Inspección del stack en llamadas a ensamblador (TP GINI)

Agustín Grupo SdeC 2025

18 de abril de 2025

1. Introducción

Este informe demuestra, mediante **GDB**, cómo inspeccionar el **stack** durante la llamada desde un programa *C* de 32 bits a una rutina en ensamblador (*cdecl*). El caso de prueba convierte un valor flotante (índice GINI) a entero (`floor`) y le suma 1.

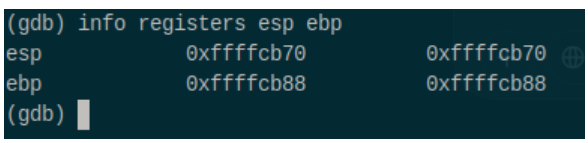
2. Entorno de trabajo

- **Arquitectura:** x86 (32 bits).
- **Compiladores:** GCC (`-m32`), NASM (`-f elf32`).
- **Depurador:** GDB.
- **Entrada:** un `float` leído desde `stdin`.
- **Función ASM:** `calculate_gini_int`.

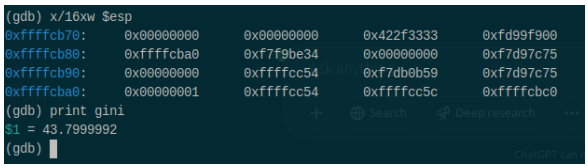
3. Inspección del stack

3.1. Estado previo a la llamada

La fig. 1 muestra el estado del stack y del registro `esp` antes de ejecutar la llamada a ensamblador.



(a) Registros `esp` y `ebp`.



(b) Valor flotante en memoria (`0x422f3333` ≈ 43.8).

Figura 1: Estado del programa antes de la llamada a ASM.

3.2. Ejecución en la rutina ASM

La fig. 2 resalta las instrucciones críticas `fld`, `fistp`, y `add eax, 1` dentro de la función.

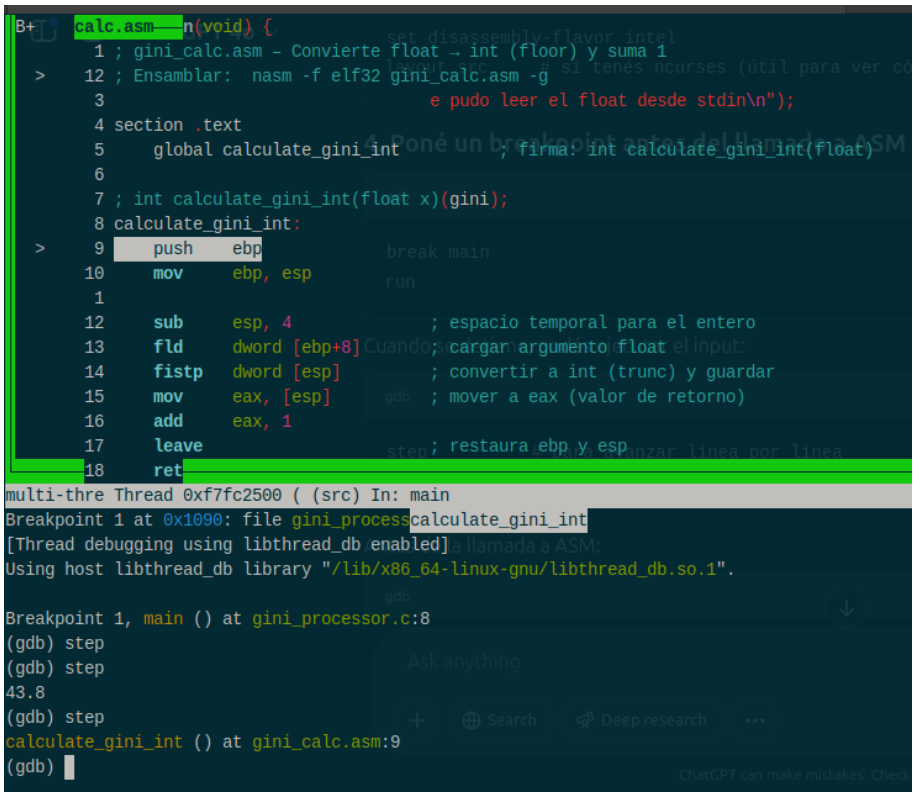
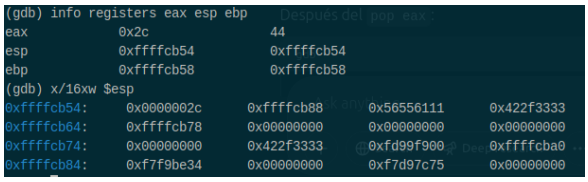


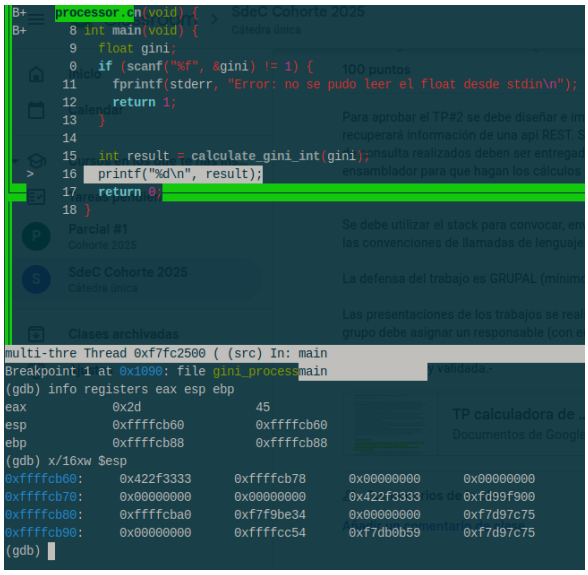
Figura 2: Vista paso a paso dentro de `calculate_gini_int`.

3.3. Retorno a `C`

Tras el `ret`, el resultado entero reside en `EAX` y se copia a `result` (fig. 3).



(a) `EAX` = 0x2D (45).



(b) Variable `result` con el mismo valor.

Figura 3: Estado del programa después de la llamada.

Para mayor detalle, fig. 4 muestra la dirección del `float` en el stack y corrobora el valor devuelto.

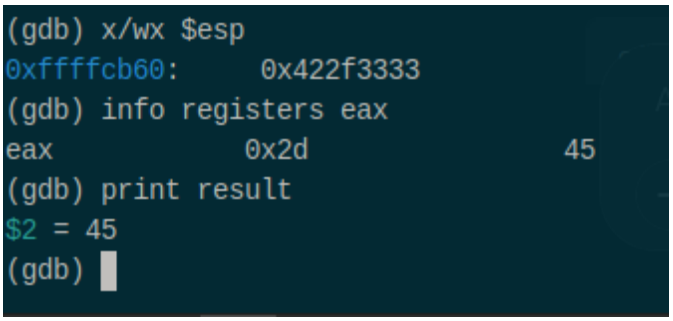


Figura 4: Dirección del argumento y verificación de `EAX`.

4. Diagrama de secuencia completo

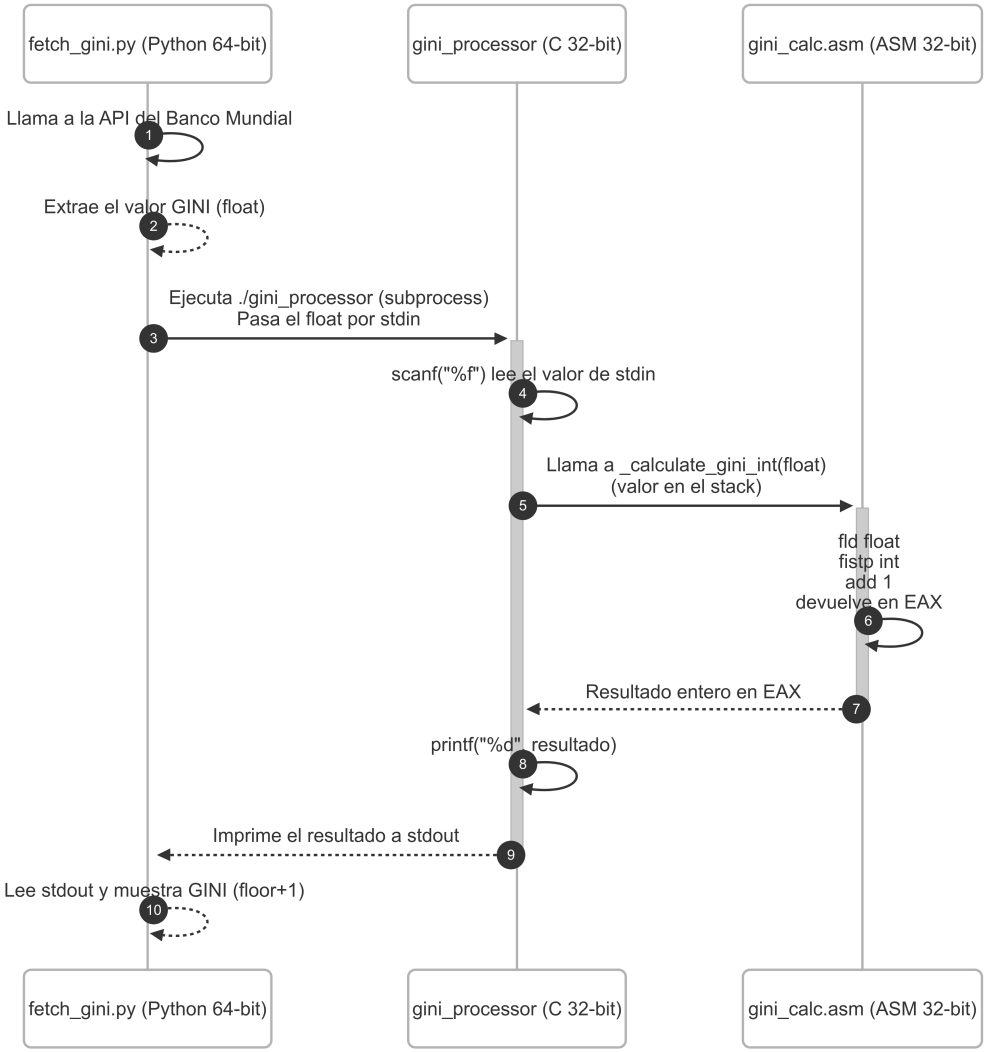


Figura 5: Flujo end-to-end desde Python hasta la rutina ASM.

5. Conclusiones

GDB permitió rastrear el flujo de datos entre C y ensamblador: *argumento en el stack* $\rightarrow$ *FPU* $\rightarrow$ *retorno en* `EAX`. La rutina en ASM cumple con la convención *cdecl* y la variable `result` refleja el valor correcto (GINI truncado + 1).