       .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 0x08000000,0x08000000,0x08000000,0x08000000
                #       .int 0xFC000000,0xFC000000,0xFC000000,0xFC000000
                #       .int 0xF0000000,0xE0000000,0xE0000000,0xD0000000
        .endm
lista:.irpc i,12345678
                linea
        .endr
        longlista:   .int (.-lista)/4
        resultado: .quad -1
formato:            .ascii "suma=%lld hex=%llx  \n \0"

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

        mov  $lista, %ebx           #Ubicamos la lista
        mov  longlista, %ecx        #Guardamos el tamaño de la lista
        call suma                   #Llamada a la función suma
        mov  %edi, resultado+4              #Metemos los resultados de suma en resultado para mostrarlo
        mov  %ebp, resultado

        #Metemos el resultado en orden inverso y el formato
        push resultado+4
        push resultado
        push resultado+4
        push resultado
        push $formato
        call printf                                 #Llamada a print para mostrar
        add  $20, %esp                              #Regresamos la pila a su posición inicial (hacemos 3 push 4*5)
        mov  $1, %eax                               #Datos para la salida
        mov  $0, %ebx
        int  $0x80                                  #Llamada a exit
suma:
        push %edx                                   #Guardamos el valor que contenga
        mov $0, %eax                                #Ponemos a 0 los registros a usar
        mov $0, %edx
        mov $0, %esi                                #Índice del bucle
        mov $0, %ebp                                #Primer registro (derecha)
        mov $0, %edi                                #Segundo registro (izquierda)
bucle:
        mov (%ebx,%esi,4), %eax             #Cogemos el primer valor
        cdq                                 #Extensión del signo de EAX a EDX (Importante no usarlos para nada mas)
        add %eax,%ebp                       #Realizamos la suma
        adc %edx,%edi
        inc %esi                                    #Incremento del índice del bucle
        cmp %esi,%ecx                               #¿Fin del bucle?
        jne bucle

        pop %edx                                    #Devuelve EDX a su estado original
        ret
```

Salida:

5.2 Ejemplos

Ejemplo	Resultado	Comentario
[-1,...]	-32	Todos al mismo valor
[1,-2,1,-2,…]	-16	Cíclico de los valores
[1,2,-3,-4,…]	-32	Cíclico de los valores
[0x7FFFFFF,…]	68719476704	Todos al mismo valor
[0x80000000,…]	-68719476736	Todos al mismo valor
[0x04000000,…]	2147483648	Todos al mismo valor
[0x08000000,…]	4294967296	Todos al mismo valor
[0xFC000000,…]	-2147483648	Todos al mismo valor
[0xF0000000,0xE0..,0xE0..,0xD0..,….]	-17179869184	Cíclico de los valores

5.3.- Media de N enteros con signo de 32 bits en una plataforma de 32 bits

Código:

```
#SECCION DE DATOS (.data, variables globales inicializadas)
.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 32,-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
        resultado: .quad -1

#Sección de formato para la salida por pantalla
formato:
        .ascii "cociente=%8d \t resto=%8d  \n"
        .ascii "hex=0x%08x \t resto=0x%08x  \n \0"

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

        mov  $lista, %ebx               #Ubicamos la lista
        mov  longlista, %ecx            #Guardamos el tamaño de la lista
        call suma                       #Llamada a la función suma
        mov  %ebp,%eax                  #Movemos el resultado a los registros necesarios para hacer la división con IDIV
        mov  %edi,%edx
        idiv %ecx                       #Realizamos la división
        mov  %edx,resultado+4           #Movemos el resultado para la salida
        mov  %eax,resultado

        #Metemos el resultado en orden inverso y el formato
        push resultado+4
        push resultado
        push resultado+4
        push resultado
        push $formato                                   #Llamada a print para mostrar
        call printf
        add  $20, %esp                                  #Regresamos la pila a su posición inicial (hacemos 3 push 4*5)
        mov  $1, %eax                                   #Datos para la salida
        mov  $0, %ebx
        int  $0x80                                      #Llamada a exit
suma:
        push %edx                                       #Guardamos el valor que contenga
        mov $0, %eax                                    #Ponemos a 0 los registros a usar
        mov $0, %edx
        mov $0, %esi            #Índice del bucle
        mov $0, %ebp            #Primer registro (derecha)
        mov $0, %edi            #Segundo registro (izquierda)
```

```
bucle:
        mov (%ebx,%esi,4), %eax          #Cogemos el primer valor
        cdq                                                  #Extensión del signo de EAX a EDX
(Importante no usarlos para nada mas)
        add %eax,%ebp                    #Realizamos la suma
        adc %edx,%edi
        inc %esi                         #Incremento del índice del bucle
        cmp %esi,%ecx                    #¿Fin del bucle?
        jne bucle
        pop %edx                         #Devuelve EDX a su estado original
        ret
```

Salida:

5.3 Ejemplos

Ejemplo	Cociente	Resto	Comentario
[1,-2,1,-2,…]	0	-16	Cíclico de los valores
[1,2,-3,-4,…]	-1	0	Cíclico de los valores
[0x7FFFFFFF,…]	2147483647	0	Todos al mismo valor
[0x80000000,…]	-2147483648	0	Todos al mismo valor
[0xF0000000,0xE0..,0xE0..,0xD0..,….]	-536870912	0	Cíclico de los valores
[-1,...]	-1	0	Todos al mismo valor
[0,-1,-1,-1,…]	0	-24	Cíclico de los valores
[0,-2,-1,-1,…]	-1	0	Cíclico de los valores
[1,-2,-1,-1,…]	0	-24	Cíclico de los valores
[32,-2,-1,-1,…]	7	16	Cíclico de los valores
[64,-2,-1,-1,…]	15	16	Cíclico de los valores
[95,-2,-1,-1,…]	23	8	Cíclico de los valores
[-31,-2,-1,-1,…]	-8	-8	Cíclico de los valores

Cuestionarios

Cuestiones sobre Suma Unsigned

1.-

- 5 bits, esto se debe a que el resultado de la suma con todos los valores a 1 (F en todos) devuelve como resultado 1FFFFFFFE0.
- El valor indicado seria el fffffff000000000
- La suma de elementos (todos a F) produce un total de 31 acarreos.

2.- El valor en hexadecimal es 8000000, el acarreo se produce cada vez que se suma dos veces el valor indicado.

3.-

- La suma de los valores cíclicamente produce el resultado 32212254720 (en hexadecimal: 780000000).
- Los acarreos se producen cuando se empieza una suma de un nuevo "ciclo" de números.

Cuestiones sobre Suma Signed

1.- El mayor entero positivo que puede representarse es 7FFFFFFF (todos los números a 1 menos el primero que indica el signo). La suma de estos valores daría como resultado el hexadecimal FFFFFFFE0.

2.- El menor valor negativo seria 80000000 (en hexadecimal)

3.- El valor máximo que se puede sumar sin producir un acarreo es el 4000000 (hexadecimal)

4.- El razonamiento es incorrecto, ya que no existe una consecución de elementos positivos iguales que al sumar produzca un único acarreo de los 32 bits inferiores a los superiores. Existe un ejemplo con el número 8000000 en hexadecimal, que produce un único acarreo.

5.- El valor es el 80000000 que al sumarlo nos devuelve como resultado fffffff000000000, esto es para -2^31, en el caso de -2^{32} no sería posible almacenarlo ya que necesitaríamos al menos 33 bits.

6.- El valor obtenido seria el fffffffc00000000

Cuestiones sobre Media

1.- Por que la división que realizamos es entera, por eso además tenemos un resto de -24, que no es posible dividirlo entre el total de elementos que es 32.

2.-

- Caso 1: -16 de resto, cociente 0
- Caso 2: -8 de resto, cociente 0
- Caso 3: 0 de resto, cociente 0
- Esto continua en el rango 3/35, ya que al llegar el valor a 35,-1,-1,-1 daría cociente 1 resto 0.
- Los restos van creciendo desde 1 hasta 31.

3.- Cualquier consecución de elementos que al salir de la suma nos dé 32

4.- El rango de valores seria el que al dividir entre 32 (número de sumas) ofreciera como cociente 1, por lo que sería entre 32 y 63 (64 nos daría 2)

5.- El igual que en el caso anterior como la división se realiza entre 32, los valores que podemos tener que nos dan como cociente -1 (media) serían los que van de -32 a -63. Estos producirían un resto creciente de 0 hasta -31