

The logo consists of the text "< Teach Me Skills />" in a white, sans-serif font. The less-than and greater-than symbols are yellow. The background of the logo is a dark navy blue rectangle.

< Teach
Me
Skills />

Reverse engineering

Собираемся и отмечаемся

Вопросы по предыдущим темам или ДЗ

Mini-quizе по прошлым темам:

1. Какие существуют политики резервного копирования?
2. В чем заключаются достоинства и недостатки групп в политиках администрирования AD?
3. Как работает механизм доменных имен AD, в целом?
4. Если в компании развернуты только ОС Linux, какую AD вы бы использовали?

Mini-quiz по новой теме:

1. В чем разница между интерпретатором и компилятором?
2. C# с фреймворком .NET является компилируемым или интерпретируемым ЯП?
3. Какие программы для написания кода вы знаете?

План занятия

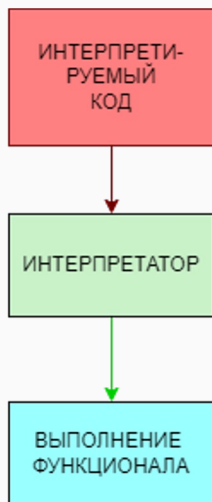
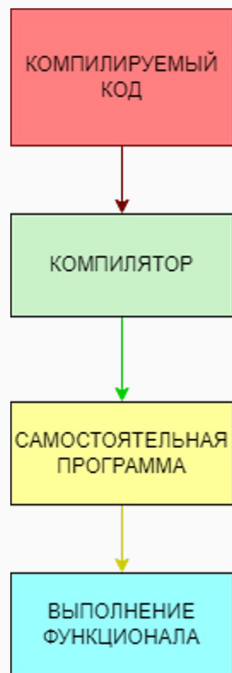
1. Разберемся как работают компиляторы
2. Изучим основные опкоды и команды процессора
3. Подробнее изучим механизм подключения зависимостей PE файлов
4. Рассмотрим отличия в архитектуре x86 ia64

Что такое реверс-инжиниринг?

```

; var uint32 t var ch @ rbp-0xc
; var uint32 t var 0h @ rbp-0x8
; var int64 t var 4h @ rbp-0x4
0x00001135 55          leah rbp
0x00001136 4889e5      mov rbp, rsp
0x00001139 4883ec10    sub rsp, 0x10
0x0000113d c745fc0000. mov dword [var 4h], 0
0x00001144 c745f8010000. mov dword [var 0h], 1
0x0000114b c745f4020000. mov dword [var ch], 2
0x00001152 8b45fc      mov eax, dword [var 4h]
0x00001155 3b45f8      cmp eax, dword [var 0h]
0x00001158 7522        jne 0x117c
0x0000115a 837df400    cmp dword [var ch], 0
0x0000115e 750e        jne 0x116e
0x00001160 488d3d9d0e90. lea rdi, str.z_0_x_y ; 0x2004 : "z = 0; x = y" ; const char *s
0x00001167 e8c4fe1177. call sym.imp.puts ; int puts(const char *s)
0x0000116c eb2e        jmp 0x119c
; 0000 0007 from main @ 0x1136
0x0000116e 488d3d9c0e90. lea rdi, str.z_0_x_y ; 0x2004 : "z = 0; x = y" ; const char *s
0x00001175 e8b6fe1177. call sym.imp.puts ; int puts(const char *s)
0x0000117a eb20        jmp 0x119c
; 0000 0007 from main @ 0x1136
0x0000117c 837df400    cmp dword [var ch], 0
0x00001180 750e        jne 0x1190
0x00001182 488d3d960e90. lea rdi, str.z_zero_and_x_y ; 0x2017 : "z = zero and x = y." ; const char *s
0x00001189 e8a2fe1177. call sym.imp.puts ; int puts(const char *s)
0x0000118e eb0c        jmp 0x119c
; 0000 0007 from main @ 0x1136
0x00001190 488d3d9d0e90. lea rdi, str.z_non_zero_and_x_y ; 0x2024 : "z non-zero and x = y." ; const char *s
0x00001197 e894fe1177. call sym.imp.puts ; int puts(const char *s)
; 0000 0007 from main @ 0x1136
0x0000119c 90          nop
0x0000119d c9          leave
0x0000119e c3          ret

```



Компилируемые ЯП

Swift
C++/C
Java
Objective-c
Rust
Go

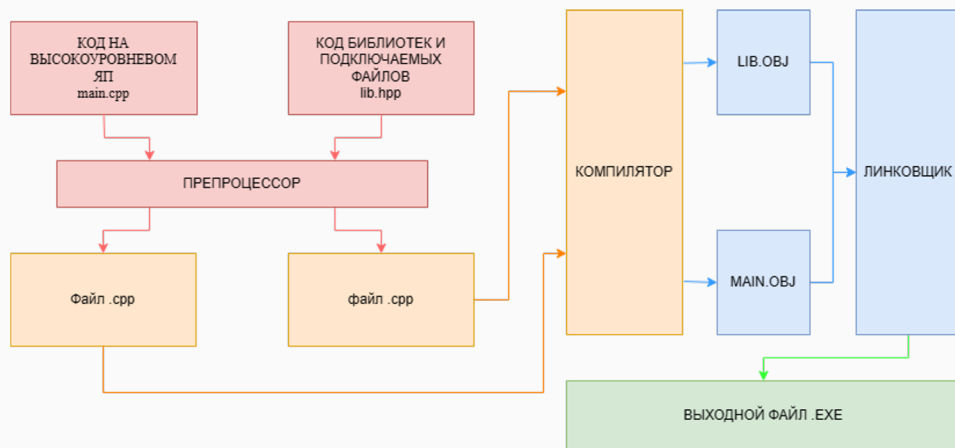
Интерпретируемые ЯП

Python
JavaScript
PHP
Perl
Ruby
C#

[Компилятор VS интерпретатор](#)

Процесс сборки приложения компилятором

Общая схема формирования исполняемого файла



Преппроцессор — это компьютерная программа, принимающая данные на входе и выдающая данные, предназначенные для входа другой программы (например, компилятора).

Основная задача - обработка исходного кода перед передачей его на следующий шаг компиляции

Код до препроцессора

```
#define A 10
#define B 40
#define TEST

// MAIN FUNCTION
int main(){
    #ifdef TEST
        std::cout << 10 << std::endl;
    #else
        std::cout << 40 << std::endl;
    #endif

    return 0;
}
```

Код после оптимизации

```
int main(){
    std::cout << 10 << std::endl;

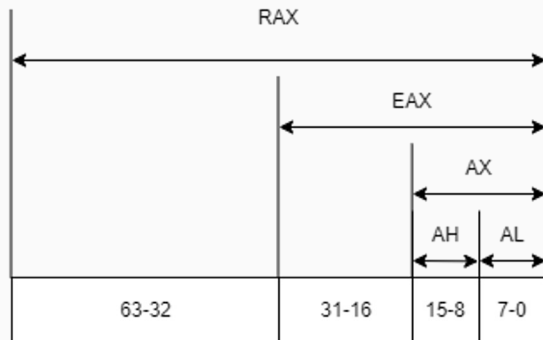
    return 0;
}
```

Память ПК

Пирамидка памяти



Масштабирование регистров



Регистр — устройство для записи, хранения и считывания n -разрядных двоичных данных и выполнения других операций над ними.

Иерархия памяти

Регистры данных

AH	AL	Аккумулятор
BH	BL	Базовый регистр
CH	CL	Счетчик
DH	DL	Регистр данных

Регистры-указатели

SI	Индекс источника
DI	Индекс приемника
BP	Указатель базы
SP	Указатель стека

Сегментные регистры

CS	Регистр сегмента команд
DS	Регистр сегмента данных
ES	Регистр дополнительного сегмента данных
SS	Регистр сегмента стека

Прочие регистры

IP	Указатель команд
FLAGS	Регистр флагов

Регистры

Архитектура процессора это основные принципы, структура и организация внутренних элементов процессора, который является центральным элементом компьютерной системы. Архитектура процессора определяет, как процессор выполняет инструкции программы, управляет данными и взаимодействует с другими компонентами компьютера.

Инструкции

ISA описывает список поддерживаемых инструкций. бывает CISC или RISC.
RISC - Простые команды
CISC - Сложные команды



Данные и АЛУ

функционал относительно арифметических и логических операций, передача данных, управление шинами данных, адресация.



Регистры

Минимальные ячейки памяти, внутри процессора. Используются для временного хранения данных.



Управление потоком

Указатель команд, счетчик команд, стек вызова и переходы.



Регистры x86

SSE 128 бит		64 бита	
XMM0		RAX	EAX
XMM1		RBX	EBX
XMM2		RCX	ECX
XMM3		RDY	EDX
XMM4		RSI	ESI
XMM5		RDI	EDI
XMM6		RBP	EBP
XMM7		RSP	ESP
XMM8		R8	
XMM9		R9	
XMM10		R10	
XMM11		R11	
XMM12		R12	
XMM13		R13	
XMM14		R14	
XMM15		R15	

Регистры Arm

Общие регистры и режимы					
User32	FIQ32	Supervisor32	Abort32	IRQ32	Undefined32
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_flg	R8	R8	R8	R8
R9	R9_flg	R9	R9	R9	R9
R10	R10_flg	R10	R10	R10	R10
R11	R11_flg	R11	R11	R11	R11
R12	R12_flg	R12	R12	R12	R12
R13	R13_flg	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_flg	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)
Регистры состояния программы					
CPSR	CPSR SPSR_flg	CPSR SPSR_svc	CPSR SPSR_abt	CPSR SPSR_irq	CPSR SPSR_und

Ассемблер и его язык

Язык ассемблера x86 — это название семейства языков ассемблера, которые обеспечивают некоторый уровень обратной совместимости с процессорами вплоть до микропроцессора Intel 8008

Команды ассемблера состоят из: кодов операций и операндов.

Операнды — это адреса, из которых процессор будет брать данные для вычислений и в которые будет помещать результат. Адресами могут быть ячейки оперативной памяти и регистры.

Операции (опкоды) в языке ассемблера — мнемонические, для удобного чтения людьми.

Машинный код vs Ассемблер

Машинный код (побайтовый)

```
B8 22 11 00 FF
01 CA
31 F6
53
8B 5C 24 04
8D 34 48
39 C3
72 EB
C3
```

Код на Ассемблере

```
foo:
movl $0xFF001122, %eax
addl %ecx, %edx
xorl %esi, %esi
pushl %ebx
movl 4(%esp), %ebx
leal (%eax,%ecx,2), %esi
cmpl %eax, %ebx
jnae foo
retl
```

ADD - сложение

SUB - вычитание

MOV - перемещение

INC - инкрементация

DEC - декрементация

JMP — безусловный прыжок (переход)

Типы данных

Тип	Размер	Тип	Размер
int	2	long	4
unsigned int	2	signed long	4
signed int	2	unsigned long	4
short	2	float	4
unsigned short	2	double	8
signed short	2	long double	8
char	1	signed char	1
unsigned char	1	bool	1

signed int	-127...127	8-bit
------------	------------	-------

unsigned int	0...255	8-bit
--------------	---------	-------

Типы данных, которыми оперирует ПК

Пример хранения данных в памяти

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZh.яя..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	ë.....@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00ш...
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	e .r.H!ë[]LH!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is.program.canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t.be.run.in.DOS.
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode.....\$.....
00000080	0F	95	11	73	4B	F4	7F	20	4B	F4	7F	20	4B	F4	7F	20	*sK .K .K .
00000090	5E	8B	7E	21	49	F4	7F	20	5E	8B	7A	21	56	F4	7F	20	^<~!I .^<z!V .
000000A0	5E	8B	7B	21	40	F4	7F	20	5E	8B	7C	21	48	F4	7F	20	^<{ @ .^< !H .
000000B0	AF	84	7E	21	4C	F4	7F	20	4B	F4	7E	20	05	F4	7F	20	I„~!L .K . .
000000C0	7D	74	7A	21	4A	F4	7F	20	7D	74	80	20	4A	F4	7F	20	}tz!J .}tE.J .
000000D0	4B	F4	E8	20	4A	F4	7F	20	7D	74	7D	21	4A	F4	7F	20	K .J .}t!J .
000000E0	52	69	63	68	4B	F4	7F	20	00	00	00	00	00	00	00	00	RichK
000000F0	00	00	00	00	00	00	00	00	50	45	00	00	64	86	0A	00FE..dt..
00000100	04	12	0A	65	00	00	00	00	00	00	00	00	F0	00	22	00	.e.....p."
00000110	0B	02	0E	25	00	86	00	00	00	3A	02	00	00	00	00	00	%..t....:
00000120	D0	12	01	00	00	10	00	00	00	00	40	01	00	00	00	00	E@ . . .

Сбор информации о файле

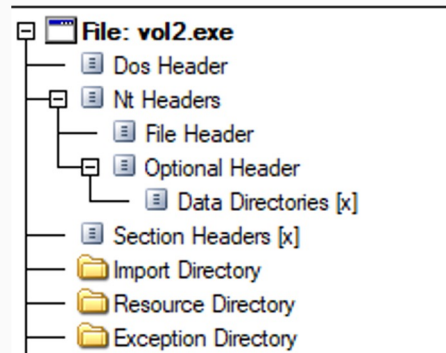
Любой, сложный, файл имеет:

- 1) Заголовок
- 2) Данные

!Заголовок хранит информацию о файле и информацию о том, как с ним взаимодействовать.

Исполняемый файл имеет в заголовке:

- 1) Информацию о сборке файла
- 2) Данные о внешних зависимостях
- 3) Данные о размерах
- 4) Информацию о вложенных структурах
- 5) Данные необходимые для корректного запуска файла



Введение в ELF-файлы

Соглашение о вызове

stdcall или winapi



Аргументы - передаются **через стек, справа налево**.
Очистку стека производит вызываемая подпрограмма.

Применяется в ОС Windows для вызова функций WinAPI.

fastcall



Аргументы - передаются через регистры, первые два параметра **слева направо** в регистры `ecx` и `edx`, а остальные - передаются **справа налево** в стек.
Очистку стека производит **вызываемая** подпрограмма.

cdecl



Аргументы - передаются через стек, справа налево.
Аргументы, меньше 4 байт, расширяются до 4 байт.
За сохранение регистров **EAX, ECX, EDX** и стека отвечает вызывающая программа, за остальные — вызываемая функция.
Очистку стека производит **вызывающая** программа.
Это основной способ вызова функций с переменным числом аргументов (например, `printf()`, `scanf()`)

Практика анализа общей информации об исполняемых файлах.

- `objdump -h /bin/ps`
- `readelf -S /bin/ps`

Задача:

при помощи инструментов – максимально изучить что делает файл, и общую информацию о нем, чем больше тем лучше.

Интересуют:

- 1) используемые библиотеки,
- 2) функции,
- 3) найденные строки,
- 4) как когда был собран файл,
- 5) для какой версии ос.

*Логика выполнения кода, а именно - мини описание что делает это приложение.

Анализировать лучше всего статически в **IDA. CFFexplorer.**

!Запуск производить только на VM!

Результатом выполнения дз должен быть мини-отчет в произвольном формате в файле .docx



Спасибо за внимание!