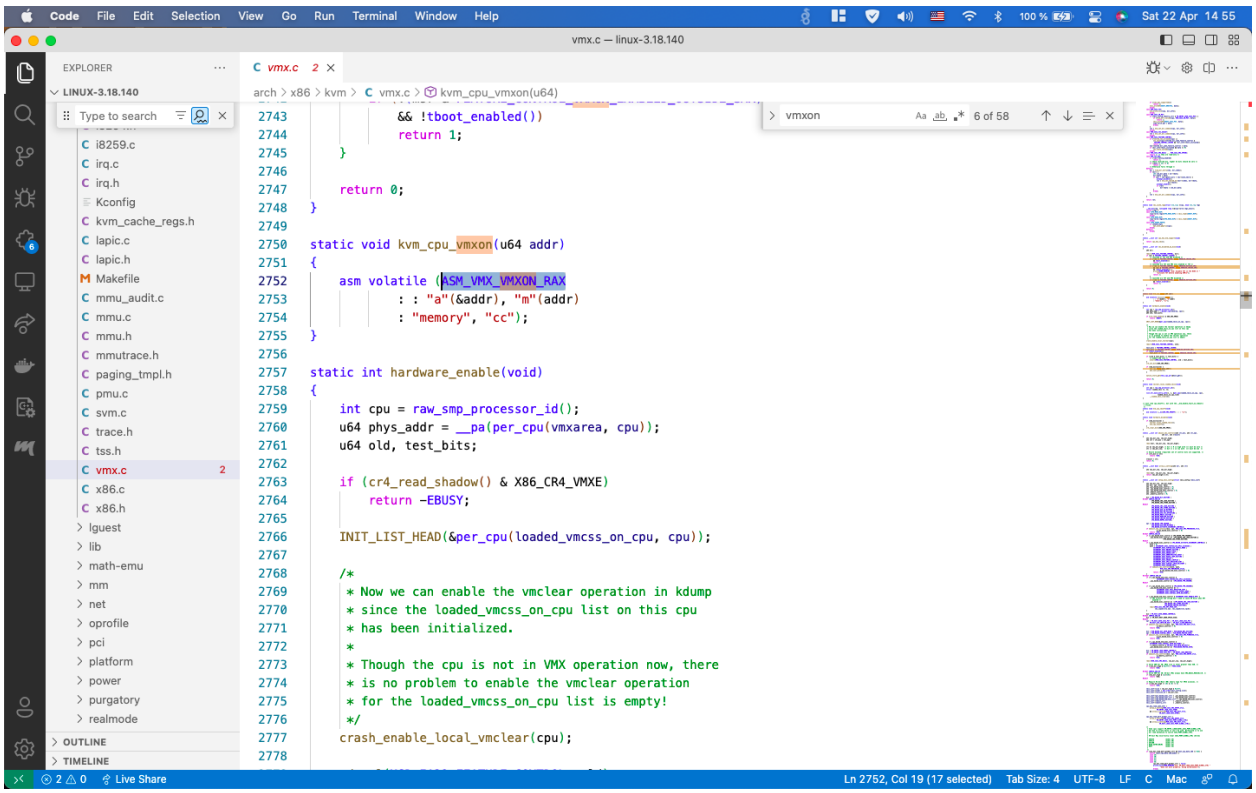
Artem Chernitsa, B20-AI-01, a.chernitsa@innopolis.university

1. Main execution cycle
    a. infinite loop
    b. guest enter
    c. return to VM
2. #PF(page fault) processing

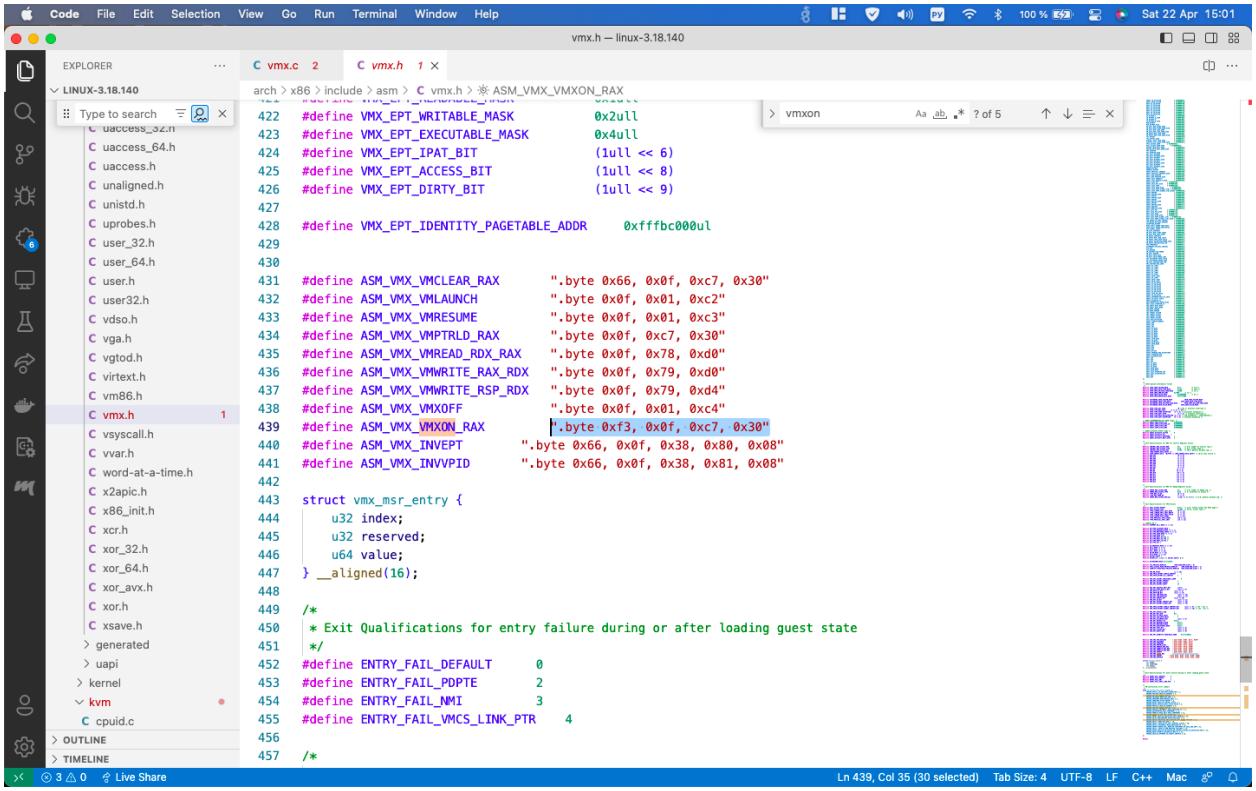
Main execution cycle

Для начала, хочется поискать где вообще есть VMXON, VMXOFF, разные функции, в названии которых фигурирует VMX. Т.к. были даны послабления, например, поиска для конкретной архитектуры, то поиск был начат в папке с логичным названием arch/x86. Там же есть каталог kvm, что напрямую относится к KVM (по собственным предположениям). При дальнейшем рассмотрении в файле vmx.c можно найти следующие строки:



Picture: linux-3.18.140/arch/x86/kvm/vmx.c

При переходе в заголовочный файл vmx.h, можно наблюдать объявление этого имени. Ему соответствует ассемблерный код (инструкция), который выглядит так:



Picture: linux-3.18.140/arch/x86/include/asm/vmx.h

Он помещает адрес 0xc0f300f в регистр esi, а затем делает jo на адрес 0x37 (https://defuse.ca/online-x86-assembler.htm#disassembly2):

```
0:   be 0f 30 0f 0c          mov    esi,0xc0f300f
5:   70 30                   jo     0x37
```

**Remark. jo => jumps to the specified location if the previous instruction sets the overflow flag (OF).*

Здесь же, чуть выше, можно наблюдать объявление имени ASM_VMX_VMLAUNCH, что может означать как раз точку входа. Разберём соответствующий ассемблерный код:

```
.text:
vmlaunch
```

Так как у меня не получилось дизассемблировать код в предыдущем редакторе, я воспользовался другим (https://godbolt.org/). Соответственно для ASM_VMX_VMXOFF код будет такой:

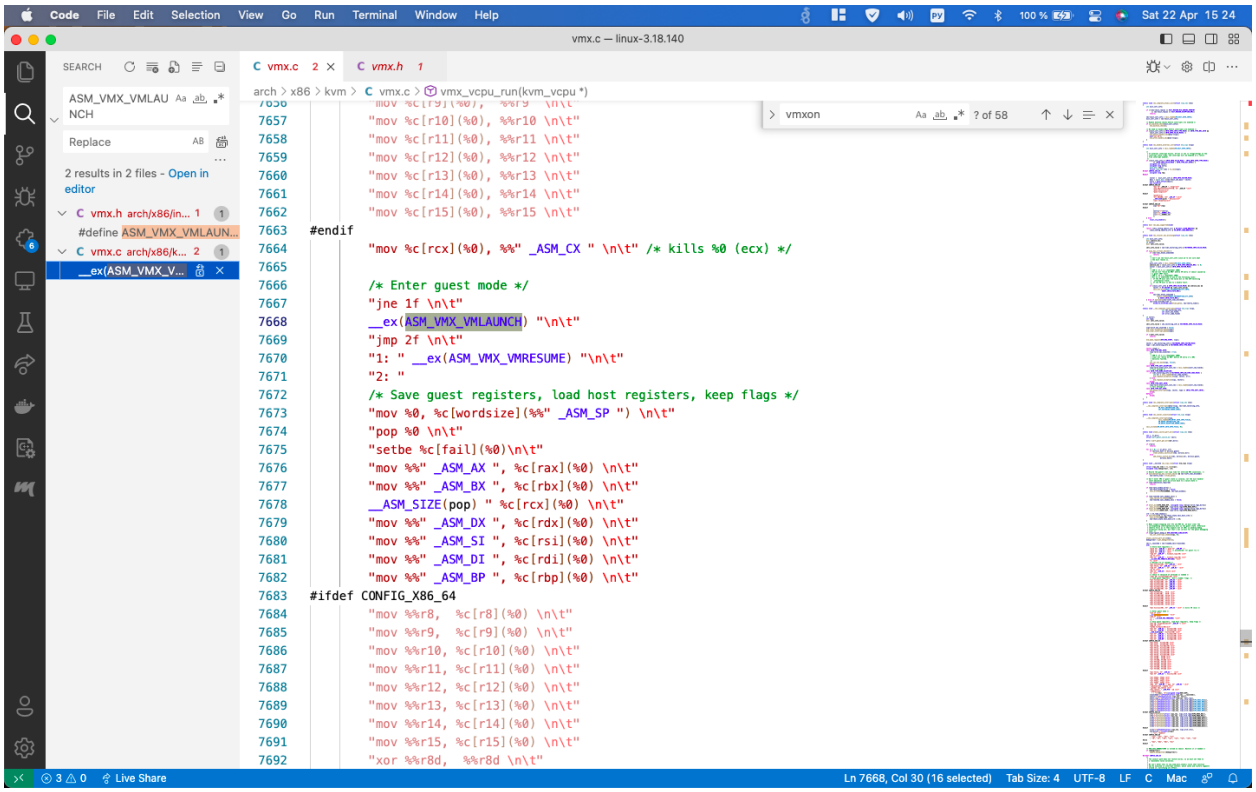
```
.text:
vmxoff
```

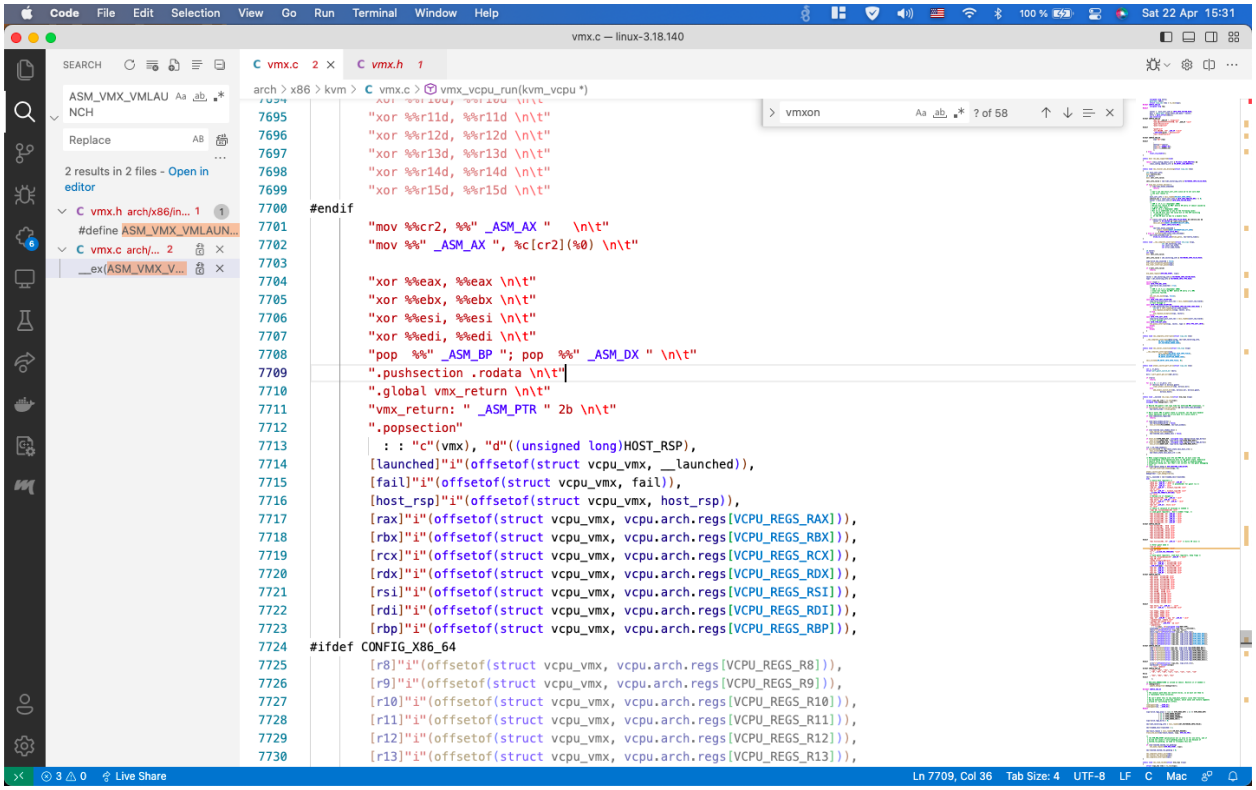
Схема там получается примерно такая:

1. Программное обеспечение переходит в режим работы VMX, выполняя команду VMXON.
2. VMM выполняет вход в виртуальную машину с помощью инструкций VMLAUNCH и VMRESUME;
3. Виртуальная машина завершает передачу управления в точку входа, указанную VMM.
4. В конце концов, VMM может решить завершить работу и прекратить работу VMX. Он делает это, выполняя инструкцию VMXOFF.

Ref: https://docs.hyperdbg.org/tips-and-tricks/considerations/basic-concepts-in-intel-vt-x

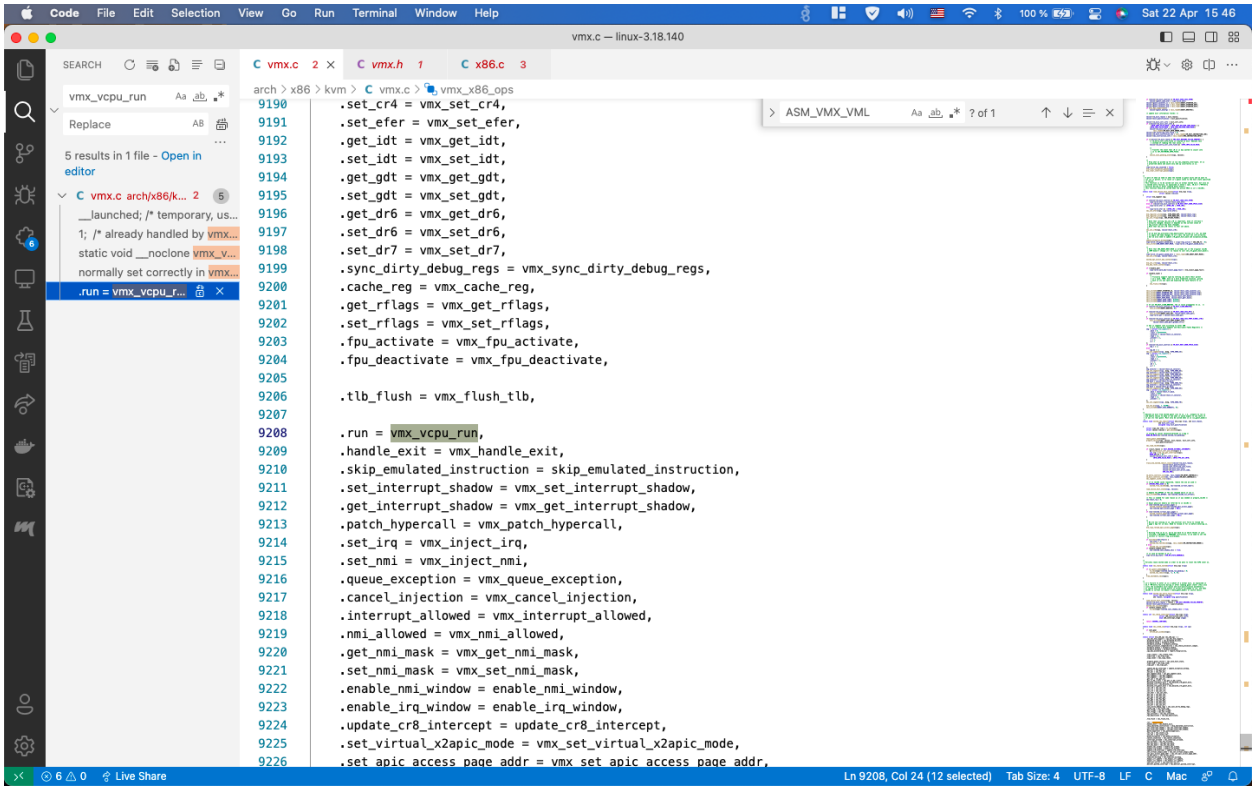
В итоге, надо поискать где вызываются эти инструкции, вызов может быть найден в одном файле, который мы видели ранее: в файле vmx.c. Здесь же видна ещё одна ассемблерная вставка (в функции vmx_vcpu_run):





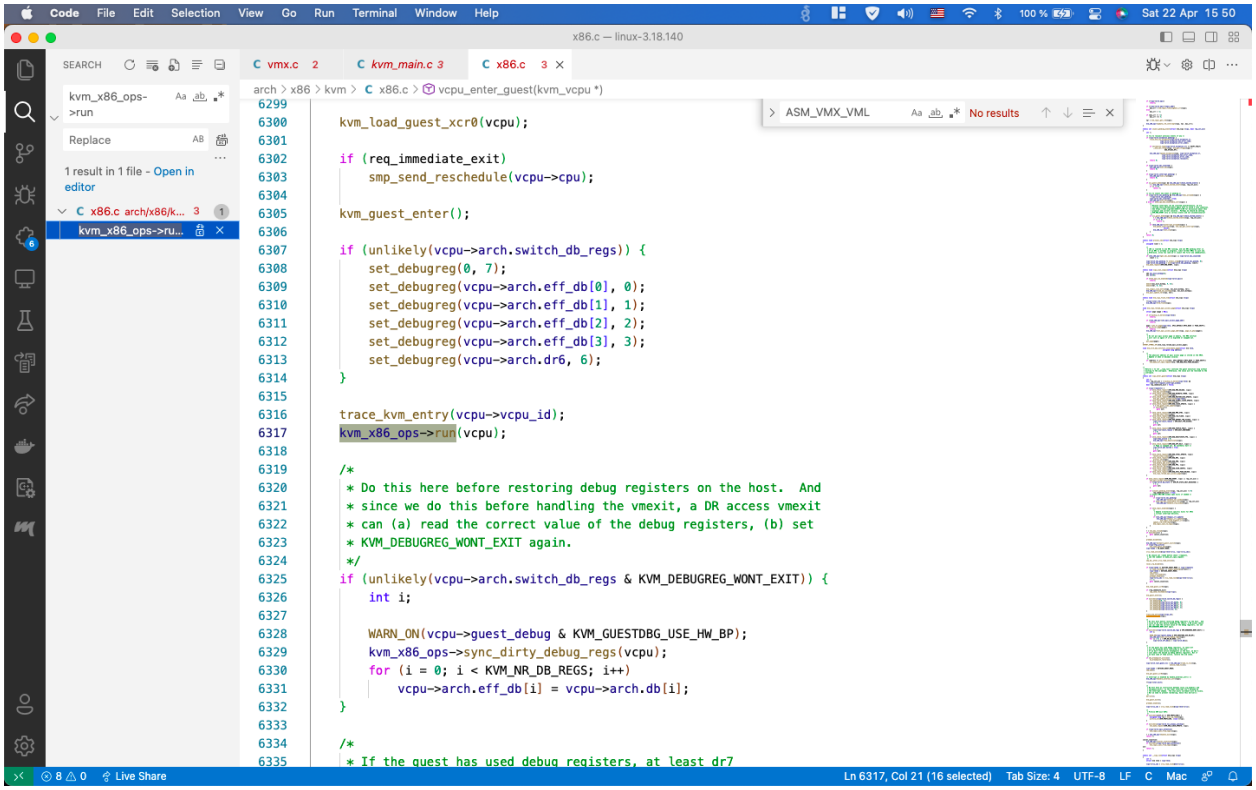
Picture: linux-3.18.140/arch/x86/kvm/vmx.c

Отсюда мы получаем информацию о структуре vcpu_vmx, в частности о структуре kvm_vcpu. А ещё находим место использования vmx_vcpu_run в объявлении структуры kvm_x86_ops vmx_x86_ops:



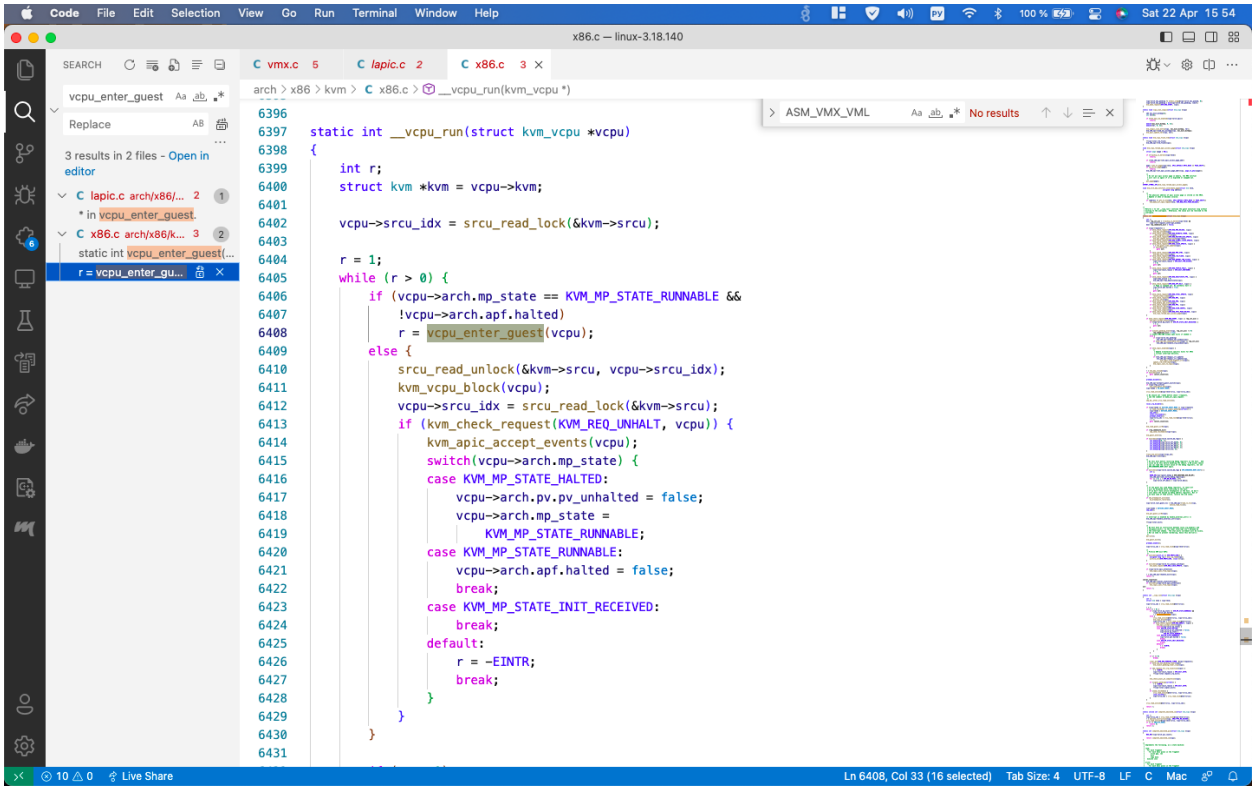
Picture: linux-3.18.140/arch/x86/kvm/vmx.c

Значит где-то будет вызываться vmx_x86_ops->run или kvm_x86_ops->run, поищем это место:



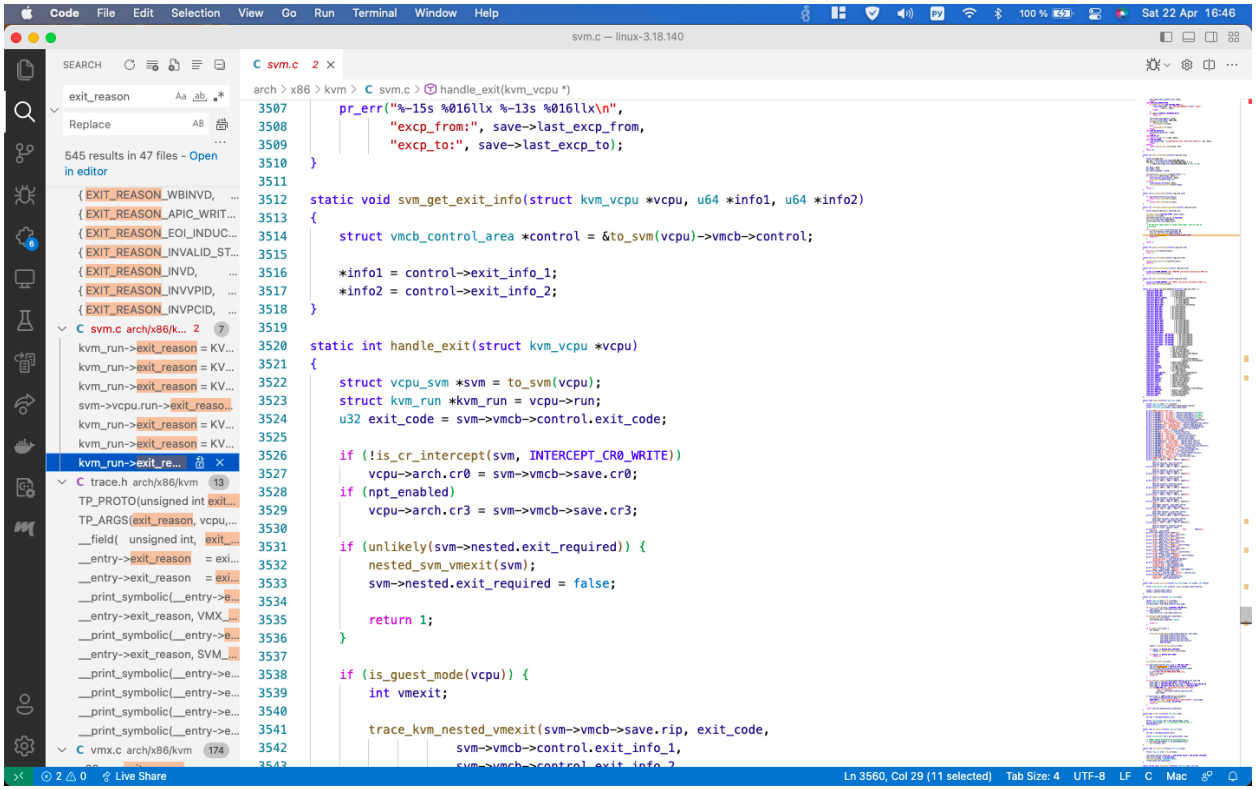
Picture: linux-3.18.140/arch/x86/kvm/x86.c

Вызов происходит в функции vcpu_enter_guest. Вызов vcpu_enter_guest происходит в функции __vcpu_run того же файла.



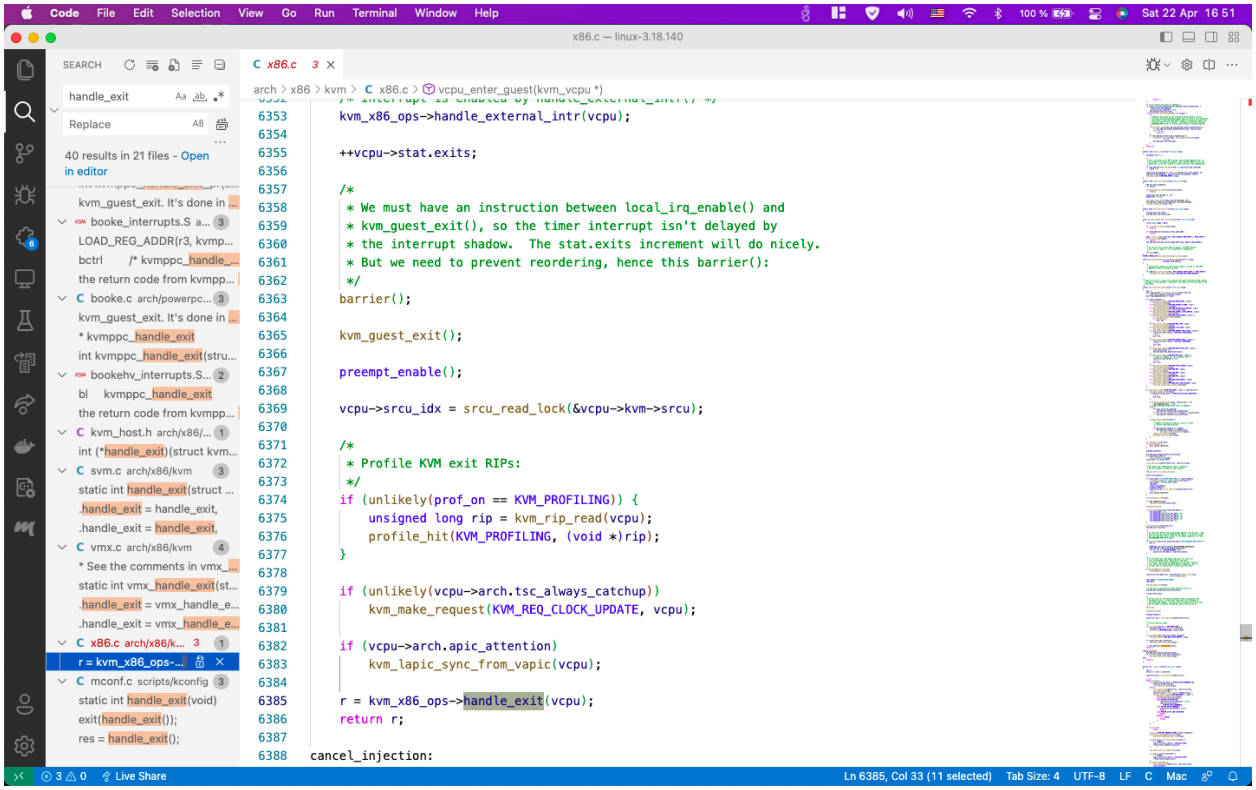
Picture: linux-3.18.140/arch/x86/kvm/x86.c

Вероятно, это и есть main loop. А работу по передаче управления и “входа” в гостя осуществляет vcpu_enter_guest. По поводу выхода постараемся воспользоваться тем, что было и начать, например, с VM_EXIT_REASON. По поиску exit_reason можно найти следующие строки:

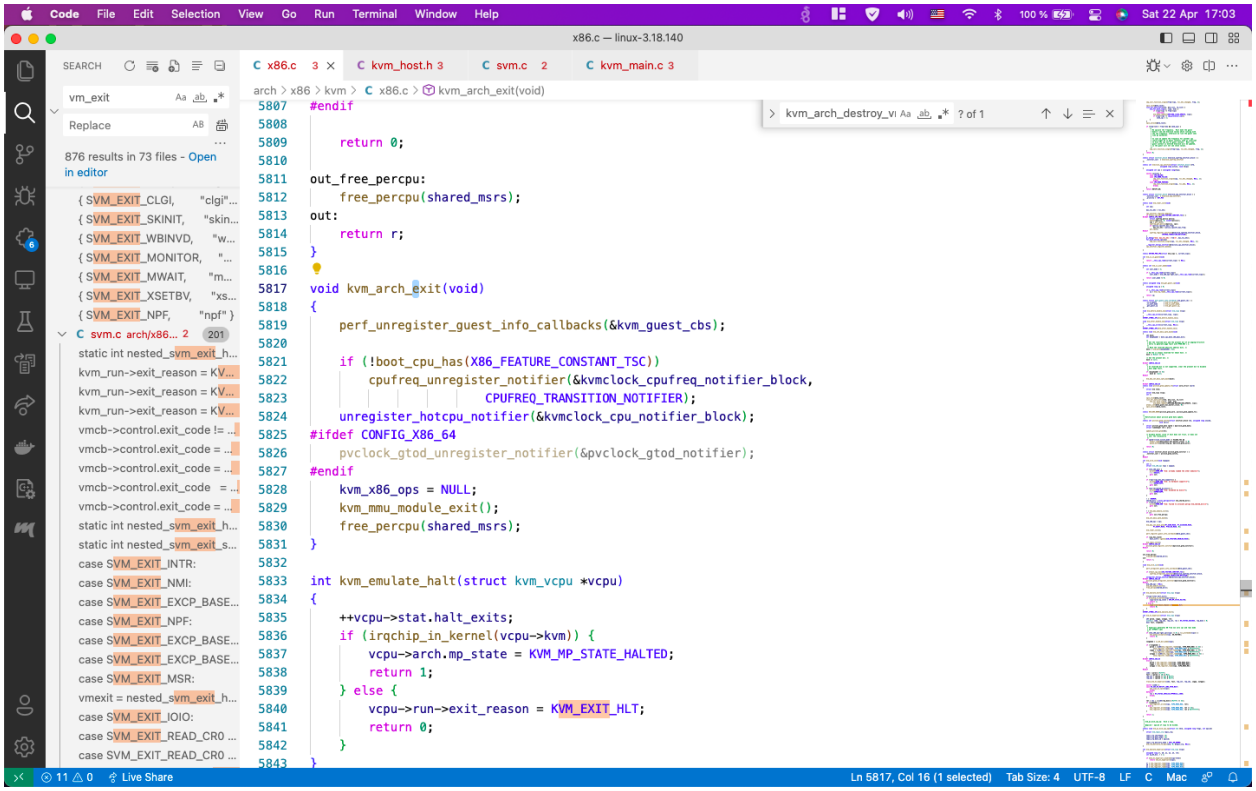


Picture: linux-3.18.140/arch/x86/kvm/svm.c

Функция называется handle_exit, которая обрабатывает прерывание и ставит причину выхода, возвращает необходимый обработчик. Через неё можно попасть в vmx_handle_exit файла vmx.c, который мы видели ранее, а handle_exit и kvm_guest_exit() можно увидеть снова в vcpu_enter_guest:



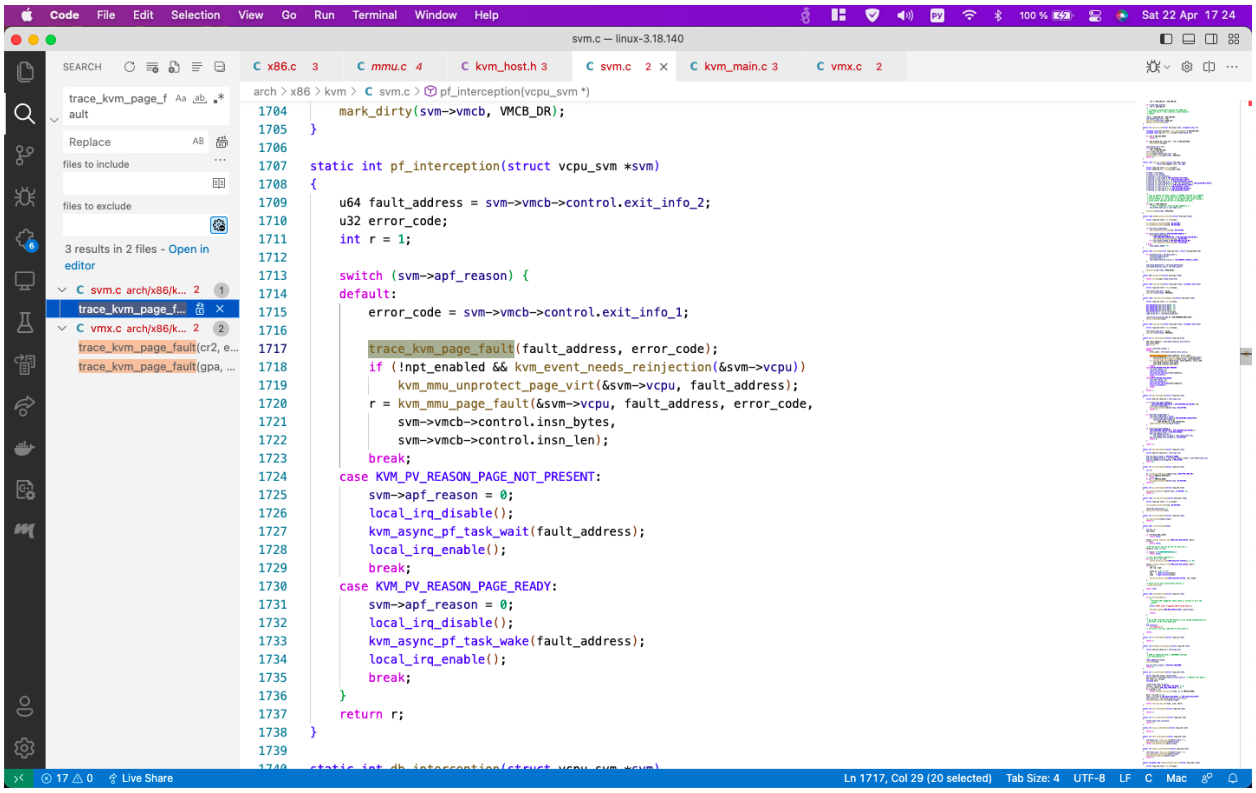
Picture: linux-3.18.140/arch/x86/kvm/x86.c



Picture: linux-3.18.140/arch/x86/kvm/x86.c

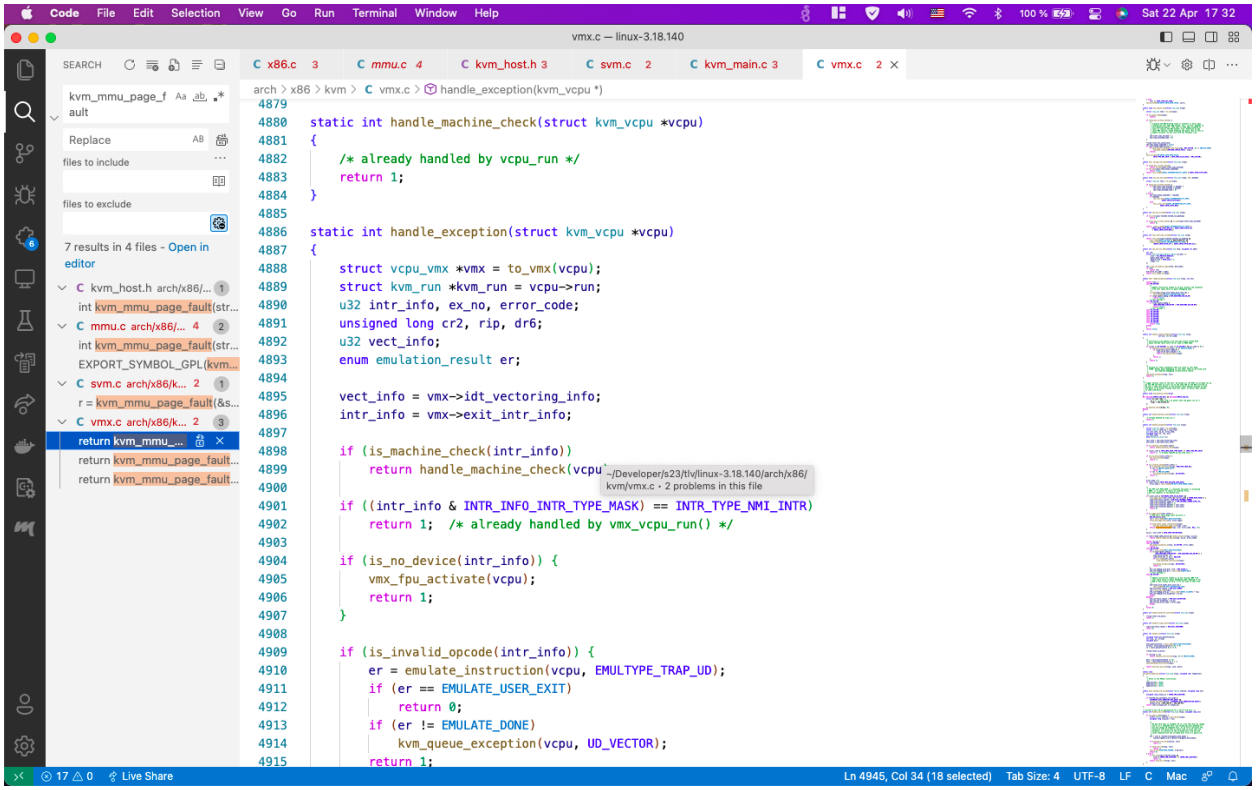
Предположу, что так как и запускали handle_exit будет вызываться от kvm_x86_ops, и видеть мы его можем снова в vcpu_enter_guest.

Page Fault Processing

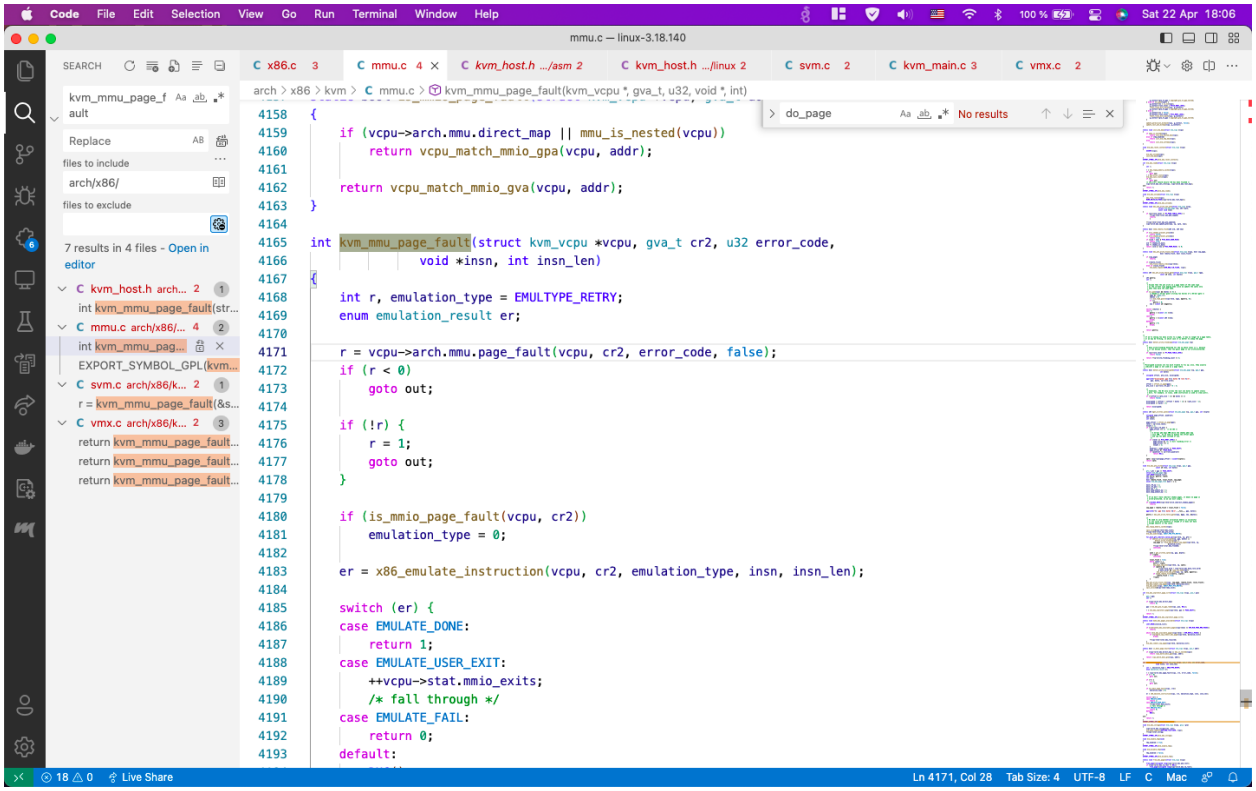


Picture: linux-3.18.140/arch/x86/kvm/svm.c

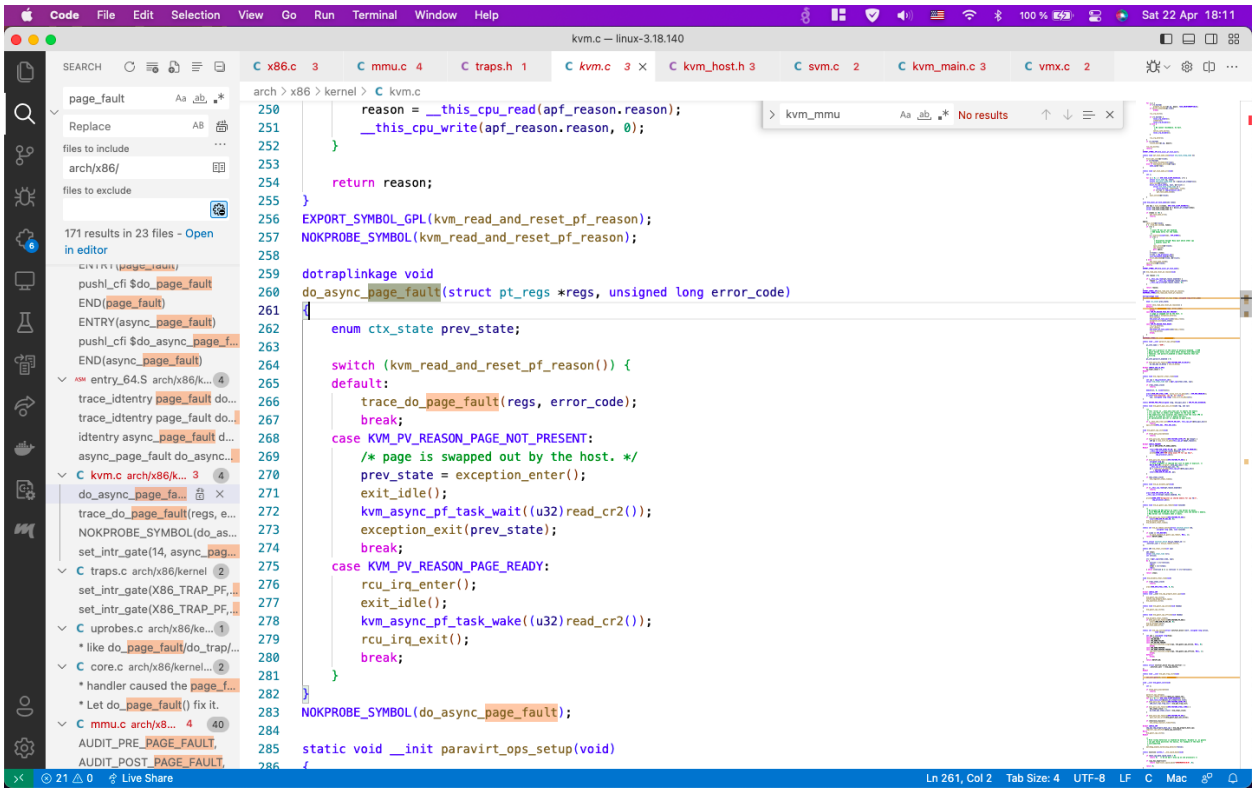
Ещё есть kvm_mmu_page_fault в файле kvm/mmu.c. В vmx.c вызов этой функции находится в функции handle_exception:



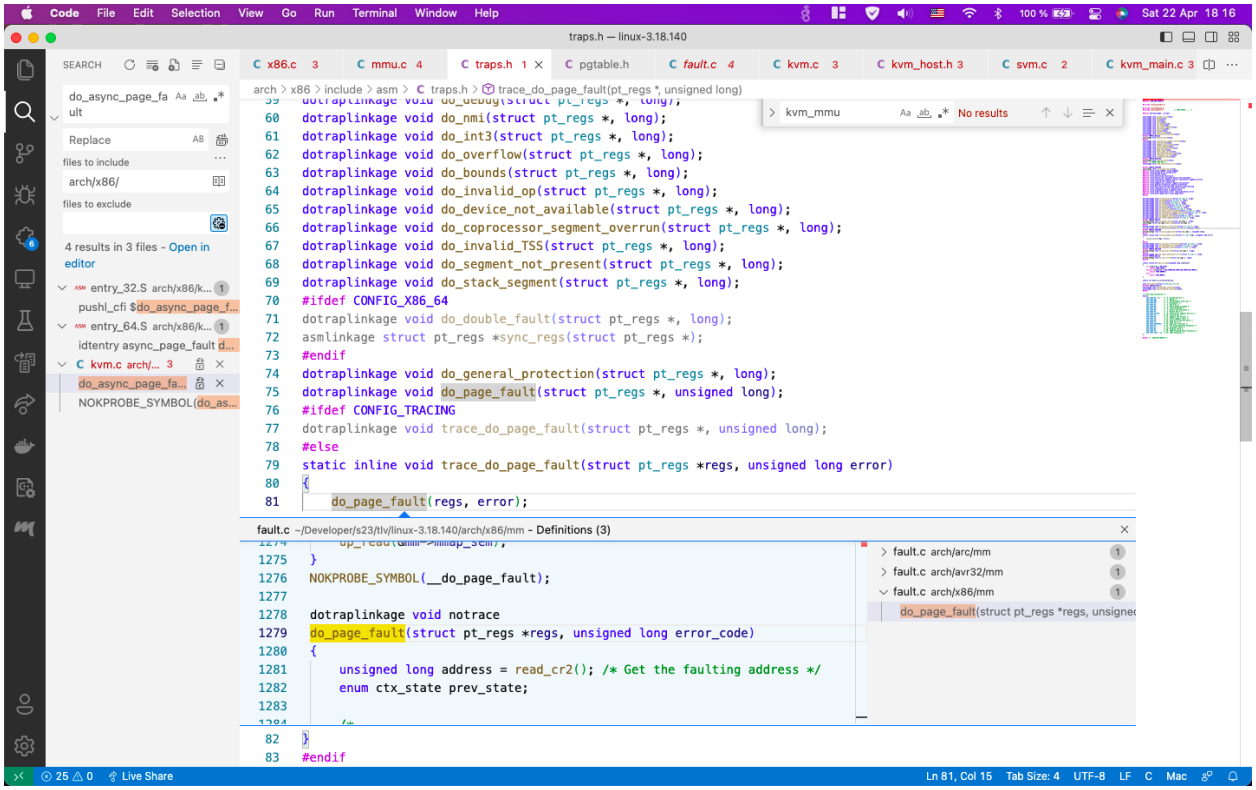
Picture: linux-3.18.140/arch/x86/kvm/vmx.c



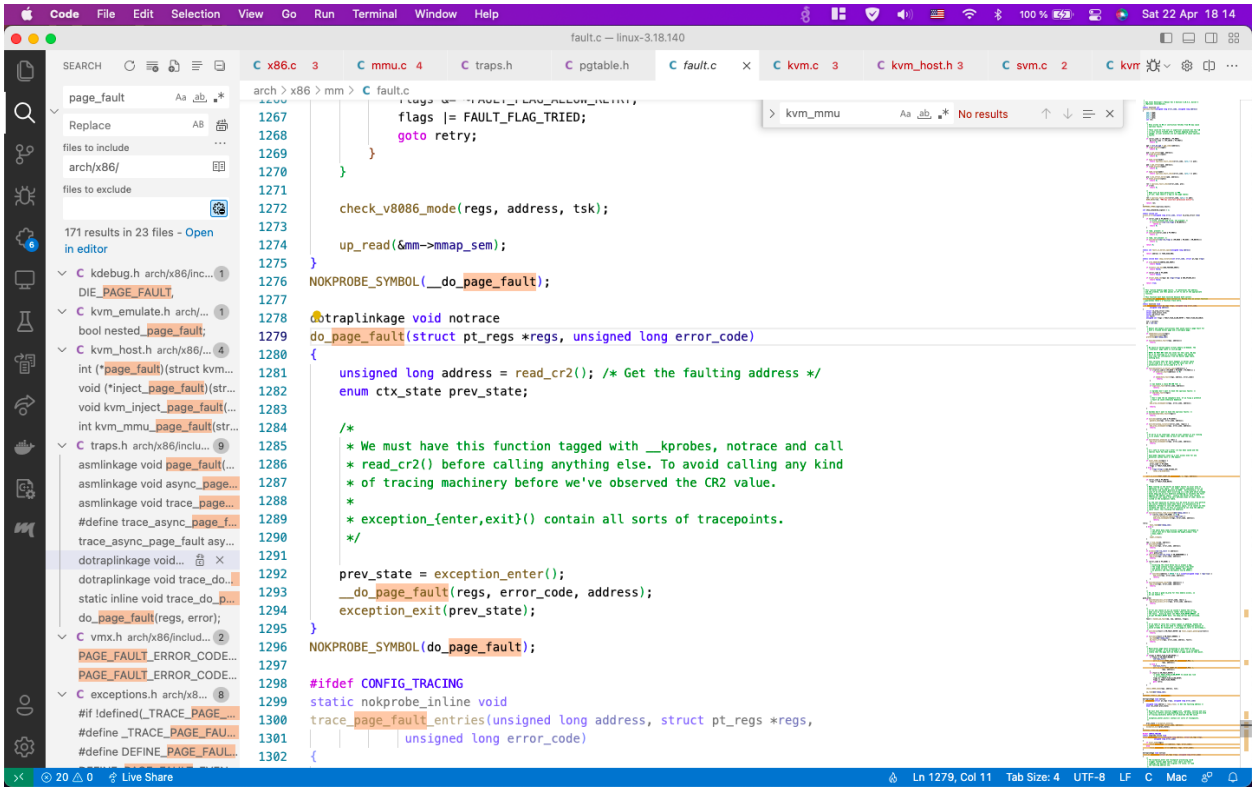
Думаю, здесь и должна происходить обработка. А он ссылается на обработчик для этой платформы, который можно искать в trace, mm/fault.c с названием __do_page_fault.



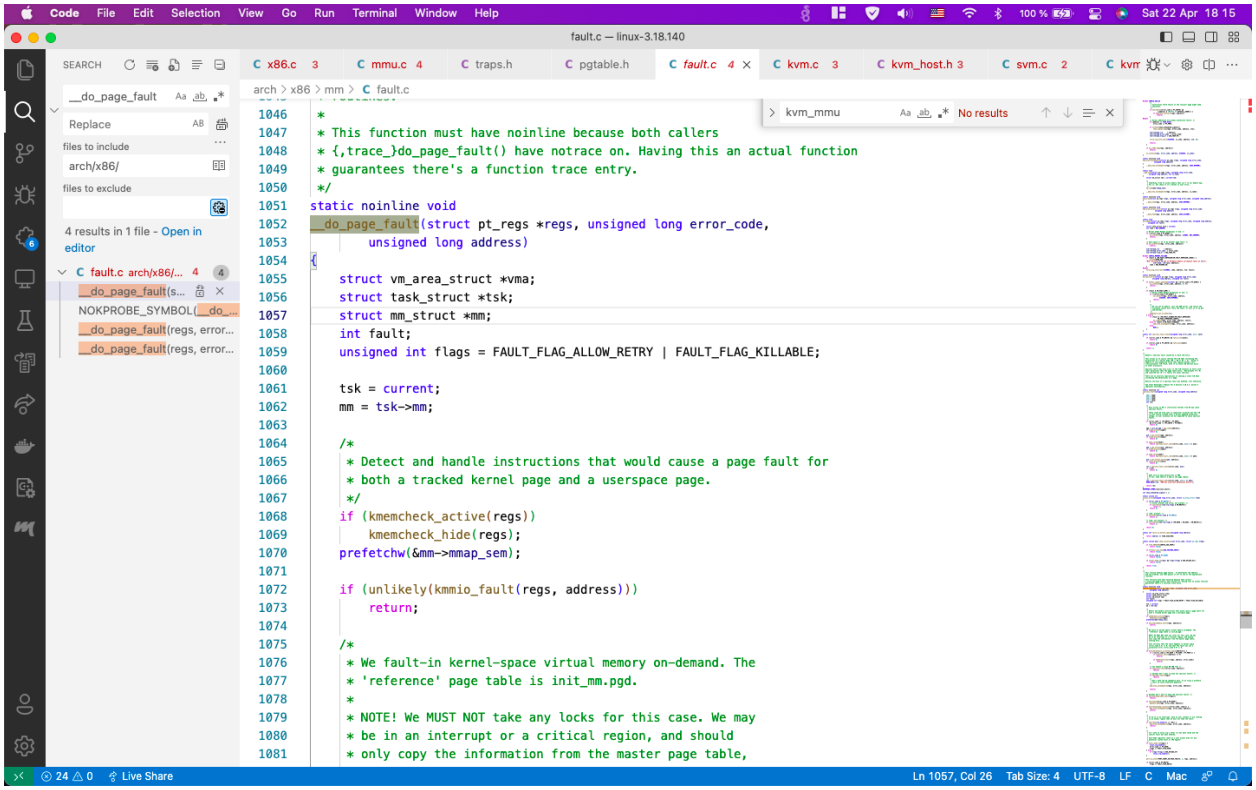
Picture: linux-3.18.140/arch/x86/kernel/kvm.c



Picture: linux-3.18.140/arch/x86/include/asm/traps.h



Picture: linux-3.18.140/arch/x86/mm/fault.c



Picture: linux-3.18.140/arch/x86/mm/fault.c