

# Понемножку обо всём

Мельник Богдан

Все писали код

C++

C#

Python

Java

JavaScript

```
1 #include <string>
2 #include <iostream>
3
4 const std::string& F(const std::string& s) {
5     return s;
6 }
7
8 template <class C, class F>
9 inline void ForEach(const C& c, const F& f) {
10    for (auto& x : c) {
11        f(x);
12    }
13 }
14
15 int main() {
16    ForEach(F("123"), [] (char x) {
17        std::cout << x << std::endl;
18    });
19
20    for (char x : std::string("123")) {
21        std::cout << x << std::endl;
22    }
23 }
```

JavaApplication1 - NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

SMART Ink Search (Ctrl+I)

Proj... Files Services

JavaApplication1

Source Packages <default package> MyFirstProgram.java Libraries

Start Page MyFirstProgram.java

```
1
2 class MyFirstProgram
3 {
4     public static void main(String args[] ) {
5         System.out.print("Hello world");
6     }
7 }
8
```

MyFirstProgram - Navigator

Members View

MyFirstProgram main(String[] args)

Tasks

run: Hello worldBUILD SUCCESSFUL (total time: 0 seconds)

Output - JavaApplication1 (run)

1 | 1 INS

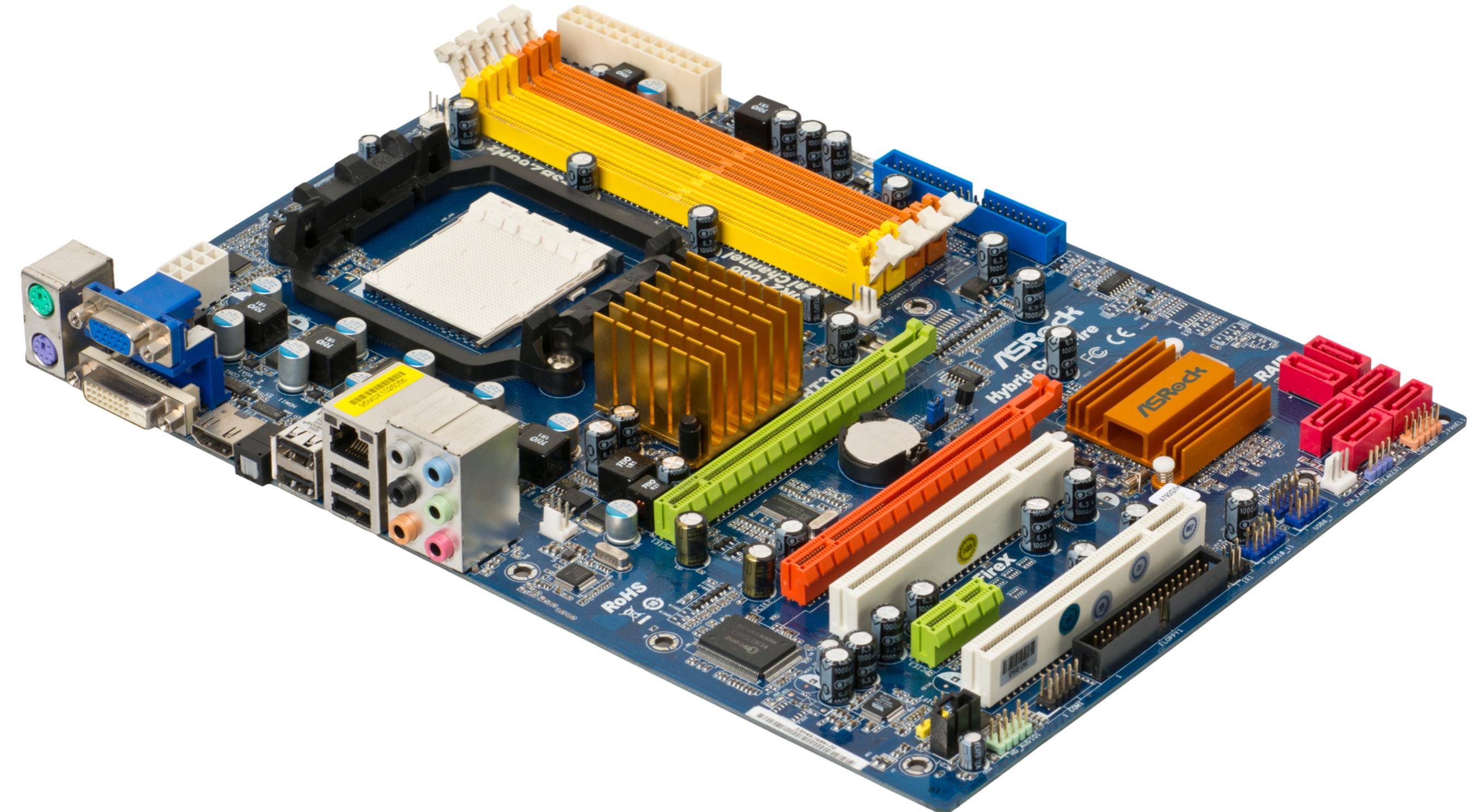
The screenshot shows the NetBeans IDE interface with the title "JavaApplication1 - NetBeans IDE 7.0.1". The menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar has various icons for file operations like New, Open, Save, and Build. The Project Explorer on the left shows a project named "JavaApplication1" with a source package containing "MyFirstProgram.java". The code editor window displays the following Java code:

```
1
2 class MyFirstProgram
3 {
4     public static void main(String args[] ) {
5         System.out.print("Hello world");
6     }
7 }
8
```

The Navigator pane shows the class "MyFirstProgram" and its main method. The Output pane shows the results of running the program, displaying "Hello world" and a build success message. The bottom status bar indicates "1 | 1 INS".

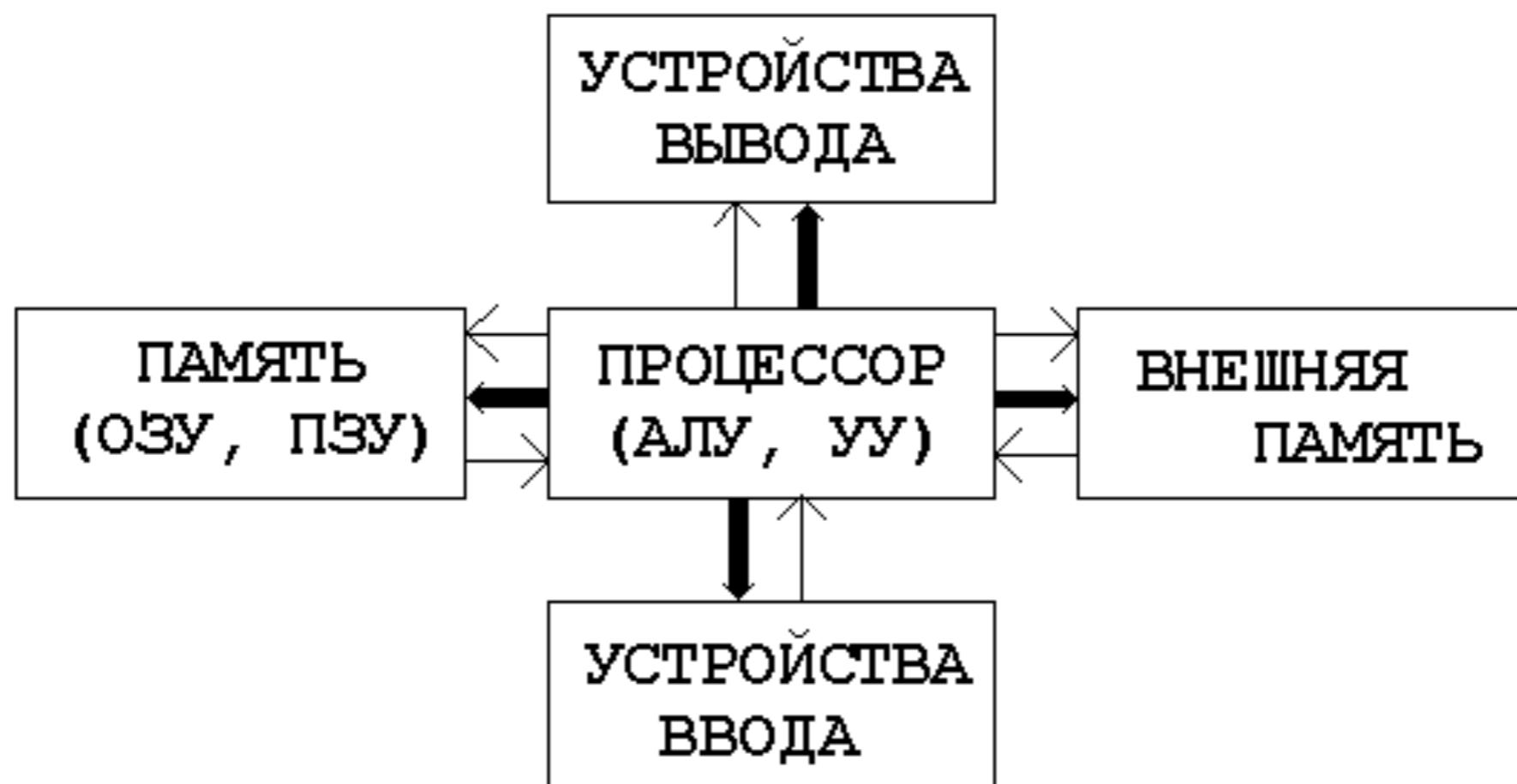
```
5
6 namespace Inheritance
7 {
8     public class Parent
9     {
10         public string name;
11         public Parent(string s)
12         {
13             Console.WriteLine("Parent constructor was called");
14             name = s;
15         }
16     }
17     public class Child : Parent
18     {
19         public Child():base()
20         {
21             Console.WriteLine("Child constructor was called");
22             this.name = "bob";
23         }
24     }
25     class Program
26     {
27         static void Main(string[] args)
28         {
29             Child c = new Child();
30             Console.WriteLine(c.name);
31         }
32     }
33 }
34 }
```

Что объединяет все языки?



Как это работает?

# Принципы фон Неймана



# Принципы фон Неймана

Однородности памяти

Адресуемости памяти

Программного управления

Двоичного кодирования

# Архитектуры повсюду

x86

x86-64

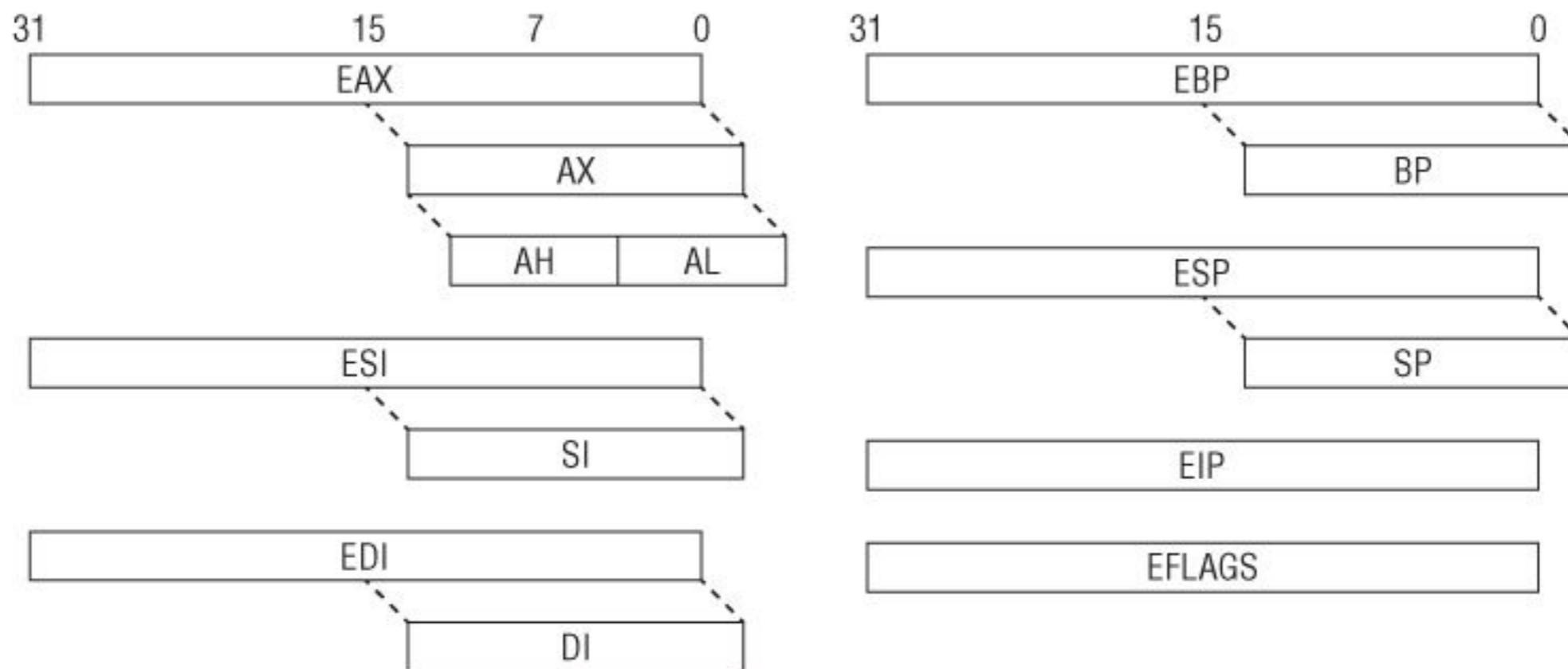
ARM

MIPS

...

Только x86 архитектура

# Регистры



# Инструкции

- Перемещение данных
- Арифметические инструкции
- Управление потоком исполнения
- Вызов функций, операции со стеком

# Синтаксис

Intel

```
mov eax, 12345678h  
mov eax, dword ptr [ebx]  
mov eax, ecx
```

AT&T

```
movl $0x12345678, %eax  
movl (%ebx), %eax  
movl %ecx, %eax
```

# Перемещение данных

```
mov eax, 12345678h
; eax = 0x12345678
mov ebx, eax
; ebx = eax
mov dword ptr [eax], 1
; *eax = 1
mov ecx, dword ptr [eax]
; ecx = *eax
```

# Перемещение данных

```
mov dword ptr [eax], ebx  
; *eax = ebx  
  
mov dword ptr [esi+34h], eax  
; *(esi+34h) = eax  
  
mov eax, dword ptr [esi+34h]  
; eax = *(esi+34h)  
  
mov edx, dword ptr [ecx+eax]  
; edx = *(ecx+eax)
```

# Арифметические операции

add eax, ebx	; eax = eax + ebx
sub eax, ebx	; eax = eax - ebx
inc eax	; eax = eax + 1
dec eax	; eax = eax - 1
or eax, 0xFFFFFFFFh	; eax = eax   0xFFFFFFFF
and eax, 0FFh	; eax = eax & 0xFF
xor eax, eax	; eax = eax ^ eax
not eax	; eax = ~eax
shl eax, 2	; eax = eax << 2
shr eax, 2	; eax = eax >> 2
mul ecx	; edx:eax = eax * ecx
div ecx	; eax = edx:eax / ecx

# Управление потоком

Все условные конструкции `if/else`, `switch` и `while/for` транслируются в несколько типов инструкций.

Инструкции влияющие на регистр флагов:

- `cmp`
- `test`
- Все арифметические инструкции.

Инструкции меняющие поток исполнения:

- `jmp`
- `jcc`

# Регистр флагов

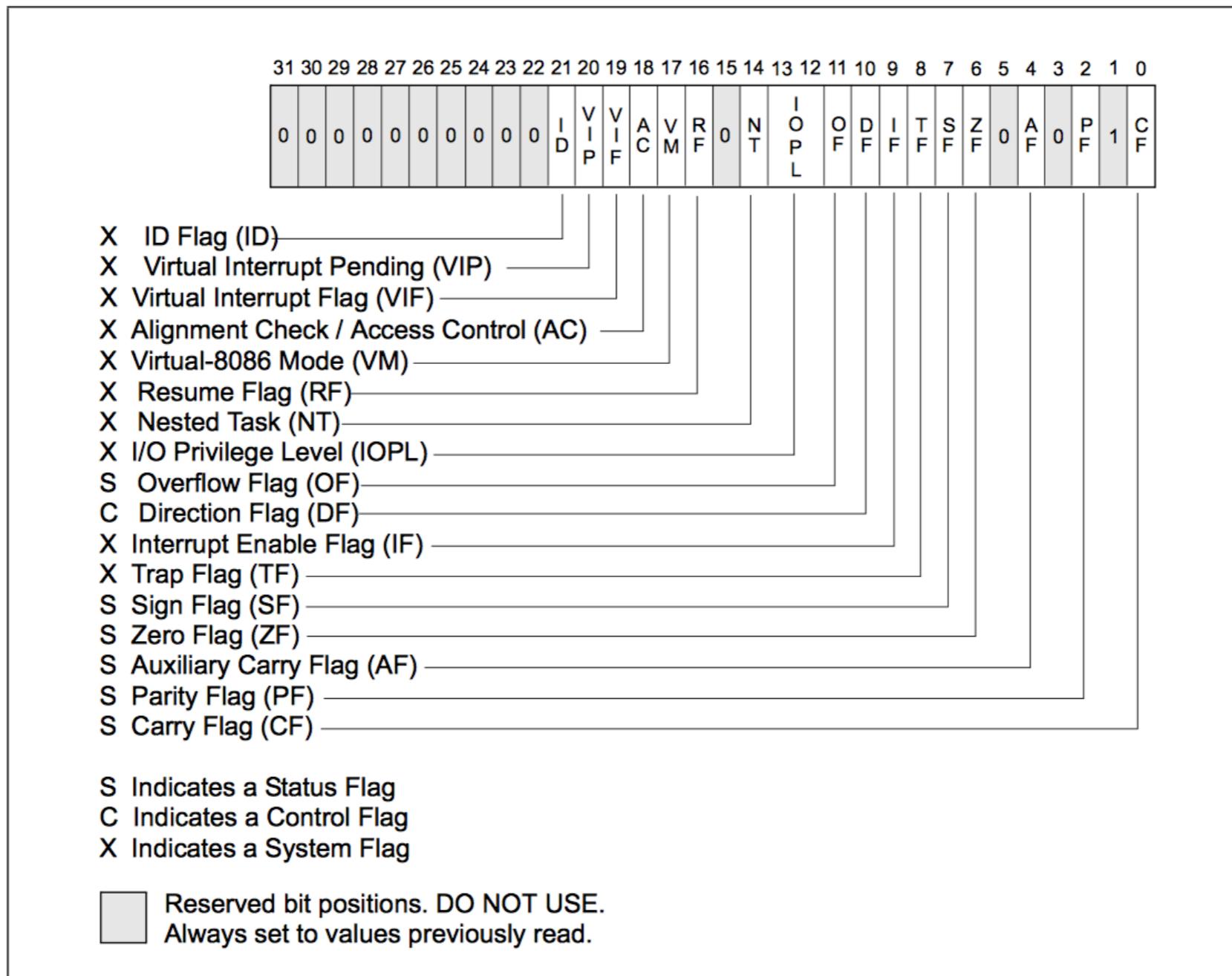


Figure 3-8. EFLAGS Register

# jmp

Используется для безусловного перехода, аналог goto в С.

```
mov eax, 1
jmp a
mov ebx, 10      ; Это не будет исполнено
a:
mov ebx, 5
```

## jXX

Используются для перехода в зависимости от значения регистра флагов.

- JB/JNAE — Меньше, SF == 1
- JNB/JAE — Не меньше, SF == 0
- JE/JZ — Равно, ZF == 1
- JNE/JNZ — Не равно, ZF == 0
- JL — Меньше знаково, (SF ^ OF) == 1
- JGE/JNL — Не меньше знаково, (SF ^ OF) == 0
- JG/JNLE — Больше знаково, ((SF ^ OF) | ZF) == 0

# Пример кода

```
mov eax, 12345678h
xor ecx, ecx
mov edx, 32
label:
    test eax, 1
    je a
    inc ecx
a:
    shr eax, 1
    dec edx
    jne label
```

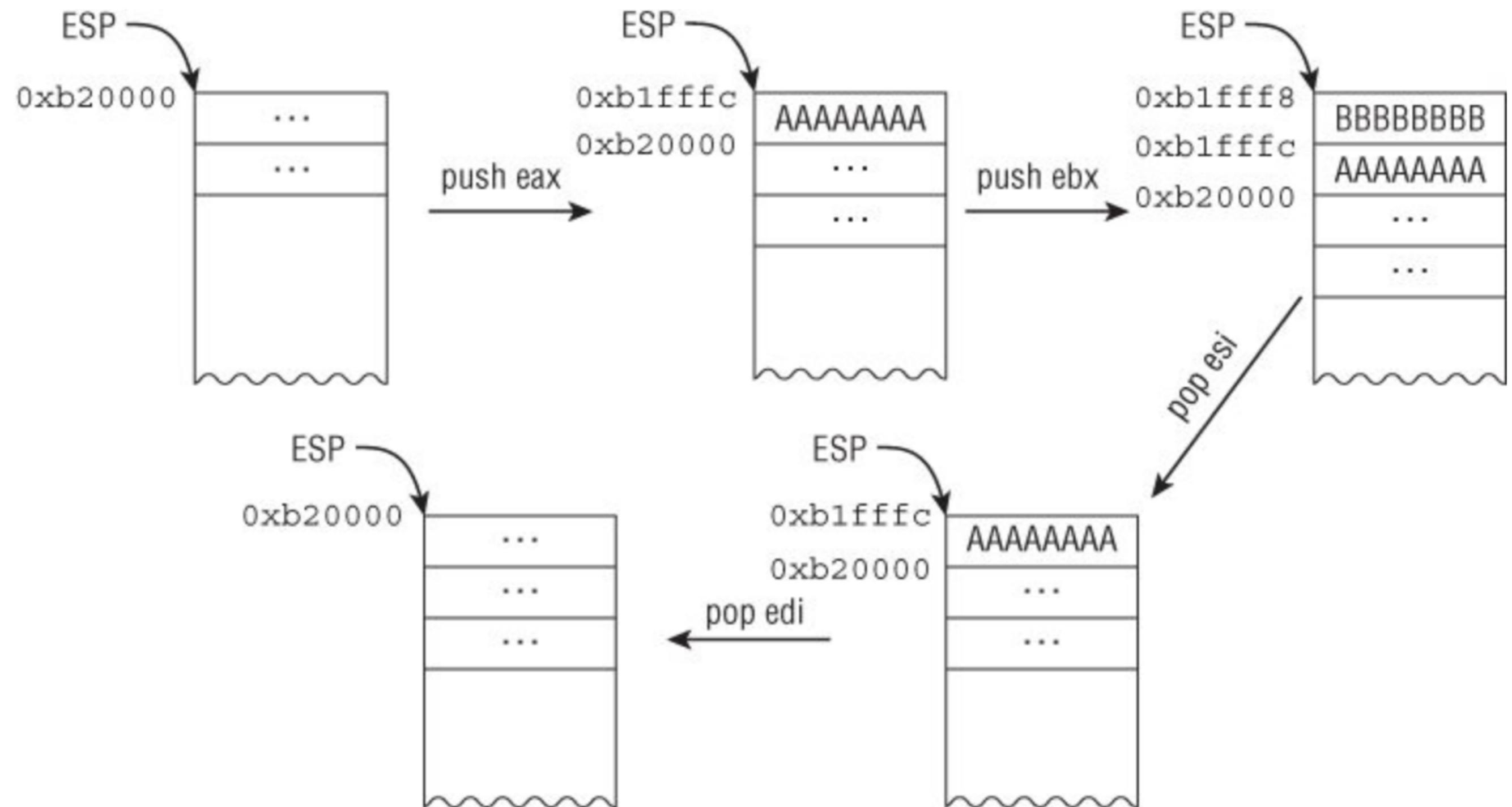
# Стек

**Стек** (англ. *stack* – стопка; читается *стэк*) – структура данных, представляющая собой список элементов, организованных по принципу *LIFO* (англ. *last in – first out*, «последним пришёл – первым вышел»).

Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

Аппаратная поддержка стека

# Стек



# Вызов функций

Для вызова функций было придумано несколько инструкций.

Самые важные это **CALL** и **RET**

# CALL и RET

**0x40000000 call foo**

0x40000005 mov ebx, 1

0x40001000 foo:

0x40001000 mov eax, 1

0x40001005 ret

0x7C81776F

0x7C910228

# CALL и RET

```
0x40000000 call foo
0x40000005 mov ebx, 1

0x40001000 foo:
0x40001000 mov eax, 1
0x40001005 ret
```

0x40000005

0x7C81776F

0x7C910228

# CALL и RET

```
0x40000000 call foo
0x40000005 mov ebx, 1

0x40001000 foo:
0x40001000 mov eax, 1
0x40001005 ret
```

0x40000005

0x7C81776F

0x7C910228

# CALL и RET

```
0x40000000 call foo  
0x40000005 mov ebx, 1
```

```
0x40001000 foo:  
0x40001000 mov eax, 1  
0x40001005 ret
```

0x7C81776F

0x7C910228

# Передача аргументов

Самый простой и логичный способ передавать аргументы через регистры, но это не всегда возможно.

# Передача аргументов

	<b>CDECL</b>	<b>STDCALL</b>	<b>FASTCALL</b>
Parameters	Pushed on the stack from right-to-left. Caller must clean up the stack after the call.	Same as CDECL except that the callee must clean the stack.	First two parameters are passed in ECX and EDX. The rest are on the stack.
Return value	Stored in EAX.	Stored in EAX.	Stored in EAX.
Non-volatile registers	EBP, ESP, EBX, ESI, EDI.	EBP, ESP, EBX, ESI, EDI.	EBP, ESP, EBX, ESI, EDI.



**I know kung fu.**

Я знаю ассемблер

Но зачем?

モニタを用いて、モニタの映像を直接工場内に表示する。モニタは、モニタ用の映像を直接工場内に表示する。モニタは、モニタ用の映像を直接工場内に表示する。

# Бинарный код

```
eb 03 5e eb 05 e8 f8 ff ff ff 83 c6 0f 31 c9 66
b9 1c 02 80 36 03 46 e2 fa ea b8 03 03 03 03 2c 61
6a 6d 2c 70 6b 03 7d 03 93 93 93 93 93 93 93 93 93
93 93 93 93 93 93 93 93 7f 79 75 76 76 77 72
6e 69 68 6d 70 76 76 72 6e 69 64 65 60 63 5e 58
58 5e 5d 5d 5e 5c 61 64 6e 70 78 a5 af b1 b5 bf
bc c0 c6 c6 c1 c2 bd be b9 bb b6 ac aa a0 6d 6f
66 6a 69 64 55 26 33 4d 54 5d 43 38 35 46 40 63
60 61 55 48 49 4d 5d 66 6e 6a 64 63 59 54 63 69
6f 70 7a 7b 77 7b 79 75 71 6d 72 72 72 70 71 72 6d
```

# Reverse engineering

# Reverse engineering

**Обратная разработка (обратный инжиниринг, реверс-инжиниринг; англ. *reverse engineering*)** – исследование некоторого устройства или программы, а также документации на него с целью понять принцип его работы; например, чтобы обнаружить недокументированные возможности (в том числе «программные закладки»), сделать изменение, или воспроизвести устройство, программу или иной объект с аналогичными функциями, но без копирования как такового.

# Анализ вредоносных программ

- Вирусы
- Трояны
- Руткиты
- Черви

# Анализ вредоносных программ

- Что делает вредоносная программа?
- Кто ей управляет?
- Как обезвредить её?
- Как создать защиту от будущих атак?

# Поиск и использование уязвимостей

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	
1	<a href="#">CVE-2014-6335</a>	<a href="#">94</a>		DoS Exec Code Mem. Corr.	2014-11-11	2014-12-02	<span style="background-color: red; color: white; padding: 2px;">9.3</span>	
Microsoft Word 2007 SP3, Word Viewer, and Office Compatibility Pack SP3 allow remote attackers to execute arbitrary code or cause a denial of service via crafted properties in a Word document, aka "Microsoft Office Invalid Pointer Remote Code Execution Vulnerability."								
2	<a href="#">CVE-2014-6334</a>	<a href="#">94</a>		DoS Exec Code Mem. Corr.	2014-11-11	2014-12-02	<span style="background-color: red; color: white; padding: 2px;">9.3</span>	
Microsoft Word 2007 SP3, Word Viewer, and Office Compatibility Pack SP3 allow remote attackers to execute arbitrary code or cause a denial of service via crafted properties in a Word document, aka "Microsoft Office Bad Index Remote Code Execution Vulnerability."								
3	<a href="#">CVE-2014-6333</a>	<a href="#">94</a>		Exec Code	2014-11-11	2014-12-02	<span style="background-color: red; color: white; padding: 2px;">9.3</span>	
Microsoft Word 2007 SP3, Word Viewer, and Office Compatibility Pack SP3 allow remote attackers to execute arbitrary code or cause a denial of service via crafted properties in a Word document, aka "Microsoft Office Invalid Pointer Remote Code Execution Vulnerability."								
4	<a href="#">CVE-2014-4117</a>	<a href="#">20</a>		Exec Code	2014-10-15	2014-10-31	<span style="background-color: red; color: white; padding: 2px;">9.3</span>	
Microsoft Office 2007 SP3, Word 2007 SP3, Office 2010 SP1 and SP2, Word 2010 SP1 and SP2, Office for Mac 2011, Office Communicator 2010 SP1 and SP2, and Word Web Apps 2010 Gold, SP1, and SP2 allow remote attackers to execute arbitrary code via crafted properties in a Word document, aka "Office Object Embedding Vulnerability."								
5	<a href="#">CVE-2014-2778</a>	<a href="#">119</a>		DoS Exec Code Overflow Mem. Corr.	2014-06-11	2014-06-18	<span style="background-color: red; color: white; padding: 2px;">9.3</span>	
Microsoft Word 2007 SP3 and Office Compatibility Pack SP3 allow remote attackers to execute arbitrary code or cause a denial of service via crafted properties in a Word document, aka "Embedded Font Vulnerability."								
6	<a href="#">CVE-2014-1761</a>	<a href="#">119</a>		DoS Exec Code Overflow Mem. Corr.	2014-03-25	2014-12-08	<span style="background-color: red; color: white; padding: 2px;">9.3</span>	
Microsoft Word 2003 SP3, 2007 SP3, 2010 SP1 and SP2, 2013, and 2013 RT; Word Viewer; Office Compatibility Pack SP3; Office 2010 SP1 and SP2; Office 2013; Office Web Apps 2010 SP1 and SP2; and Office Web Apps Server 2013 allow remote attackers to execute arbitrary code via crafted properties in a Word document, aka "Office Object Embedding Vulnerability." This vulnerability was disclosed publicly in March 2014.								
7	<a href="#">CVE-2014-1758</a>	<a href="#">119</a>		Exec Code Overflow	2014-04-08	2014-04-09	<span style="background-color: red; color: white; padding: 2px;">9.3</span>	

for fun and profit



**PCTF**  
2011

# Интересно?

Bogdan Melnik

[ld@hackerdom.ru](mailto:ld@hackerdom.ru)

<https://ld86.org/re/14/>