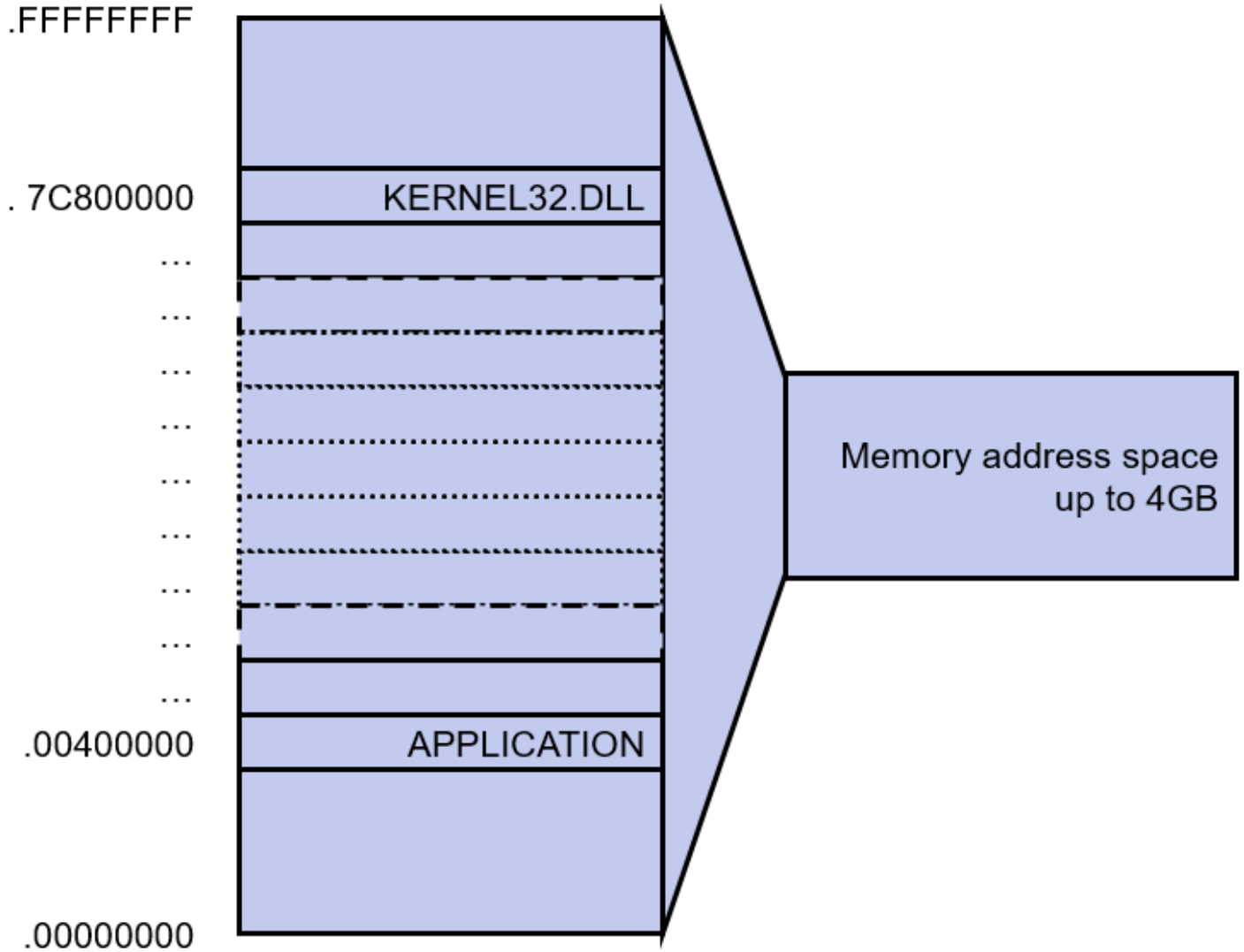
Win32 Assembly

Concepts

- Run in 32-bit segments
- Use FLAT memory model
- Use Application Programming Interface (API)
- Stored in DLLs
- Three main DLLs – Kernel32, GDI32, User32

Flat Memory Model



How is C looks like?

```
#include <header>

<variables, structures, constants, etc.>

function(){
        <program code here>
}
```

How Assembly source code looks like?

```
.586p                   ; supported processor
.model flat
.data                   ; data section
        <variables, constants, structures, etc.>
.code                   ; code section
_entrypoint:            ; code wrapper
        <program code here>
end _entrypoint     ; code wrapper / entrypoint
```

Win32 API Programming

APIs

- Two types of API
  - ANSI – ‘A’ (e.g. ShellExecuteA)
  - Unicode – ‘W’ (e.g. ShellExecuteW)
- Require some parameters to be pushed on the stack before calling them
- Return values is often stored in EAX
- Error will occurred when return value is 0
- Case sensitive

Example

API: MessageBoxA()
Creates, displays, and operates a message box

PARAMETERS:

```
int MessageBox(
    HWND hWnd,                  // handle of owner window
    LPCTSTR lpText, // ptr to the string inside message box
    LPCTSTR lpCaption,          // ptr to the string w/c is the title of message box
    UINT uType                  // style of message box
   );
```

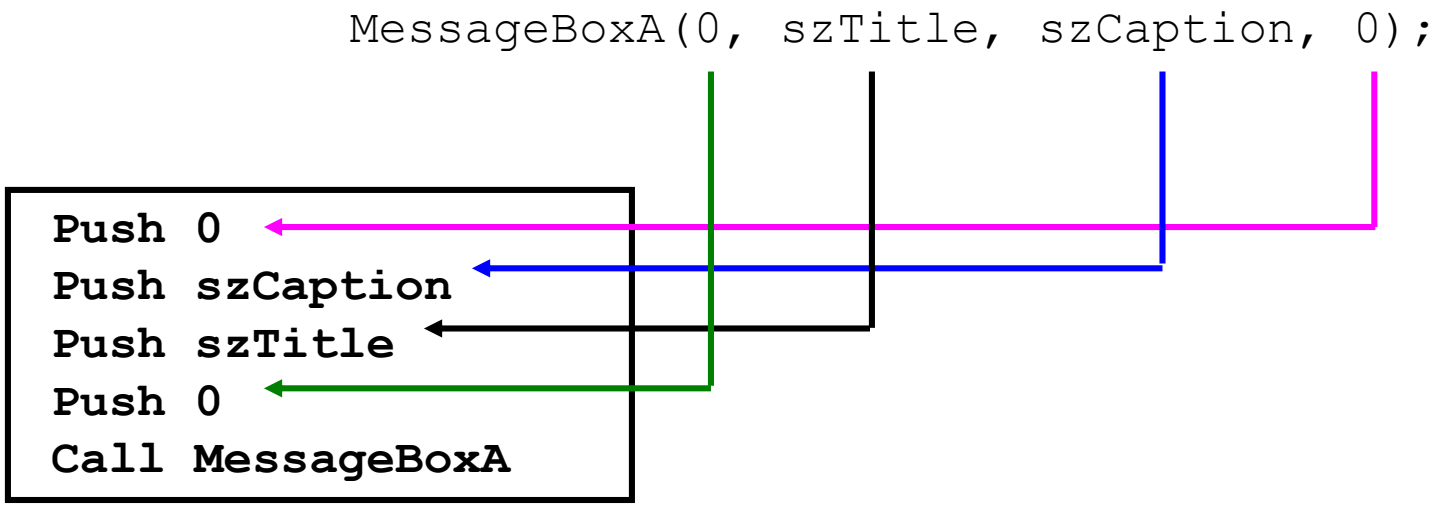
RETURN VALUES:
EAX = 0 (if error occurs)
EAX = value corresponding to user response (If there's no error…)

Calling Convention

API Call in High-Level (in C)

```
MessageBoxA(0, szTitle, szCaption, 0);
```

API Call in Low-Level (in Assembly)



FASM

Flat Assembler

Compiling Assembly Source Code

- Open fasmw.exe and paste or write your code in the pad.
- Follow by compile.
- Once compile, you can straight run the application.

Easy right?

Let’s write something…