Wszystkie potrzebne pliki są w repo

Zadane 1

Na początek zestaw sobie środowisko do testowania plików efi pod qemu. Pod linuxem zwykle wystarczy zainstalować pakiet `qemu-system-x86_64`. Do testowania aplikacji będziemy wykorzystywać TianoCore, które podamy do qemu jako bios. Tu drogi są 2 dla odważnych i mających stalowe nerwy można skompilować projekt edk2 albo poprostu wziąć ten plik z repo (OVMF.fd).

Teraz korzystając z tej instrukcji stwórz obraz dysku z plikiem zad1.efi, następnie uruchom pod qemu i załącz screenshota.

Dalej znajdź kawałek pustego pendrive'a stwórz na nim partycje EFI (to jest fajna nazwa na fat32), umieść plik zad1.efi na pendrivie jako `/EFI/BOOT/BOOTX64.efi` i spróbuj zabootować z niego swojego laptopa. Uwaga jak nie działa, bo ostatnio się przekonałem że np laptopy HP elitebook wolą jak się to nazwie `/EFI/BOOT/mmx64.efi`, to szukać w necie / pisać będziemy debugować. Jak się uda to załącz zdjęcie ekranu.

Zadanie 2 kompilujemy Hello World

Pisząc pod UEFI będziemy korzystać z biblioteki gnu-efi. Pobierz paczkę z linku, rozpakuj i skompiluj bibliotekę (wywołaj `make` w rozpakowanym katalogu).

Po skompilowaniu kopiujemy z katalogu bilioteki:

1. Katalog `inc` z plikami nagłówkowymi
2. Pliki `./x86_64/lib/libefi.a` i `./x86_64/gnuefi/libgnuefi.a` zastępują nam standardową bibliotekę języka C
3. Plik `./x86_64/gnuefi/crt0-efi-x86_64.o` zastąpi nam kod który normalnie wykonuje się przed funkcją "main" i inicjalizuje bibliotekę języka C. U nas zainicjalizuję bibliotekę libefi.
4. Plik `./gnuefi/elf_x86_64_efi.lds` jest skryptem linkera zawierającym opis struktury pliku efi (adresy sekcji, adresy tablicy z symbolami itd)

Pliki, katalogi skopiuj do folderu, gdzie będziesz kompilować aplikacje.

Utwórz plik main.c (na razie C) z Hellow Worldem:

```
#include <efi.h>
#include <efilib.h>

EFI_STATUS
EFIAPI
efi_main (EFI_HANDLE ImageHandle, EFI_SYSTEM_TABLE *SystemTable) {
   InitializeLib(ImageHandle, SystemTable);
   Print(L"Hello, world!\n");

   return EFI_SUCCESS;
}
```

Całość skompiluj korzystając z instrukcji z strony osdev. Jak się skompiluje i uda Ci się odpalić to wypisz na ekran "WWW16 Imię Nazwisko", odpal na laptopie i zrób zdjęcie ekranu.

Zadanie 3

Jako że żyjemy w krainie kompatybilności wstecznej każda karta graficzna powinna obsługiwać tekstowy tryb, jest to prosty tryb pracy, którym mamy do dyspozycji 80x25 znaków i 16 kolorów. Korzystając z funkcji:

```
void putstr(wchar_t * ptr) {
    uefi_call_wrapper((void*)systab->ConOut->OutputString, 2, systab->ConOut,
ptr);
}

void setcolor(Color fg = Color::White, Color bg = Color::Black, std::size_t
blink = 0) {
    std::size_t attr = blink << 7 | ((std::size_t)bg & 0x7) << 4 |
(std::size_t)fg;
    uefi_call_wrapper((void*)systab->ConOut->SetAttribute, 2, systab->ConOut,
attr);
}

void setpos(std::size_t x, std::size_t y) {
    uefi_call_wrapper((void*)systab->ConOut->SetCursorPosition, 3, systab-
>ConOut, x, y);
}
```

Stwórz aplikacje EFI, z prostym efektem graficznym np. spadające kolorowe literki lub coś podobnego. Oprócz tego przydać się może:

```
void sleep(std::size_t us) {
    uefi_call_wrapper((void*)BS->Stall, 1, us);
}
```

Pamiętaj, że nie możesz korzystać z funkcji z standardowej biblioteki C (a przynajmniej nie z wszystkich). Użycie np. takiego malloca lub free na 100% skończy się wybuchem co w qemu objawi się jako zawieszenie się wirtualnej maszyny, a na prawdziwym sprzęcie najpewniej skończy się to resetem. Dlatego staraj się jak najmniej korzystać standardowych funkcji.

Do generatora pseudolosowego można wykorzystać instrukcje rdtsc, która zwraca liczbę cykli procesora:

```
std::size_t rdtsc() {
    std::size_t high = 0;
    std::size_t low = 0;
    __asm__ ("rdtsc" : "=a"(low), "=d"(high));
    return high << 32 | low;
}
```

W katalogu zad3 znajdziesz szablon projektu z plikiem `CMakeLists.txt` do automatycznej kompilacji, instrukcje w katalogu zad3.

W rozwiązaniu wyślij kod źródłowy, wystarczą stworzone przez ciebie pliki źródłowe.

Kontakt

mail: `michalszak@gmail.com`

Pisać jak coś nie działa to będziemy debugować, proszę pamiętać że jak coś działa pod qemu to raczej na pewno nie działa na żywym sprzęcie.