Reverse engeneering

Teach
Me
Skills

Собираемся и отмечаемся

Teach Me Skills

Вопросы по предыдущим темам или ДЗ

Teach Me Skills

Mini-quize 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?**

Teach Me Skills

Mini-quize по новой теме:

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

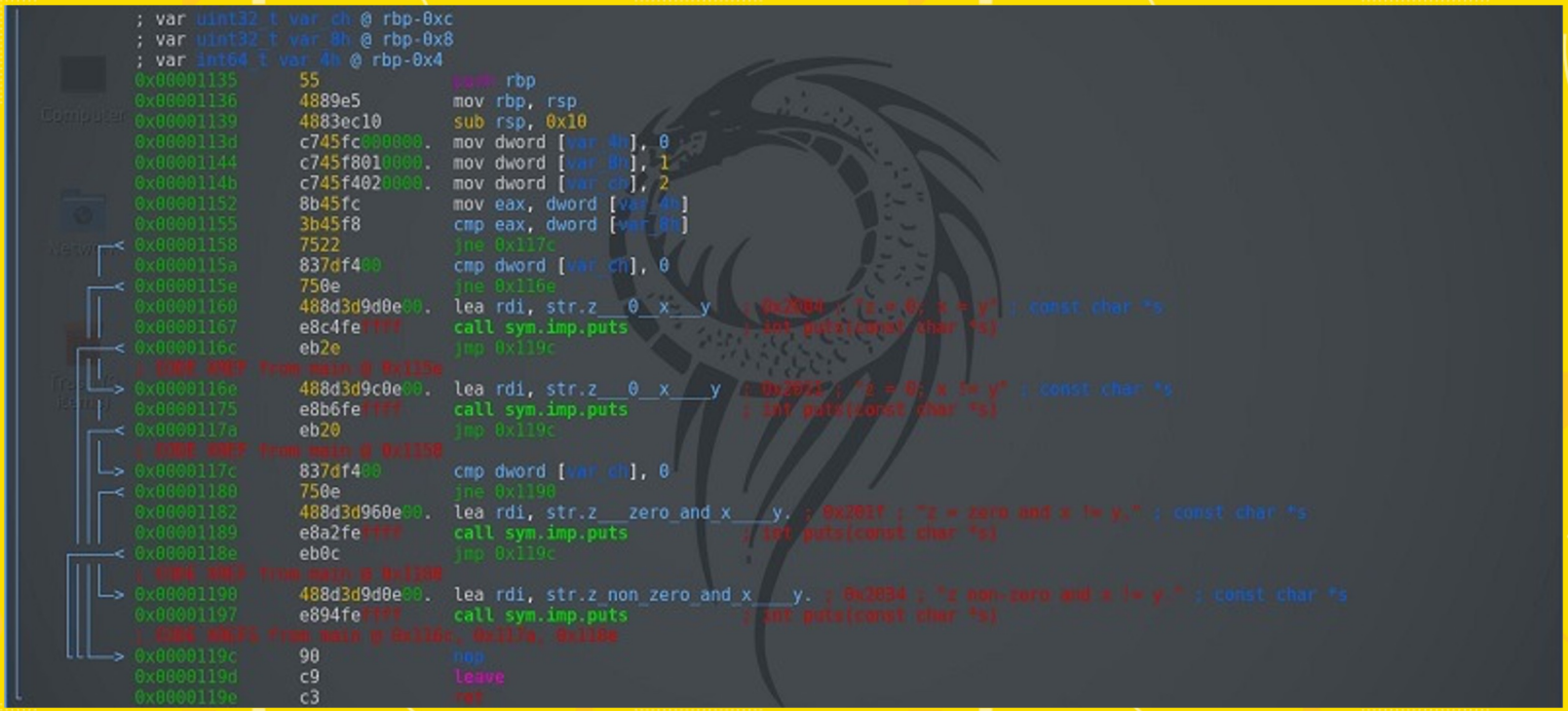
Teach Me Skills

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

Teach
Me
Skills

Что такое реверс-инжиниринг?



Компиляторы и Интерпретаторы

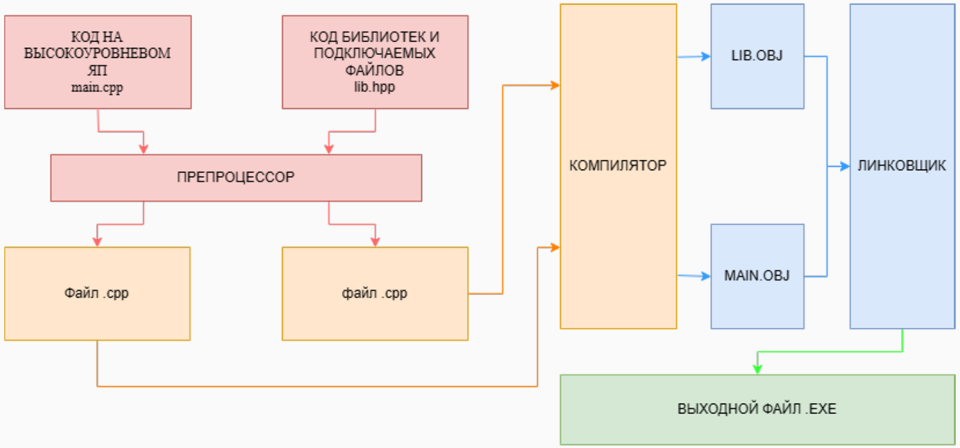
Компилируемые ЯП
Swift
C++/C
Java
Objective-c
Rust
Go

Интерпретируемые ЯП
Python
JavaScript
PHP
Perl
Ruby
C#

[Компилятор VS интерпретатор]

Процесс сборки приложения компилятором

Общая схема формирования исполняемого файла



Препроцессор — 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

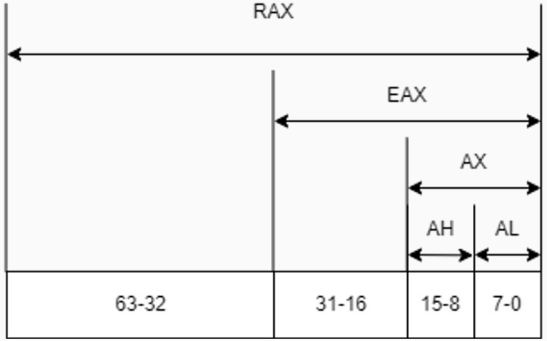
```
int main(){
    std::cout << 10 << std::endl;

    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

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.

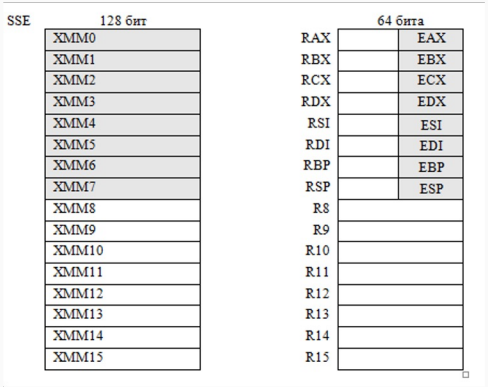
Регистры

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

Управление потоком

Указатель комманд, счетчик комманд, стек вызова и переходы.

Регистры x86



SSE	128 бит
	XMM0
	XMM1
	XMM2
	XMM3
	XMM4
	XMM5
	XMM6
	XMM7
	XMM8
	XMM9
	XMM10
	XMM11
	XMM12
	XMM13
	XMM14
	XMM15

64 бита	
RAX	EAX
RBX	EBX
RCX	ECX
RDX	EDX
RSI	ESI
RDI	EDI
RBP	EBP
RSP	ESP
R8	
R9	
R10	
R11	
R12	
R13	
R14	
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_fig	R8	R8	R8	R8
R9	R9_fig	R9	R9	R9	R9
R10	R10_fig	R10	R10	R10	R10
R11	R11_fig	R11	R11	R11	R11
R12	R12_fig	R12	R12	R12	R12
R13	R13_fig	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fig	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)

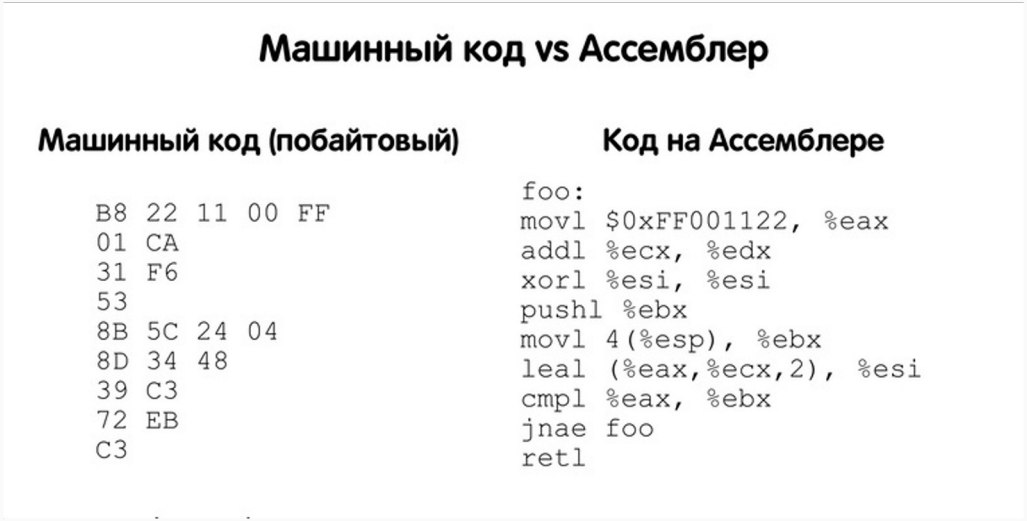
Регистры состояния программы

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fig	SPSR_svc	SPSR_abt	SPSR_irq	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

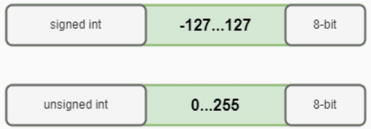
Команды ассемблера состоят из: кодов операций и операндов. **Операнды** — это адреса, из которых процессор будет брать данные для вычислений и в которые будет помещать результат. Адресами могут быть ячейки оперативной памяти и регистры.
Операции (опкоды) в языке ассемблера – мнемонические, для удобного чтения людьми.



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



[Типы данных, которыми оперирует ПК]

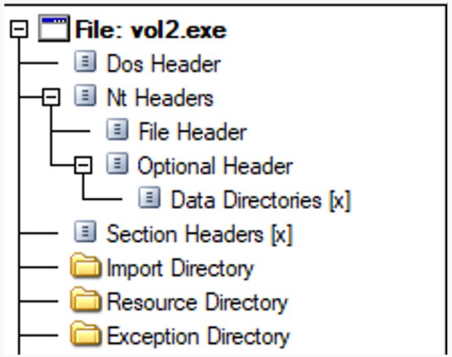
Пример хранения данных в памяти

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	MZђ.▯...▯...яя..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	ё.......@.......
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	F8	00	00	00	............m...
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	▯ e▯.r.H!ё▯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	▯•▯sKф▯.Kф▯.Kф▯.
00000090	5E	8B	7E	21	49	F4	7F	20	5E	8B	7A	21	56	F4	7F	20	^‹~!Iф▯.^‹z!Vф▯.
000000A0	5E	8B	7B	21	40	F4	7F	20	5E	8B	7C	21	48	F4	7F	20	^‹{!@ф▯.^‹|!Hф▯.
000000B0	AF	84	7E	21	4C	F4	7F	20	4B	F4	7E	20	05	F4	7F	20	Ї„~!Lф▯.Kф~.▯ф▯.
000000C0	7D	74	7A	21	4A	F4	7F	20	7D	74	80	20	4A	F4	7F	20	}tz!Jф▯.}tЂ.Jф▯.
000000D0	4B	F4	E8	20	4A	F4	7F	20	7D	74	7D	21	4A	F4	7F	20	Kфи.Jф▯.}t}!Jф▯.
000000E0	52	69	63	68	4B	F4	7F	20	00	00	00	00	00	00	00	00	RichKф▯.........
000000F0	00	00	00	00	00	00	00	00	50	45	00	00	64	86	0A	00	........PE..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	▯ ▯%.†...: .....
00000120	D0	12	01	00	00	10	00	00	00	00	00	40	01	00	00	00	Р▯▯..▯.....@▯...

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 по дпрограмма.

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 стеке.

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

Практика анализа общей информации об исполняемых файлах.

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

ДЗ

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

Teach
Me
Skills

Спасибо за внимание!