HERVÉ SCHAUER CONSULTANTS
Cabinet de Consultants en Sécurité Informatique depuis 1989
Spécialisé sur Unix, Windows, TCP/IP et Internet

GS Days 2010

Return Oriented Programming

Jean-Baptiste Aviat
<Jean-Baptiste.Aviat@hsc.fr>

Hervé Schauer Consultants

- Société de conseil en sécurité informatique depuis 1989
- Prestations intellectuelles d'expertise en toute indépendance
  - Pas de distribution, ni intégration, ni infogérance, ni investisseurs, ni délégation de personnel
- Prestations : conseil, études, audits, tests d'intrusion, formations
- Domaines d'expertise
  - Sécurité Windows / Unix et Linux / embarqué
  - Sécurité des applications
  - Sécurité des réseaux
    - TCP/IP, téléphonie, réseaux opérateurs, réseaux avionique, VoIP, etc.
  - Organisation de la sécurité
- Certifications
  - CISSP, ISO 20000-1 Lead Auditor, ISO 27001 Lead Auditor, ISO 27001 Lead Implementor, ISO 27005 Risk Manager, ITIL, ProCSSI, GIAC GCFA

Buffer overlow : le principe

- Il s'agit d'une erreur de programmation !
- Touche les langages bas niveau...
- ...ou les **bibliothèque** des langages haut niveau !
  - (ou leur coeur..., par exemple PHP : http://www.suspekt.org/)

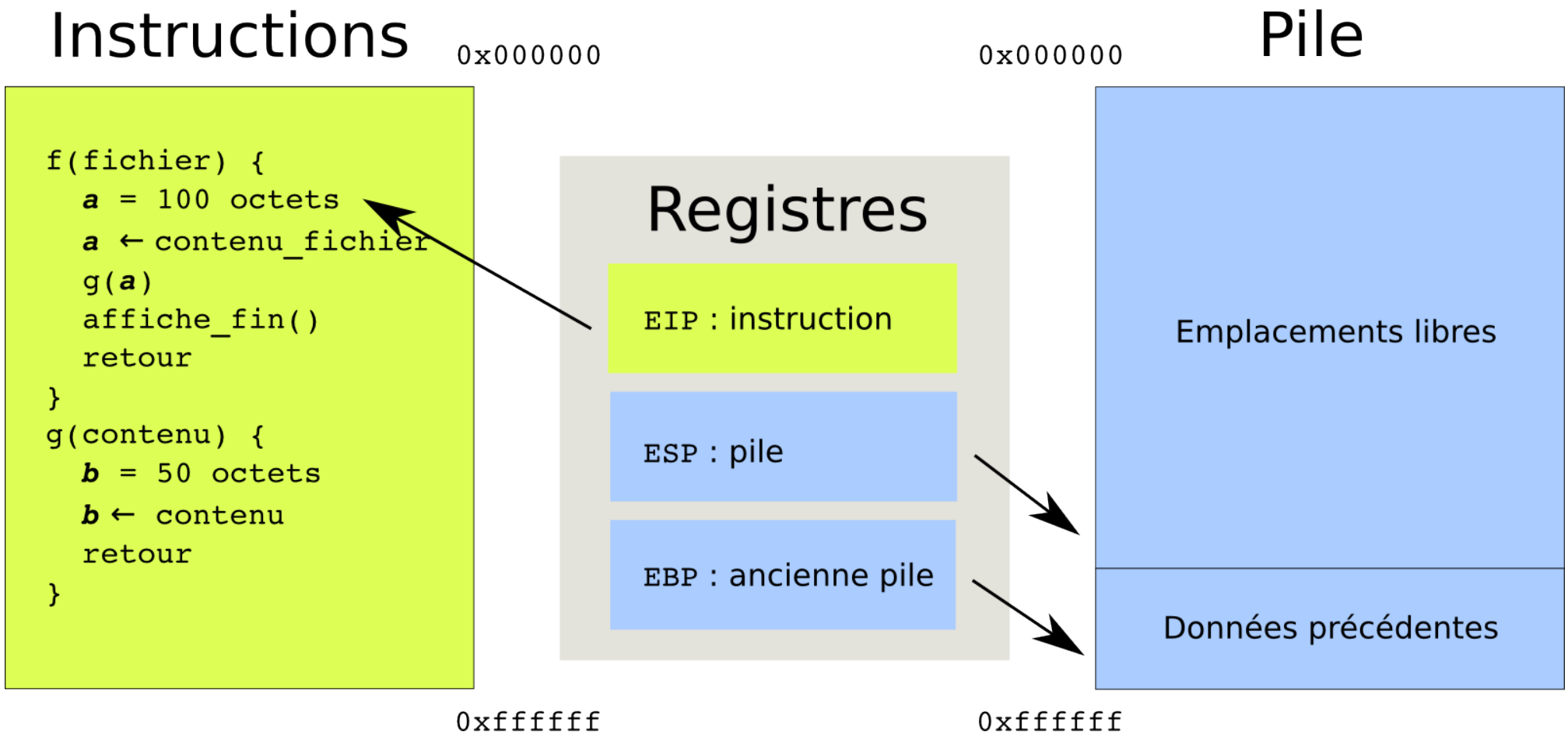
- Une mémoire de capacité *n* octets est réservée
- On écrit y plus de *n* octets
- On empiète sur la mémoire voisine !
- Des données alentour sont écrasées.

Une menace réelle

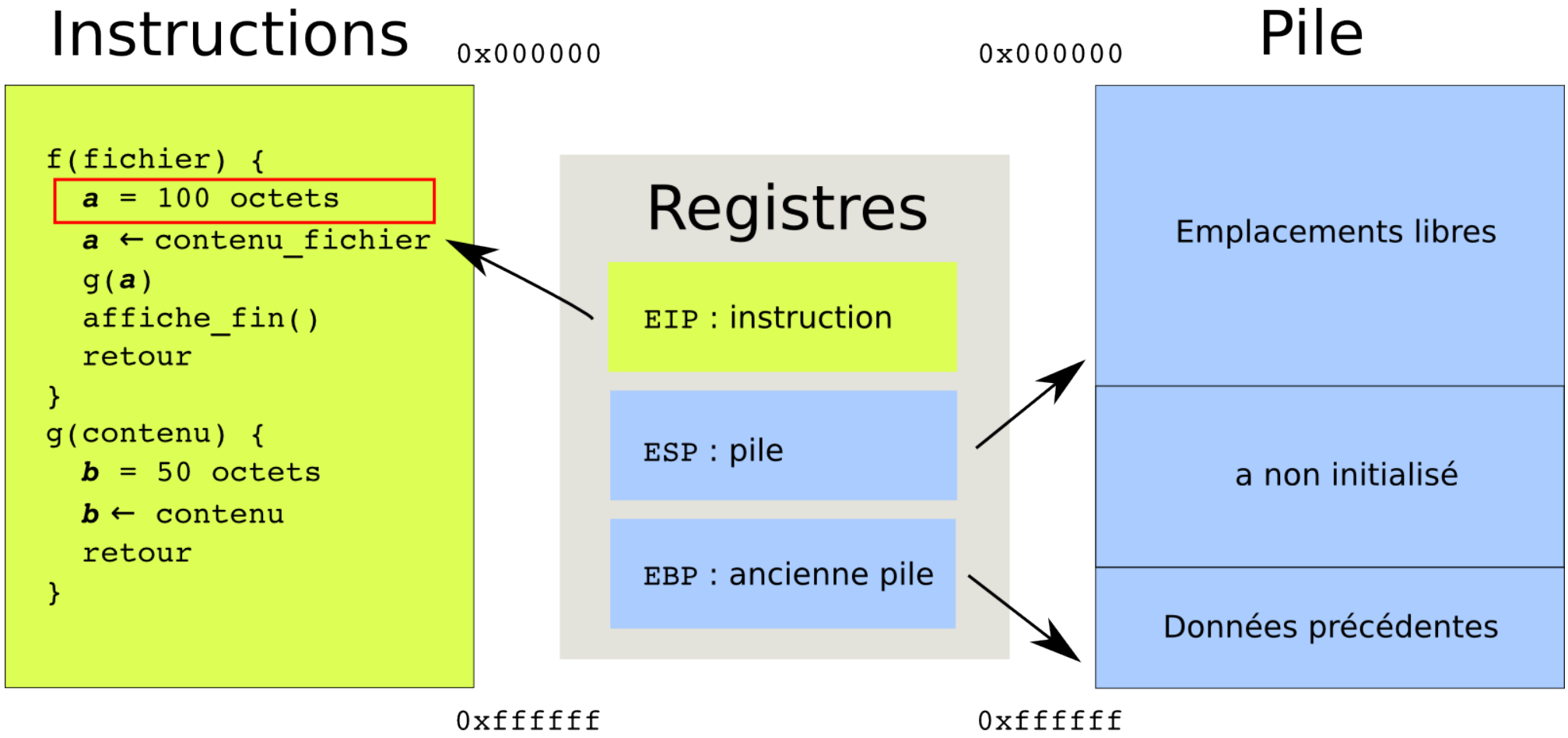
- Code Red (2001), IIS 5.0
- Slammer (2003), MSSQL 2000
- Twilight Hack (2008), Nintendo Wii
- iPhone Jailbreak (2010), jailbreakme.com

- Et de nombreuses vulnérabilités dans la nature...

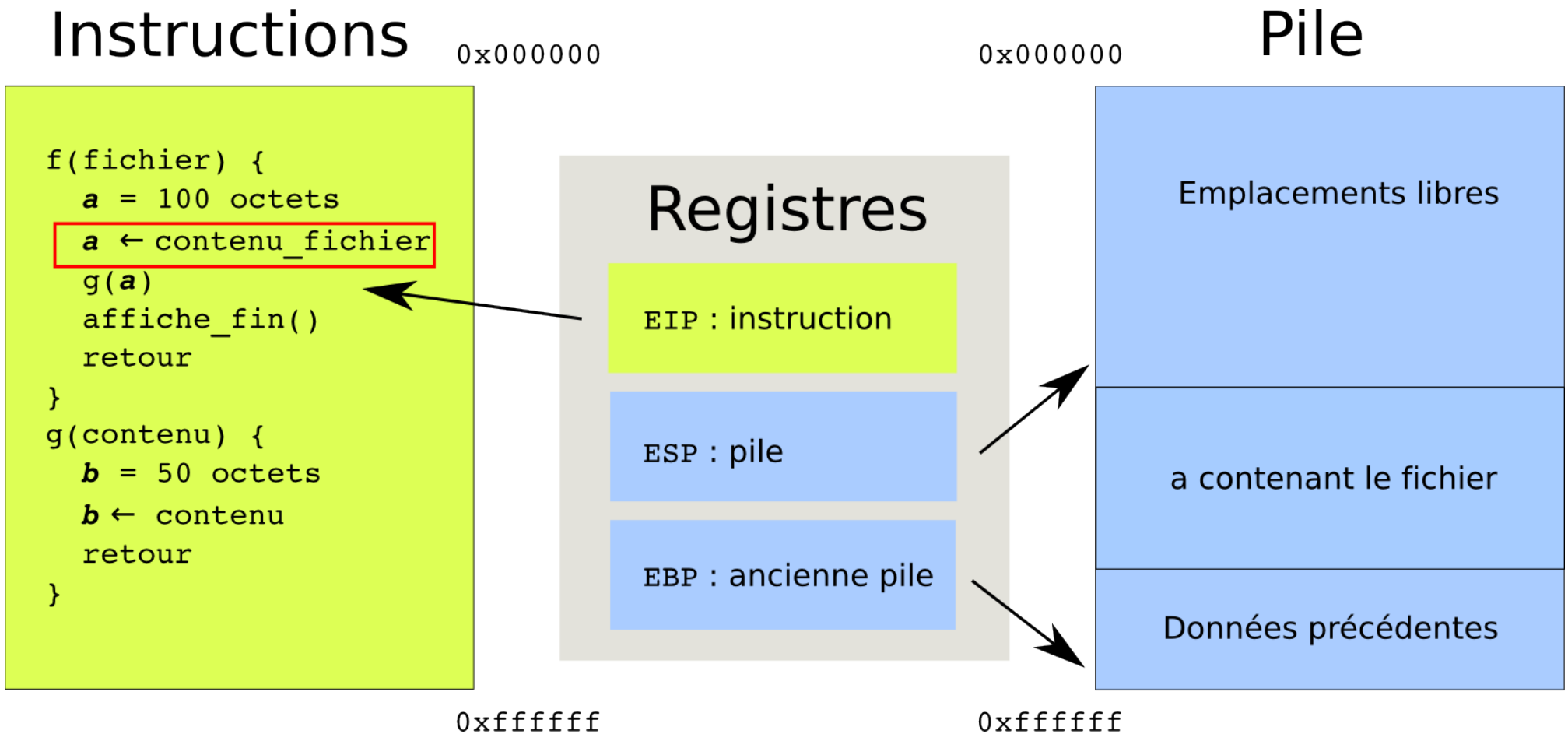
Exécution légitime



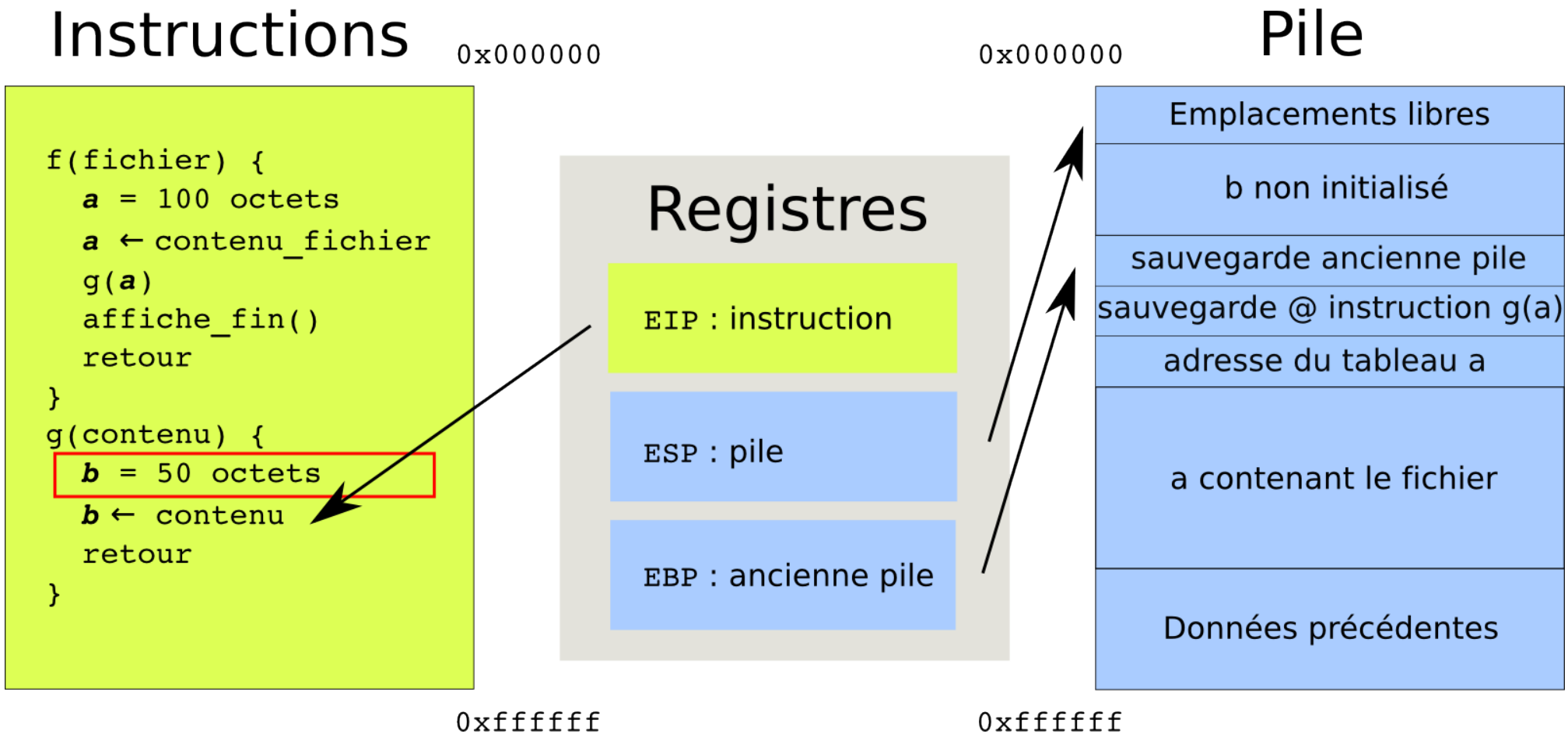
Exécution légitime



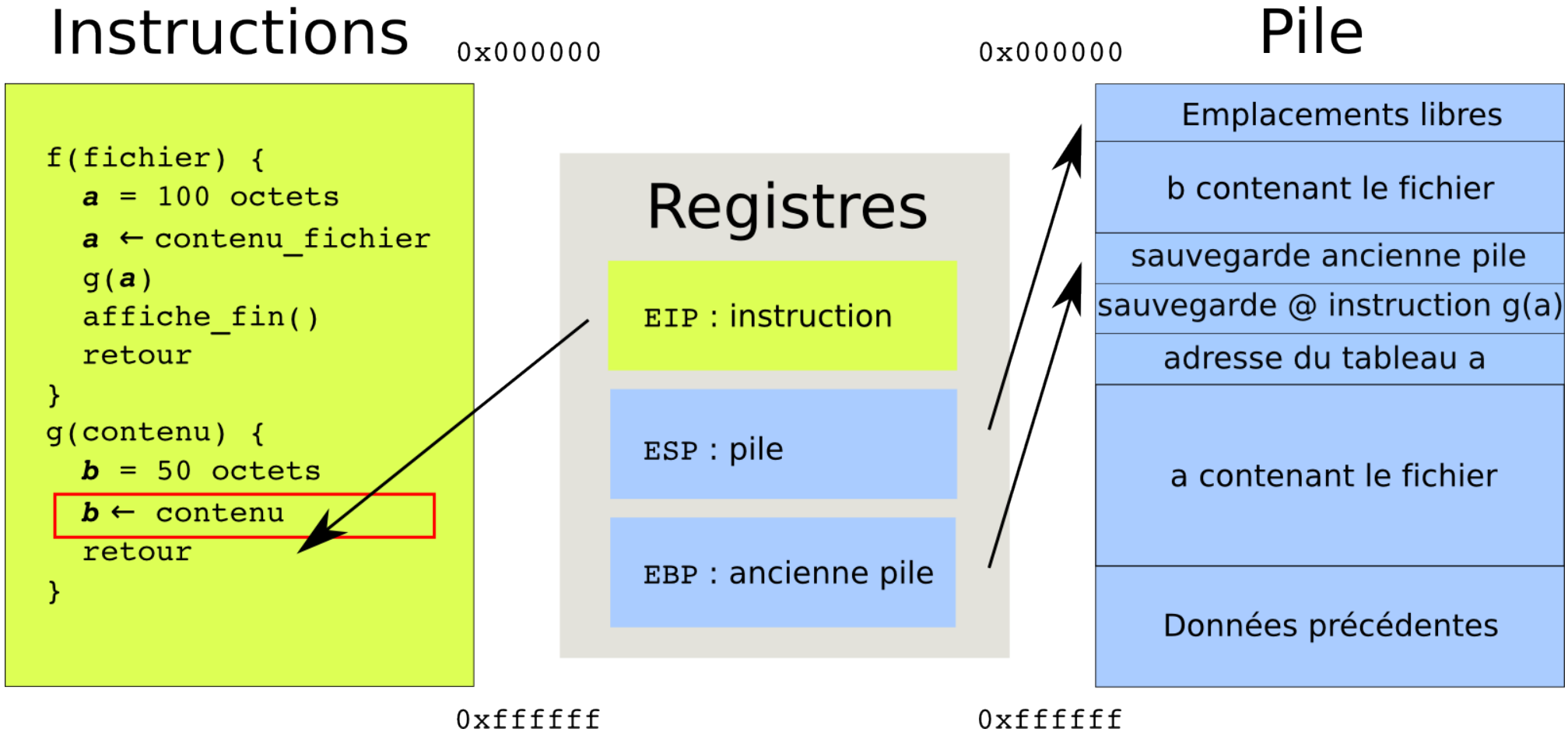
Exécution légitime



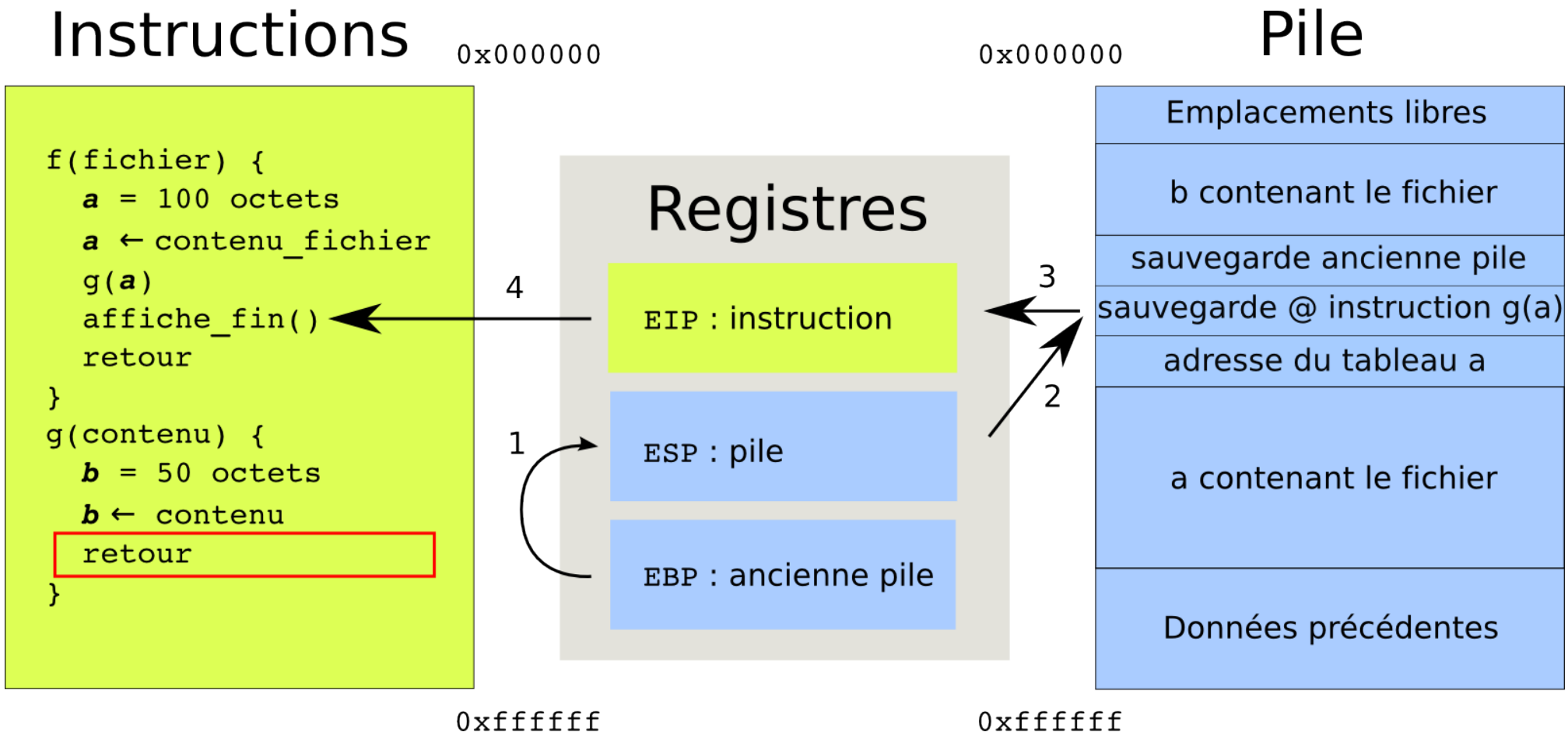
Exécution légitime



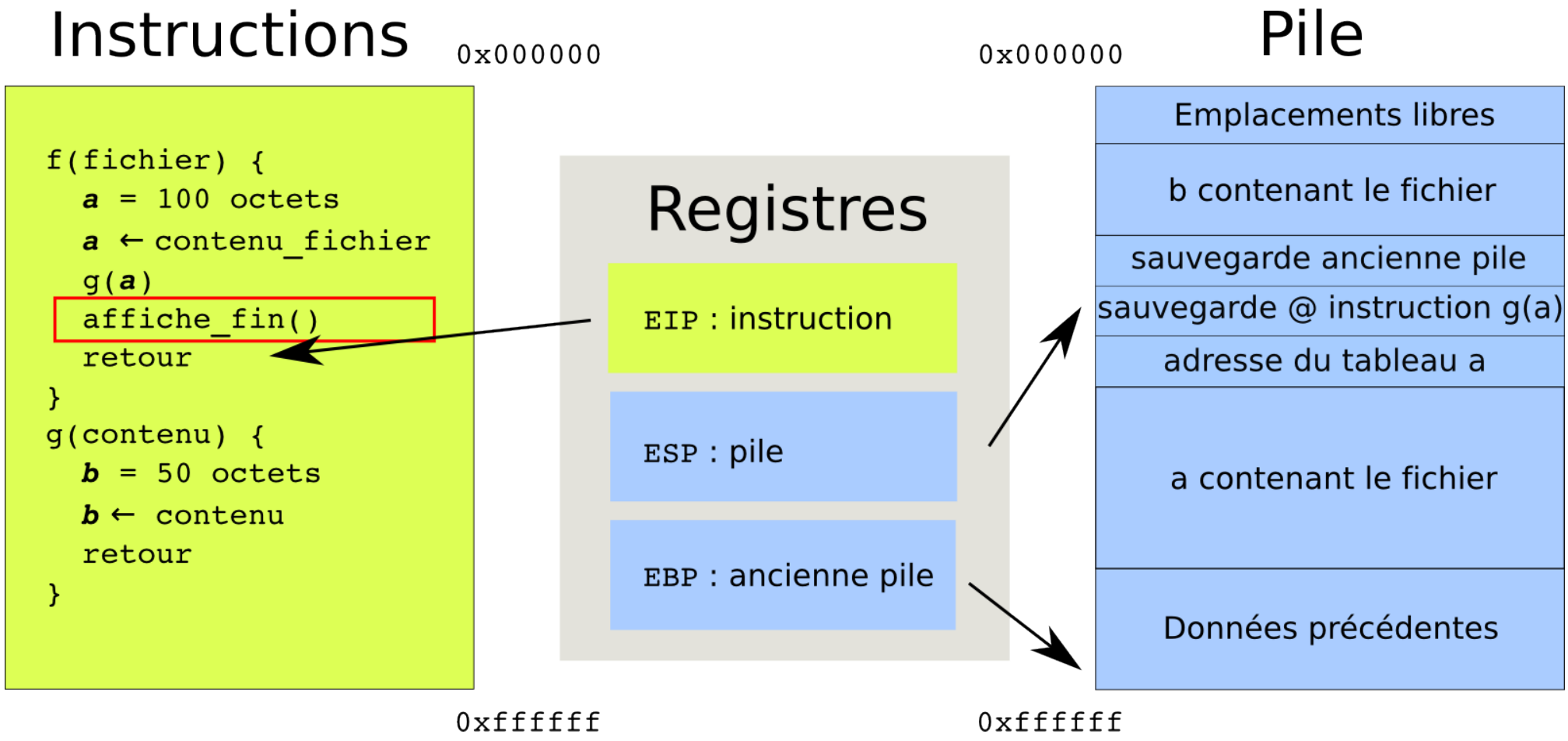
Exécution légitime



Exécution légitime

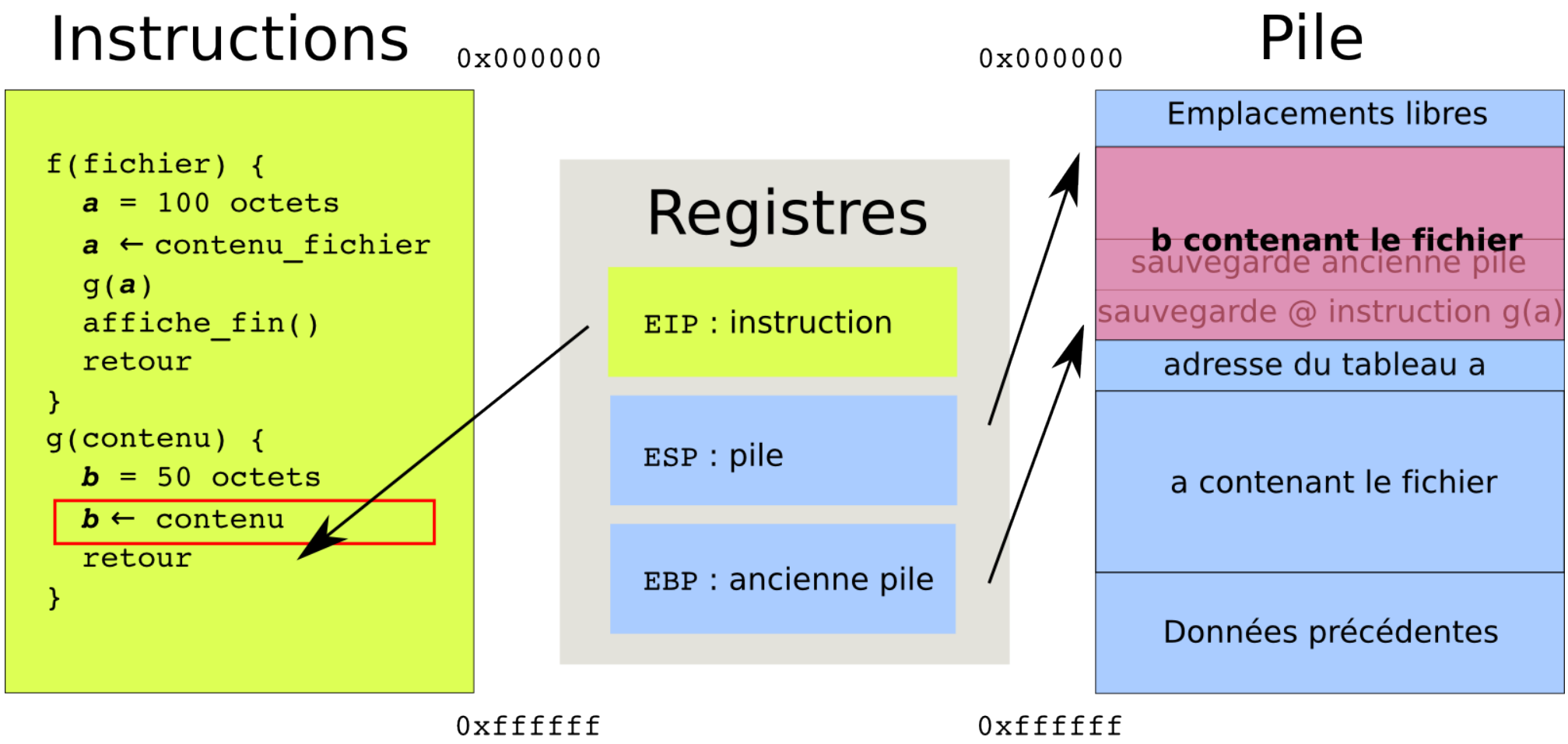


Exécution légitime



Si le fichier contient plus de 50 octets...

Que se passe-t-il ?

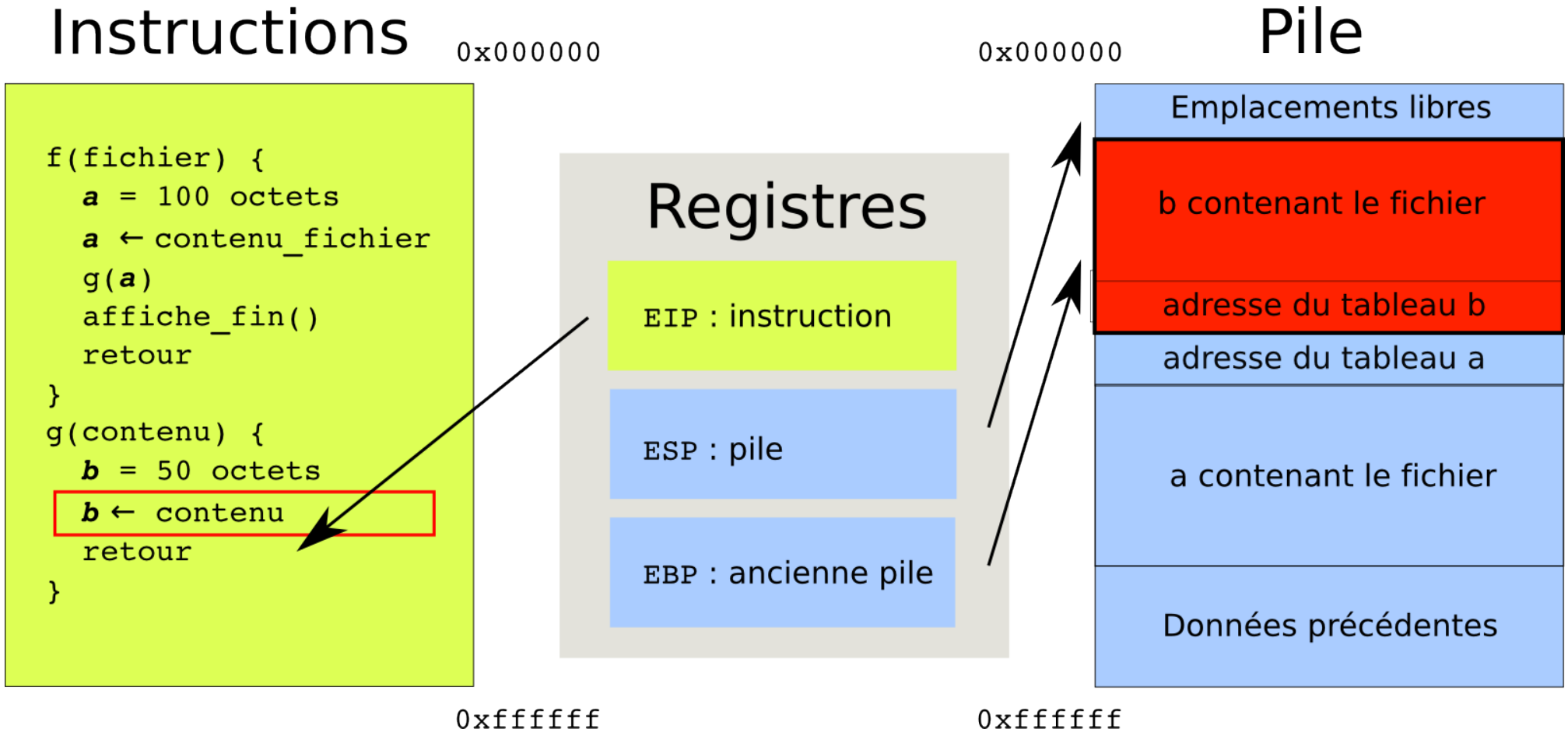


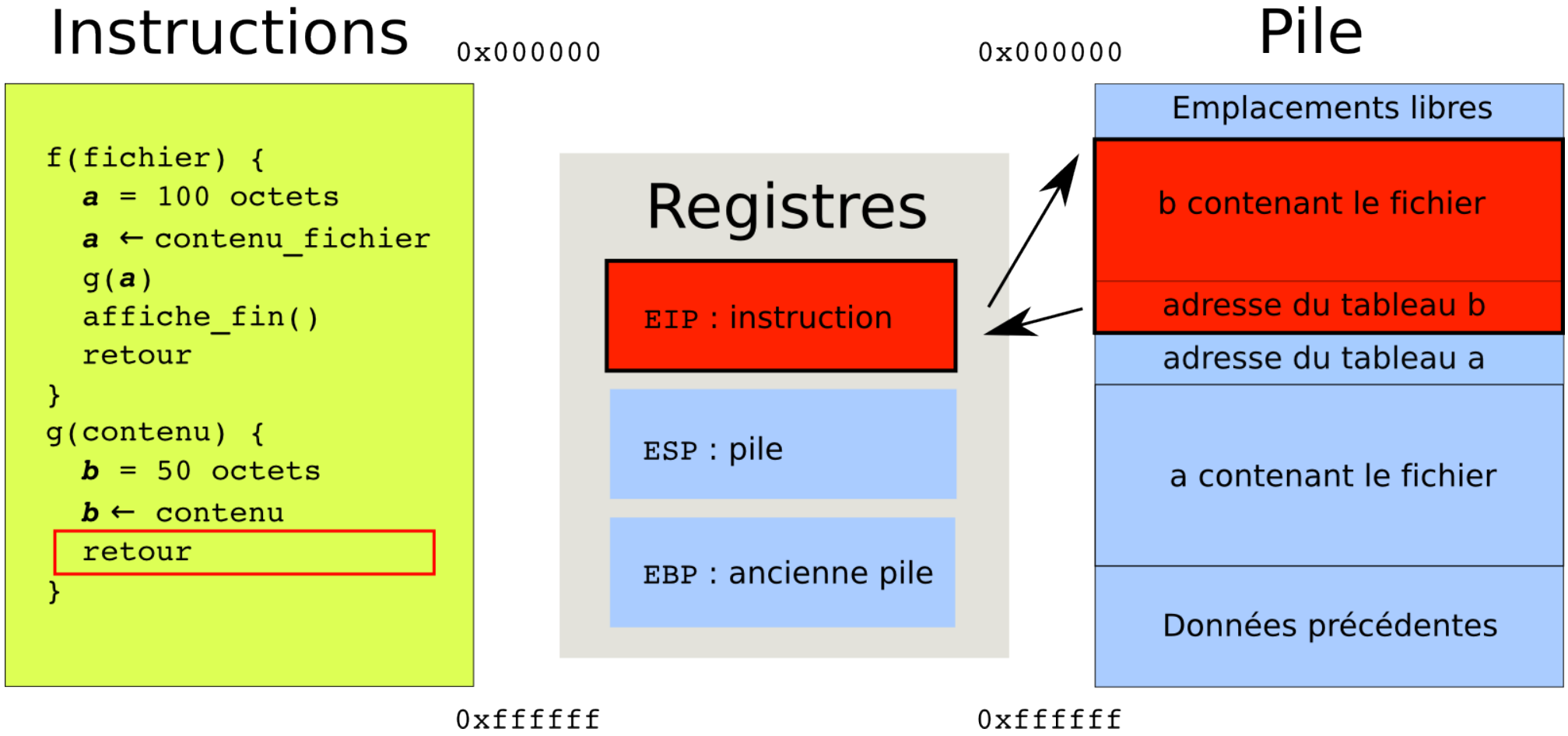
- On écrase l'adresse de retour !
- Le programme plante...

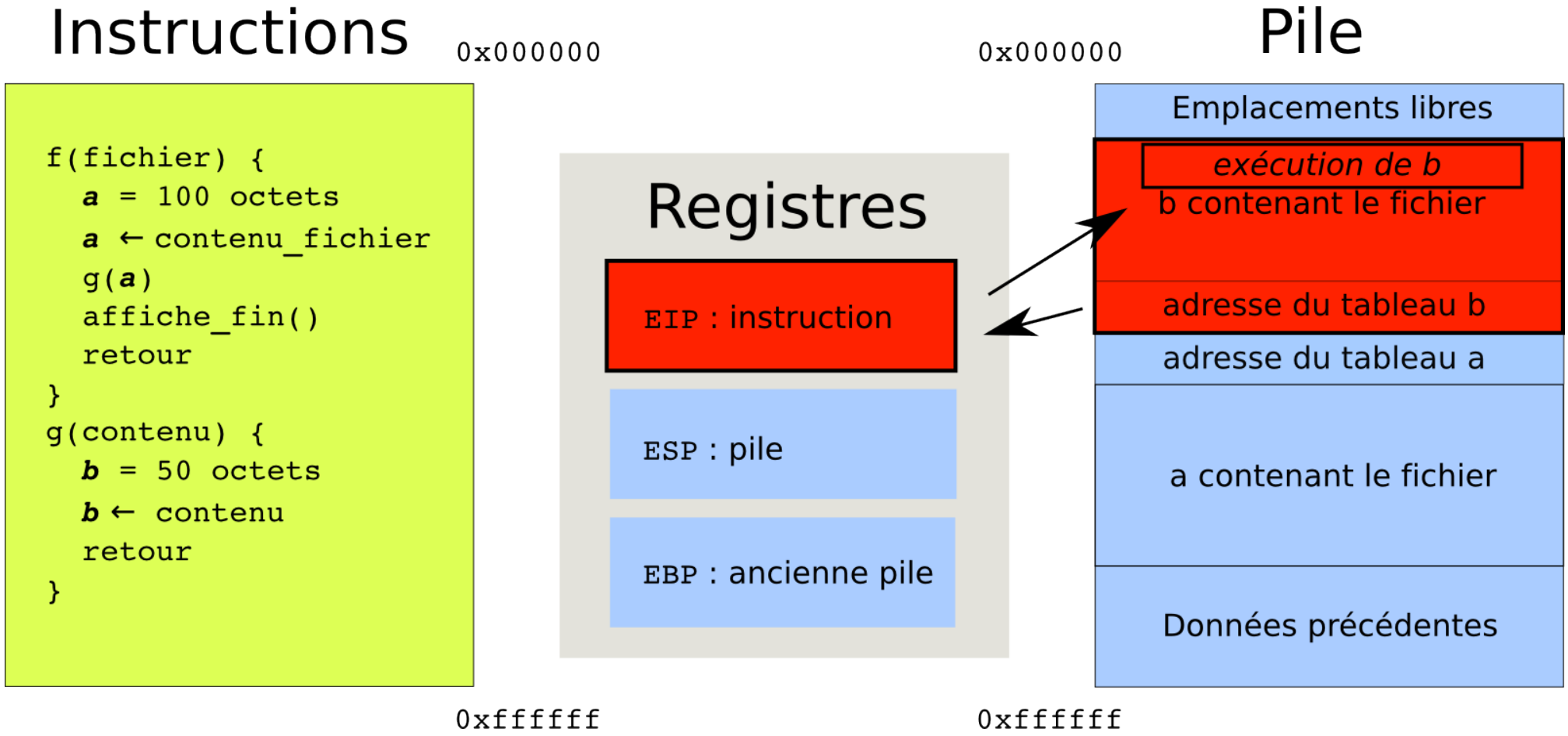
De l’écrasement à l’exécution

- Insertion du code exécutable (compilé) dans le fichier
- La sauvegarde d’EIP est écrasé par l’adresse de ce code
  - c’est à dire le début du tableau ***b***
- Au retour de la fonction...
  - EIP reçoit l’adresse de ***b***

Et notre code est exécuté !





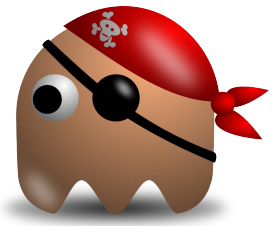


Contres-mesures aux buffers overflows

- Ajouter des « canaris »
  - Valeurs aléatoires présentes sur la pile
  - Vérifiées avant le retour de la fonction,
  - Le programme s’arrête si elles ont été écrasées
- Rendre la pile non exécutable !
  - Le processeur refusera d’exécuter du code situé dans une zone mémoire correspondant à une pile

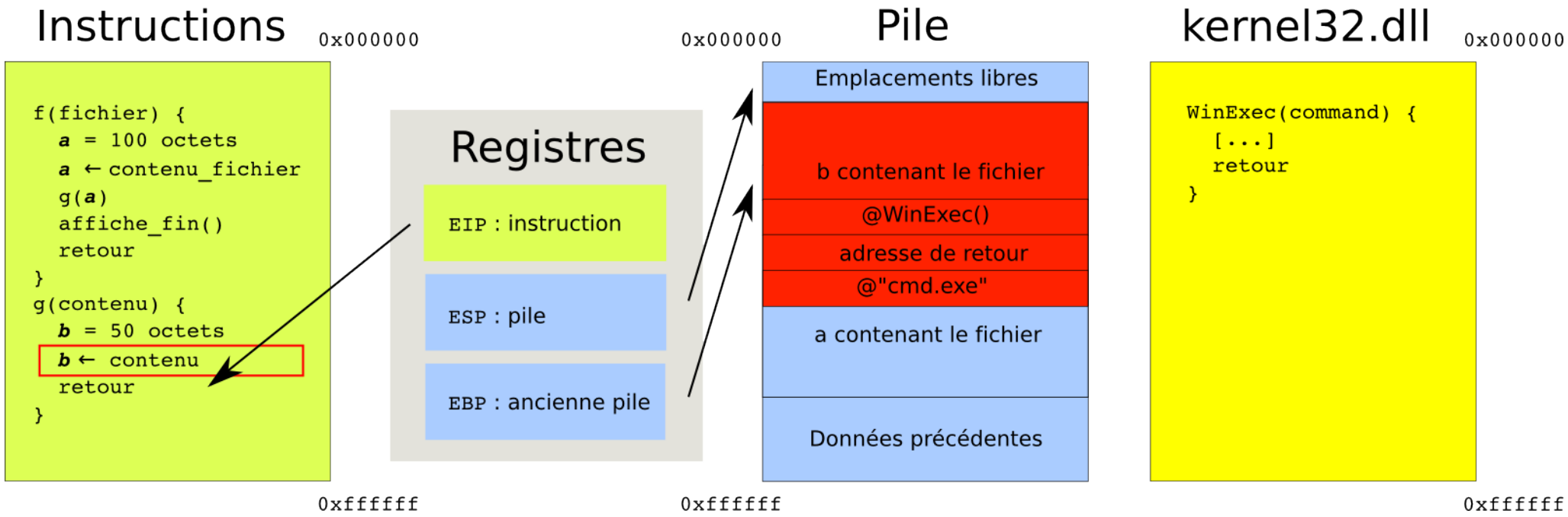
Contournement

- Appeler des fonctions du binaire :
  - System()
  - WinExec()
  - …
- Possibilité d'exécuter des commandes !
- En n'écrivant que les arguments sur la pile
- Et l'adresse de la fonction à appeler

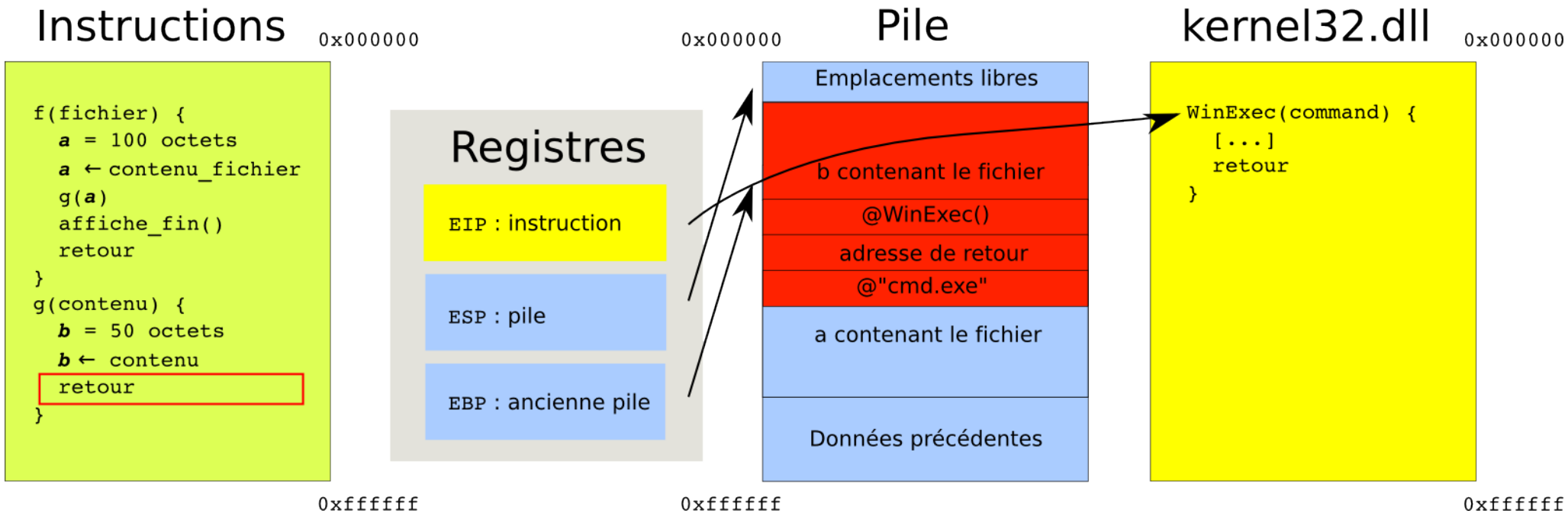


→ « return into libc »

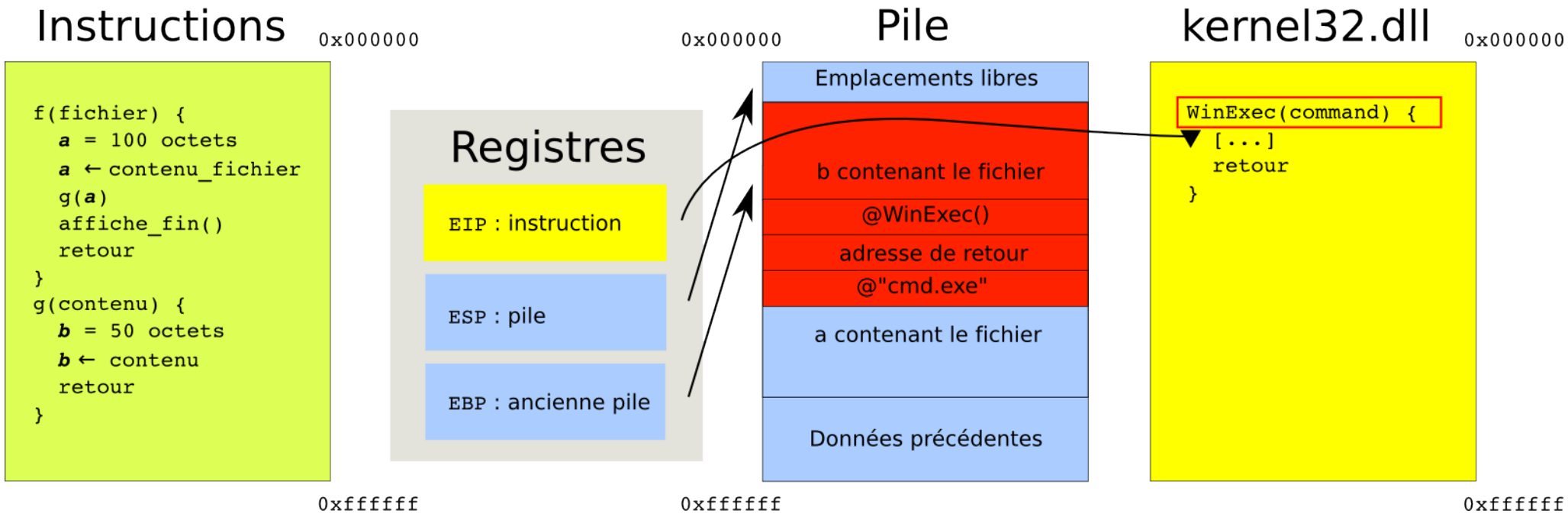
return into libc



return into libc



return into libc

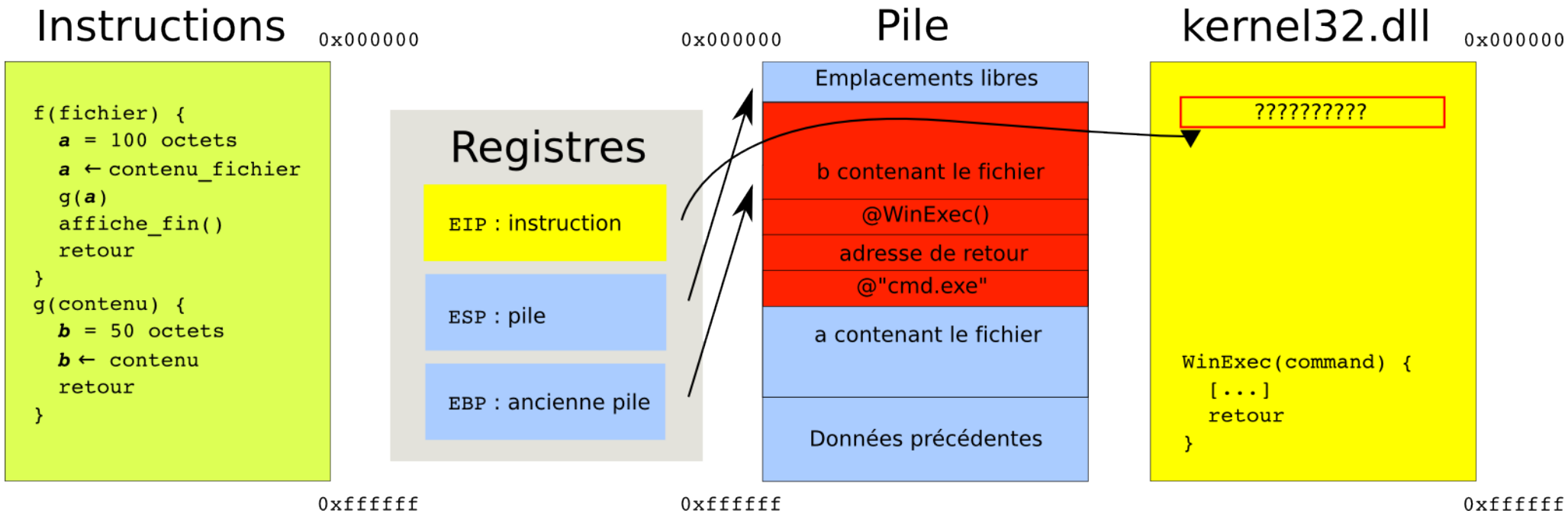


Contres-mesures au ret-into-libc

- Compiler les librairies sans les fonctions dangereuses
  - Complexe, pas toujours possible
- Inclure des 0x00 dans les adresses des fonctions dangereuses
  - Ne protège pas dans tous les cas
- Meilleure solution :
  - Rendre aléatoire l'adresse des fonctions !

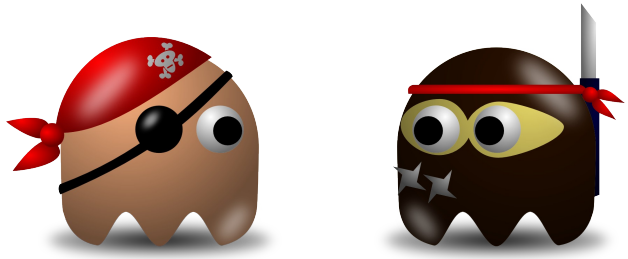
- Pour ce faire, les librairies sont chargées avec une adresse aléatoire
- L'attaquant ne connaît plus l'adresse de la fonction

Utilité de l'ASLR



Contournement : ROP

- Return Oriented Programming ! (enfin)
- Principe :
  - Utiliser des bouts de code en mémoire
    - Nommés « gadgets »
  - Terminés par des : `RET ↔ POP EIP`
- Nécessite :
  - au moins une librairie sans ASLR
  - suffisamment grande

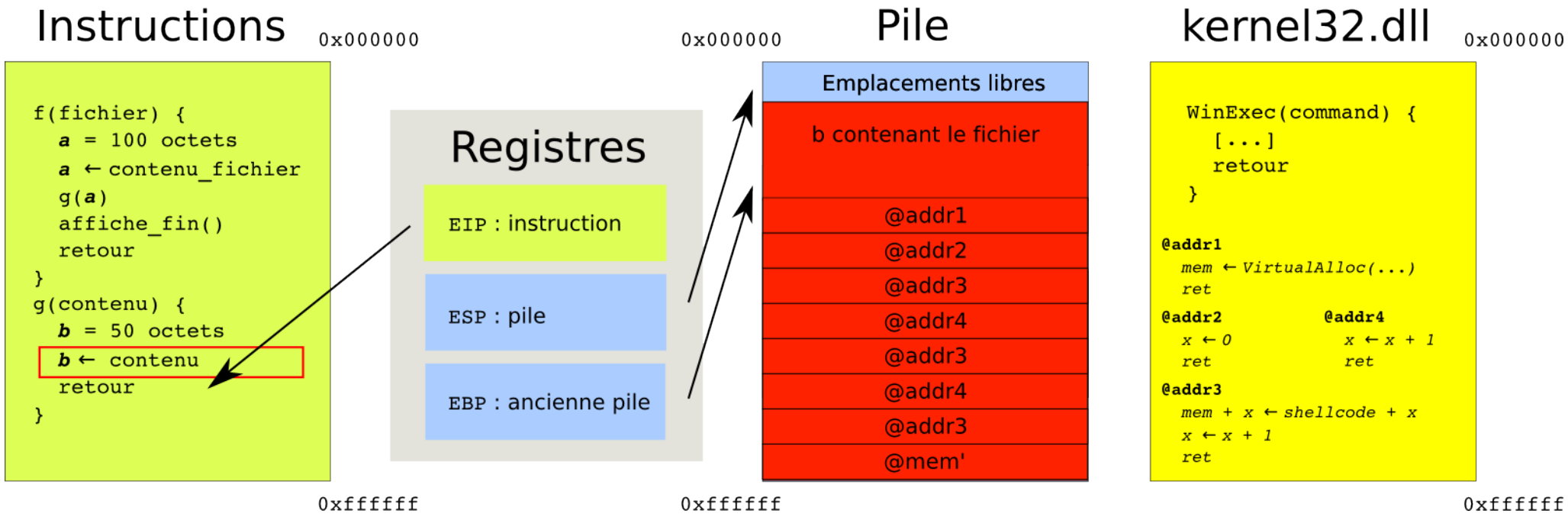


ROP : Exemple de fonctionnement

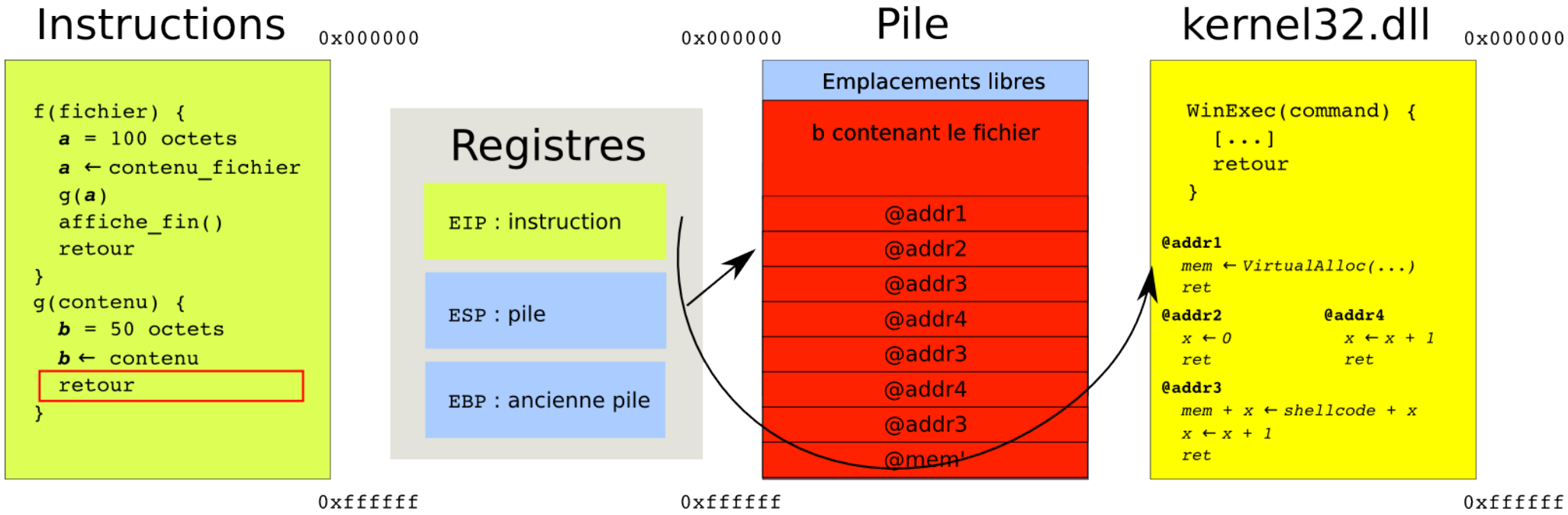
- Copie du shellcode en zone non exécutable
  - Simplement l'overfow
- Transformation d'une zone NX en zone X
  - Avec VirtualAlloc()
- Copie du shellcode dans cette zone
- Exécution du shellcode !

```
VirtualAlloc(0, 0x200, MEM_COMMIT, PAGE_EXECUTE_READWRITE) ;
```

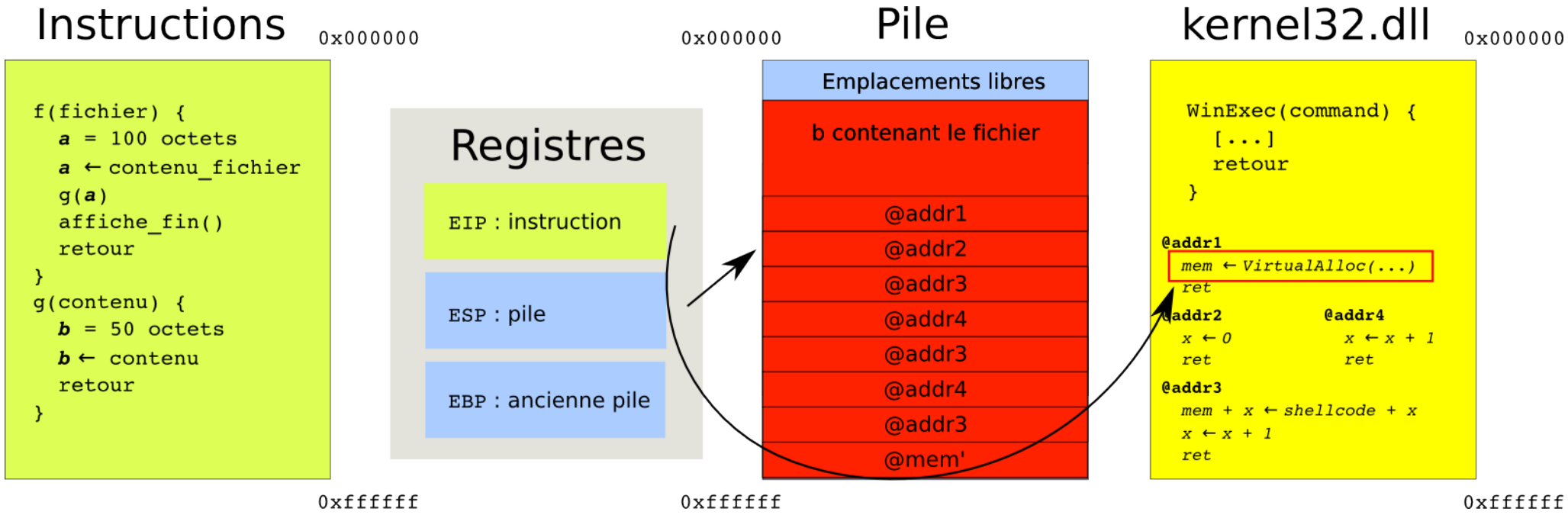
ROP : Exemple de fonctionnement



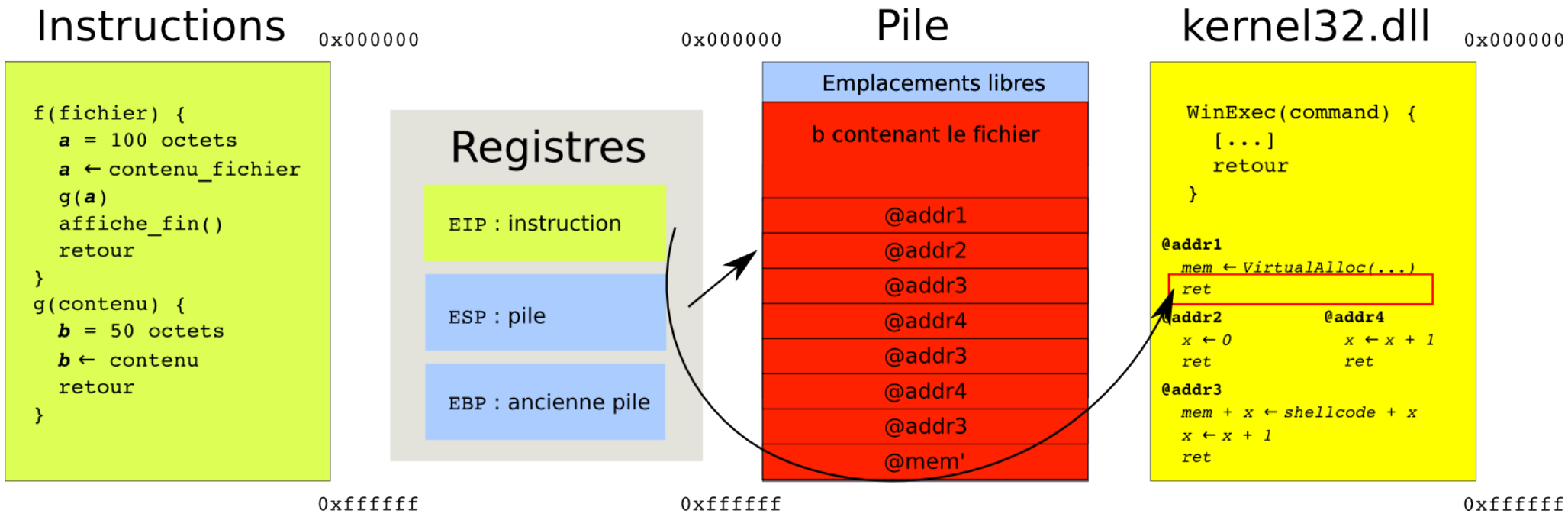
ROP : Exemple de fonctionnement



ROP : Exemple de fonctionnement



ROP : Exemple de fonctionnement



ROP : caractéristiques

- Permet d'introduire des conditions ou des boucles
  - Dans le code injecté
  - Preuve qu'il est inutile de minimiser les bibliothèques
- Autorise l'exécution avec :
  - NX (OS « récents »)
  - zones mémoires signées (iPhone)
- Nécessite :
  - De la place sur la pile
  - Une construction plus ou moins complexe

Recherche des gadgets

- Besoin de lister les instructions RET = 0xC3
- Puis les instructions précédentes
- Puis les ordonner afin de pouvoir les choisir

Combien de gadgets ?

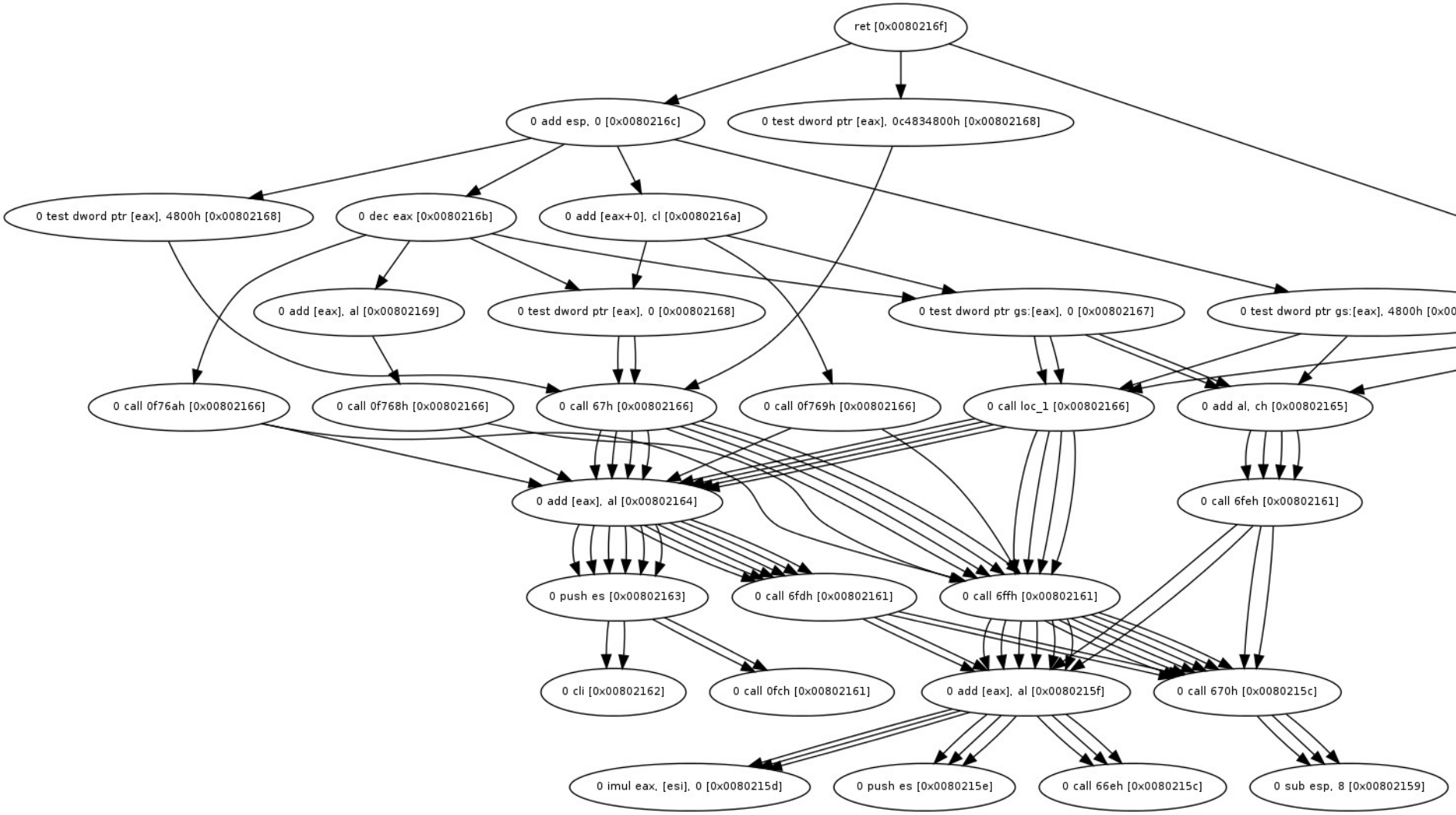
- Pour **une** instruction RET

```
entrypoint_0:
    or al, ch
    imul eax, [esi], 0
    add al, ch
    cli
    push es
    add [eax], al
    call 0f775h
    dec eax
    add esp, 8
    ret
```

```
; @0   08e8
; @2   6b0600
; @5   00e8
; @7   fa
; @8   06
; @9   0000
; @0bh   e865f70000   x:0f775h
; @10h   48
; @11h   83c408
; @14h   c3   x:unknown
```

Combien de gadgets ?

- Pas 1 ;)



Recherche des gadgets

- Outil interne HSC
  - Ruby
    - Basé sur Metasm (https://github.com/jjyg/metasm/)
  - Multi-architectures
    - Comme Metasm : ELF, PE, MachO...
  - Profondeur de recherche arbitraire
- Autres outils
  - RopeME, www.vnsecurity.net/2010/08/ropeme-rop-exploit-made-easy
    - Linux uniquement
  - Immunity Debugger plugins, http://seanhn.wordpress.com/
    - non public
  - Metasploit ?

Choix des gadgets

- Méthode Zynamics :
  - Une thèse !
  - Et un compilateur de gadgets
- Méthode « standard » :
  - À la main !
  - Grep et petits scripts

Merci !

Des questions ?