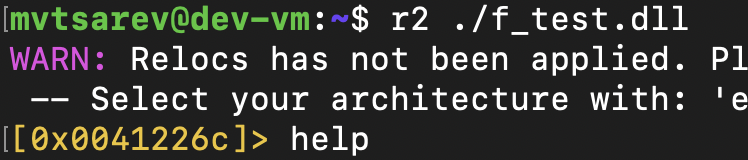
Лабораторная работа

«reverse engineering DLL»

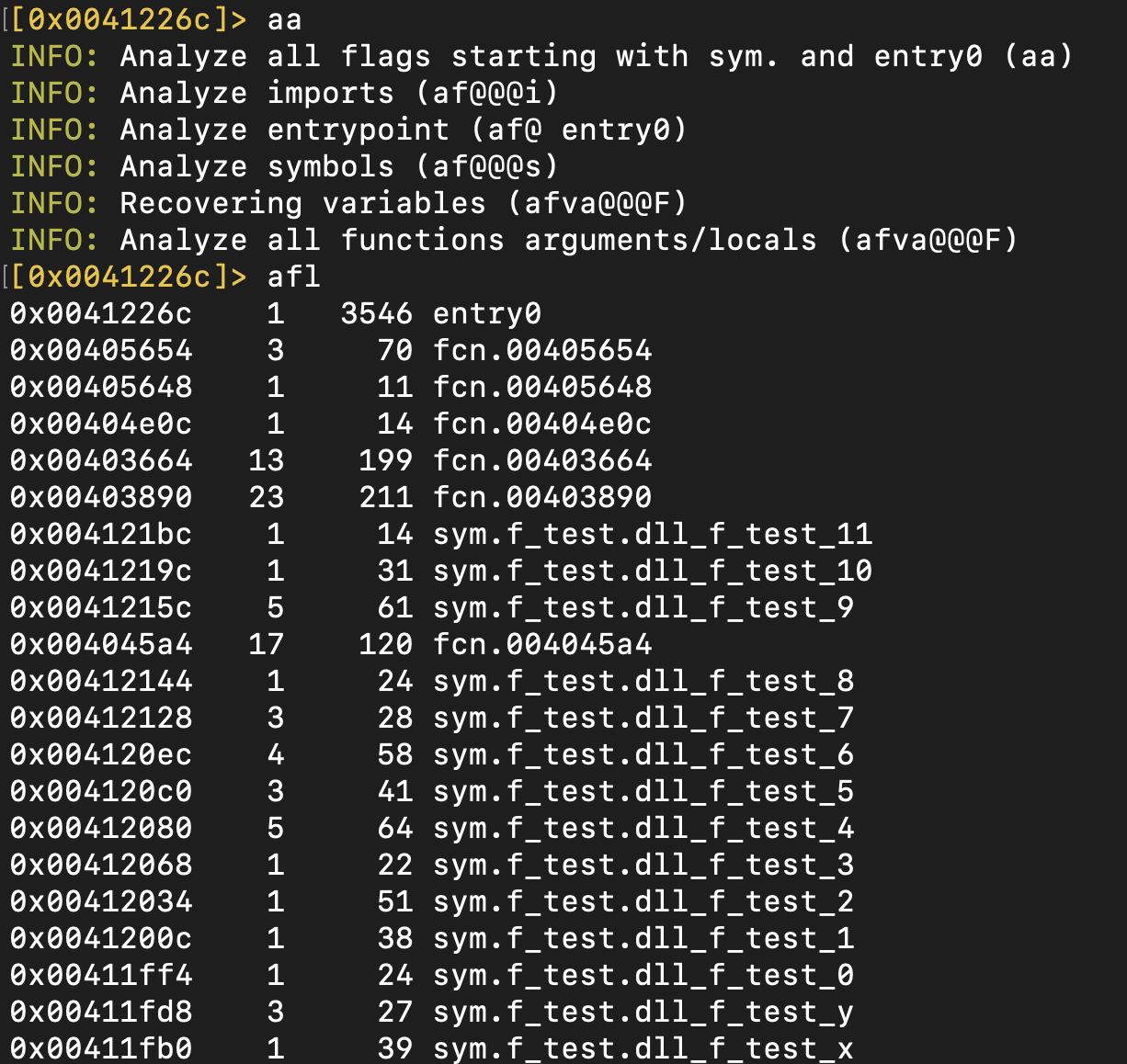
Задание: определить, что выполняет функция f\_test\_2 в заданном файле f\_test.dll

Воспользуемся программой Radare2 для выполнения задания:

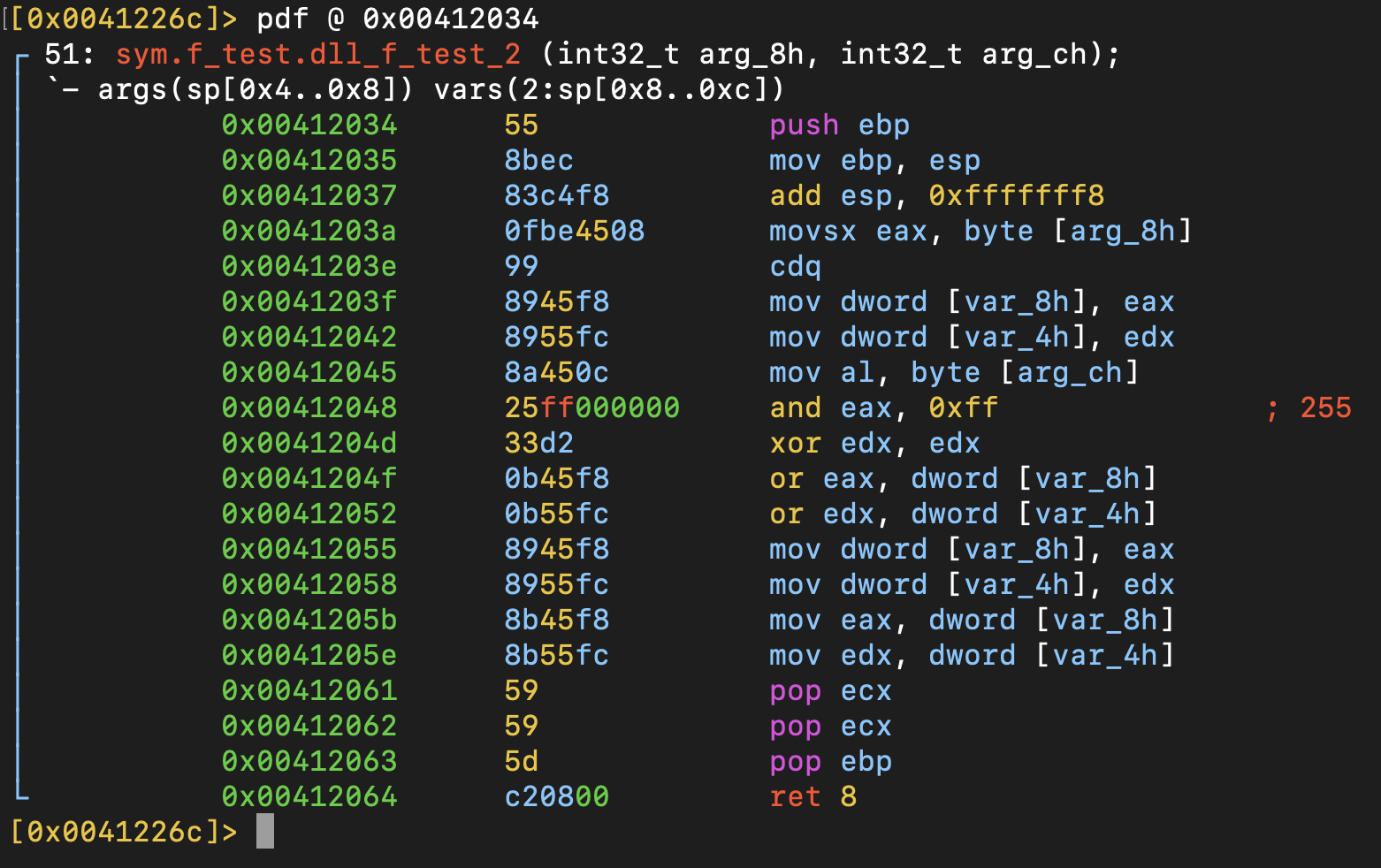
1. Открываем файл



1. Анализируем файл и выводим все найденные имена



1. Теперь смотрим на дизассемблированный код f\_test\_2 (здесь: sym.f\_test.dll\_f\_test\_2



Первое замечание: по использованию регистров esp, ebp заключаем, что переменные передаются через память стека; использование byte[] говорит о том, что входные данные размером в 1 байт.

Add esp, 0xfffffff8 резервирует 8 байт в стеке для двух локальных переменных (здесь: var\_8h, var\_4h)

Команда movsx eax, byte [arg8\_h] помещает данные первого аргумента в eax (arg8\_h означает смещение 8 байт относительно вершины стека, то есть аргумент на расстоянии 2 слов от вершины – первый из двух 32-битных аргументов)

Cdq помещает превращает знаковый 32-битный eax в знаковое 64-битное число в паре регистров edx:eax, копируя старший бит eax во все биты edx.

Следующие две команды mov помещают значения eax и edx в секцию данных.

Mov al, byte [arg\_ch] помещает в al второй аргумент

And eax, 0xff обнуляет старшие 16 бит.

Xor edx, edx обнуляет edx.

Or eax, dword [var8\_h] вычисляет побитовое или между аргументами функции, так как в eax теперь значение второго аргумента, а в [var8\_h] хранится значение первого аргумента

Or edx, dword [var\_4h] помещает в edx, предыдущее значение edx до обнуления, так как edx теперь 0.

Следующие две пары команд mov делают приблизительно ничего )

Двойной pop ecx освобождает в стеке 2 локальные переменные

Ret 8 возвращает значение, содержащееся в eax, освобождая верхние 8 байт стека, которые использовались для передачи аргументов.

Таким образом, мы получаем побитовое или двух аргументов.