ESTRUCTURA DE COMPUTADORES: PRACTICA 2

Carlos Morales Aguilera 2ºB B3

Diario de trabajo

Día 1 : Planteamiento del primer programa de sumar sin signo con acarreo, intentando entender las distintas operaciones que se realizan en ensamblador.

Día 2 : Realizacion del primer ejercicio, con ayuda del profesor, que me ha resulto las dudas que tenia sobre las distintas operaciones que puedo hacer, y como funciona cada una.

Día 3 : Planteamiento y realización del segundo programa de sumar con signo y acarreo.

Día 4 : Planteamiento y realización del tercer programa de la media y comprobación de que funcionan todos los ejercicios.

Programa 1

```
# Datos
.section .data
lista:          .int 1,2,3,4,8    # binario 0b / hex 0x
longlista:      .int (.-lista)/4   #.= contador posiciones (numero de enteros que hay)
resultado:      .quad 0 # Para ver el resultado.
formato:        .string "La suma es %llu en decimal o 0x%llx en hexadecimal.\n"

# Codigo
.section .text
        .global main # Programa
        .extern printf # Busca printf fuera del fichero
#_start:
```

```
main:
        mov $lista, %ebx        # Direccion del array lista
        mov longlista, %ecx     # Cantidad de elementos a sumar
        call suma               # llamar suma(&lista, longlista);
        mov %eax, resultado     # Salvar mitad del resultado en el registro eax
        mov %edi, resultado + 4 # Salvar la otra mitad del resultado en el registro edi

        # Formato en decimal
        push resultado + 4
        push resultado

        # Formato en hexadecimal
        push resultado +4
        push resultado

        # Apilar el formato
        push $formato

        # printf (&formato, resultado)
        call printf
        add $20, %esp       # dejar pila intacta (hemos hecho 5 push, 20Bytes), equivale a los 5 pop

        ret

        # void _exit(int status);

        mov $1, %eax # exit: servicio 1 kernel Linux
        mov $0, %ebx # status: código a retornar (0->OK)
        int $0x80    # llamar _exit(0);

suma:
        push %edx      # Lo usamos como indice
        mov $0, %eax # poner a 0 acumulador 1
        mov $0, %edi # poner a 0 acumulador 2
        mov $0, %edx # poner a 0 índice (contador)

bucle:
        add (%ebx,%edx,4), %eax     # Acumulacion de la suma
        adc $0, %edi            # Acarreo, si hay

        # Iteracion del bucle

        inc      %edx           # Incrementar indice
        cmp  %edx,%ecx                  # Comparar con longitud
        jne bucle               # Si no son iguales, vuelve al bucle

        pop %edx                # Recuperar anterior %edx
        ret
```

A este nuevo programa le he incorporado el adc para el acarreo y el nuevo registro %edi comparandolo con el suma.s

Programa 2

```
# Datos
.section .data

		.macro linea
		#	.int -1,-1,-1,-1
		#	.int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
		#	.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 0xf8000000,0xf8000000,0xf8000000,0xf8000000
			.int 0xf0000000,0xe0000000,0xe0000000,0xd0000000
		.endm



lista:
		.irpc i,12345678
			linea
		.endr
longlista:	.int (.-lista)/4 # .= contador posiciones (numeros enteros que hay)
resultado:	.quad 0 # Para ver el resultado
formato:	.string "La suma es %llu en decimal o 0x%llx en hexadecimal.\n"
# Codigo
.section .text
	.global main # Programa
	.extern printf # Busca printf fuera del fichero
#_start:
main:
	mov $lista, %ebx		# Direccion del array lista
	mov $longlista, %ecx		# Cantidad de elementos a sumar
	call suma			# Llamar suma(&lista,longlista)
	mov %eax, resultado		# Salvar mitad del resultado en el registro eax
	mov %edi, resultado + 4		# Salvar la otra mitad del resultado en el registro edi

	# Formato en decimal
	push resultado + 4
	push resultado
```

```
	# Formato en hexadecimal
	push resultado +4
	push resultado

	# Apilar el formato
	push $formato

	# printf (&formato, resultado)
	call printf
	add $20, %esp		# Dejar pila intacta (hemos hecho 5 push, 20Bytes), equivale a los 5 pop

suma:

	mov $0, %edi
	mov $0, %ebp
	mov $0, %esi

bucle:

	mov (%ebx,%esi,4), %eax
	cltd			# Extender signo
	add %eax,%edi
	adc %edx,%ebp		# Acarreo, si hay
	inc %esi		# Incrementar índice
	cmp %esi, %ecx			# Comparar con longitud
	jne bucle		# Si no son iguales, vuelve al bucle

	mov %edi,%eax		# Guarda el resultado en el registro %eax
	mov %ebp,%edx			# Guarda el resultado de acarreo en el registro %edx
	ret
```

Programa 3

```
# SECCIÓN DE DATOS (.data, variables globales inicializadas)
.section .data

lista:
		.macro linea
		#	.int -1,-1,-1,-1
		#	.int 0xffffffff,0xffffffff,0xffffffff,0xffffffff
		#	.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 0xf8000000,0xf8000000,0xf8000000,0xf8000000
                        .int 0xf0000000,0xe0000000,0xe0000000,0xd0000000
                .endm

longlista:
        .int (.-lista)/4 # .= contador posiciones (número de enteros). Aritmética de etiquetas.

resultado:
        .quad 0 #Para ver la media.

formato:
        .string "La media es %ld en decimal.\n"

# SECCIÓN DE CÓDIGO (.text, instrucciones máquina)
.section .text
        .global main # PROGRAMA PRINCIPAL - se puede abreviar de esta forma
        .extern printf # busca printf fuera de este fichero

#_start:
main:

        mov $lista, %ebx    # dirección del array lista
        mov longlista, %ecx # número de elementos a sumar
        call suma           # llamar suma(&lista, longlista);


        idiv %ecx   #equivale a dividir suma  entre longlista  y guarda en %eax el resultado
        mov %eax,resultado
        push resultado


        push $formato       # apila formato
        call printf         # llamada a función printf(&formato, media)
        add $8, %esp        # dejar pila  (hemos hecho 2 push, 8Bytes)

        ret

        # void _exit(int status);
        mov $1, %eax # exit: servicio 1 kernel Linux
        mov $0, %ebx # status: código a retornar (0=OK)
        int $0x80    # llamar _exit(0);



suma:
        mov $0, %edi  #poner a 0 acumuladores.
        mov $0, %ebp
        mov $0, %esi   # poner a 0 el índice.


bucle:
        mov (%ebx,%esi,4), %eax  # mueve i-ésimo elemento
```

```
cltd     # extender signo
add %eax,%edi
adc %edx,%ebp
inc %esi        # incrementamos índice.
cmp %esi, %ecx  # comparar con longitud
jne bucle       # si no son iguales, volver al bucle

mov %edi,%eax  #Obtenemos resultados en %eax y %edx
mov %ebp,%edx

ret
```