



# TP2

## Stack Frame

### Debianitas

Carreras: Ingeniería en computación , Ingeniería Electrónica

Profesor:- Javier Jorge  
- Solinas Miguel Angel

Alumnos:  
- Landaeta Ezequiel Matias, Legajo: 41092007  
- Agüero David Gabriel, Legajo: 39621362  
- Cherino Macarena, Legajo: 41809793

# Índice

<b>1 Consigna</b>	<b>3</b>
<b>2 Desarrollo</b>	<b>3</b>
<b>2.1 Código</b>	<b>3</b>
<b>2.2 Comandos para compilar</b>	<b>5</b>
<b>2.3 Depuración con gdb</b>	<b>7</b>
<b>2.3.1 Análisis del Stack Frame</b>	<b>9</b>
<b>2.3.2 Dissasembling main:</b>	10
<b>3 Bibliografía</b>	<b>13</b>

# 1 Consigna

Para aprobar el TP#2 se debe diseñar e implementar una Calculadora de Cotización de Criptomonedas. La capa superior recuperará la cotización de al menos dos criptomonedas que pueden ser obtenidas de alguna de "Las 12 mejores API del mercado de valores para crear productos financieros". Se recomienda el uso de API Rest y python. Los datos de consulta realizados deben ser entregados a un programa en C que convocará rutinas en ensamblador para que hagan los cálculos de conversión y devuelvan las cotizaciones en pesos, u\$s y euros. Luego el programa en C o python mostrará los cálculos obtenidos.

Se debe utilizar el stack para convocar, enviar parámetros y devolver resultados. O sea utilizar las convenciones de llamadas de lenguajes de alto nivel a bajo nivel.

## 2 Desarrollo

### 2.1 Código

En el github <https://github.com/TheRabbitt/Sistemas-de-Computacion> se encuentra el código de un programa escrito en lenguaje C que obtiene el precio de dos criptomonedas (Bitcoin y Ethereum) usando la API de Binance y luego imprime el precio de estas criptomonedas en diferentes monedas (ARS, USD y EUR).

El programa utiliza la biblioteca libcurl para realizar solicitudes HTTP a la API de Binance y obtener los precios de las criptomonedas. También utiliza la función "write\_data" como una función de retorno de llamada para procesar los datos recibidos de la API.

Se define una variable llamada BTC y se le da el valor 0. Por otro lado, se define una variable llamada ETH y se le da el valor 1. Estas dos variables tienen el propósito de facilitar el manejo de las diferentes monedas. Luego se definen dos matrices de caracteres btc\_price y eth\_price, que se utilizarán para almacenar los precios de Bitcoin y Ethereum, respectivamente. current\_currency es una variable entera que se utilizará para indicar qué moneda se está procesando en un momento dado. Por último, se definen las variables valorArsDolar y valorEurDolar, que se utilizarán como factores de conversión para convertir los precios de Bitcoin y Ethereum en diferentes monedas.

A continuación, se declara la función `calc_conversion`, la cual se implementa en un archivo de ensamblador externo y se utilizará para realizar la conversión de precios de USD a otras monedas.

Después de la declaración de la función externa, se define la función `write_data()`. Esta función, es utilizada para procesar los datos recibidos de la API en la solicitud HTTP GET. Esta función se llamará automáticamente por la biblioteca libcurl cada vez que reciba datos de la API.

La función `write_data()` toma cuatro argumentos:

- `ptr`: un puntero al bloque de memoria que contiene los datos recibidos.
- `size`: el tamaño en bytes de cada elemento recibido.
- `nmemb`: el número de elementos recibidos.
- `stream`: un puntero al objeto que llama a la función.

La función devuelve el número total de bytes recibidos. Si la moneda actual es BTC, los datos recibidos se copiarán en `btc_price`; si es ETH, los datos se copiarán en `eth_price`.

A continuación, comienza la función `main()`.

En esta función, se inicializan las variables, se configuran las opciones de CURL, se realiza la solicitud a la API, se procesa la respuesta y se imprimen los resultados.

Aquí está la explicación detallada de lo que hace cada línea de la función `main()`:

1. Se declara un puntero de tipo CURL, que se usará para realizar las solicitudes HTTP a la API.
2. Se declara una variable de tipo CURLcode, que se utiliza para verificar si la solicitud se realizó con éxito.
3. Se inicializa el puntero CURL con la función `curl_easy_init()`.
4. Se verifica si el puntero CURL se inicializó correctamente.
5. Se establece la solicitud HTTP GET para obtener el precio de BTC con la función `curl_easy_setopt()`. Esto se realiza configurando las opciones `CURLOPT_URL` y `CURLOPT_WRITEFUNCTION`. La opción `CURLOPT_URL` especifica la URL de la API que proporciona el precio de BTC. La opción `CURLOPT_WRITEFUNCTION` establece la función de retorno de llamada que procesará los datos recibidos de la API.
6. Se realiza la solicitud HTTP GET para obtener el precio de BTC con la función `curl_easy_perform()`. Si la solicitud se realizó con éxito, se copian los datos recibidos en `"btc_price"` usando la función de retorno de llamada `write_data()`.
7. Se establece la solicitud HTTP GET para obtener el precio de ETH con la función `curl_easy_setopt()`. Esto se realiza configurando las opciones `CURLOPT_URL` y `CURLOPT_WRITEFUNCTION`. La opción `CURLOPT_URL` especifica la URL de la API que

proporciona el precio de ETH. La opción `CURLOPT_WRITEFUNCTION` establece la función de retorno de llamada que procesará los datos recibidos de la API.

8. Se realiza la solicitud HTTP GET para obtener el precio de ETH con la función `curl_easy_perform()`. Si la solicitud se realizó con éxito, se copian los datos recibidos en "eth\_price" usando la función de retorno de llamada `write_data()`.
9. Se imprime el precio de BTC en diferentes monedas usando la función `obtenerCotizacion()`. La función obtiene la cotización actual del BTC en pesos argentinos y en euros, y utiliza la función `calc_conversion()` (externa en archivo .asm) para convertir el precio de BTC a las diferentes monedas.
10. Se imprime el precio de ETH en diferentes monedas usando la función `obtenerCotizacion()`. La función obtiene la cotización actual del ETC en pesos argentinos y en euros, y utiliza la función `calc_conversion()` (externa en archivo .asm) para convertir el precio de ETH a las diferentes monedas.
11. Se libera la memoria y los recursos utilizados por CURL con la función `curl_easy_cleanup()`.

La función `main()` devuelve 0, lo que indica que el programa finalizó correctamente.

La función "obtenerCotizacion" se utiliza para convertir el precio de una criptomoneda en diferentes monedas. Esta función toma dos parámetros: `moneda`, que es la cotización actual de la moneda de referencia en relación a una tercera moneda (USD en este caso), y `monedaDigital`, que es un arreglo de caracteres que contiene el precio de una criptomoneda en relación a USD en formato de cadena.

La función comienza buscando la letra 'e' en `monedaDigital` utilizando la función `strchr()`. Si se encuentra la letra 'e', ya que en el formato que retorna la API hay una 'e' antes de que comience el valor de la cotización de la moneda digital respectiva. Si no se encuentra la letra 'e', la función termina con un error.

Luego, la función avanza cuatro caracteres más allá de la letra 'e' y convierte el resto de la cadena en un número de punto flotante utilizando la función `strtod()`. Este número representa el precio de la criptomoneda en USD.

La función luego llama a la función `calc_conversion()` que se define en otro archivo .asm y toma como parámetros el valor del precio de la criptomoneda en USD y la cotización actual de la moneda de referencia en relación a USD. La función `calc_conversion()` devuelve el valor de la criptomoneda en la moneda de referencia.

## 2.2 Comandos para compilar

1- `nasm -f elf64 -d ELF_TYPE -o Conversor5.o -d NO_STACK_PROTECTOR Conversor5.asm`

2- `gcc -o simpleget simplev3.c Conversor5.o -lcurl -z noexecstack`

El primer comando empleado (1), utiliza el ensamblador NASM para compilar un archivo de código fuente llamado "Conversor5.asm" en un archivo objeto de 64 bits con formato ELF llamado "Conversor5.o", con la adición de dos macros definidas en tiempo de compilación: "ELF\_TYPE" y "NO\_STACK\_PROTECTOR".

**-f elf64:** establece el formato de salida a ELF de 64 bits, que es un formato de archivo binario comúnmente utilizado en sistemas operativos basados en Unix/Linux.

**-d ELF\_TYPE:** define una macro llamada "ELF\_TYPE" con un valor vacío. Esta macro podría usarse en el código fuente de "Conversor5.asm" para realizar operaciones condicionales o para diferenciar entre diferentes tipos de ELF.

**-o Conversor5.o:** establece el nombre de salida del archivo objeto a "Conversor5.o".

**-d NO\_STACK\_PROTECTOR:** establece una macro de preprocesador llamada NO\_STACK\_PROTECTOR con un valor vacío durante la compilación del código fuente NASM. Esta macro puede ser utilizada en el código fuente para desactivar la protección de pila en el programa final compilado.

La protección de pila es una técnica de seguridad utilizada en la programación para detectar y prevenir los desbordamientos de pila. Los desbordamientos de pila ocurren cuando un programa escribe datos en una ubicación de la pila que está más allá del final de la región de memoria reservada para la pila, lo que puede llevar a comportamientos inesperados o vulnerabilidades de seguridad.

Sin embargo, en algunos casos, la protección de pila puede interferir con el funcionamiento de un programa, especialmente si se está escribiendo código de bajo nivel o se está interactuando con hardware. Por lo tanto, la opción -d NO\_STACK\_PROTECTOR permite desactivar la protección de pila en el programa final compilado si se considera que es necesario en un contexto particular. Es importante tener en cuenta que desactivar la protección de pila puede aumentar el riesgo de explotación de vulnerabilidades de seguridad en el programa, por lo que esta opción debe usarse con precaución y solo en casos donde se justifique.

El segundo comando empleado (2), utiliza el compilador GCC para compilar y vincular dos archivos de código fuente: "simplev3.c" y "Conversor5.o", generando un archivo ejecutable llamado "simpleget". Además, la línea de comando especifica que el archivo ejecutable no debe tener una pila ejecutable, y que debe vincularse con la biblioteca estática "curl" durante la compilación.

**-o simpleget:** establece el nombre de salida del archivo ejecutable a "simpleget".

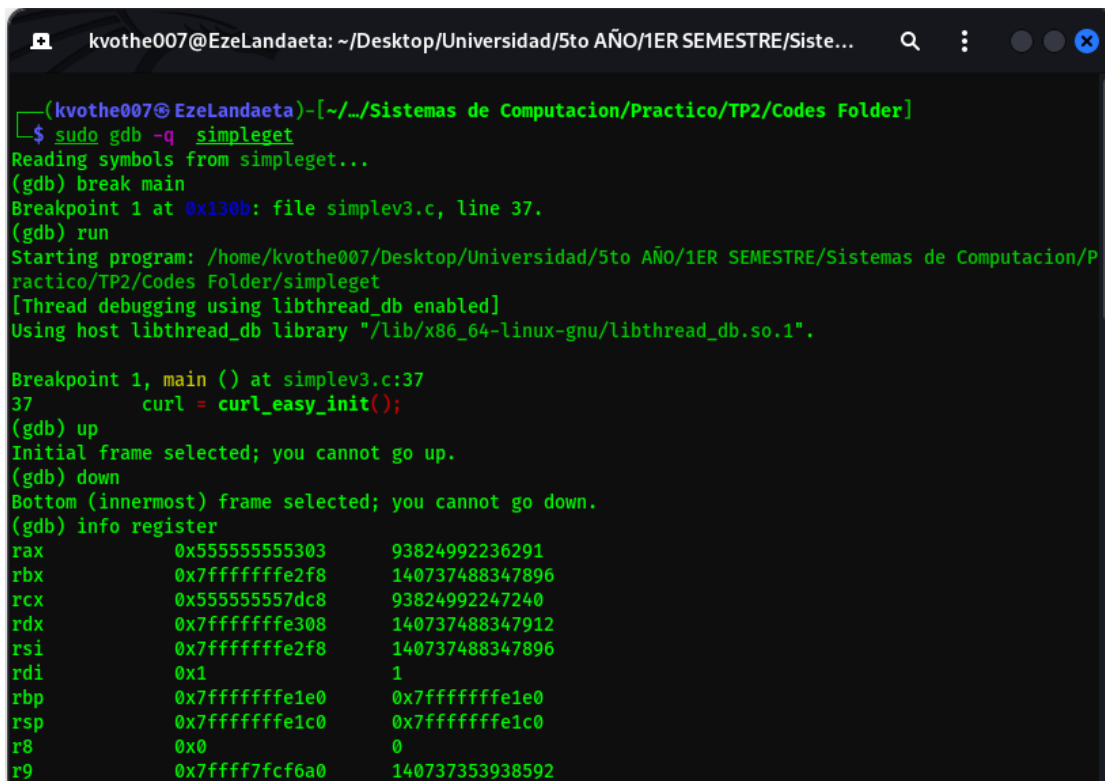
**simplev3.c:** es el nombre del archivo de código fuente de C que se compilará y vinculará con el archivo objeto "Conversor5.o" para crear el archivo ejecutable "simpleget".

**Conversor5.o:** es el nombre del archivo objeto compilado previamente que se vinculará con el archivo de código fuente "simplev3.c" para crear el archivo ejecutable "simpleget".

**-lcurl:** especifica que la biblioteca estática "curl" debe vincularse con el archivo ejecutable durante la compilación. Esto significa que el archivo ejecutable tendrá acceso a las funciones y variables definidas en la biblioteca estática "curl" durante su ejecución.

**-z noexecstack:** establece la pila del programa para que no sea ejecutable. Esto significa que se evita que la pila del programa se use para ejecutar código, lo que puede ayudar a prevenir ciertos tipos de ataques de seguridad.

## 2.3 Depuración con gdb



```
kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Siste...
(kvothe007@EzeLandaeta)-[~/Sistemas de Computacion/Practico/TP2/Codes Folder]
$ sudo gdb -q simpleget
Reading symbols from simpleget...
(gdb) break main
Breakpoint 1 at 0x130b: file simplev3.c, line 37.
(gdb) run
Starting program: /home/kvothe007/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Sistemas de Computacion/Practico/TP2/Codes Folder/simpleget
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at simplev3.c:37
37      curl = curl_easy_init();
(gdb) up
Initial frame selected; you cannot go up.
(gdb) down
Bottom (innermost) frame selected; you cannot go down.
(gdb) info register
rax             0x55555555303          93824992236291
rbx             0x7fffffff2f8          140737488347896
rcx             0x555555557dc8          93824992247240
rdx             0x7fffffff308          140737488347912
rsi             0x7fffffff2f8          140737488347896
rdi             0x1                      1
rbp             0x7fffffff1e0          0x7fffffff1e0
rsp             0x7fffffff1c0          0x7fffffff1c0
r8              0x0                      0
r9              0x7fff7fcf6a0          140737353938592
```

```
kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Siste...
eflags      0x206      [ PF IF ]
cs          0x33      51
ss          0x2b      43
ds          0x0       0
es          0x0       0
fs          0x0       0
gs          0x0       0
(gdb) step
38      if (curl) {
(gdb) next
41          current_currency = BTC;
(gdb) next
42          curl_easy_setopt(curl, CURLOPT_URL, "https://api.binance.com/api/v3/ticker/price?sym
bol=BTCUSDT");
(gdb) next
43          curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, write_data);
(gdb) next
44          res = curl_easy_perform(curl);
(gdb) print
The history is empty.
(gdb) ptype variable
No symbol "variable" in current context.
(gdb) next
[New Thread 0x7ffff6da86c0 (LWP 24332)]
[Thread 0x7ffff6da86c0 (LWP 24332) exited]
45          if (res != CURLE_OK) {
(gdb) next
51          current_currency = ETH;
(gdb) ptype current_currency
```

```
kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Siste...
(gdb) next
51          current_currency = ETH;
(gdb) ptype current_currency
type = int
(gdb) display current_currency
1: current_currency = 0
(gdb) next
52          curl_easy_setopt(curl, CURLOPT_URL, "https://api.binance.com/api/v3/ticker/price?sym
bol=ETHUSDT");
1: current_currency = 1
(gdb) next
53          res = curl_easy_perform(curl);
1: current_currency = 1
(gdb) next
54          if (res != CURLE_OK) {
1: current_currency = 1
(gdb) next
65          printf("BTCenARS: %lf\n", obtenerCotizacion (valorArsDolar,btc_price));
1: current_currency = 1
(gdb) i r sp
sp          0x7fffffff1c0      0x7fffffff1c0
(gdb) info locals
curl = 0x55555558ec80
res = CURLE_OK
(gdb) next
BTCenARS: 6462243.210000
66          printf("BTCenUSD: %lf\n", obtenerCotizacion (1,btc_price));
1: current_currency = 1
(gdb) □
```



```
kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Siste...
(gdb) next
BTCenARS: 6462243.210000
66      printf("BTCenUSD: %lf\n", obtenerCotizacion (1,btc_price));
1: current_currency = 1
(gdb) step
obtenerCotizacion (moneda=1,
monedaDigital=0x5555555580a0 <btc_price> "{\"symbol\":\"BTCUSD\",\"price\":\"30339.17000000\"}"
) at simplev3.c:84
84      char* punteroChar = strchr(monedaDigital, 'e');
1: current_currency = 1
(gdb) up
#1 0x0000555555555487 in main () at simplev3.c:66
66      printf("BTCenUSD: %lf\n", obtenerCotizacion (1,btc_price));
(gdb) down
#0  obtenerCotizacion (moneda=1,
monedaDigital=0x5555555580a0 <btc_price> "{\"symbol\":\"BTCUSD\",\"price\":\"30339.17000000\"}"
) at simplev3.c:84
84      char* punteroChar = strchr(monedaDigital, 'e');
(gdb) down
Bottom (innermost) frame selected; you cannot go down.
(gdb) continue
Continuing.
BTCenUSD: 30339.170000
BTCenEUR: 27305.253000
ETHenARS: 445657.770000
ETHenUSD: 2092.290000
ETHenEUR: 1883.061000
[Inferior 1 (process 24279) exited normally]
(gdb) □
```

## Análisis del Stack Frame

```
kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Sistemas de Computacion/Practico/TP2/Codes Folder

---(kvothe007@EzeLandaeta):~/Sistemas de Computacion/Practico/TP2/Codes Folder
$ sudo gdb -q simpleget
Reading symbols from simpleget...
(gdb) break calc_conversion
Breakpoint 1 at 0x1030
(gdb) run
Starting program: /home/kvothe007/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Sistemas de Computacion/Practico/TP2/Codes Folder/simpleget
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7ffff6da80c0 (LWP 29727)]
[Thread 0x7ffff6da80c0 (LWP 29727) exited]

Thread 1 "simpleget" hit Breakpoint 1, 0x0000555555555630 in calc_conversion ()
(gdb) info register
rax      0x40daceb1eb851eb8  4673274826906214072
rbx      0x7fffff2fe2f8    140737488347896
rcx      0xaceb1          708273
rdx      0x0
rsi      0x40daceb100000000  4673274822954846160
rdi      0x7fffffdd70      140737488346488
r10      0x7fffff2fe1b0    0x7fffff2fe1b0
r12      0x7fffff2fe178    0x7fffff2fe178
r13      0x0
r14      0x0
r15      0x1
r16      0x2
r17      0x0
r18      0x0
r19      0x1
r20      0x2
r21      0x0
r22      0x0
r23      0x7fffff2fe308    140737488347912
r24      0x555555557dc8     93824992247240
r25      0x7fffff7fd020    140737354125344
rip      0x555555555630    0x555555555630 <calc_conversion>
eflags   0x206            [ PF IF ]
cs       0x33           51
ss       0x2b           43
ds       0x0
fs       0x0
gs       0x0
(gdb) print ($xmm0)
$1 = {v8_bfloat16 = {1.948e-20, -3.216e+26, -1.485e+00, 6.812, 0, 0, 0, 0}, v8_half = {0.0065613, -3850, -26.766, 2.4258, 0, 0, 0, 0}, v4_float = {-3.21864398e+26, 6.83773888, 0, 0}, v2_double = {27450.779999999999, 0}, v16_int8 = {-72, 30, -123, -21, -79, -50, -38, 64, 0, 0, 0, 0, 0, 0, 0, 0}, v8_int16 = {7864, -5243, -12623, 16602, 0, 0, 0, 0}, v4_int32 = {-343597384, 1088081585, 0, 0}, v2_int64 = {4673274826906214072, 0}, uint128 = 4673274826906214072}
(gdb) print ($xmm1)
$2 = {v8_bfloat16 = {0, 0, -1.084e-19, 3.656, 0, 0, 0, 0}, v8_half = {0, 0, -0.0078125, 2.287, 0, 0, 0, 0}, v4_float = {0, 3.66601562, 0, 0}, v2_double = {213, 0}, v16_int8 = {0, 0, 0, 0, 0, 0, -96, 106, 64, 0, 0, 0, 0, 0, 0, 0}, v8_int16 = {0, 0, -24576, 16490, 0, 0, 0, 0}, v4_int32 = {0, 1080729600, 0, 0}, v2_int64 = {4641698287819161600, 0}, uint128 = 4641698287819161600}
(gdb) next
Single stepping until exit from function calc_conversion,
which has no line number information.
obtenerCotizacion (moneda=213, monedaDigital=0x5555555580a0 <bt_price> "\symbol\":"BTCUSD", "\price\":"27450.78000000") at simplev3.c:92
92      return value;
(gdb) print ($xmm0)
$3 = {v8_bfloat16 = {-71.5, 1.475e-33, 5.243e+08, 13.38, 0, 0, 0, 0}, v8_half = {-3.2793, 0.00015128, 23.906, 2.668, 0, 0, 0, 0}, v4_float = {1.47911416e-33, 13.3948372, 0, 0}, v2_double = {5847016.1399999999, 0}, v16_int8 = {-113, -62, -11, 8, -6, 77, 86, 65, 0, 0, 0, 0, 0, 0, 0, 0}, v8_int16 = {-15729, 2293, 19962, 16726, 0, 0, 0, 0}, v4_int32 = {158323855, 1096175098, 0, 0}, v2_int64 = {4708036196749918863, 0}, uint128 = 4708036196749918863}
(gdb) 
```

Como podemos observar, al hacer un break en la función `calc_conversion`, el registro `xmm0` en su vector `v2_double` se carga con el valor en dólares que se obtiene de la api y el registro `xmm1` en su vector `v2_double` se carga con el valor 213 que es la cotización del dólar oficial del día. Al avanzar con `next` en el debugger podemos observar que el registro `xmm0` ahora tiene el resultado de la multiplicación de ambos registros (`v2_double`) y que se terminó guardando en `xmm0`.

A continuación vamos a hacer un disassembling de nuestro programa (función `main`) donde se pueden ver todos los registros involucrados y sus respectivas operaciones.

## Dissassembling main:

```
kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Sistemas de Computacion/Practic...
(kvothe007@EzeLandaeta)~[~/Sistemas de Computacion/Practico/TP2/Codes Folder]
$ sudo gdb -q simpleget
[sudo] password for kvothe007:
Reading symbols from simpleget...
(gdb) disas main
Dump of assembler code for function main:
0x0000000000001303 <+0>:  push  %rbp
0x0000000000001304 <+1>:  mov   %rsp,%rbp
0x0000000000001307 <+4>:  sub   $0x20,%rsp
0x000000000000130b <+8>:  call  0x10a0 <curl_easy_init@plt>
0x0000000000001310 <+13>: mov   %rax,-0x8(%rbp)
0x0000000000001314 <+17>: cmpq  $0x0,-0x8(%rbp)
0x0000000000001319 <+22>: je     0x1595 <main+658>
0x000000000000131f <+28>: movl  $0x0,0x2deb(%rip)      # 0x4114 <current_currency>
0x0000000000001329 <+38>: movl  $0x2712,-0xc(%rbp)
0x0000000000001330 <+45>: mov   -0xc(%rbp),%ecx
0x0000000000001333 <+48>: mov   -0x8(%rbp),%rax
0x0000000000001337 <+52>: lea   0xcca(%rip),%rdx      # 0x2008
0x000000000000133e <+59>: mov   %ecx,%esi
0x0000000000001340 <+61>: mov   %rax,%rdi
0x0000000000001343 <+64>: mov   $0x0,%eax
0x0000000000001348 <+69>: call  0x1070 <curl_easy_setopt@plt>
0x000000000000134d <+74>: movl  $0x4e2b,-0x10(%rbp)
0x0000000000001354 <+81>: mov   -0x10(%rbp),%ecx
0x0000000000001357 <+84>: mov   -0x8(%rbp),%rax
0x000000000000135b <+88>: lea   -0xd5(%rip),%rdx      # 0x128d <write_data>
0x0000000000001362 <+95>: mov   %ecx,%esi
0x0000000000001364 <+97>: mov   %rax,%rdi
--Type <RET> for more, q to quit, c to continue without paging--RET
0x0000000000001367 <+100>: mov   $0x0,%eax
0x000000000000136c <+105>: call  0x1070 <curl_easy_setopt@plt>
0x0000000000001371 <+110>: mov   -0x8(%rbp),%rax
0x0000000000001375 <+114>: mov   %rax,%rdi
0x0000000000001378 <+117>: call  0x10b0 <curl_easy_perform@plt>
0x000000000000137d <+122>: mov   %eax,-0x14(%rbp)
0x0000000000001380 <+125>: cmpl  $0x0,-0x14(%rbp)
0x0000000000001384 <+129>: je     0x13bb <main+184>
0x0000000000001386 <+131>: mov   -0x14(%rbp),%eax
0x0000000000001389 <+134>: mov   %eax,%edi
0x000000000000138b <+136>: call  0x10d0 <curl_easy_strerror@plt>
0x0000000000001390 <+141>: mov   %rax,%rdx
0x0000000000001393 <+144>: mov   0x2ce6(%rip),%rax      # 0x4080 <stderr@GLIBC_2.2.5>
0x000000000000139a <+151>: lea   0xca7(%rip),%rcx      # 0x2048
0x00000000000013a1 <+158>: mov   %rcx,%rsi
0x00000000000013a4 <+161>: mov   %rax,%rdi
0x00000000000013a7 <+164>: mov   $0x0,%eax
0x00000000000013ac <+169>: call  0x10c0 <fprintf@plt>
0x00000000000013b1 <+174>: mov   $0x1,%eax
0x00000000000013b6 <+179>: jmp   0x159a <main+663>
0x00000000000013bb <+184>: movl  $0x1,0x2d4f(%rip)      # 0x4114 <current_currency>
0x00000000000013c5 <+194>: movl  $0x2712,-0x18(%rbp)
0x00000000000013cc <+201>: mov   -0x18(%rbp),%ecx
0x00000000000013cf <+204>: mov   -0x8(%rbp),%rax
0x00000000000013d3 <+208>: lea   0xc8e(%rip),%rdx      # 0x2068
0x00000000000013da <+215>: mov   %ecx,%esi
```

```
kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Sistemas de Computacion/Practic...
0x00000000000013da <+215>: mov     %ecx,%esi
0x00000000000013dc <+217>: mov     %rax,%rdi
0x00000000000013df <+220>: mov     $0x0,%eax
0x00000000000013e4 <+225>: call    0x1070 <curl_easy_setopt@plt>
0x00000000000013e9 <+230>: mov     -0x8(%rbp),%rax
0x00000000000013ed <+234>: mov     %rax,%rdi
0x00000000000013f0 <+237>: call    0x10b0 <curl_easy_perform@plt>
0x00000000000013f5 <+242>: mov     %eax,-0x14(%rbp)
0x00000000000013f8 <+245>: cmpl    $0x0,-0x14(%rbp)
0x00000000000013fc <+249>: je       0x1433 <main+304>
0x00000000000013fe <+251>: mov     -0x14(%rbp),%eax
0x0000000000001401 <+254>: mov     %eax,%edi
0x0000000000001403 <+256>: call    0x10d0 <curl_easy_strerror@plt>
0x0000000000001408 <+261>: mov     %rax,%rax
0x000000000000140b <+264>: mov     0x2c0e(%rip),%rax # 0x4080 <stderr@GLIBC_2.2.5>
0x0000000000001412 <+271>: lea     0xc2f(%rip),%rcx # 0x2048
0x0000000000001419 <+278>: mov     %rcx,%rsi
0x000000000000141c <+281>: mov     %rax,%rdi
0x000000000000141f <+284>: mov     $0x0,%eax
0x0000000000001424 <+289>: call    0x10c0 <fprintf@plt>
0x0000000000001429 <+294>: mov     $0x1,%eax
0x000000000000142e <+299>: jmp     0x159a <main+663>
0x0000000000001433 <+304>: mov     0x2c2e(%rip),%rax # 0x4068 <valorArsDolar>
0x000000000000143a <+311>: lea     0x2c5f(%rip),%rdx # 0x40a0 <btc_price>
0x0000000000001441 <+318>: mov     %rdx,%rdi
0x0000000000001444 <+321>: movq    %rax,%xmm0
0x0000000000001449 <+326>: call    0x159c <obtenerCotizacion>
0x000000000000144e <+331>: movq    %xmm0,%rax
0x0000000000001453 <+336>: movq    %rax,%xmm0
--Type <RET> for more, q to quit, c to continue without paging--RET
0x0000000000001458 <+341>: lea     0xc44(%rip),%rax # 0x20a3
0x000000000000145f <+348>: mov     %rax,%rdi
0x0000000000001462 <+351>: mov     $0x1,%eax
0x0000000000001467 <+356>: call    0x1030 <fprintf@plt>
0x000000000000146c <+361>: mov     0xc8d(%rip),%rax # 0x2100
0x0000000000001473 <+368>: lea     0x2c26(%rip),%rdx # 0x40a0 <btc_price>
0x000000000000147a <+375>: mov     %rdx,%rdi
0x000000000000147d <+378>: movq    %rax,%xmm0
0x0000000000001482 <+383>: call    0x159c <obtenerCotizacion>
0x0000000000001487 <+388>: movq    %xmm0,%rax
0x000000000000148c <+393>: movq    %rax,%xmm0
0x0000000000001491 <+398>: lea     0xc1a(%rip),%rax # 0x20b2
0x0000000000001498 <+405>: mov     %rax,%rdi
0x000000000000149b <+408>: mov     $0x1,%eax
0x00000000000014a0 <+413>: call    0x1030 <fprintf@plt>
0x00000000000014a5 <+418>: mov     0x2bc4(%rip),%rax # 0x4070 <valorEurDolar>
0x00000000000014ac <+425>: lea     0x2bed(%rip),%rdx # 0x40a0 <btc_price>
0x00000000000014b3 <+432>: mov     %rdx,%rdi
0x00000000000014b6 <+435>: movq    %rax,%xmm0
0x00000000000014bb <+440>: call    0x159c <obtenerCotizacion>
0x00000000000014c0 <+445>: movq    %xmm0,%rax
0x00000000000014c5 <+450>: movq    %rax,%xmm0
0x00000000000014ca <+455>: lea     0xbf0(%rip),%rax # 0x20c1
0x00000000000014d1 <+462>: mov     %rax,%rdi
0x00000000000014d4 <+465>: mov     $0x1,%eax
```

```
kvothe007@EzeLandaeta: ~/Desktop/Universidad/5to AÑO/1ER SEMESTRE/Sistemas de Computacion/Practic...
0x000000000000149b <+408>: mov     $0x1,%eax
0x00000000000014a0 <+413>: call    0x1030 <fprintf@plt>
0x00000000000014a5 <+418>: mov     0x2bc4(%rip),%rax # 0x4070 <valorEurDolar>
0x00000000000014ac <+425>: lea     0x2bed(%rip),%rdx # 0x40a0 <btc_price>
0x00000000000014b3 <+432>: mov     %rdx,%rdi
0x00000000000014b6 <+435>: movq    %rax,%xmm0
0x00000000000014bb <+440>: call    0x159c <obtenerCotizacion>
0x00000000000014c0 <+445>: movq    %xmm0,%rax
0x00000000000014c5 <+450>: movq    %rax,%xmm0
0x00000000000014ca <+455>: lea     0xbf0(%rip),%rax # 0x20c1
0x00000000000014d1 <+462>: mov     %rax,%rdi
0x00000000000014d4 <+465>: mov     $0x1,%eax
0x00000000000014d9 <+470>: call    0x1030 <fprintf@plt>
0x00000000000014de <+475>: mov     0x2b83(%rip),%rax # 0x4068 <valorArsDolar>
0x00000000000014e5 <+482>: lea     0x2bf4(%rip),%rdx # 0x40e0 <eth_price>
0x00000000000014ec <+489>: mov     %rdx,%rdi
0x00000000000014ef <+492>: movq    %rax,%xmm0
0x00000000000014f4 <+497>: call    0x159c <obtenerCotizacion>
0x00000000000014f9 <+502>: movq    %xmm0,%rax
0x00000000000014fe <+507>: movq    %rax,%xmm0
0x0000000000001503 <+512>: lea     0xbc6(%rip),%rax # 0x20d0
0x000000000000150a <+519>: mov     %rax,%rdi
0x000000000000150d <+522>: mov     $0x1,%eax
0x0000000000001512 <+527>: call    0x1030 <fprintf@plt>
0x0000000000001517 <+532>: mov     0xbe2(%rip),%rax # 0x2100
0x000000000000151e <+539>: lea     0x2bbb(%rip),%rdx # 0x40e0 <eth_price>
0x0000000000001525 <+546>: mov     %rdx,%rdi
0x0000000000001528 <+549>: movq    %rax,%xmm0
0x000000000000152d <+554>: call    0x159c <obtenerCotizacion>
0x0000000000001532 <+559>: movq    %xmm0,%rax
0x0000000000001537 <+564>: movq    %rax,%xmm0
0x000000000000153c <+569>: lea     0xb9c(%rip),%rax # 0x20df
0x0000000000001543 <+576>: mov     %rax,%rdi
0x0000000000001546 <+579>: mov     $0x1,%eax
0x000000000000154b <+584>: call    0x1030 <fprintf@plt>
0x0000000000001550 <+589>: mov     0x2b19(%rip),%rax # 0x4070 <valorEurDolar>
0x0000000000001557 <+596>: lea     0x2b82(%rip),%rdx # 0x40e0 <eth_price>
0x000000000000155e <+603>: mov     %rdx,%rdi
0x0000000000001561 <+606>: movq    %rax,%xmm0
0x0000000000001566 <+611>: call    0x159c <obtenerCotizacion>
0x000000000000156b <+616>: movq    %xmm0,%rax
--Type <RET> for more, q to quit, c to continue without paging--RET
0x0000000000001570 <+621>: movq    %rax,%xmm0
0x0000000000001575 <+626>: lea     0xb72(%rip),%rax # 0x20ee
0x000000000000157c <+633>: mov     %rax,%rdi
0x000000000000157f <+636>: mov     $0x1,%eax
0x0000000000001584 <+641>: call    0x1030 <fprintf@plt>
0x0000000000001589 <+646>: mov     -0x8(%rbp),%rax
0x000000000000158d <+650>: mov     %rax,%rdi
0x0000000000001590 <+653>: call    0x1090 <curl_easy_cleanup@plt>
0x0000000000001595 <+658>: mov     $0x0,%eax
0x000000000000159a <+663>: leave
0x000000000000159b <+664>: ret
End of assembler dump.
(gdb) 
```

En las capturas se pueden ver

- los diferentes tipos de registros de 64 bits,
- cómo son llamadas las funciones con call,
- el movimiento de valores entre registros,
- las impresiones,
- las direcciones de cada instrucción que se ejecuta (a la izquierda en azul)
- los saltos a otra instrucción con jmp
- los saltos condicionales con la instrucción je (jump if equal) que realiza el salto a la correspondiente instrucción si los dos valores que compara son iguales

# 3 Bibliografía

## **Documentación Api Rest:**

<https://binance-docs.github.io/apidocs/spot/en/#general-api-information>

## **Ejemplos de Paul Carter:**

<http://pacman128.github.io/pcasm/>

## **Link del repo:**

<https://github.com/TheRabbitt/Sistemas-de-Computacion>