# 贪吃蛇
# 反汇编代码分析报告

THINCT

December 9, 2023

# SnakeGame::update

# EBX 代替当前的函数栈底

```
.text:004079C0    push    ebx
.text:004079C1    mov     ebx, esp
.text:004079C3    sub     esp, 8
.text:004079C6    and     esp, -8
.text:004079C9    add     esp, 4
.text:004079CC    push    ebp
.text:004079CD    mov     ebp, [ebx+4]
.text:004079D0    mov     [esp+4], ebp
.text:004079D4    mov     ebp, esp
```

1. 当 eip 在.text:004079C0 处，esp 所指向的是 ret addr.
2. 当 eip 在.text:004079C1 处，ebx 所指向的是 esp-4. 此时:
   - ebx+4 指向的是 ret addr
   - ebx+8 指向的是第一个参数

# EBX 代替当前的函数栈底

```
.text:004079C3    sub     esp, 8
.text:004079C6    and     esp, 0FFFFFFF8h
.text:004079C9    add     esp, 4
.text:004079CC    push    ebp
```

esp 实现了向下最近的 8 的倍数取证。比如 12 取整就是 8，16 取整就是 16，18 取整就是 16. 因为是针对栈结构地址取整，所以越是往小的方向越安全，因为对于栈结构来讲，越小的地址是没有用过的地址。所以后面的 ebp,esp, ebp 只能作为局部变量的索引，而对于参数的索引，用 ebx 比较合适。

**总结:**

对于这个函数来讲，并不是按照套路 *ebp* 作为局部变量和函数参数的唯一参考.