

逆向工程

CheatEngine入門

\$ whoami

- ▶ 林昆立 / Inndy
- ▶ 奧義智慧科技資安研究員
 - ▶ 惡意程式分析、Windows 安全攻防技術
- ▶ 逆向工程愛好者
 - ▶ 從國中開始學逆向工程
 - ▶ 累積十年以上 Programming 經驗，分析過數百個程式
- ▶ 演講經歷
 - ▶ 2020 BlackHat USA
 - ▶ 2020 HITB Lockdown 002
 - ▶ 2019 ROOTCON

遊戲修改入門



遊戲修改入門

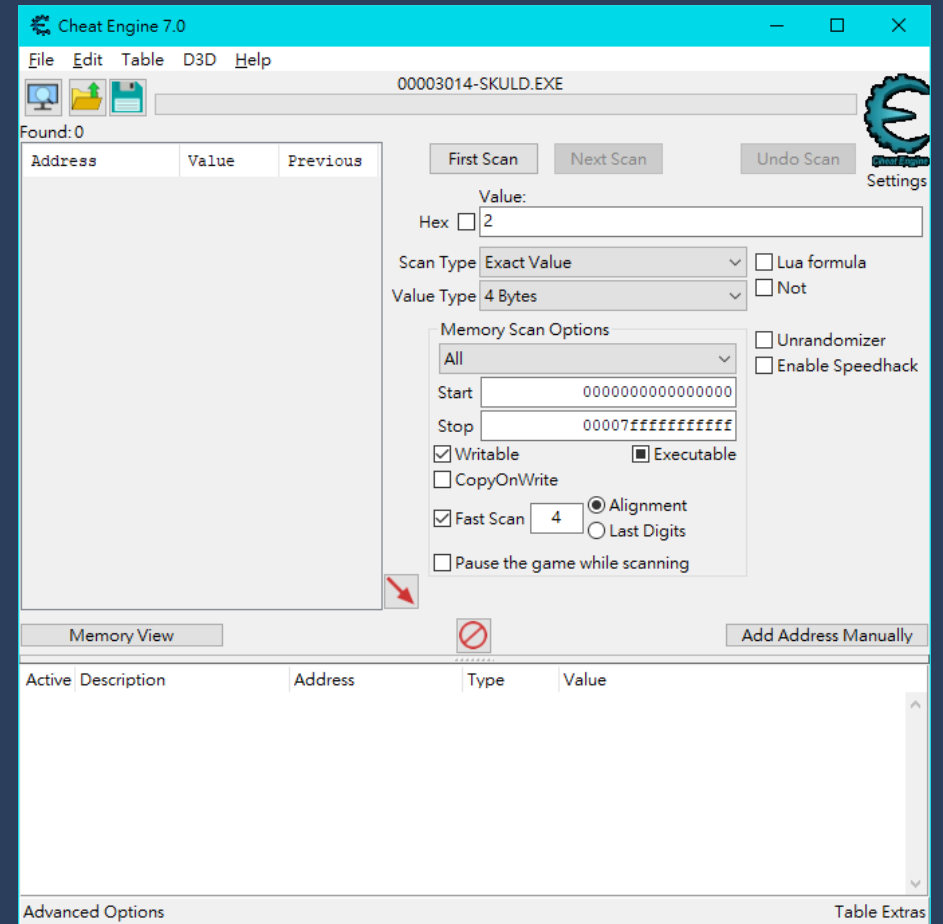
- ▶ 修改遊戲內的數值，如金錢、血量等
- ▶ 修改遊戲的組合語言指令碼，改變程式行為

遊戲修改入門

- ▶ 從已知或容易觀察的數值開始搜索 (金錢、生命值、魔力值)
 - ▶ 直接修改數值 → 結案
- ▶ 利用 "Find Out What {Write, Access} To This Address" 找到讀寫特定記憶體指令
 - ▶ 把該指令換成 "nop"，使該數值不再變動 → 結案
- ▶ 使用 Auto Assemble 功能寫客製化的 hook，自己做更多的行為修改

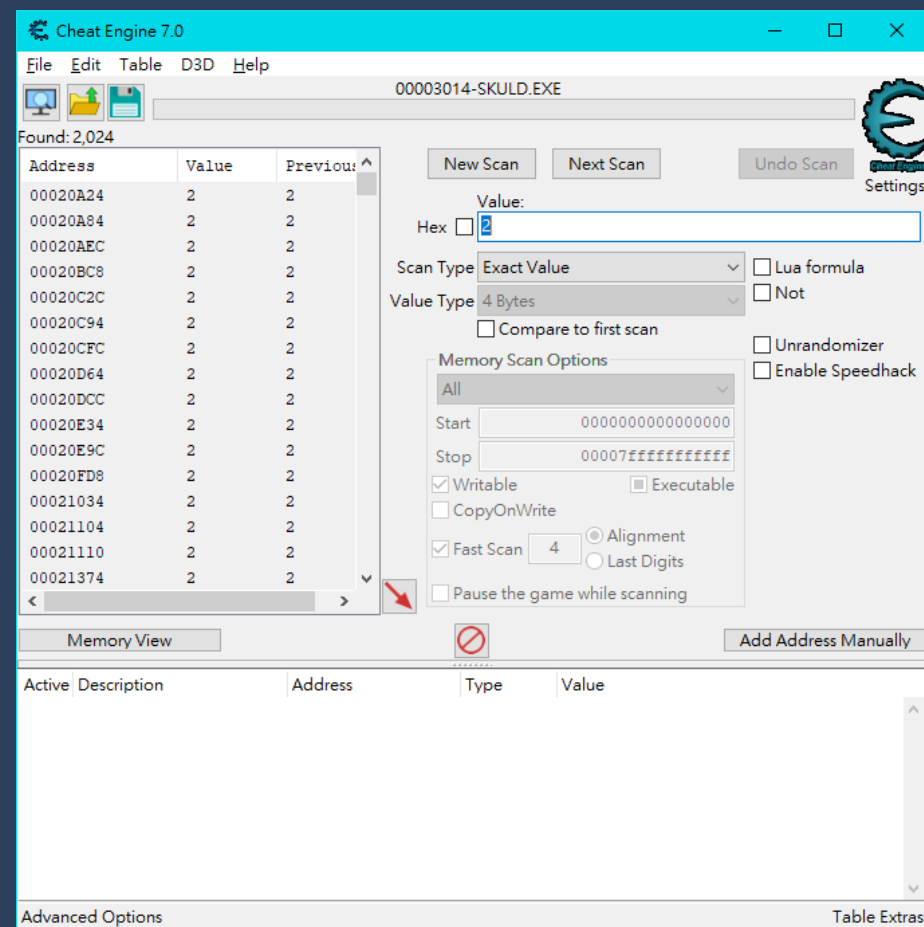
搜索整數

- ▶ 輸入要搜尋的數值，按下 First Scan
- ▶ Cheat Engine 會搜尋記憶體中所有的整數
- ▶ 並且記錄一份清單



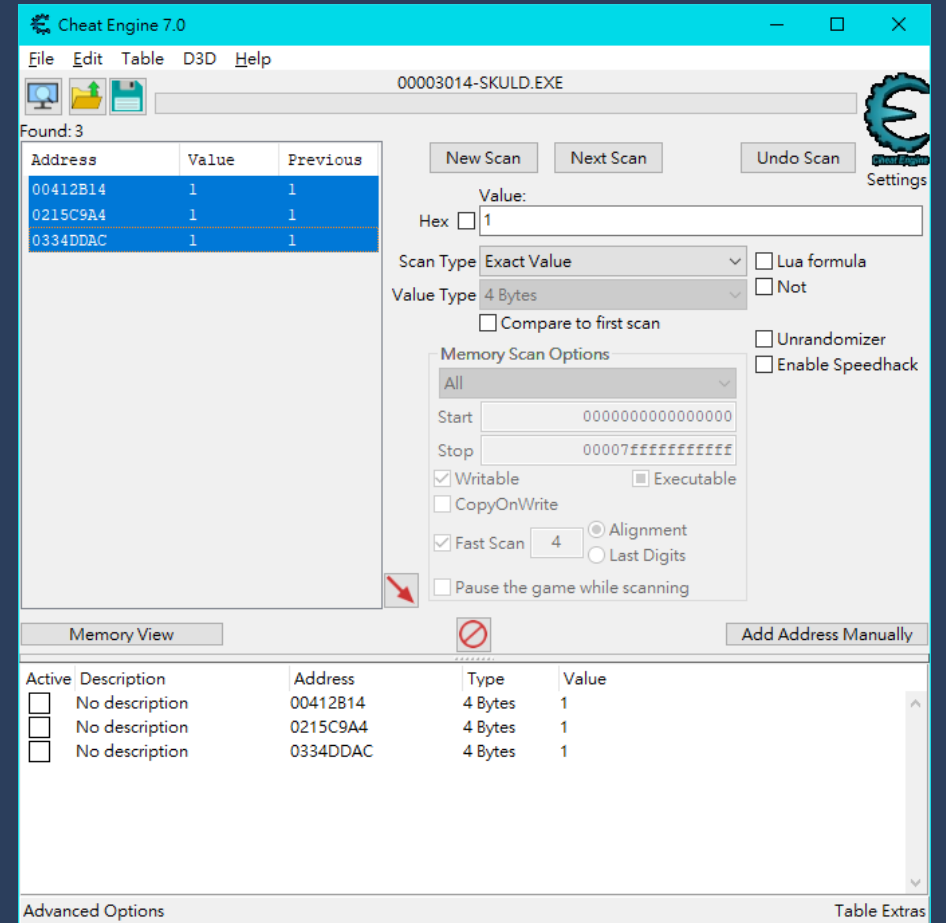
搜索整數

- ▶ 此時可以看到有 2,000 多個位址此時儲存著整數 2



搜索整數

- ▶ 待遊戲中的數值發生變化，在 Value 輸入新的數值
- ▶ 按下 Next Scan
- ▶ Cheat Engine 會比對上一次的清單中，有誰的數值變化成了新的數值
- ▶ 從左邊的列表點兩下，可以把項目加入底下的清單
- ▶ 底下的清單最右側的數值，點兩下可以進行修改
- ▶ 修改完畢後回到遊戲觀察是否發生變化



生命值減少相關程式碼

Address	Bytes	Opcode
00409F61	A1 142B4100	mov eax, [00412B14]
00409F66	85 C0	test eax, eax
00409F68	7E 40	jle 00409FAA
00409F6A	C7 05 182B4100 010...	mov [00412B18], 00000001
00409F74	48	dec eax
00409F75	C7 07 01000000	mov [edi], 00000001
00409F7B	68 FC030000	push 000003FC
00409F80	A3 142B4100	mov [00412B14], eax

Lab#0 : 不減少血量

Address	Bytes	Opcode
00409F61	A1 142B4100	mov eax, [00412B14]
00409F66	85 C0	test eax, eax
00409F68	7E 40	jle 00409FAA
00409F6A	C7 05 182B4100 010...	mov [00412B18], 00000001
00409F74	48	dec eax
00409F75	C7 07 01000000	mov [edi], 00000001
00409F7B	68 FC030000	push 000003FC
00409F80	A3 142B4100	mov [00412B14], eax

Lab#1 : 減少生命值變成增加

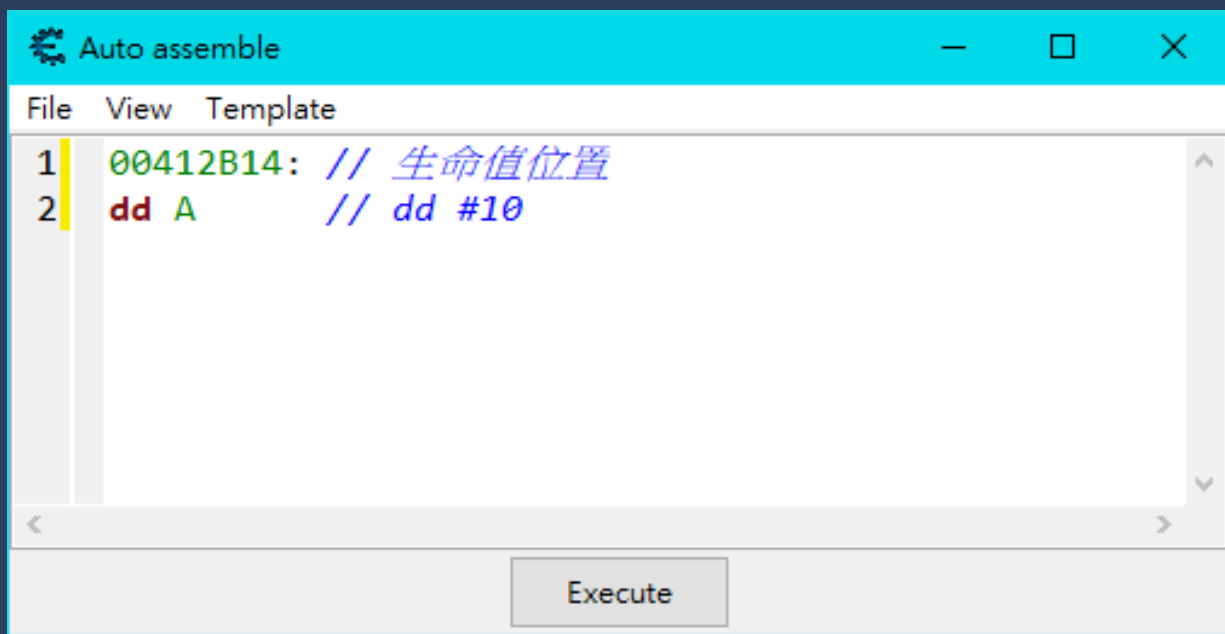
Address	Bytes	Opcode
00409F61	A1 142B4100	mov eax , [00412B14]
00409F66	85 C0	test eax , eax
00409F68	7E 40	jle 00409FAA
00409F6A	C7 05 182B4100 010...	mov [00412B18], 00000001
00409F74	48	inc eax
00409F75	C7 07 01000000	mov [edi], 00000001
00409F7B	68 FC030000	push 000003FC
00409F80	A3 142B4100	mov [00412B14], eax

Auto Assemble

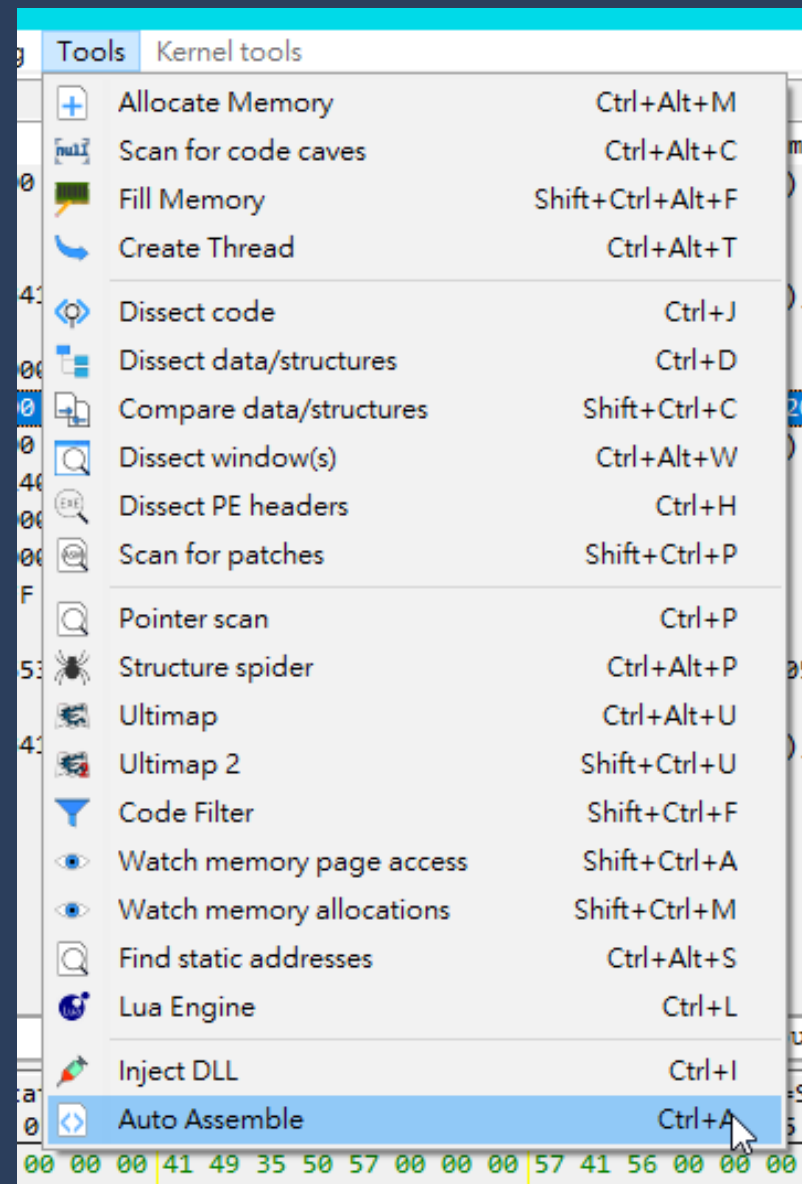


Auto Assemble

- ▶ Cheat Engine 的 Script 機制，[文件在官方 Wiki](#)
- ▶ 功能相當豐富，可以在裡面寫 hook 或者操作記憶體
- ▶ 註解語法同 C 語言，使用 // 與 /* 多行註解 */



```
File View Template
1 00412B14: // 生命值位置
2 dd A // dd #10
Execute
```



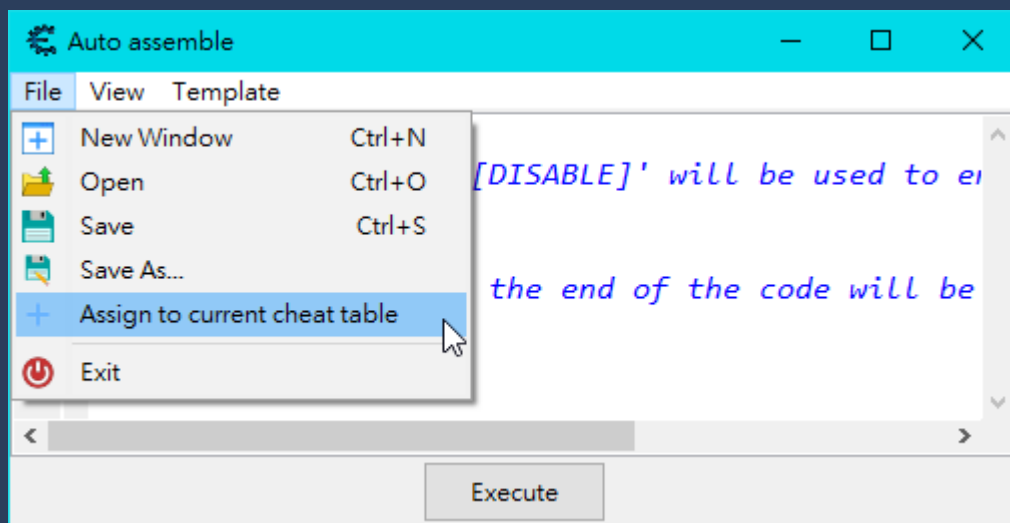
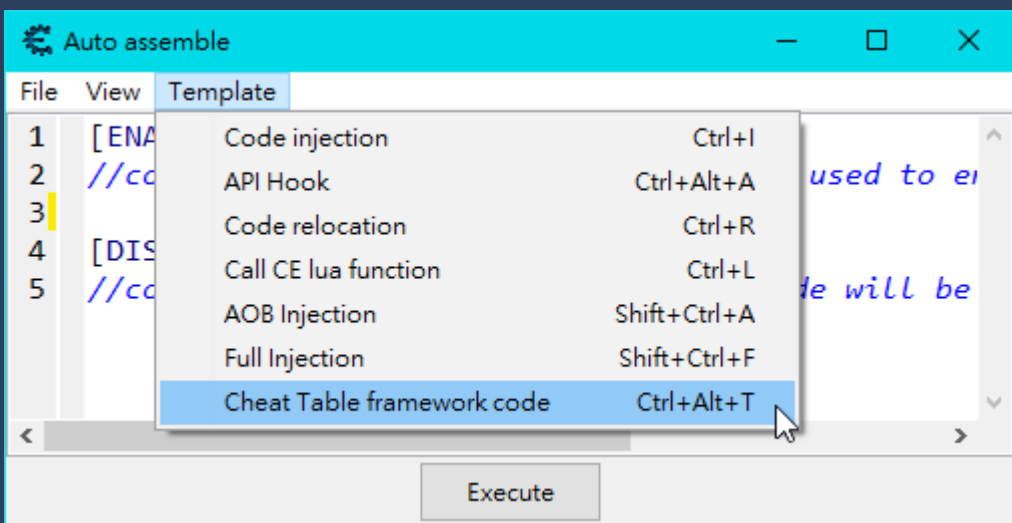
Auto Assemble

- ▶ 執行以下 AutoAsm，然後去 memory view，按 Ctrl-G 跳到 "memlab"

```
1 // 分配 1024 bytes
2 alloc(memlab, 1024)
3 // 讓 Ctrl-G 可以輸入 "memlab" 然後跳過去
4 registersymbol(memlab)
5
6 memlab:
7 dq 1122334455667788
8 dd 12345678 abcdabcd
9 dw 1122 3344 5566 7788
10 db 12 34 56 78 9a bc de f0
```

Auto Assemble

- ▶ 如何把 Auto Assemble 加入打勾列表
- ▶ [ENABLE] 後面表示打勾啟用時要做的事情
- ▶ [DISABLE] 後面表示取消打勾時，還原要做的事情

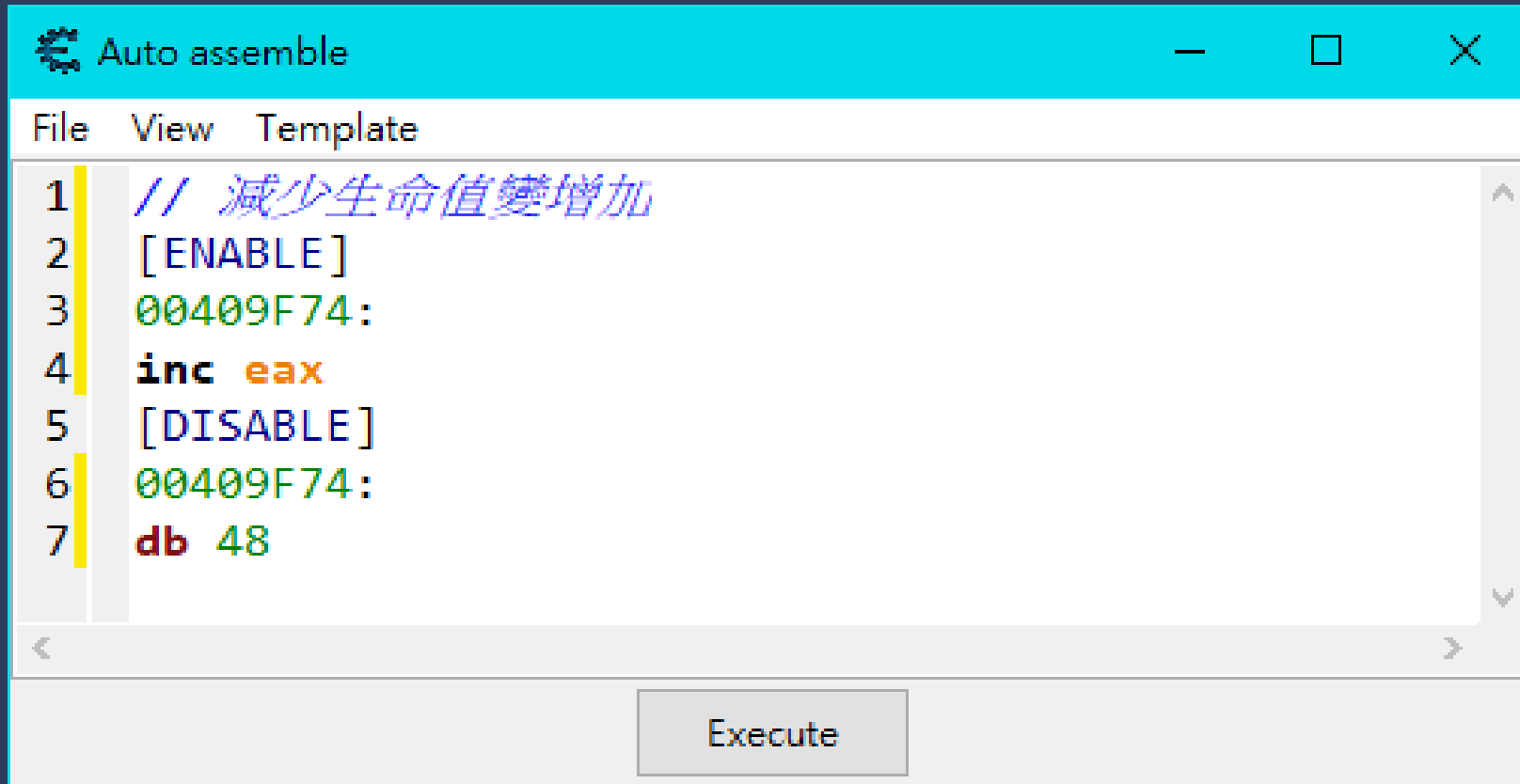


Auto Assemble

```
1 // 打勾HP變10，取消則自殺
2 [ENABLE]
3 412B14:
4 dd #10
5 [DISABLE]
6 412B14:
7 dd #-1
```

Execute

Auto Assemble



```
1 // 減少生命值變增加
2 [ENABLE]
3 00409F74:
4 inc eax
5 [DISABLE]
6 00409F74:
7 db 48
```

Execute

x86 指令長度不定

- ▶ 每個指令的長度不同，改成相同長度、更短的指令可以直接寫
 - ▶ 如果你想要寫更多的指令來做到更複雜的功能要怎麼做？

Address	Bytes	Opcode
00100000	90	nop
00100001	40	inc eax
00100002	83 C0 0A	add eax , 0A
00100005	2D 34120000	sub eax , 00001234
0010000A	81 05 78563412 21436587	add [12345678], 87654321
00100014	A1 21433412	mov eax , [12344321]
00100019	29 58 08	sub [eax+08], ebx

Lab#2 : 生命值每次加3

```
1  [ENABLE]
2  alloc(IncLife, 128) // 分配 128byte 空間
3
4  IncLife: // 在剛剛分配的空間寫指令
5  add eax, 4 // 已經被 -1 了，所以 +4
6  mov [00412B14],eax
7  jmp 00409F85
8
9  00409F80:
10 jmp IncLife
11 [DISABLE]
12 dealloc(IncLife) // 釋放剛剛的空間
13
14 00409F80:
15 db A3 14 2B 41 00
```

保護暫存器

- ▶ 暫存器的值不能隨便更動，如果要使用暫存器得要把原本的值存起來，用完之後還原
 - ▶ 除非你確定你改的位置前後文不使用這個暫存器

```
27  push  eax
28  mov  eax, [mobX]
29  mov  [ecx+24], eax
30  mov  eax, [mobY]
31  mov  [ecx+28], eax
32  mov  [ecx+58], 2
33  mov  [ecx+64], 0
34  pop  eax
```

遊戲修改到逆向工程

- ▶ 想要修改的東西沒有明顯的數值，或者無法以記憶體搜索的方式找到...
- ▶ 從其他已知的東西找前後文關係
 - ▶ 想要修改攻擊力 → 發動技能會減少魔力值
 - 所以往前往後找會存在施放技能的程式碼
 - ▶ 想要無敵 → 受到怪物攻擊會減少生命值
 - 所以往前找可以找到受到攻擊的程式碼
 - ▶ 想要全圖打怪 → 需要讀取人物座標才可以計算怪物是否在擊中區域內
 - 所以觀察攻擊怪物時，會有那些程式碼讀取了人物座標

遊戲修改到逆向工程

- ▶ 想要修改的東西沒有明顯的數值，或者無法以記憶體搜索的方式找到...
- ▶ 從已知的外部 API 呼叫行為找
 - ▶ Hook 網路相關的 API，例如 send, recv
 - 找到遊戲內處理封包的程式碼，再追到特定的封包是從那裡送出的
 - ▶ Hook 檔案操作 API，例如 fopen, open, CreateFileW
 - 找到遊戲內那裡負責讀取遊戲資源，再追到遊戲參數存放在那裡