

## Sesión de suma64signed.s

1-¿Cual es el maximo entero positivo que puede representarse (escribirlo en hexadecimal)? Si se sumaran los  $N=32$  elementos de la lista inicializados a ese valor ¿que resultado se obtendría (en hexadecimal)? ¿Que valor aproximado tienen el elemento y la suma (indicarlo en multiplos de potencias binarias, ki, Mi, Gi)? Comprobarlo usando ddd

El máximo entero positivo que se puede representar es el 0x7FFFFFFF. La suma obtenida de los 32 elementos con el valor anterior es la siguiente:

```
juanka1995@juanka-laptop ~/practicass/ec/5.2 $ ./suma_con_signo
suma = ffffffff0
```

La suma 0xFFFFFFFF0  $\approx 2^{36}$  y el elemento 0x7FFFFFFF  $\approx 2^{31}$ .

2-.Misma pregunta respecto a negativos: menor valor negativo en hexadecimal, suma, valores decimales aprox., usar ddd

El numero más pequeño que puede representarse en hexadecimal es el 0x80000000. La suma obtenida de los 32 elementos inicializados a ese número es de:

```
juanka1995@juanka-laptop ~/practicass/ec/5.2 $ ./suma_con_signo
suma = ffffffff000000000
```

La suma 0xFFFFFFFF000000000  $\approx 2^{64}$  y el elemento 0x80000000  $\approx 2^{31}$ .

3-.Si nos proponemos obtener solo 1 acarreo con una lista de 32 elementos positivos iguales, se podría pensar que el objetivo es que la suma alcance  $2^{31}$  (que ya no cabe en 32 bits como número positivo en complemento a dos). Aparentemente, cada elemento debe valer por tanto  $2^{31}/32 = 2^2/2^7 = ?$ . ¿Como se escribe ese valor en hexadecimal? Inicializar los 32 elementos de la lista con ese valor y comprobar si se produce el acarreo.

$$2^{31}/32 = 0x04000000$$

```
juanka1995@juanka-laptop ~/practicass/ec/5.2 $ ./suma_con_signo
suma = 800000000
```

<pre> mov \$lista, %ebx mov longlista, %ecx call suma mov %esi, resultado1 mov %edi, resultado2  push resultado2 push resultado1 push \$formato call printf  mov \$1, %eax mov \$0, %ebx </pre>	<pre> eax      0x40000000  67108864 ecx      0x20       32 edx      0x0        0 ebx      0x804a01c   134520860 esp      0xffffd0cc  0xffffd0cc ebp      0x0        0x0 esi      0x80000000  -2147483648 edi      0x0        0 eip      0x8048421   0x8048421 &lt;main&gt; eflags   0x246      [ PF ZF IF ] cs       0x23       35 </pre>
---	---

En la imagen anterior vemos que el acarreo se produce debido a que los primeros 32 bits del número toman el valor 0x80000000 y la segunda parte de 32 bits esta a 0. De todas formas podriamos comprobarlo llenando a la ultima interacción del bucle y viendo el flag de acarreo CF.

5-.Respecto a negativos,  $-2^{31}$  sí cabe en 32 bits como numero negativo en complemento a dos. Calcular que valor de elemento se requiere para obtener como suma  $-2^{31}$ , y para obtener  $-2^{32}$ . Comprobarlo usando ddd

$$-2^{31} = 0x80000000 = 0x04000000 * 32$$

$$-2^{32} = 0x100000000 = 0x08000000 * 32$$

6-.Por probar valores intermedios: si la lista se inicializara con los valores 0xF0000000, 0xE0000000, 0xE0000000, 0xD0000000, repetidos ciclicamente, ¿Que valor tomaría la suma de los 32 elementos (en hexadecimal)? Comprobarlo con ddd.

```
juanka1995@juanka-laptop ~/practicass/ec/5.2 $ ./suma_con_signo
suma = ffffffff00000000
```

The screenshot shows the DDD debugger interface. On the left, the assembly code is displayed, including the definition of a list of 32 integers and the calculation of their sum. The registers window on the right shows the current state of the registers, with the value of edi highlighted as -4.

Register	Value	Comment
eax	0xd0000000	-805306368
ecx	0x20	32
edx	0xffffffff	-1
ebx	0x804a01c	134520860
esp	0xffffd0cc	0xffffd0cc
ebp	0x0	0x0
esi	0x0	0
edi	0xffffffffc	-4
eip	0x8048427	0x8048427 <mai
eflags	0x246	[ PF ZF IF ]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43

El valor que tomaría la suma es de 0xFFFFFFFFC0000000.