Ejemplo sobre complemento a dos

Arquitectura del Computador - LCC - FCEIA-UNR

Agosto 2021

En este ejemplo se pretende mostrar algunas cuestiones sobre las operaciones en complemento a dos y las banderas *overflow* y *carry*. También es útil para familiarizarse con GDB.

Dado el siguiente código en Assembler X86-64 en el archivo `complemento_dos.s`:

```
.data

a: .byte 56
b: .byte 84

.text

.global main
main:
    movb a, %al
    movb b, %bl
    addb %bl, %al
    retq
```

Compilar utilizando la opción `-g` para poder *debuggear* utilizando GDB:

```
$ gcc -g complemento_dos.s
```

Luego ejecutar utilizando GDB:

```
gdb ./a.out
```

Una vez que estamos dentro de la sesión de *debugging*, ponemos un *breakpoint* en `main` y ejecutamos el comando `run`:

```
(gdb) br main
Breakpoint 1 at 0x4004b6: file complemento_dos.s, line 11.
(gdb) r
Starting program: /home/dferoldi/2020/a.out

Breakpoint 1, main () at complemento_dos.s:11
```

Luego vamos ejecutando línea a línea utilizando el comando `next`:

```
11 movb a, %al
(gdb) n
12 movb b, %bl
(gdb) n
13 addb %bl, %al
```

Ahora podemos ver el contenido de los subregistros `al` y `bl`:

```
(gdb) i r al bl
al             0x38 56
bl             0x54 84
```

ejecutamos una línea más para realizar la suma y vemos el resultado en `al`:

```
(gdb) n
(gdb) i r al
al             0x8c -116
```

Vemos que el resultado no es el que esperábamos, dado que $56 + 84 = 140$ y no -116 como estamos viendo. Es más, vemos que sumamos dos números positivos y obtuvimos como resultado un número negativo. ¿Qué sucedió? Estamos trabajando con datos de un byte (8 bits). Por lo tanto, el rango de números representables si trabajamos con signo es $-128 \leq rango \leq 127$, con lo cual el número 140 no es representable con 8 bits. Si además chequeamos el contenido del registro `eflags`, vemos que la bandera *overflow* (`OF`) se encendió lo que nos indica que el resultado es incorrecto si trabajamos con números con signo.

```
(gdb) i r eflags
eflags         0xa82 [ SF IF OF ]
```

Ahora bien, ¿por qué el resultado fue -116? Lo que hizo la ALU fue realizar la operación suma bit a bit, en este caso una suma. Por lo tanto, lo que realizó fue la siguiente operación:

$$\begin{array}{r} 00111000 \\ + \\ 01010100 \\ \hline 10001100 \end{array} \tag{1}$$

Efectivamente, la secuencia de bits $(10001100)_2$ representa el valor -116 en decimal utilizando complemento a dos: $-2^7 + 2^3 + 2^2 = -116$. Es decir, la ALU sumó las dos secuencias de bits bit a bit y luego GDB nos muestra el resultado como número con signo utilizando complemento a dos.

Notar que la misma secuencia de bits del resultado obtenido en (1) representa el valor 140 si lo interpretamos como número sin signo: $2^7 + 2^3 + 2^2 = 140$. Es decir, si estamos trabajando con número sin signos entonces el resultado es correcto. La bandera *carry* (`CF`) apagada efectivamente nos indica que el resultado es correcto si interpretamos a los números como números si signo.

Entonces, la conclusión general es que el resultado será correcto o no dependiendo de cómo interpretemos a los números. La computadora en su nivel más básico meramente hace operaciones entre secuencias de bits. Cómo se interpretan esas secuencias de bits será responsabilidad del programador.