Procedures

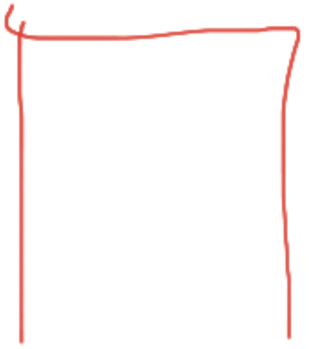
Topics

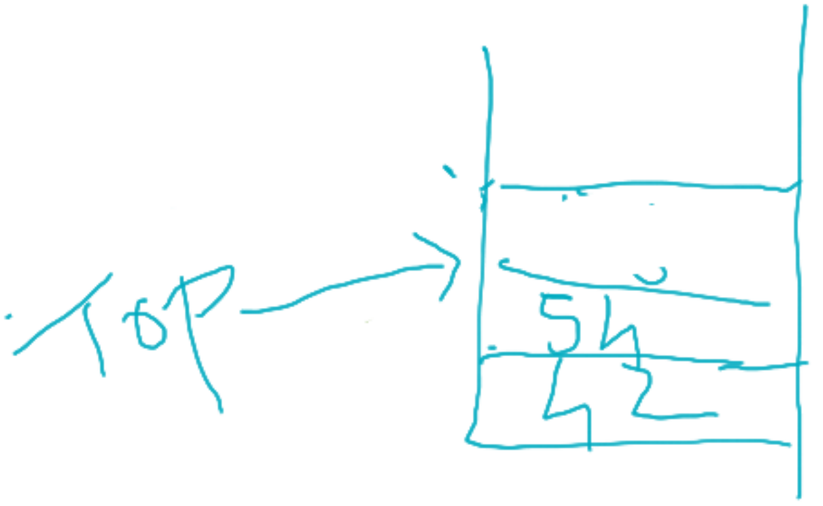
- 80x86 Stack
- 32-bit Procedures with Value Parameters
- Additional 32-bit Procedure Options
- 64-bit Procedures
- Macro Definition and Expansion

LIFO

last in first out

data structures

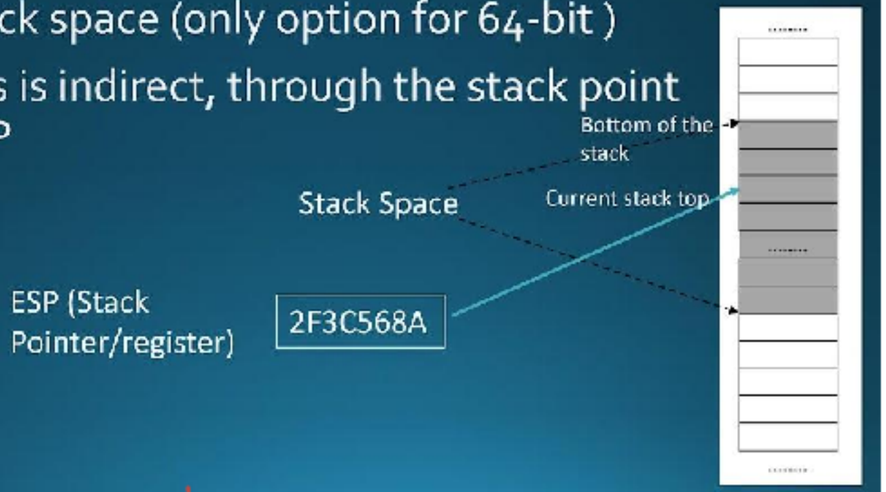




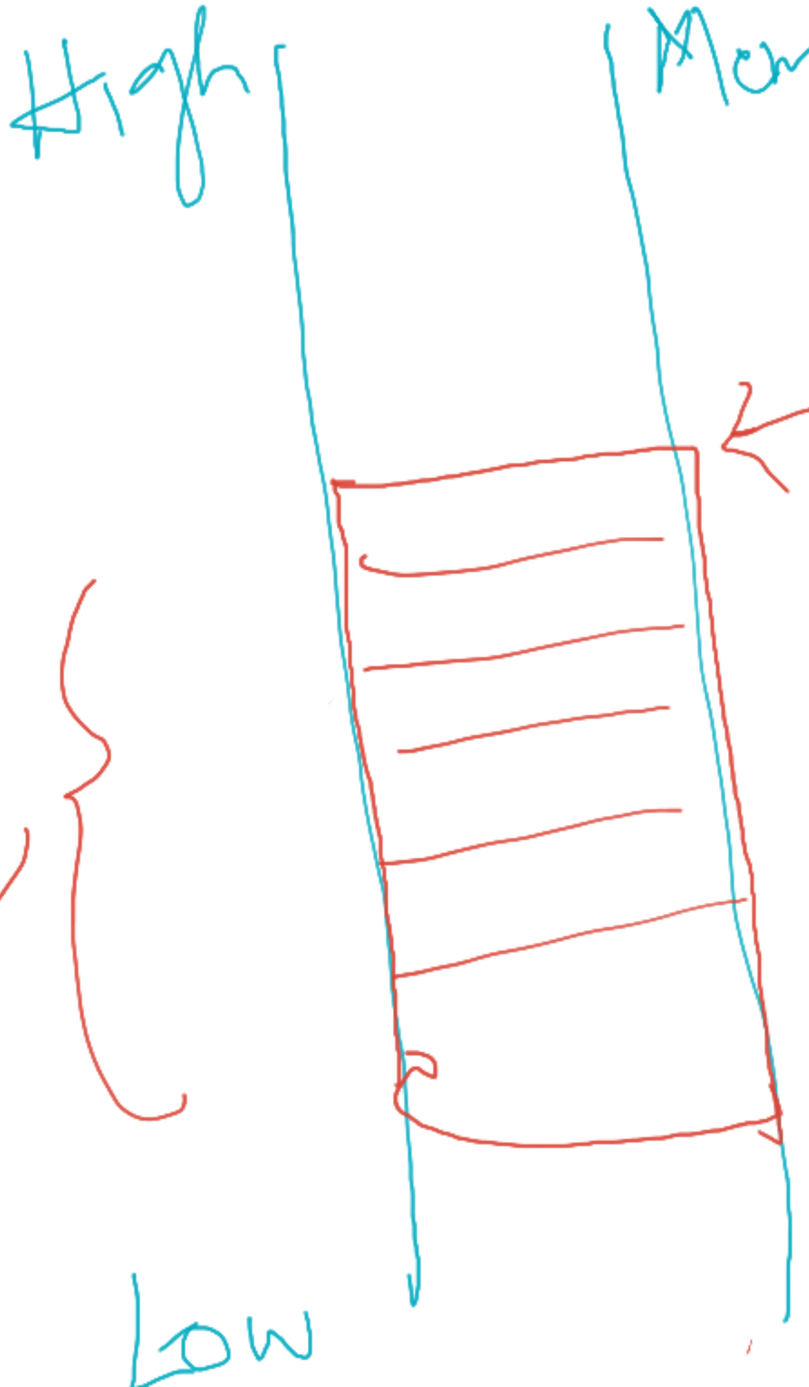
push 42
push 54
push 56
pop

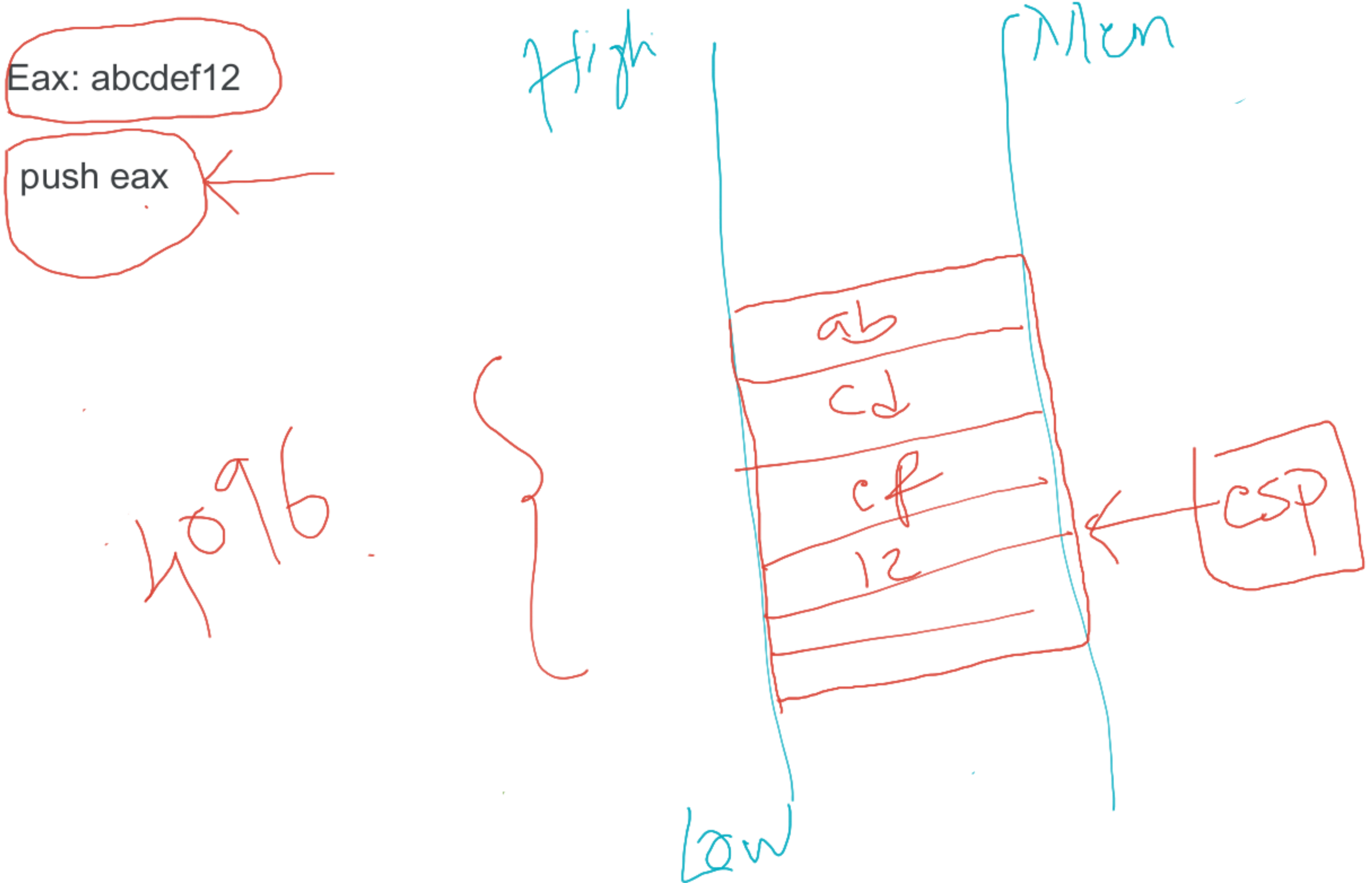
80x86 Hardware Stack

- Allocated with directive(for 32-bit only), for example
  `.STACK 4096`
  - allocates 4096 uninitialized memory bytes
- Visual Studio project setting can also be used to allocate stack space (only option for 64-bit)
- Most access is indirect, through the stack point register ESP



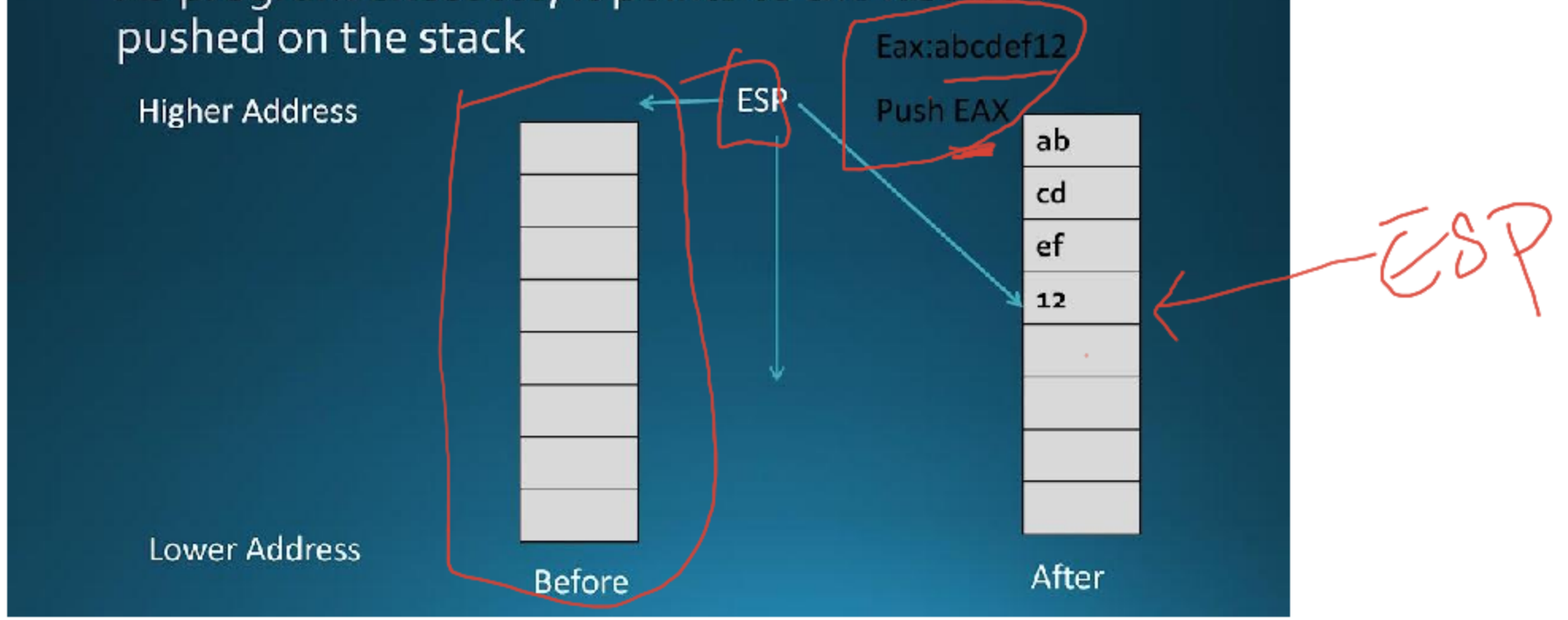
4096




Eax: abcdef12
push eax
High
Mem
ab
cd
ef
12
esp
4096
Low

80x86 Hardware Stack

- Operating system initializes ESP to point to byte above stack
- As program executes, it points to the last item pushed on the stack



Use of Stack

- Implicitly
  - Procedure call
  - Return
- Explicitly
  - Push (Instruction)
  - Pop (Instruction)

call procname

when we execute call instruction
by default it pushes the ret address

Ret

when we execute ret in procedure, it will pop the ret address from the stack and store it in EIP

call

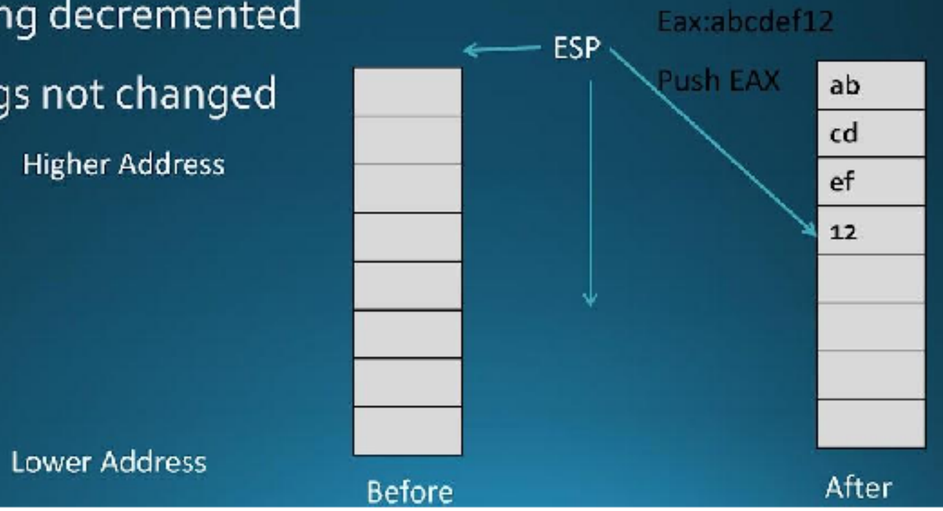
`push` instruction

- Usual format: `push` *source*
  - *source* can in memory, register or immediate
  - DWORD or WORD pushed on the stack (+QWORD for 64 bit)
- Formats to use when the assembler cannot determine operand size
  - `pushd` *source*
  - `pushw` *source*

When you don't know size of Push

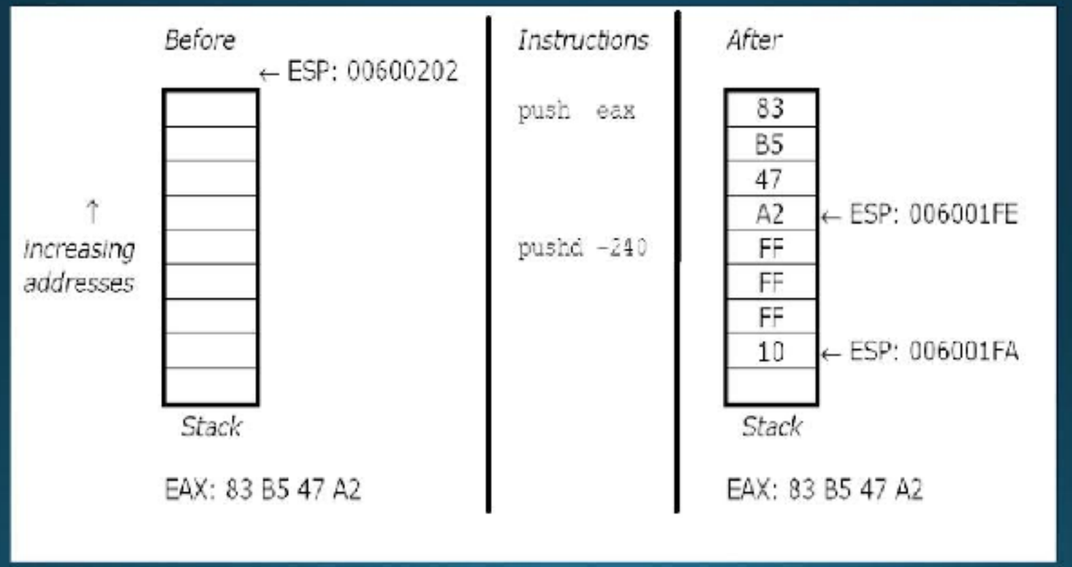
push execution

- ESP decremented by size of operand
- Operand stored in stack where ESP points after being decremented
- Flags not changed



ESP

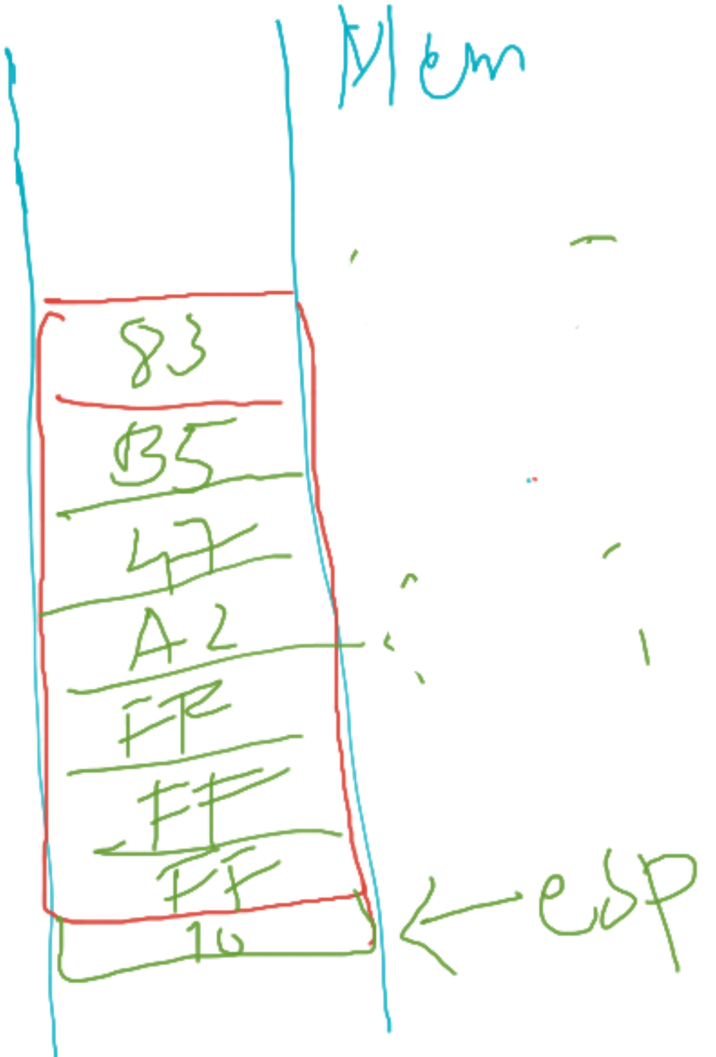
push example



stack 4096

push eax ;eax= 83 B5 47 A2

pushd -240 ;FF FF FF 10



`push` instruction encoding

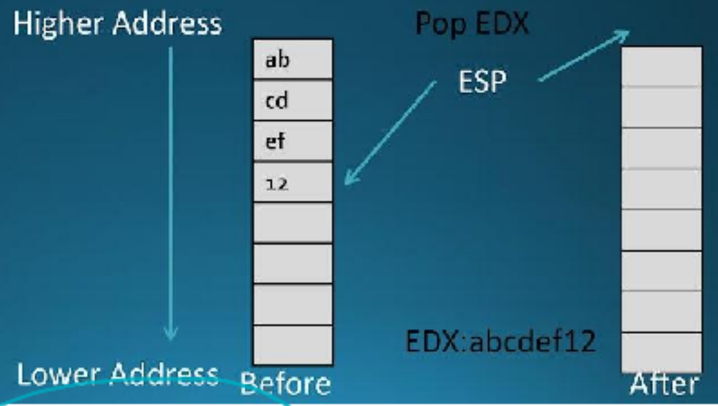
Operand	Opc ode	Byte length
EAX or AX	50	1 (or 2)
ECX or CX	51	1 (or 2)
EDX or DX	52	1 (or 2)
EBX or BX	53	1 (or 2)
ESP or SP	54	1 (or 2)
EBP or BP	55	1 (or 2)
ESI or SI	56	1 (or 2)
EDI or DI	57	1 (or 2)
Memory word	FF	3+
Memory doubleword	FF	2+
Immediate byte	6A	2
Immediate word	68	4
Immediate doubleword	68	5

```
00000025  50              push EAX
00000026  66| 50          push AX
00000028  51              push ECX
00000029  66| 51          push CX
0000002B  52              push EDX
0000002C  66| 52          push DX
0000002E  53              push EBX
0000002F  66| 53          push BX
00000031  54              push ESP
00000032  66| 54          push SP
00000034  55              push EBP
00000035  66| 55          push BP
00000037  56              push ESI
00000038  66| 56          push SI
0000003A  57              push EDI
0000003B  66| 57          push DI

0000003D  66| FF 35            push myword
      00000198 R
00000044  66| FF 33            push WORD PTR [EBX]
00000047  FF 35 00000194 R      push sum
0000004D  FF 33           push [EBX]
0000004F  6A 64           push 100
00000051  66| 68 0100          pushw 256
00000055  68 0186F3B4           push 2870156
```

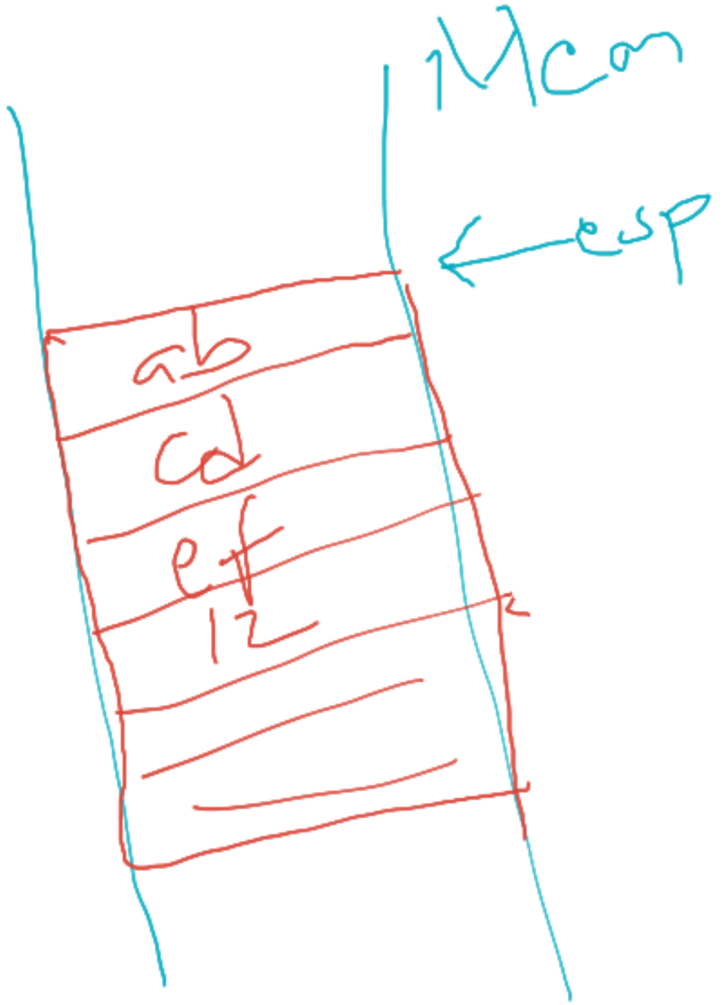
pop instruction and execution

- Usual format: `pop destination`
  - doubleword or word *destination* can in memory or register
- Operand stored in stack where ESP points is copied to destination
- ESP incremented by size of operand after the value is copied
- Flags not changed



pop destination

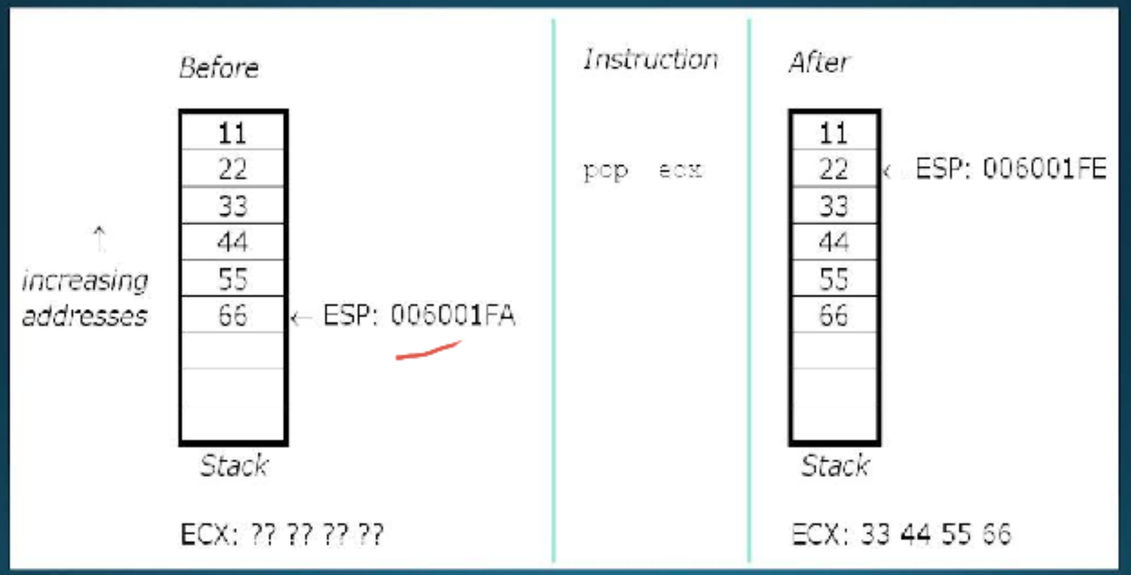
element in top of stack is stored in Destination and the esp is changed

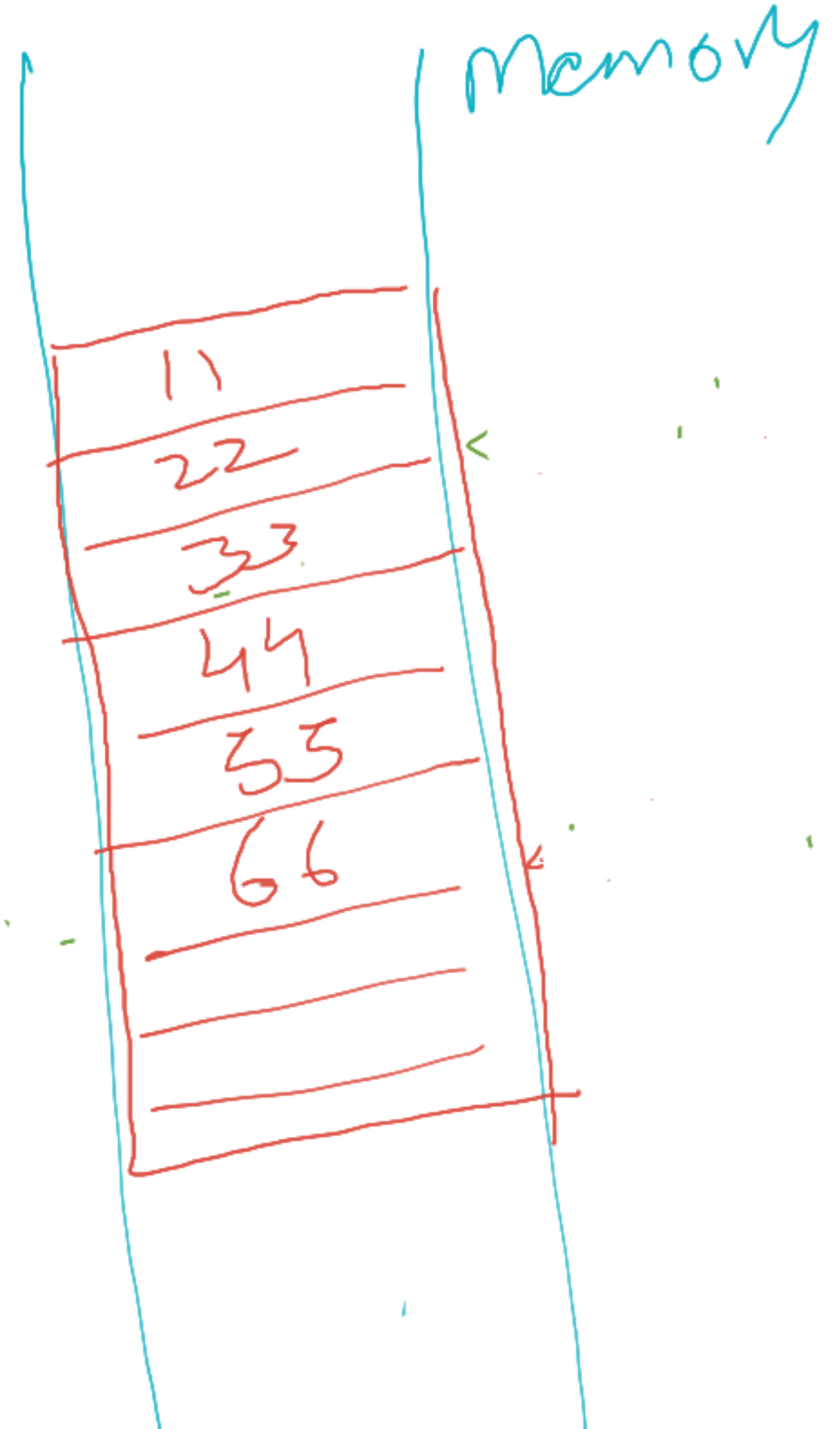


Pop EDX → DWORD → 4bytes

EDX = abcdef12

pop example





pop ecx --- > dword----> 4bytes of data

ecx-->33445566

pop instruction encoding

Operand	Opcode	Byte length
EAX or AX	58	1 (or 2)
ECX or CX	59	1 (or 2)
EDX or DX	5A	1 (or 2)
EBX or BX	5B	1 (or 2)
ESP or SP	5C	1 (or 2)
EBP or BP	5D	1 (or 2)
ESI or SI	5E	1 (or 2)
EDI or DI	5F	1 (or 2)
Memory word	8F	2+
Memory doubleword	8F	2+

```
0000005D  58                pop EAX
0000005F  66| 58             pop AX
00000060  59                pop ECX
00000061  66| 59             pop CX
00000063  5A                pop EDX
00000064  66| 5A             pop DX
00000066  5B                pop EBX
00000067  66| 5B             pop BX
00000069  5C                pop ESP
0000006A  66| 5C             pop SP
0000006C  5D                pop EBP
0000006D  66| 5D             pop BP
0000006F  5E                pop ESI
00000070  66| 5E             pop SI
00000072  5F                pop EDI
00000073  66| 5F             pop DI

00000075  66| 8F 05              pop myword
      00000198 R
0000007C  66| 8F 03              pop WORD PTR [EBX]
0000007F  8F 05 00000194 R      pop sum
00000085  8F 03              pop [EBX]
```

Pushing/Popping Flags

- `pushf` pushes FLAGS register contents onto stack.
- `pushfd` pushes EFLAGS register contents onto stack.
- `popf` pops word from top of stack into FLAGS.
- `popfd` pops doubleword from top of stack into EFLAGS.

Viewing the Stack with Debugger

- Stop at breakpoint
- View registers
  - ESP contains address of byte at the top of the stack

```
mov ebx , 012345678h
mov ecx,0FFFFFFFFH

push ebx
push ecx
pop eax
pushd 022222222h
```

EAX - FFFFFFFF