

- Steps required to crack a safe?

1. Understand the inner working
2. Use the right tools
3. Listen for the right TICK
4. Do the magic
5. Open the safe



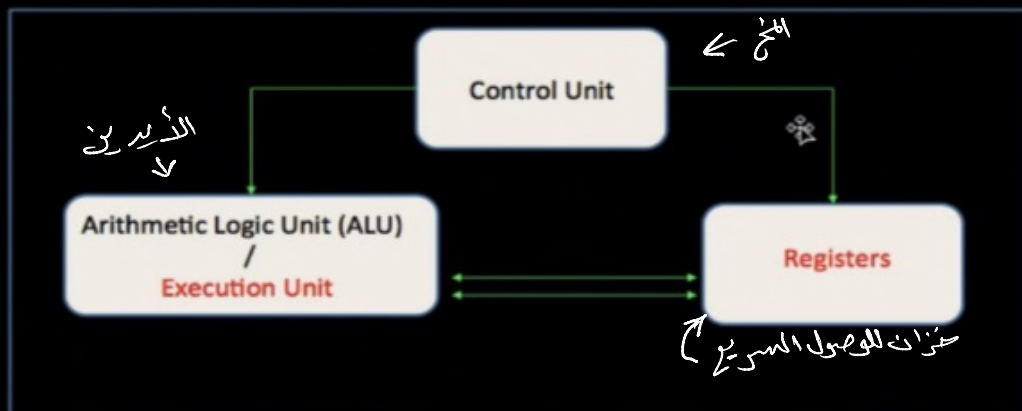
Understand the Inner Working (1)

- Input devices: Keyboard
- **Central Processing Unit: CPU**
- **Memory**
- Output Devices: Monitor



Understand the Inner Working (2)

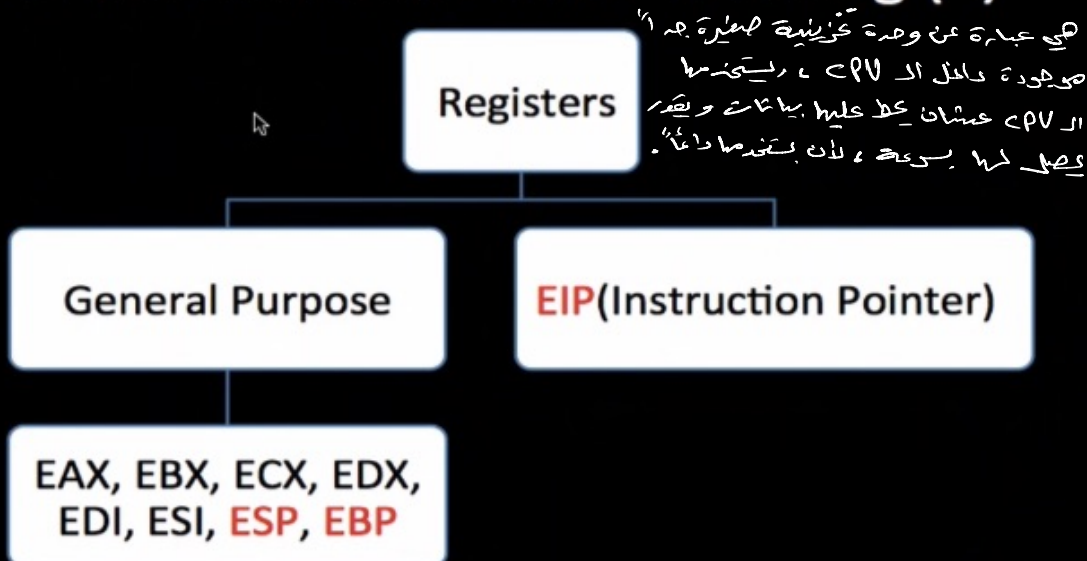
Central Processing Unit / Microprocessor



www.Nakerah.net



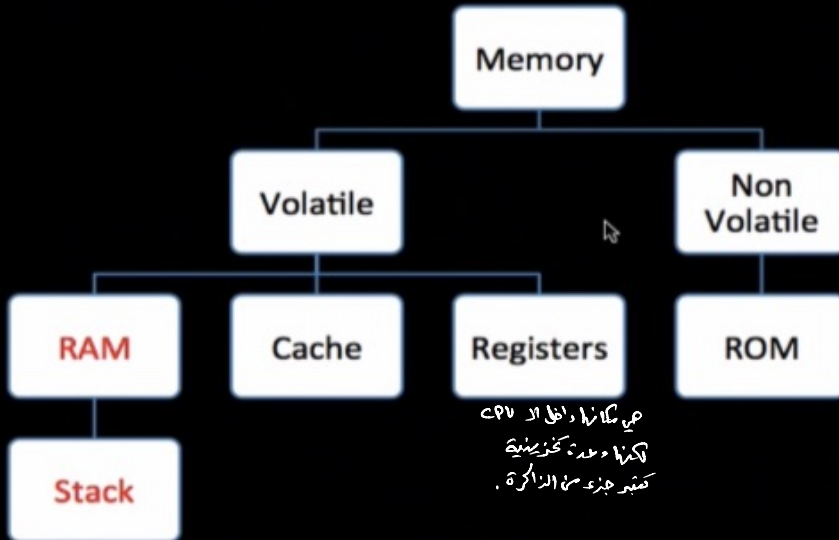
Understand the Inner Working (3)



www.Nakerah.net



Understand the Inner Working (4)

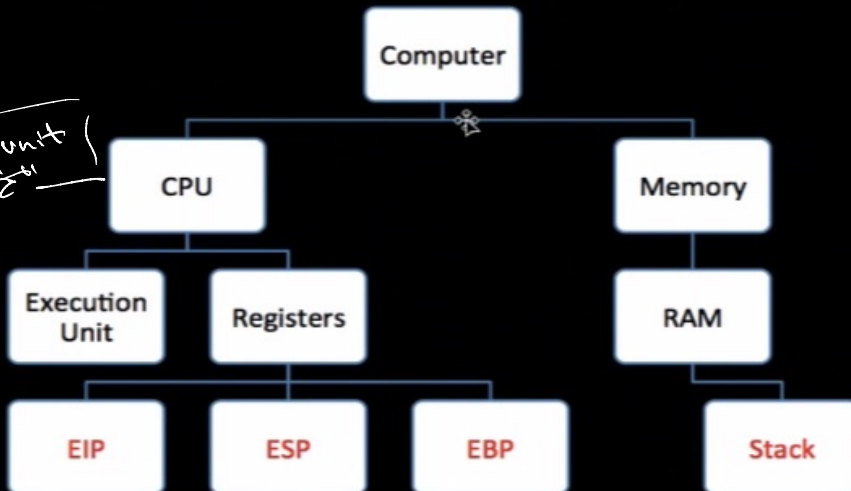


هي فيها ابريق على الماسوب
يحفظ فيها مكن في طيفه
الماسوب . وفيها ال 500

هي مكانها داخل ال CPU
لكمها وحدة تخزينية
تعتبر جزء من الذاكرة .

www.Nakerah.net

Understand the Inner Working (4)



Control unit
تحكم

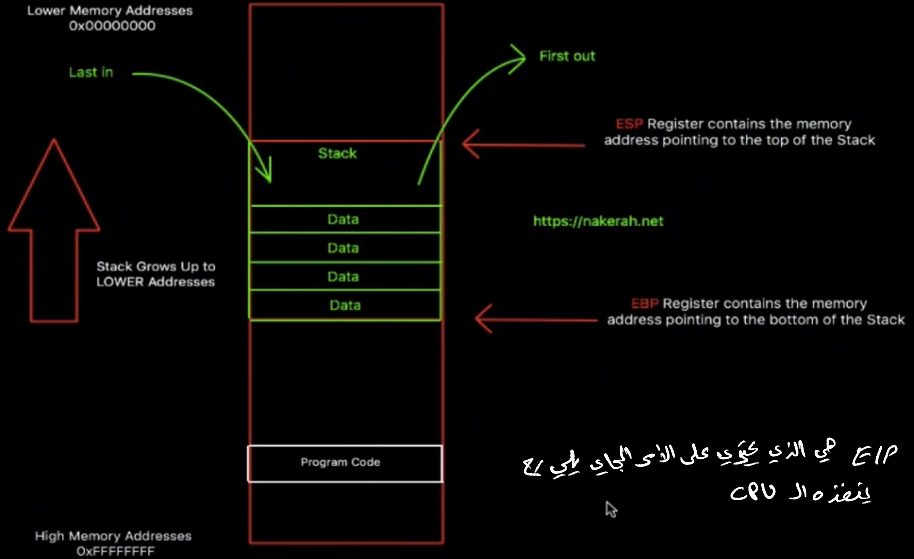
extend instruction pointer

Extend
Stack
pointer

Extend
Based
pointer



RAM



EIP هي الذي يحوي على الايدي الباي في مع
ينفذه ال CPU

مؤشر
البيانات
يخزنه في ال ESP
والذي هو ال EBP

Immunity Debugger - Sample Program.exe - [CPU - main thread module ntdll]

File View Debug Plugins ImmLib Options Window Help Jobs

Assembly:

```

775801E8 895C24 08 MOV EDWORD PTR [ESP+8],EBX
775801EC E9 69900200 JMP REGD1.77509F2A
775801C1 8DA424 00000000 LEA ESP,EDWORD PTR SS:[ESP]
775801C8 8DA424 00000000 LEA ESP,EDWORD PTR SS:[ESP]
775801C7 90 NOP
775801D0 8BD4 MOV EDX,ESP
775801D2 0F34 SYSENTER
775801D4 C3 RETN
775801D5 8DA424 00000000 LEA ESP,EDWORD PTR SS:[ESP]
775801DC 8D6424 00 LEA EDX,EDWORD PTR SS:[ESP]
775801E0 8D6424 08 LEA EDX,EDWORD PTR SS:[ESP+8]
775801E4 CD 2E INT 2E
775801E6 C3 RETN
775801E7 90 NOP
775801E8 0000 ADD BYTE PTR DS:[EAX],AL
775801EA 0000 ADD BYTE PTR DS:[EAX],AL
775801EC 7D 9A JGE SHORT REGD1.7750D158
775801EE 1E PUSH DS
775801EF 52 PUSH EDX
775801F0 0000 ADD BYTE PTR DS:[EAX],AL
775801F2 0000 ADD BYTE PTR DS:[EAX],AL
775801F4 4A DEC EDX
775801F5 51 PUSH ECX
775801F6 0100 ADD DWORD PTR DS:[EAX],EAX
775801F8 0100 ADD DWORD PTR DS:[EAX],EAX
775801FA 0000 ADD BYTE PTR DS:[EAX],AL
775801FC F1 INTR
775801FD 07 POP ES
775801FE 0000 ADD BYTE PTR DS:[EAX],AL
77580200 E9 07000010 JMP 8758020C
77580205 0201 ADD AL,BYTE PTR DS:[ECX]
77580207 0004 ADD AH,DL
77580209 2101 AND EDWORD PTR DS:[ECX],EAX

```

Modification of segment register

Stack SS:[0028FFFF]-00000000

Registers (FPU):

```

EAX 004014E0 Sample_P._ModuleEntryPoint>
ECX 00000000
EDX 00000000
EBX 7EFDE000
ESP 0028FFFF
ESI 00000000
EDI 00000000
EIP 775801E8 ntdll:775801E8

```

Stack dump:

Address	Hex dump	ASCII
00403000	0A 00 00 00 FF 00 00 00y....
00403008	FF FF FF FF FF FF FF FF	yyyyyyyy
00403010	AC 26 40 00 02 00 00 00	..6a.....
00403018	FF FF FF FF E0 25 40 00	yyyyyA..
00403020	F0 25 40 00 4E E6 40 B8	0000 NA..
00403028	E1 19 BF 44 00 00 00 00
00403030	00 00 00 00 00 00 00 00
00403038	00 00 00 00 00 00 00 00
00403040	00 00 00 00 00 00 00 00
00403048	00 00 00 00 00 00 00 00
00403050	00 00 00 00 00 00 00 00
00403058	00 00 00 00 00 00 00 00
00403060	00 00 00 00 00 00 00 00
00403068	00 00 00 00 00 00 00 00
00403070	00 00 00 00 00 00 00 00
00403078	00 00 00 00 00 00 00 00
00403080	00 00 00 00 00 00 00 00

0028FFFF 00000000

0028FFF4 004014E0 Sample_P._ModuleEntryPoint>

0028FFFB 00000000

0028FFFC 00000000

هذا الذاكرة كلها

الذاكرة

الذاكرة

هذا الذاكرة

Single step event at ntdll:775801E8 - use Shift+F7/F8/F9 to pass exception to program

Paused

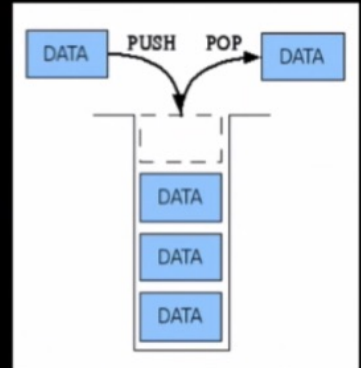
6:30 PM

Stack

Understand the Inner Working (7)

- Stack

1. Used by functions
2. PUSH and POP 4bytes for x86
3. Last input first output
4. ESP and EBP : Top and Bottom
5. Grows towards lower addresses

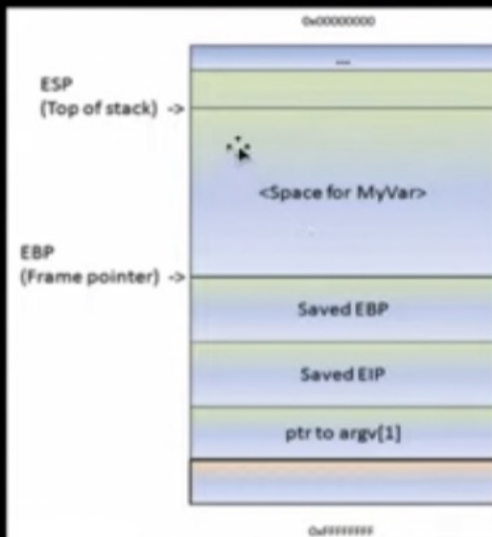


Understand the Inner Working (6)

1. Push function arguments
 - Command line arguments
2. Call function
 - PUSH return address to restore normal execution after callee terminates & jmp to the callee.
3. push ebp
 - save caller ebp to restore it after callee terminates
4. mov ebp, esp.
 - Make top and bottom of the stack equal "create new stack frame for the callee."
5. sub esp, N
 - reserve space on the stack for callee local variables
6. Callee inner working
7. Mov esp, ebp
 - restore old value of esp
8. pop ebp
 - restore old value of ebp.
9. Return
 - restore original execution flow by executing instruction located at saved return address



Producing BOF Condition (1)



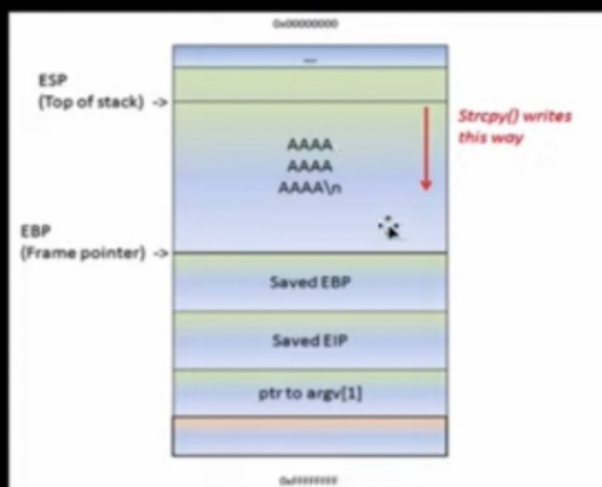
```
#include "string.h"

void do_something(char *buffer)
{
    char MyVar[0];
    strcpy(MyVar, buffer);
}

int main (int argc, char **argv)
{
    do_something(argv[1]);
    printf
}
```



Producing BOF Condition (2)



Producing BOF Condition (3)

