ELEC 377 - Operating Systems

Week 10 – Class 1

Network Types

- Local Area Networks (LAN)
  - ◊ High Speed, High Cost
  - ◊ Ethernet (1 Mbit - 1 Gigbit)
  - ◊ Token Ring, Optical Fibre
  - ◊ Short Distance (100’s Meters)
- Wide Area Network (WAN)
  - ◊ Long Distance (100’s -1000’s Km)
  - ◊ Internet (Arpanet)
  - ◊ Private Networks (IBM Global Services, UUNet)
  - ◊ Routers
  - ◊ Slow (T1 = 1.544 Mbits, T4= 28 T1 = 45 Mbits, ISDN = 128Kbits, 56K, 33k, DSL)

Distributed File Systems

- Integral Part of Distributed Operating Systems
  - ◊ Many implementations
  - ◊ Data Migration
- Concepts
  - ◊ **service** – software entity running on one or more machines providing a particular function (file access)
  - ◊ **server** – a machine running the service software
  - ◊ **client** - process that can invoke a service
  - ◊ **client interface** - operations on the service available to clients
  - ◊ Machine may be both a server and a client - Peer-to-peer

Distributed File Systems

- Ideally, a distributed file system looks the same where ever you log in
  - ◊ Suns in CASLAB - home file system is on zeus
  - ◊ Single server - relatively easy
  - ◊ Research System

    - multiple unix systems each with disk space

    - /home/stephan is located on cetus.ee.queensu.ca

    - /home/stephan is a remote mount on all other machines

    - /home/li is located on orion.

    - /home/li is a remote mount on all other machines
  - ◊ Machines are all both server and client

Distributed File Systems

- Transparency - remote and local disks look the same
  - ◊ Virtual File System abstracts interface to multiple file systems.
  - ◊ User does not know where the files are located
  - ◊ There may be more than one copy of a file - (**replication**)
  - ◊ Two components

    - Location Transparency (static)

        - name does not reveal location

    - Location Independence (stronger, dynamic)

        - name does not change if location changes

        - file migration

Distributed File Systems

- Location transparency vs independence
  - ◊ separate data from location
  - ◊ static location transparency - share files
    - location independence - share space
  - ◊ separate naming hierarchy from storage hierarchy
    - remove restrictions on system architecture

Distributed File Systems

- Diskless workstations
    - ◊ ROM loads kernel from server
    - ◊ popular in Late 80's
    - ◊ resurgence now

Distributed File Systems

- File Naming - three approaches
  - ◊ host + location
    - not location transparent not location independent
  - ◊ Attach remote directories to local directories
    - remote mount
    - permissions??
  - ◊ total integration
    - global name structure spans all files
    - problem with special files
  - Examples:
    - ◊ most current DFS tie location to mount point. Drive Z: is on files.engineering.queensu.ca
    - ◊ difficult to move a single file on Z: to be on some other server. All the files on the other server are on H:

Distributed File Systems - Caching

- Performance
  - ◊ network overhead in addition to other I/O overhead
  - ◊ Similar to cache for disk I/O
- Consistency
  - ◊ more than one client may be accessing same file
    - client initiated (check for consistency before using cached value)
    - server initiated (track clients and notify)
- Location
  - ◊ main memory
  - ◊ local disk

State

- Stateful Connections
  - ◊ Connection between client and server is persistent
  - ◊ Server keeps track of all clients
  - ◊ Amortize overhead of connection, I/O
  - ◊ AFS, AFPS
- Stateless
  - ◊ Each operation is a separate request
  - ◊ NFS
- Tradeoffs
  - ◊ Client/Server Crash?
  - ◊ Performance

Security - Three meanings

1. Protection and Authentication
   - ◊ Identify Users
   - ◊ Users only access information they have privileges for
   - ◊ secrecy
2. System Integrity
   - ◊ Only authorized users
   - ◊ prevent execution of code by outsiders
3. Information Security
   - ◊ Statistical Attacks
   - ◊ Medical/Financial

Security

- Security
  - ◊ impossible in practice
  - ◊ accidental violations (easy to protect)
  - ◊ malicious (harder)
    - – Reading of data (info theft)
    - – Modification of data
    - – Destruction of data
    - – Denial of service
  - ◊ Cost tradeoffs

Security - Informational Security

- Statistical Attacks
  - ◊ Individual pieces of information reveal nothing
  - ◊ Collectively, they reveal private information
  - ◊ statistics databases
- Statistics Canada
  - ◊ only releases information in predefined categories
- Traffic Analysis
- User Generated Queries
  - ◊ carefully crafted queries
  - ◊ refuse queries whose results are small counts
  - ◊ You and Joe are the only mid level managers
    - “what is the average of mid levels managers salaries” tells you Joes salary.

Security Levels

- Physical
  - ◊ bios on PC
- Human
  - ◊ social engineering
- Network
  - ◊ packet interception, denial of service
- OS
  - ◊ only level OS has control over

- first two are outside of OS control but necessary
- hardware protection for OS
- harder to add security than design for it

Physical Security

- Physical access to the machine
  - ◊ bios password helps a bit
  - ◊ hard drive removal
    - - encryption??
    - - OS vs Device
      - - device encryption
      - - encryption algorithm (two stage?)
      - - device access
      - - where is the key?
    - - full disk vs file encryption
    - - Trust....

Physical Security

- Personnel access
  - who has access
  - auto exec/inf files
  - hardware (firewire DMA hole - Winlockpwn)
  - PMCIA, eSATA?
- Tempest
  - ◊ CRT video - flyback transformer
  - ◊ Wireless Keyboard
  - ◊ Wired Keyboard
- Key Loggers
  - ◊ hardware (physical security)

System Threats

- Denial of Service
  - ◊ Disable the service
  - ◊ password timeouts
  - ◊ network based
    - - smurf attack
    - - zombie attack (combined with worms)
    - - oversize ICMP packet
    - - Xmas Tree Packets
- Key Loggers
  - ◊ software (permission to install?)
  - ◊ hardware (physical security)

Human Security

- Social Engineering (manipulating people)
  - ◊ Kevin Mitnick
  - ◊ Password reset on banking/credit card
- Can be more elaborate (Patch update attack)...
- phishing
  - ◊ fake email from bank/PayPal/Microsoft
  - ◊ Nigerian 411/Lotto win
  - ◊ Harvard/UC Berkely Study

    23% did not look at addr/status bar, sec indicators

    68% ignored certificate warnings

    90% were fooled by good phishing websites

    no correlation with age, sex, previous exp, comp experience

Human Security

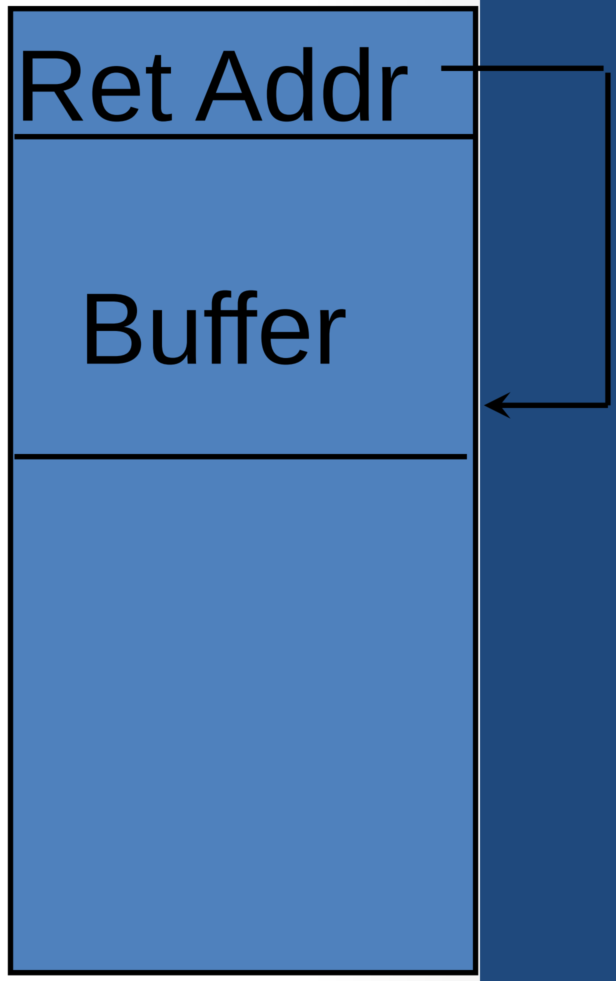
- Baiting
  - ◊ Free Screen Savers
- Quid pro quo
  - ◊ Calling back from Tech Support
- Fake Services
  - ◊ physical mail victim
  - ◊ “new” telephone banking number (1800...)
  - ◊ play back recorded prompts, record acct/pin numbers

Program Threats

- Buffer Overflow
  - ◊ Most common attack
  - ◊ exploit bug as security hole

1. Write binary code into buffer, ending with a value that overwrites the return address and points into the buffer
2. Subroutine returns into the stack instead of to calling program

Protection: don’t allow stack space to be executable!! don’t put buffers on the stack!!

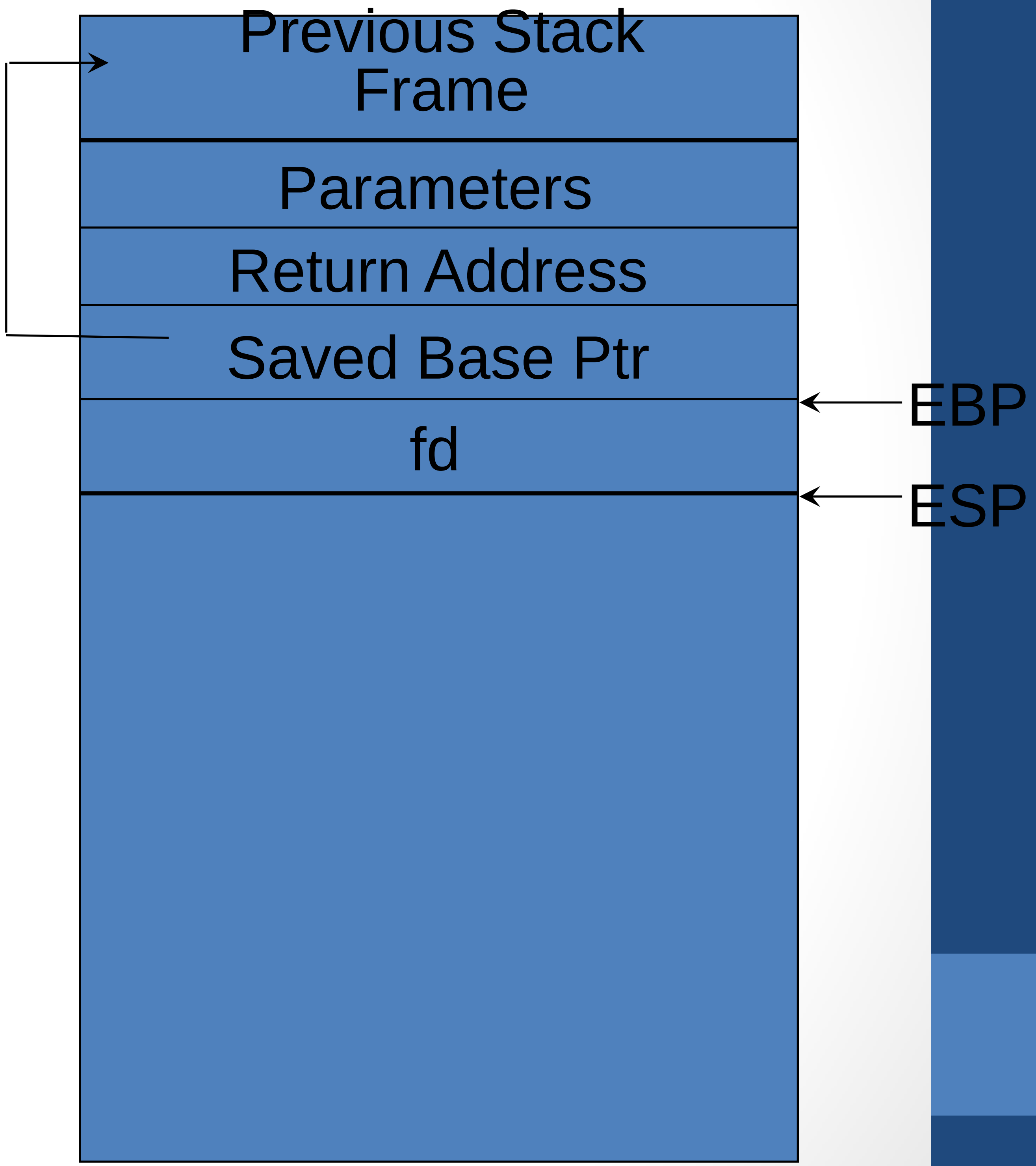


Pentium Stack Layout

```
fd = open("theFile", O_RDONLY, 0744);

push 0744
push O_RDONLY
pushd PtrToString

call open

mov [ebp-fd],eax
add esp,12
```

Pentium Stack Layout

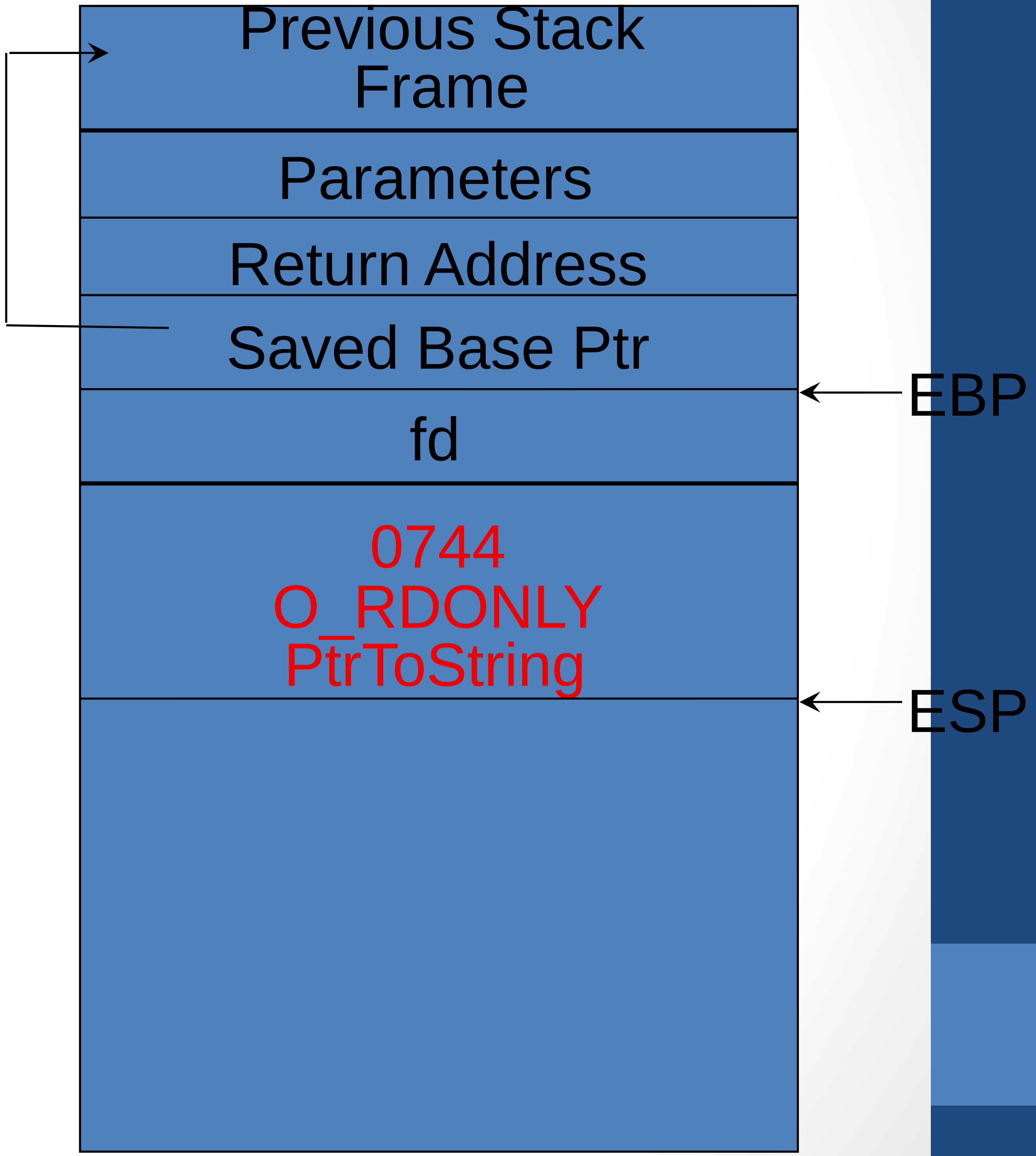
```
push 0744
push O_RDONLY
pushd PtrToString

call open

mov [ebp-fd],eax
add esp,12
```



EBP is frame pointer, ESP is stack pointer.

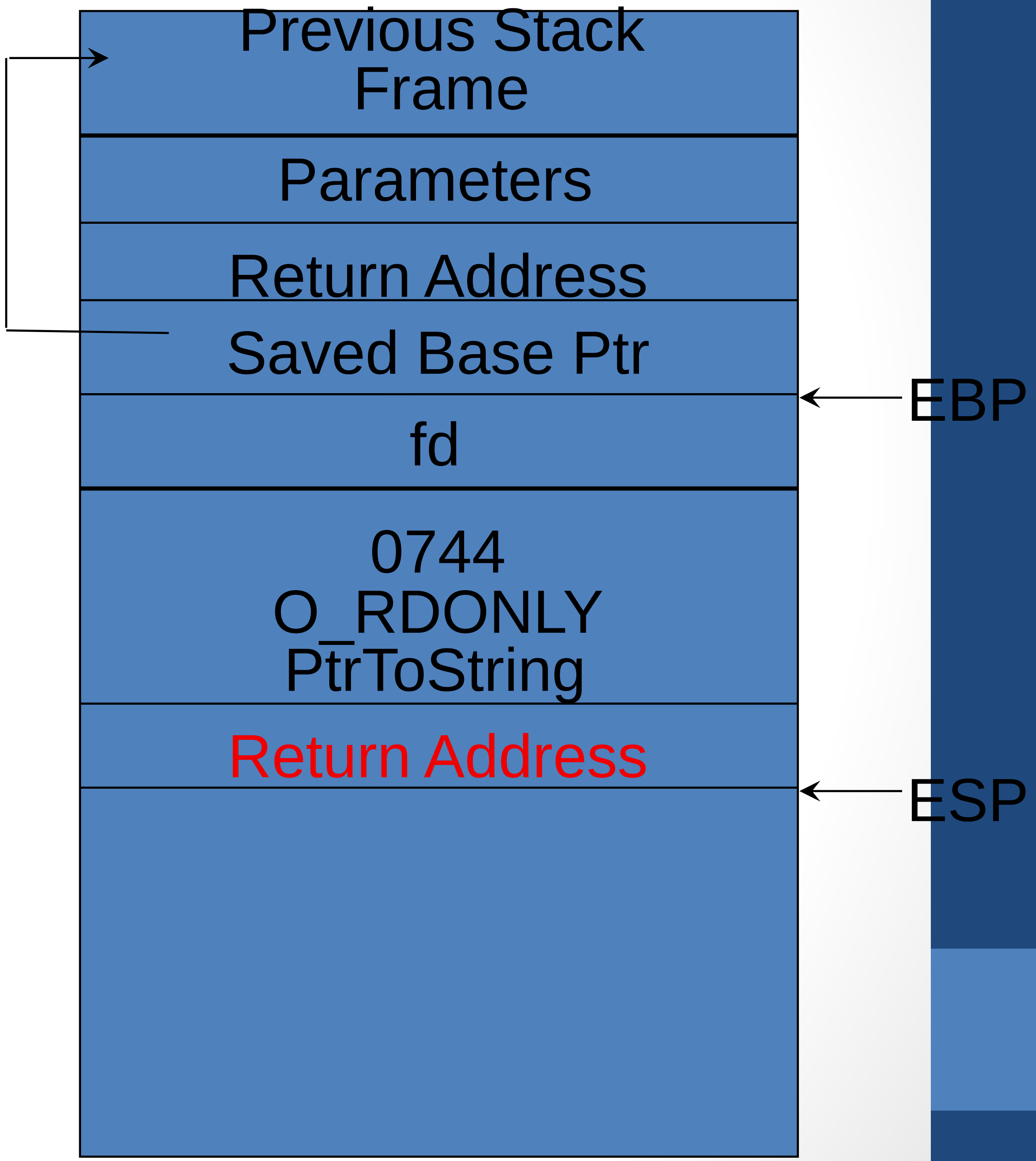
Pentium Stack Layout

```
push 0744
push O_RDONLY
pushd PtrToString

call open

mov [ebp-fd],eax
add esp,12
```



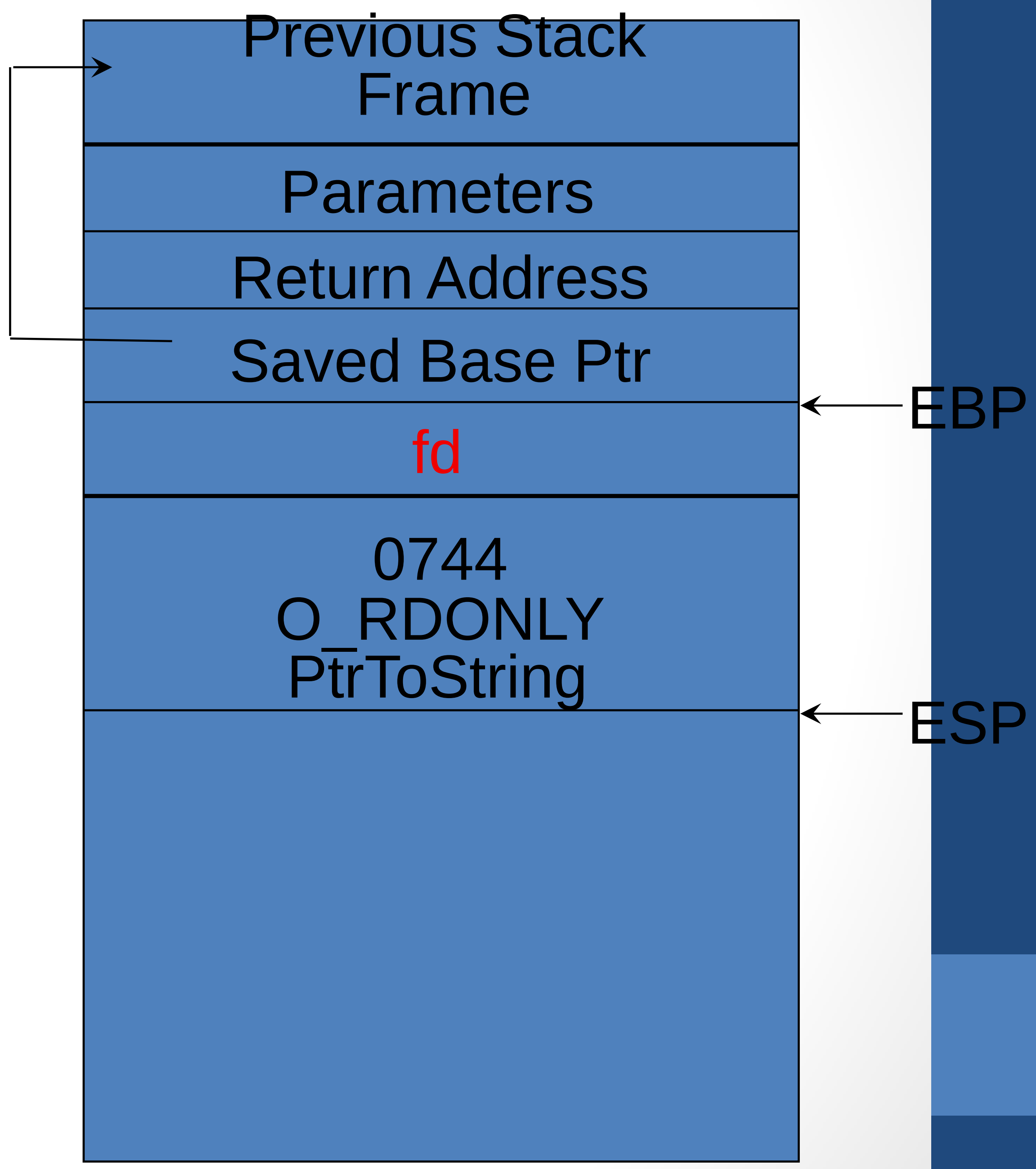
Pentium Stack Layout

```
push 0744
push O_RDONLY
pushd PtrToString

call open

mov [ebp-fd],eax
add esp,12
```



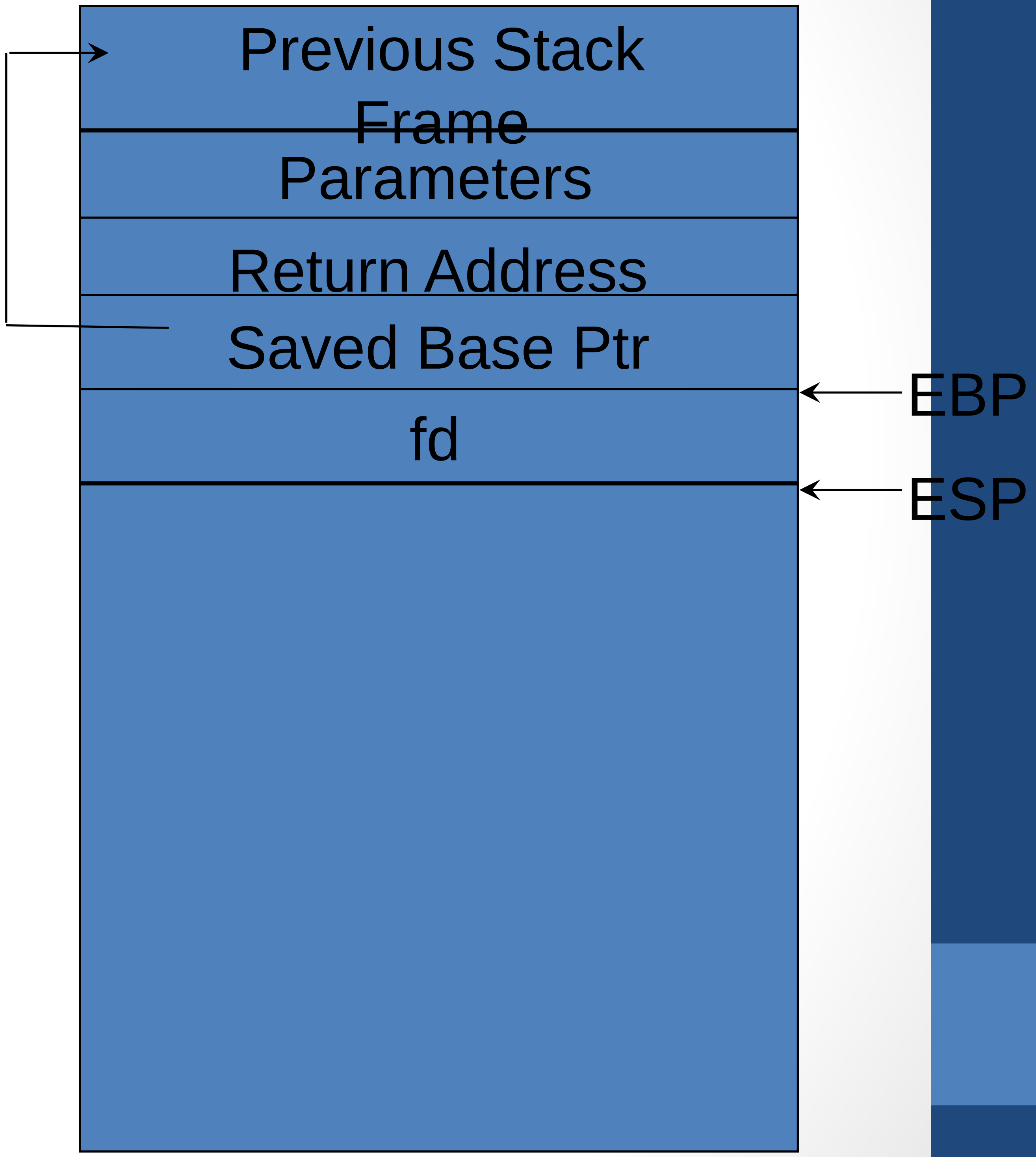
Pentium Stack Layout

```
push 0744
push O_RDONLY
pushd PtrToString

call open

mov [ebp-fd],eax
add esp,12
```



Pentium Stack Layout

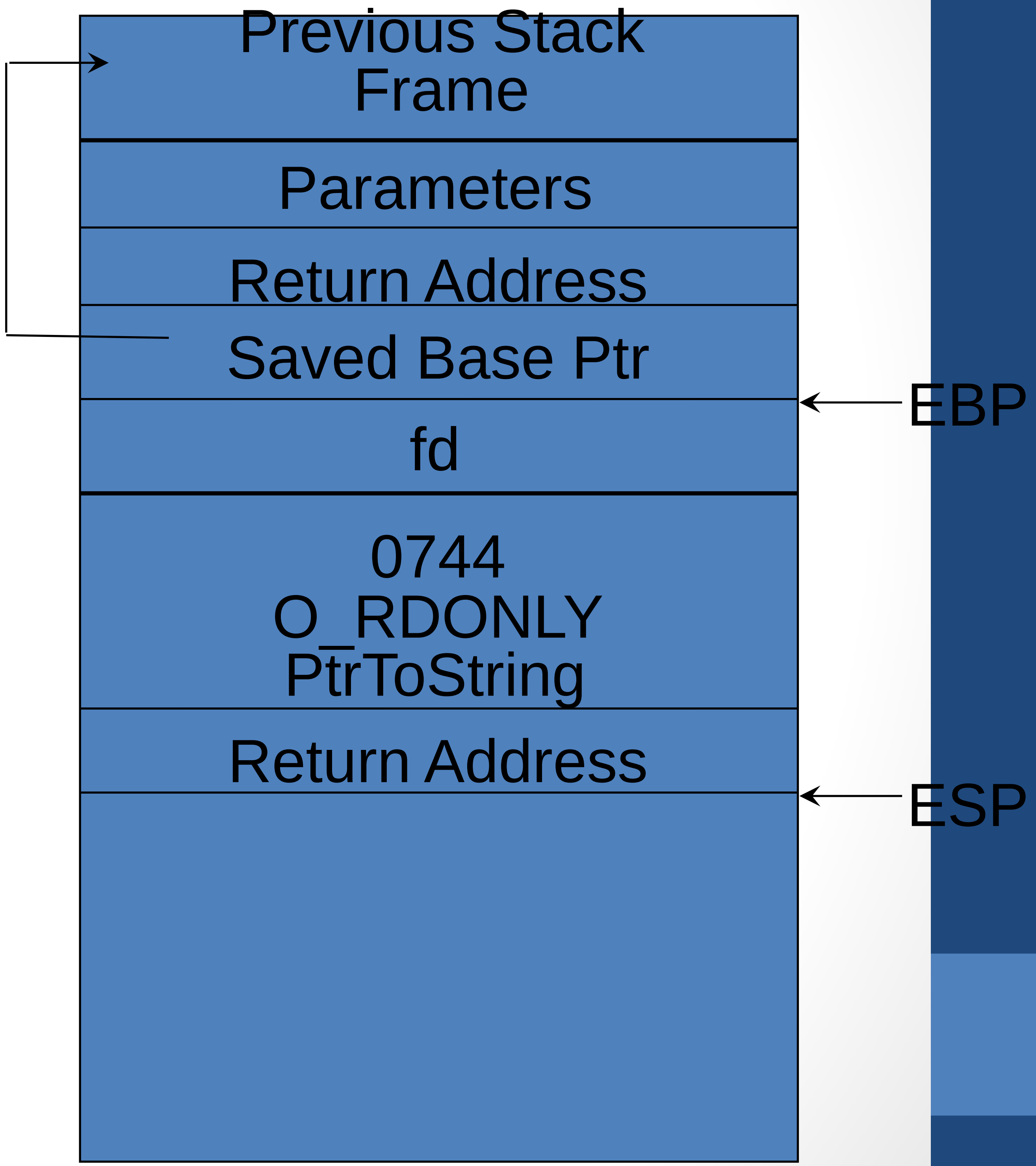
```
push 0744
push O_RDONLY
pushd PtrToString

call open

mov [ebp-fd],eax
add esp,12
```



Pentium Stack Layout

```
push ebp
mov ebp,esp
add esp,NumLocals
```

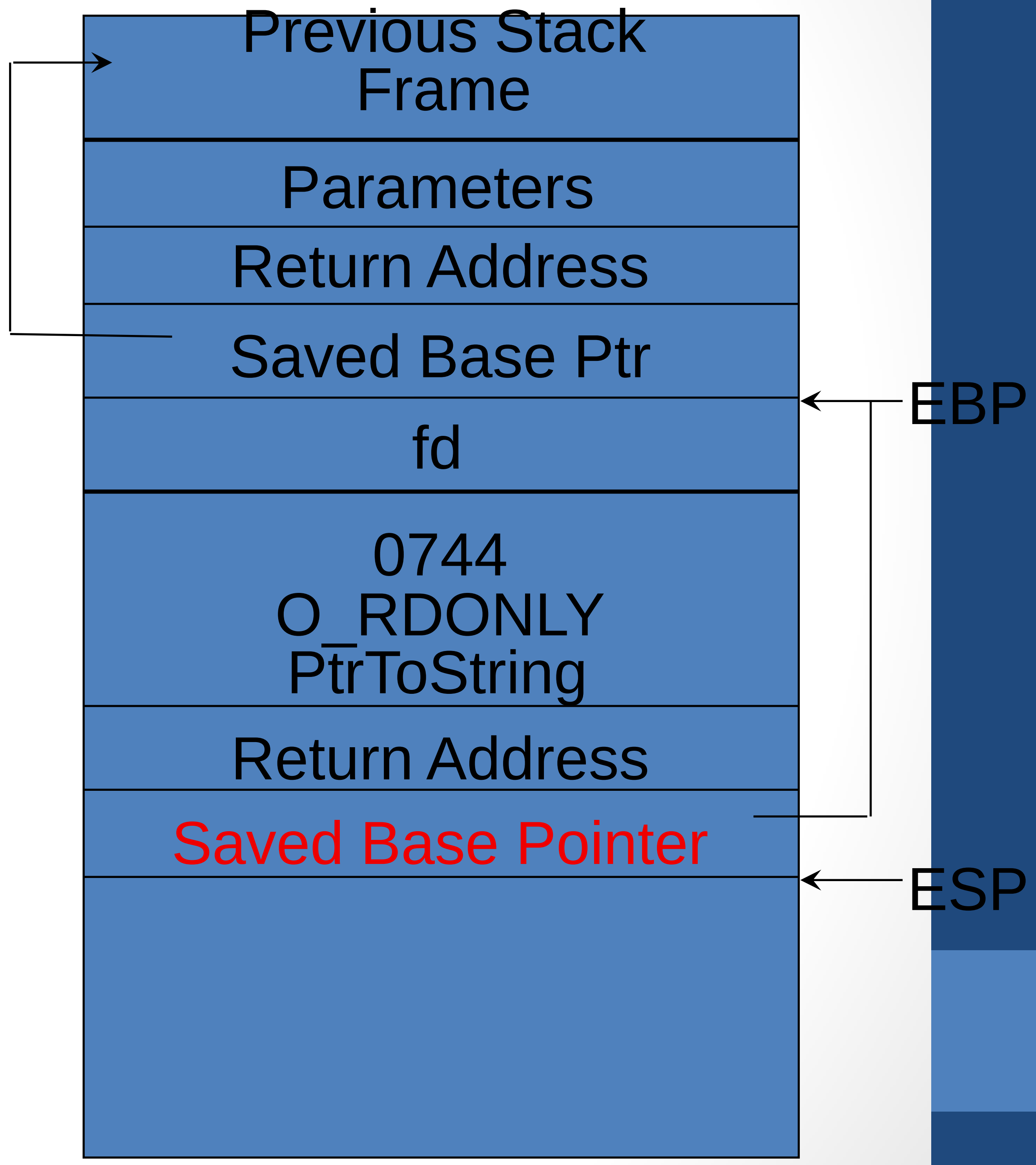
```
leave
ret
```



Pentium Stack Layout

push ebp
mov ebp,esp
add esp,NumLocals

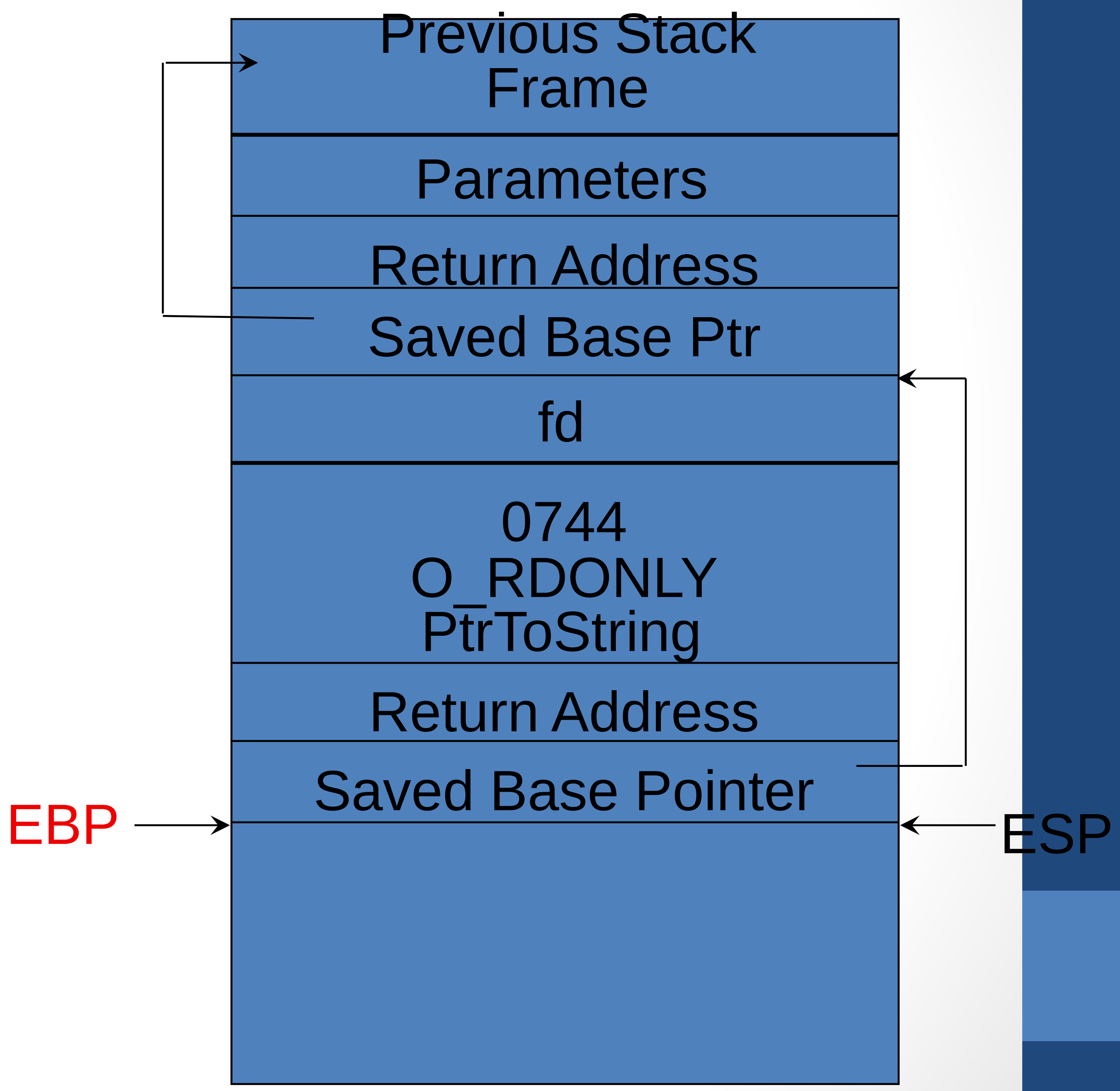
leave
ret



Pentium Stack Layout

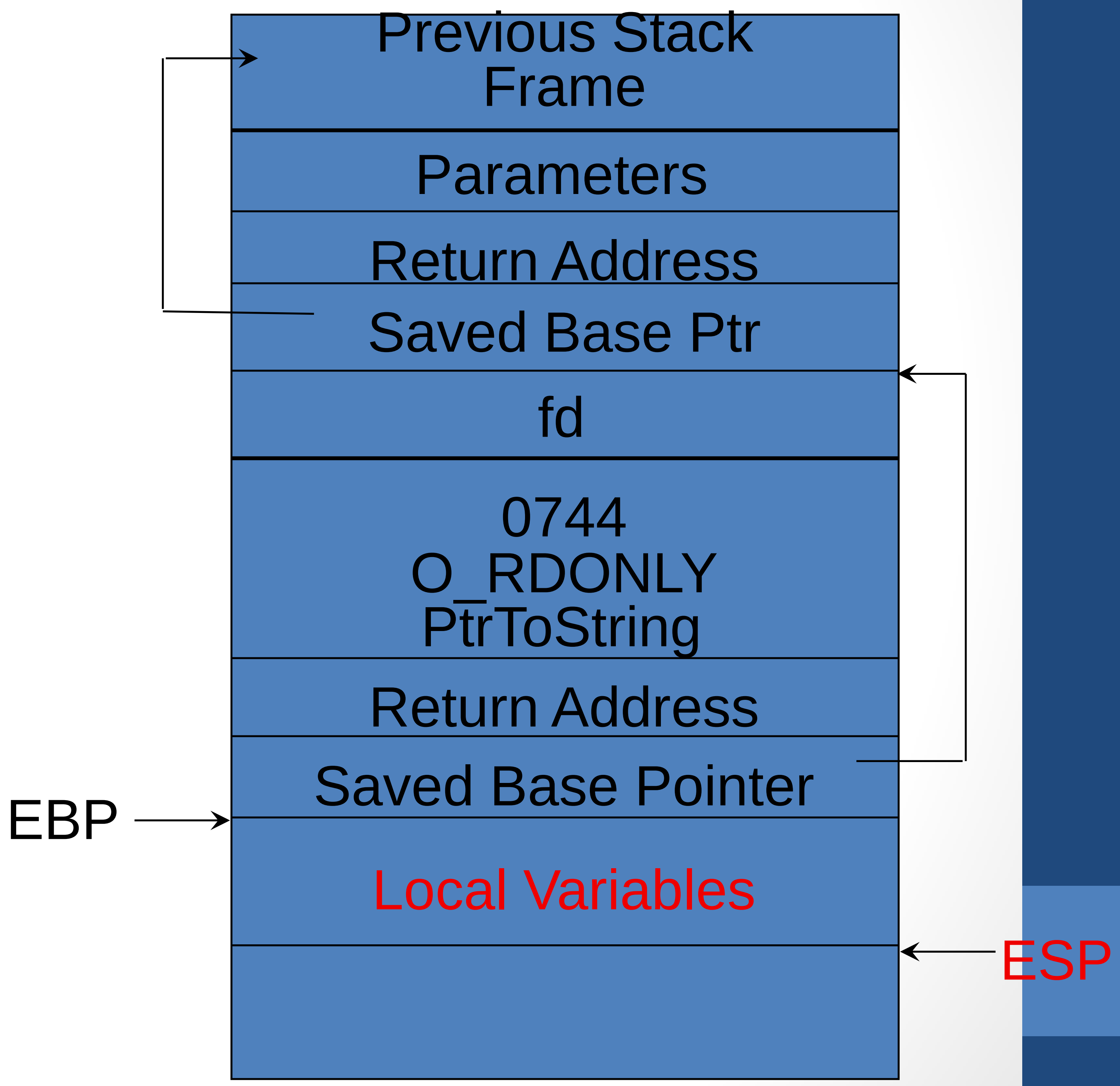
```
push ebp
mov ebp,esp
add esp,NumLocals
```

```
leave
ret
```



Pentium Stack Layout

```
push ebp
mov ebp,esp
add esp,NumLocals
```

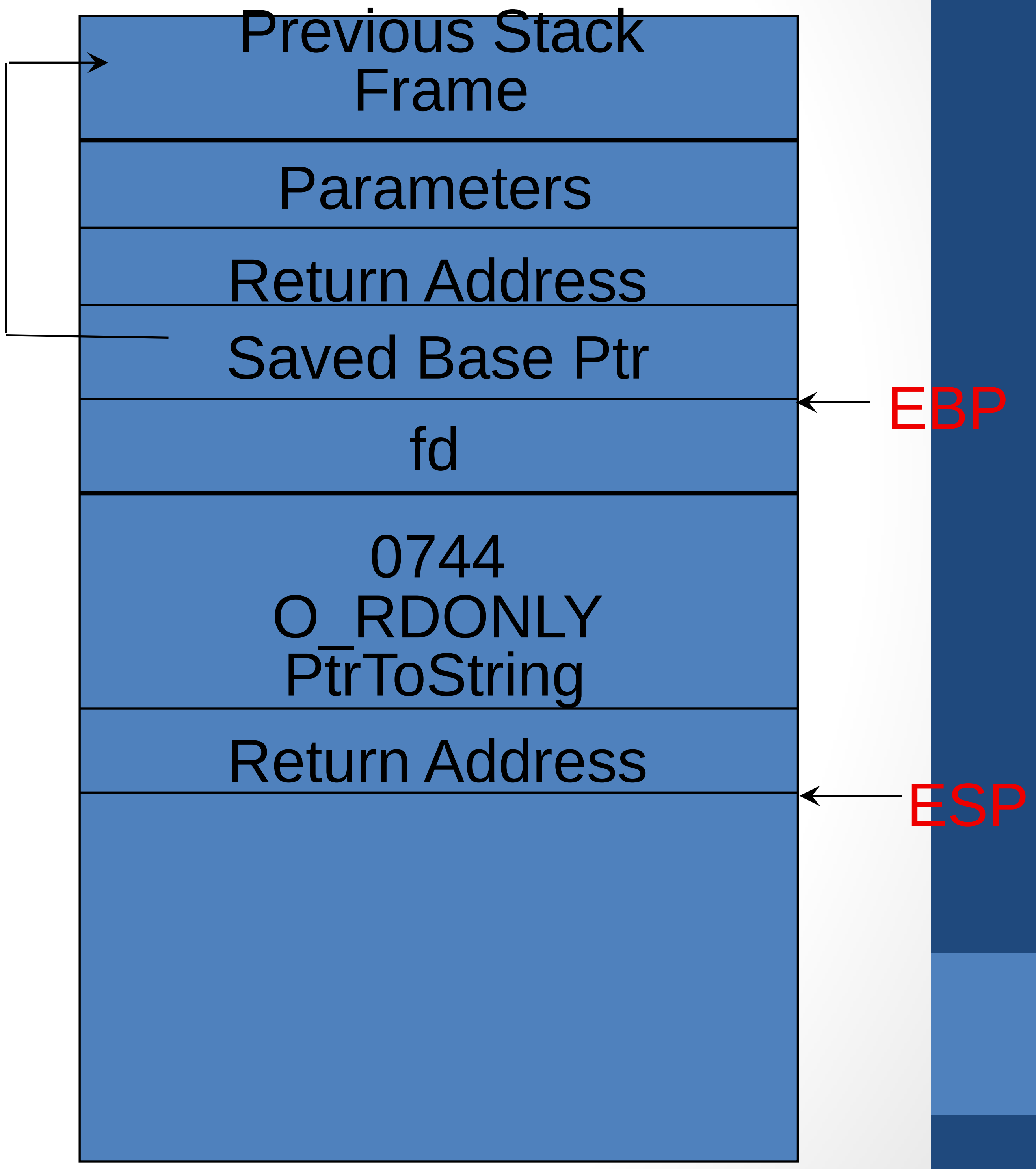


```
leave
ret
```

Pentium Stack Layout

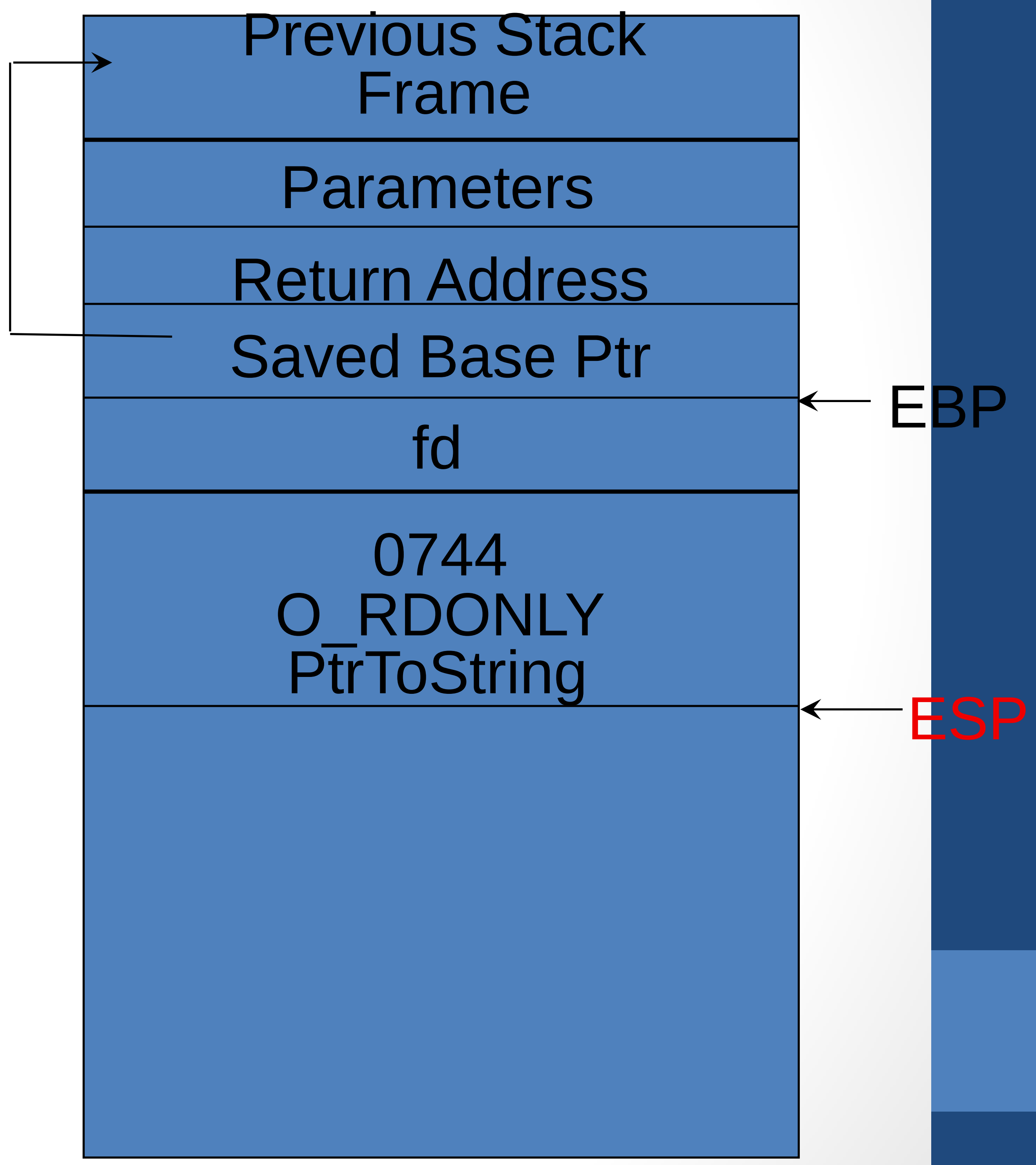
```
push ebp
mov ebp,esp
add esp,NumLocals
```

```
leave
ret
```



Pentium Stack Layout

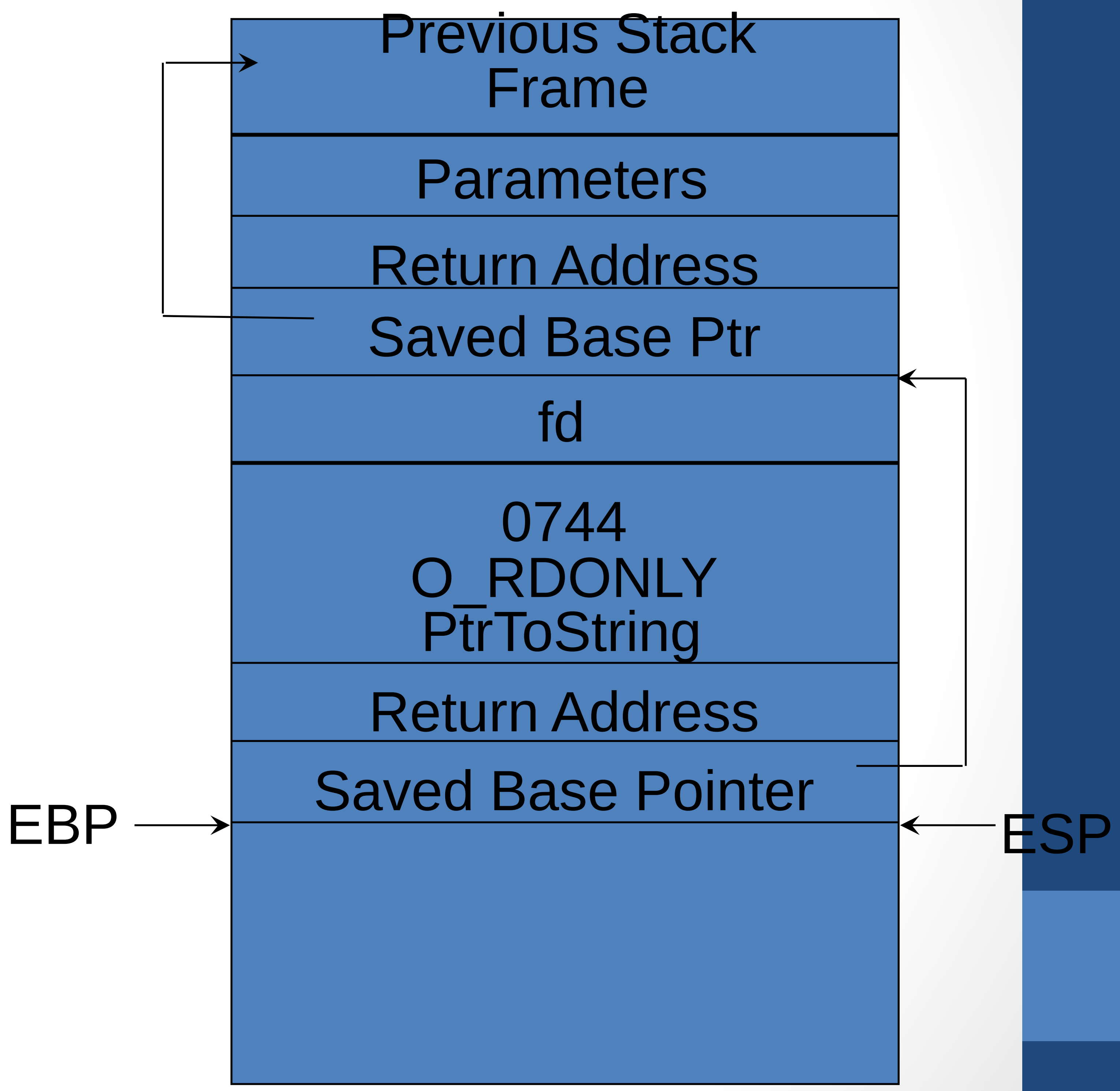
```
push ebp
mov ebp,esp
add esp,NumLocals
```



```
leave
ret
```

Open Function

```
push ebp
mov ebp,esp

mov eax,5
mov ebx,ebp+16
mov ecx,ebp+20
mov edx,ebp+24
int 0x80

ret
```


Previous Stack Frame
Parameters
Return Address
Saved Base Ptr
fd
0744
O_RDONLY
PtrToString
Return Address
Saved Base Pointer
EBP
ESP