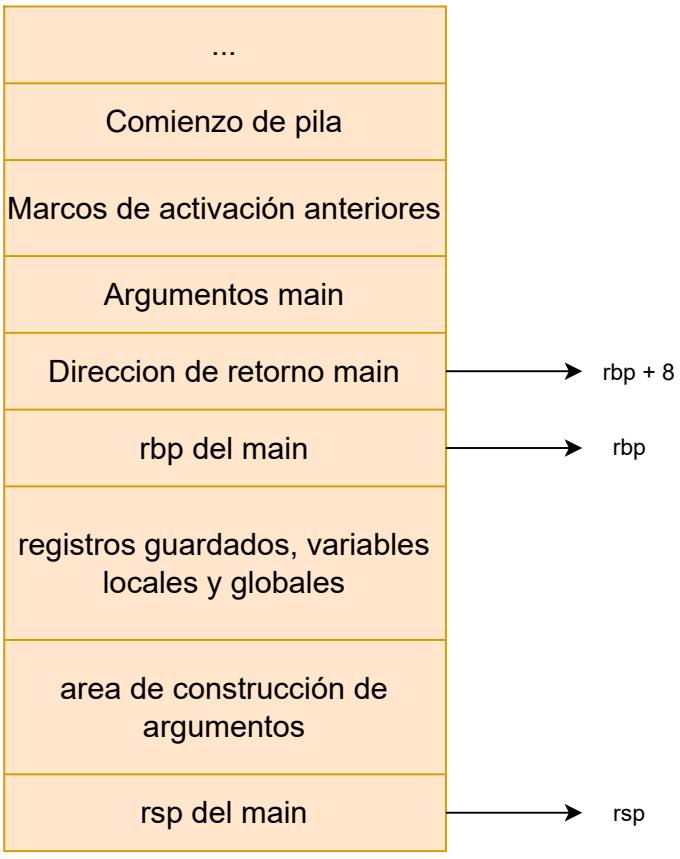
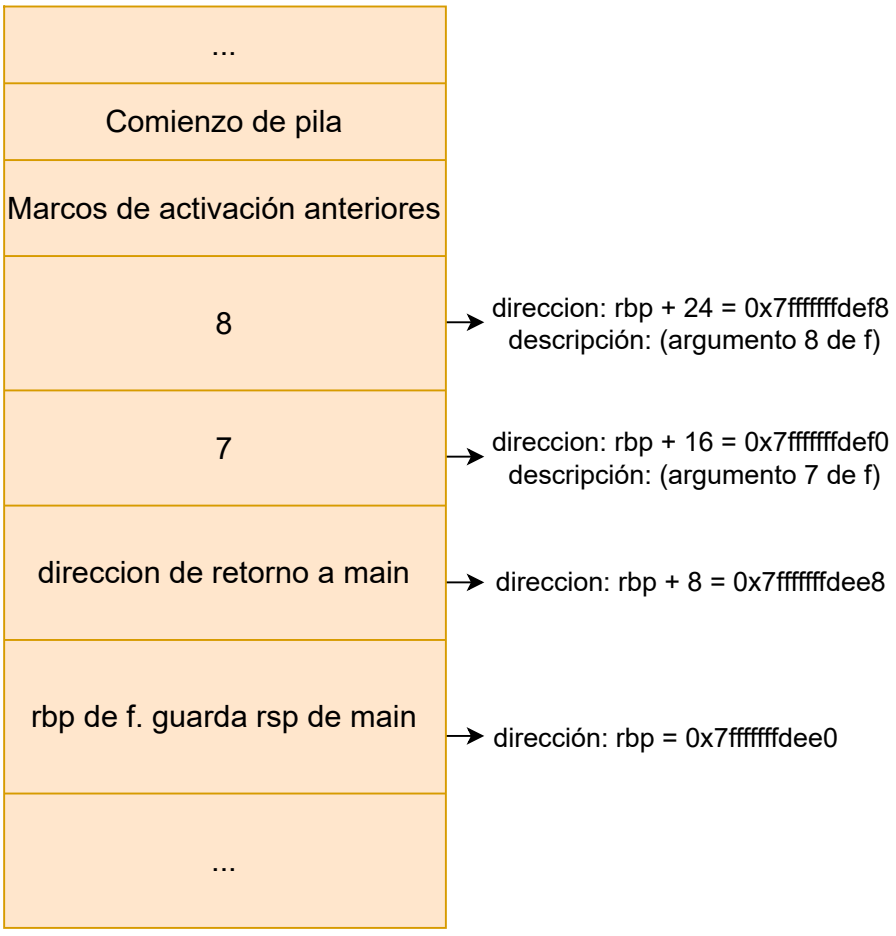
Ejecución linea 14 de 15.c.
Ejecucion lineas 101 ... 117 de 15.s
(archivo generado a partir de 15.c con comando gcc -S 15.c).



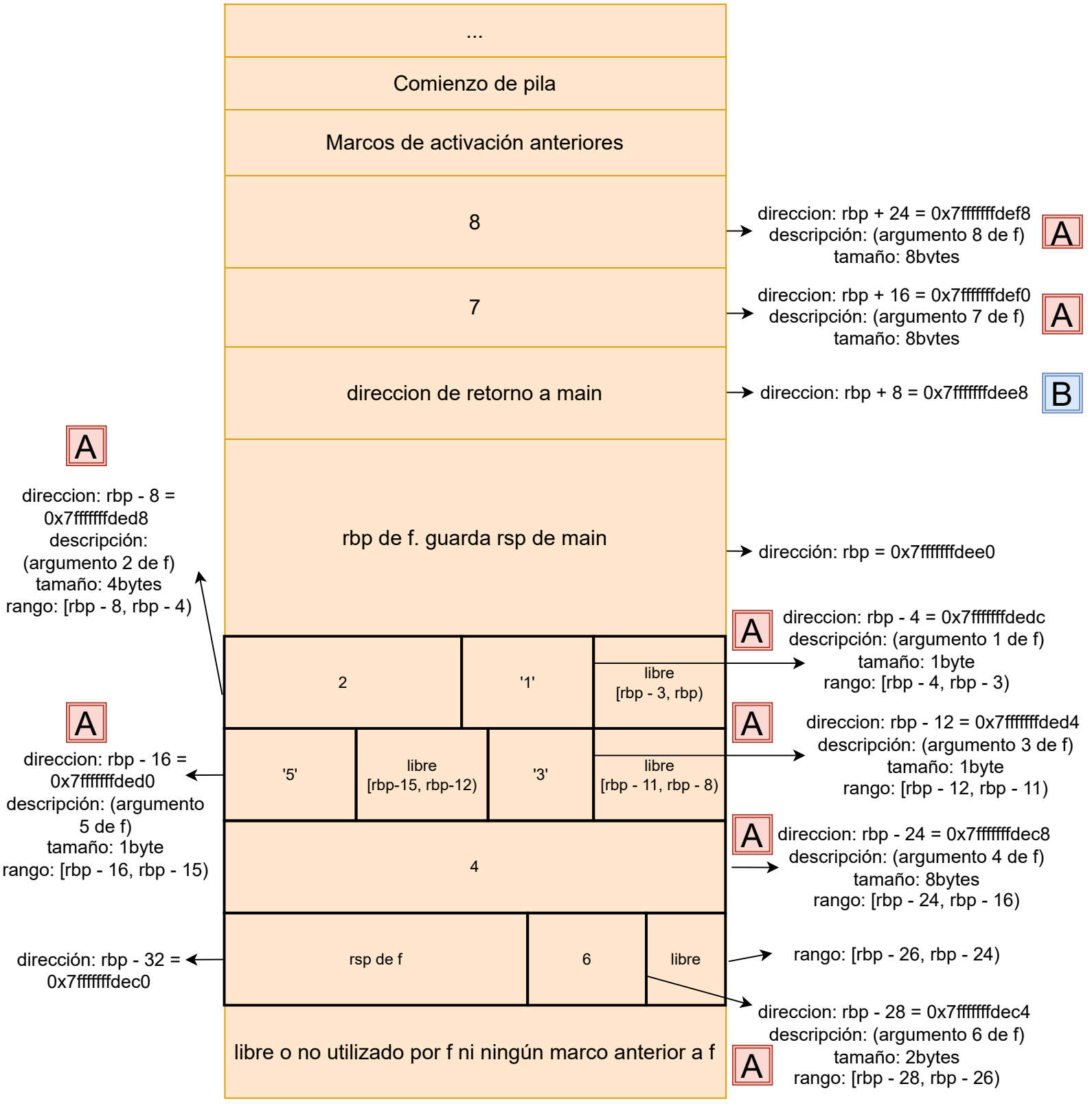
Step linea 15 de archivo 15.c
Step en linea 118 de archivo 15.s

Registros
rdi = '1'
rsi = 2
rdx = '3'
rcx =4
r8 ='5'
r9 = 6



Ejecución linea 33 a 43 de 15.s.
Sigue step linea 15 de 15.c, se podría pensar como el paso anterior al primer printf de f (linea 3 15.c).

Se pasan todos los argumentos que estan en registros (6) a memoria, caso contrario, al llamar a printf se perderian pues la mayoria de esos registros no son preservados.



Observaciones

Ejercicio 15a

Para resolver ejercicio a generamos el archivo 15.s equivalente a 15.c a través del comando "gcc -S 15.c ..."
En el archivo podemos ver que los argumentos 7 y 8 son pasados a pila con el comando pushq lo que está dandoles 8 bytes de tamaño. por otro lado los demas argumentos ocupan el espacio que deben (char 1 byte, short 2 bytes, int 4 bytes, long 8 bytes) y esto se puede ver cuando se pasan los argumentos de los registros a pila en el archivo generado.
Por otro lado, para hallar las direcciones hicimos lo siguiente:
*gdb ./ejecutable_de_15.s.
*br en la linea "call f" (br 119).
*step para meternos dentro de la funcion.
*next hasta que todos los registros sean guardados en pila
*verificación de que los registros en pila coincidan con las ubicaciones dadas en código (lineas 23 a 43 de 15.s)

Ejercicio 15b

Para resolver ejercicio b basta con ver la ubicación de la dirección de retorno en la sección convención de funciones en el apunte (rbp + 8) y con gdb verificarla.

Generales

No hace falta seguir analizando la pila luego de las lineas analizadas, ya que, las llamadas a printf, por convención de llamada, no modificaran datos por encima del rsp en la pila y nuestros argumentos estarán a salvo.
Además como las llamadas a printf solo usan dos argumentos ya no se usará la pila para argumentos de funciones conocidas, solo se usarán los registros rsi y rdi.