Západočeská Univerzita

Semestrální práce - KIV/OS

Minimalistický Operační Systém

Martin Červenka - A17N0029P
Petr Štechmüller - A17N0040P
Antonín Vrba - A17N0043P

24. listopadu 2017

Obsah

1. Návrh

Cílem této semestrální práce je navrhnout a realizovat minimalistický operační systém, který by bylo možné spouštět ve virtualizovaném prostředí Qemu[1]. Nepůjde tedy o simulaci operačního systému ve vyšším programovacím jazyce, ale o implementaci všech nutných komponent pro úspěšné zavedení a spuštění systému. Zavaděče operačních systémů jsou téměř výhradně implementovány v jazyce symbolických adres – assembleru a v rámci využívání nízkoúrovňových služeb BIOSu počítače bude celý OS naprogramován rovněž v assembleru. Aby k takovým službám byl umožněn přístup, nebude OS opouštět 16-bitový reálný režim procesoru. V tomto režimu bylo dříve možné provozovat kromě MS-DOS i Windows 3.0, ale modernější OS okamžitě přepínají do chráněného režimu. Dokonce i Linux ve verzi 0.01 v roce 1991 již plně využívá možností procesoru 80386 a přepíná do 32-bitového chráněného režimu.

Tato semestrální práce řeší v podstatě velmi triviální problém, který lze popsat jedinou větou: „Upravení souboru umístěného na disku se stále zobrazenými hodinami“. Tento problém ovšem přestává být triviální v okamžiku, kdy máme k dispozici 640 kB volné paměti a assembler.

1.1 Komponenty a funkce OS

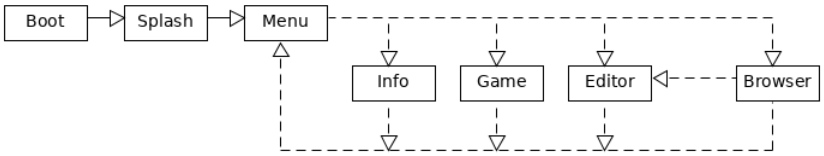
- **Zavaděč** - OS je potřeba zavést do paměti
- **Vykreslování** - bude použit 8-bitový grafický režim 320×200 px a tedy i vlastní implementace grafického rozhraní
- **Vstup z klávesnice** - nutná vlastnost pro interakci s uživatelem
- **Zobrazení hodin** - pro zobrazení aktuálního času bude implementováno nezávislé „vlákno“ pomocí systémového časovače a času uloženého v paměti BIOSu
- **Souborový systém** - organizace dat na disku společně se základními možnostmi zápisu a čtení

[1] Open source emulátor a virtualizér.

- **Prohlížeč souborů** - možnost výběru souboru pro editaci
- **Textový editor** - úprava textu, tak jak ji známe z mnoha editorů a následné uložení
- **Přepínání programů** - bezpečný mechanismus přecházení mezi jednotlivými programy např. z prohlížeče do editoru a předávání parametrů
- **Systémová volání** - skrze vytvořené obsluhy přerušení lze odkudkoli volat služby OS jako jsou vykreslovací funkce nebo čtení z disku

1.2 Navigace v OS

Na Obrázku 1.1 je znázorněno, jak je navrženo předávání řízení mezi programovým vybavením OS. Pro navigaci byl nejdříve uvažováno prostředí typu shell[2], ale bylo by zbytečné implementovat parser[3] pro zadání několika málo příkazů. Bylo tak implementováno grafické rozhraní, které umožňuje přecházet mezi jednotlivými programy.



Obrázek 1.1: Diagram navigace v OS.

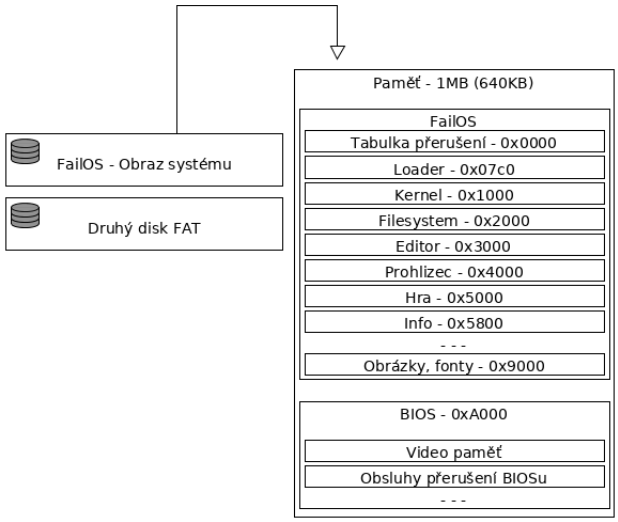
[2]Textové uživatelské rozhraní přes příkazový řádek.
[3]Syntaktická analýza textu

2. Realizace

V následujících sekcích budou popsány jednotlivé části OS:

2.1 Zavaděč

Soubor *loader.asm* je ukončený deklarací *times 510-($-$ $) db 0* a *dw 0xaa55* a splňuje tedy požadavky pro bootovatelný sektor s velikostí 512B a správným ukončením. Binární podoba tohoto souboru je po startu BIOSem rozpoznána a načtena do paměti od fyzické adresy *0x07c00*. Zavaděč nastaví datový a extra segment na hodnotu aktuálního kódového segmentu a poté začne načítat jednotlivé komponenty OS do paměti viz Obrázek 2.1.



Obrázek 2.1: Rozvržení paměti.

Vzhledem k dostatku paměti bylo každému programu vyhrazeno 64 kB paměti, tedy např. adresní prostor *0x1000:0x0000* až *0x2000:0x0000*. toto rozložení lze však dále dělit. Definice statického rozdělení paměti se nachází v souboru *consts.asm* a je nutné jej upravovat po změně rozsahu kódu jednotlivých programů. Čtení z obrazu OS zajišťuje obsluha přerušení BIOSu *int 0x13*. V následující ukázce kódu je čtena část obrazu OS na adresu *0x1000:0x0000* v operační paměti. Po načtení dojde ke skoku na jádro OS.

```
nacti_segmenty:
  mov cl,ah
  mov ah,0x02
  mov es,bx
  xor bx,bx
  int 0x13
  ret
     ...
mov ax,jadro ;start_jadro*256+velikost_jadro
mov bx,segment_jadro ;0x1000
call nacti_segmenty
```

2.2 Jádro OS

Minimalistické jádro zavěšuje obslužné rutiny pro volaná přerušení viz následující ukázka kódu, kde je nastaveno přerušení *0x23* pro spuštění programů.

```
mov word [es:0x008c],spustit_program ;nastaveni offsetu obsluhy
mov word [es:0x008e],segment_jadro ;nastaveni segmentu obsluhy
```

Výčet všech využívaných přerušení je k nalezení v Tabulce 2.1.

Tabulka 2.1: Seznam využívaných přerušení

název	**č. přerušení**	**typ přerušení**
spuštění programu	*0x23*	softwarové
vykreslovací funkce	*0x22*	softwarové
souborový systém	*0x21*	softwarové
ukončení programu	*0x05*	hardwarové
sys. časovač	*0x08*	hardwarové

V případě přerušení *0x21* a *0x22* je obslužná rutina vybrána v závislosti na parametru pomocí přerušovacího handleru implementovaného na začátku *filesystem.asm* a *images.asm*. V assembleru je tento přístup známý jako jumptables - skok na adresu rutiny podle hodnoty příslušného parametru. Jako klávesa, která ukončuje program byla zvolena *PrtSc*, která v reálném režimu generuje přerušení *int 0x05*, lze tak snadno převzít obsluhu a klávesu využít. Další klávesou, která

generuje specifické přerušení je *break*, avšak tu se nepodařilo dostat do Qemu prostředí, jelikož ji GNU/Linux jako hostovací OS zachytává dříve.

Dalším úkolem jádra operačního systému je vykreslování menu OS a inicializace hodin, které následně využívají přerušení *0x08* pro „tikání“. BIOS nastavuje po startu děličky taktu pro PIT[1], aby bylo HW přerušení IRQ0 vyvoláno každých 54.9255 ms a skrze master PIC[2] se dostalo přímo do CPU.

2.3 Souborový systém

Aby bylo možné pracovat s daty na druhém disku, bylo nutné implementovat minimální funkčnost pro čtení, zápis a formátování. Tyto funkce se nacházejí v souboru *filesystem.asm* jako obsluhy přerušení *0x21*. Souborový systém obsahuje 16 staticky vytvořených souborů o maximální velikosti 512B, tedy pro každý soubor je přiřazen jeden diskový cluster. Takový přístup výrazně zjednodušuje manipulaci s daty a je pro potřeby minimalistického OS dostatečný. V Tabulce 2.2 jsou poznamenány parametry jednotlivých obsluh. Do registru *AH* se před vyvoláním přerušení umisťuje index služby. Pro pohodlnou manipulaci s daty je ve filesystemu vyhrazena statická paměť o velikosti 512B. Tato paměť slouží jako odkládací místo, do kterého se načítá obsah root directory. Dále je souborový systém vyzbrojen funkcí *strlen*, která vypočítá velikost souboru, kterou vizualizuje browser.

Tabulka 2.2: Obsluhy přerušení *0x21* souborového systému

služba	**registr AH**	**parametry**
formátování	*0x37*	bez parametru
čtení	*0x3f*	CX - ID souboru DS:BX = adresa paměti
zápis	*0x40*	CX - ID souboru DS:BX = adresa paměti

Pro zavedení disku do virtuálního prostředí Qemu je nutné vytvořit prázdná soubor o minimální velikosti 10 kB, lze využít na Linuxu například nástroj *fallocate*. V následující ukázce dvou příkazů je disk vytvořen a následně je spuštěn virtuální stroj se zavedeným obrazem OS a datovým diskem:

```
> fallocate drive.bin -l 10 kB
> qemu-system-x86_64 -drive file=main.bin,format=raw \
              -drive file=drive.bin,cache=none,format=raw
```

Po startu OS s čistým diskem je nutné jej naformátovat, aby bylo možné dále pracovat viz kapitola 3.

[1] Programmable Interval Timer
[2] Programmable Interrupt Controller

2.4 Vykreslování

Pomocí přerušení *int 0x10* je v reálném režimu možné změnit zobrazovací režim z textového na grafický a přistupovat do části BIOSu jako do namapované videopaměti s rozlišením 320 × 200 px s možností 256 barev. Soubor *images.asm* obsahuje implementaci grafických služeb zajišťujících výpis textů a grafiky. Bylo nutné implementovat vlastní textový font a pomocí přerušení *0x22* jsou volány jednotlivé vykreslovací funkce. V následující ukázce kódu je do registrů *ES:BX* umístěna adresa videopaměti a přes všechny pixely v *CX* je celá nebo část obrazovky (v závislosti na tom, zda-li byl nastaven parametr BL) překreslena jednou hodnotou barvy pozadí.

```
vypln_obrazovku:vypln_obrazovku:
        ...
        mov ax,0xa000
        mov es,ax=
        mov cx,0xed80
        test bl,bl
        jz vyplneni_smycka
        mov cx,SIRKA_OKNA*VYSKA_OKNA
        vyplneni_smycka:
                mov bx,cx
                mov al,[cs:barva_pozadi]
                mov byte [es:bx],al
                loop vyplneni_smycka
        ...
```

Na Obrázku 2.2 jsou znázorněny implementované možnosti grafické knihovny. Konkrétně jsou zde uvedeny použité fonty. Fonty byly vytvořeny v grafickém editoru GIMP, převedeny do 8bitové barevné hloubky dle použité palety a nakonec pomocí bashového skriptu převedeny do zdrojových souborů assembleru.

Obrázek 2.2: Ukázka některých fontů.

Posledním (speciálním) fontem jsou ikony, které jsou vidět na Obrázku 2.3. Tyto ikony jsou použity v prohlížeči jako indikátory zaplněnosti jednotlivých souborů. Tyto ikony jsou inspirovány počítačovou hrou DOOM a jejich zobrazení přesně odpovídá zobrazování v původní hře. Například první ikona je vykreslena, pokud je soubor prázdný ze $100\% - 80\%$ a například poslední ikona je zobrazena, pokud je soubor již plně zaplněn.

Obrázek 2.3: Ukázka některých fontů.

V následující Tabulce 2.3 jsou vypsány jednotlivé obsluhy pro využívání služeb grafické knihovny.

Tabulka 2.3: Obsluhy přerušení *0x22* grafické knihovny.

služba	**registr AH**	**parametry**
nastavení video módu	*0x00*	bez parametrů
zobrazení textu	*0x01*	BX = pozice vykreslení DS:CX = adresa řetězce
nastavení fontu	*0x02*	BX = index fontu
zobrazení hodin	*0x03*	bez parametrů
vyplnění obdélníku	*0x04*	BX = adr. strukt. rozměrů
vyplnění obrazovky	*0x05*	BL = překreslení lišty
změna pozadí	*0x06*	BL = barva pozadí

Službě číslo $0x04$ je potřeba předávat 4 souřadnice a jednu barvu, což již bylo výhodnější předávat strukturou uvedenou v Tabulce 2.4.

Tabulka 2.4: Definice struktury kreslení obdélníku

počáteční index	**počet bajtů**	**vlastnost**
0x00	2	souřadnice Y1
0x02	2	souřadnice Y2
0x04	2	souřadnice X1
0x06	2	souřadnice X2
0x08	1	barva výplně

2.5 Funkce hodin

Počet „ticků“ od začátku dne leží v paměti BIOSu a jedná se o 4 bajty začínající na adrese $0x0040 : 0x006c$. Těchto „ticků“ nastane za den 1573040, což odpovídá frekvenci $18.206\overline{481}Hz$ při původním nastavení frekvenční děličky. Byl tedy implementován přepočet, který vydělí počet tiků hodin již uvedenou frekvencí. Obtížným cílem však bylo provést tento výpočet v celých číslech a pouze s 16bitovými registry.

```
    ...
        mov dx,[es:0x006c]
        mov cx,[es:0x006e]
        mov byte [cs:odpoledne],0
        cmp cx,0xc
        jl neopravuj_hodiny
        cmp cx,0xc
        jg urcite_oprav_hodiny
        cmp dx,0x58
        jl neopravuj_hodiny
urcite_oprav_hodiny:
        sub dx,0x58
        sbb cx,0xc
        mov byte [cs:odpoledne],12
neopravuj_hodiny:
        ...
        mov bx,1080
        mul bx
        ...
        mov bx,19663
        div bx
        ...
```

Kód výpočtu je velmi obsáhlý, proto je zde uvedena jen malá ukázka. Výše uvedenou frekvenci bylo potřeba převést na zlomek v základním tvaru, což je $\frac{1080}{199663}$. Dalším problémem bylo, že se počet sekund za den nevejde do 16bitového registru[3], což by bránilo dalšímu výpočtu. Proto byl navrhnut mechanismus, který určí, jestli je odpoledne, a pokud ano, odečte od počtu tiků hodin půl dne[4].

[3] Den má 86400 sekund, do 16bitového registru se vejde 65536 sekund

[4] Tiků za půl dne je v šestnáctkové soustavě $0xc0058$, což je v ukázce kódu vidět

3. Uživatelská příručka

V této kapitole bude představeno grafické rozhraní OS společně s jeho možnostmi ovládání.

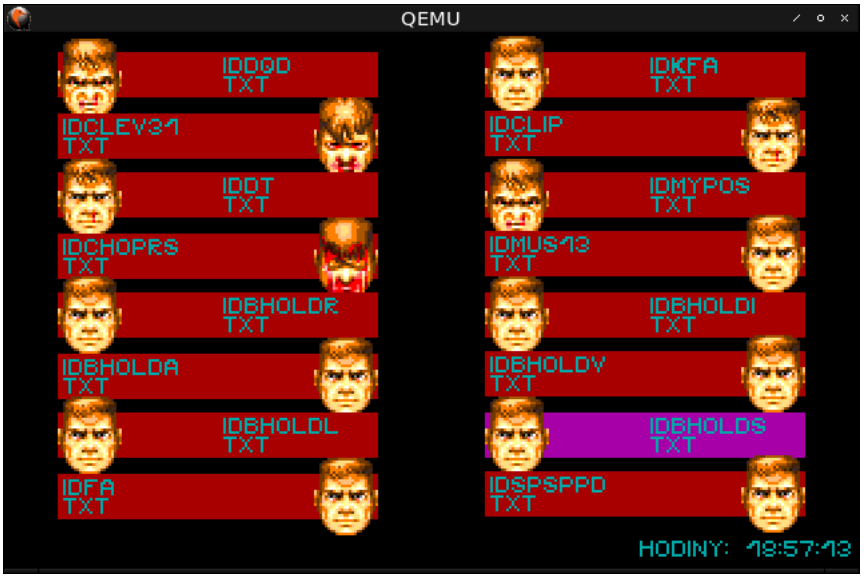
3.1 Menu

V menu se lze pohybovat pomocí šipek na klávesnici a potvrzení se provádí klávesou enter. Na Obrázku 3.1 je kromě položek menu vidět i zobrazení hodin a verze OS. Programy lze ukončit běžným způsobem klávesou escape, pro vynucený návrat do hlavního menu OS se používá klávesa *PrtSc*, která násilně ukončí program.

Obrázek 3.1: Vstupní menu OS.

3.2 Prohlížeč souborů

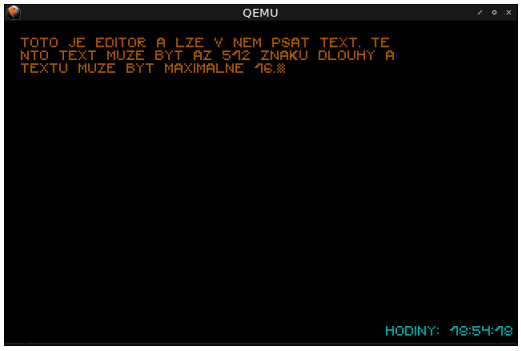
Jako první možnost v menu je prohlížeč souborů. Pokud nebude disk připojený, nebo naformátovaný, tak se nezobrazí názvy ani vlastnosti jednotlivých souborů. Je-li disk v pořádku, zobrazí se správně názvy jako na Obrázku 3.2. U každého souboru je ikona reprezentující velikost souboru. Zvolit soubor pro editaci je možné pomocí šipek a potvrzením enter.



Obrázek 3.2: Prohlížeč souborů.

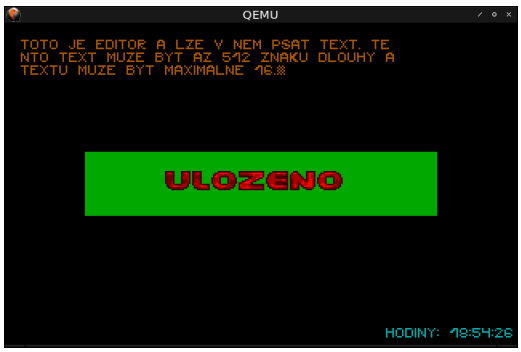
3.3 Textový editor

Podle diagramu 1.1 je možné se do textového editoru dostat z menu nebo prohlížeče. Pro první možnost je editor spuštěn s parametrem prvního souboru na disku, jinak je soubor nastavený pomocí prohlížeče. Editace probíhá stejně jako u jiných textových editorů s tím rozdílem, že pracuje s omezenou znakovou sadou [A-Z], [0-9].



Obrázek 3.3: Textový editor.

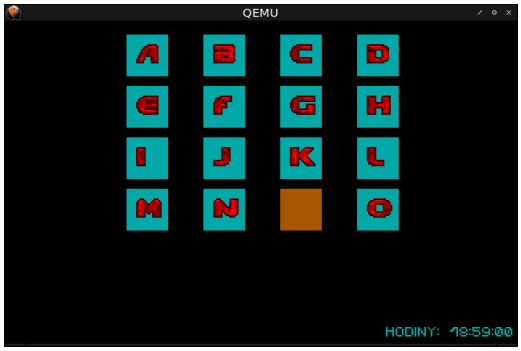
Uložení je vyvoláno klávesou enter a je oznámeno upozorněním viz Obrázek 3.4. Poté se čeká na stisk libovolné další klávesy, kterou se uzavře dialog a lze pokračovat v editaci souboru.



Obrázek 3.4: Uložený text do souboru.

3.4 Hra

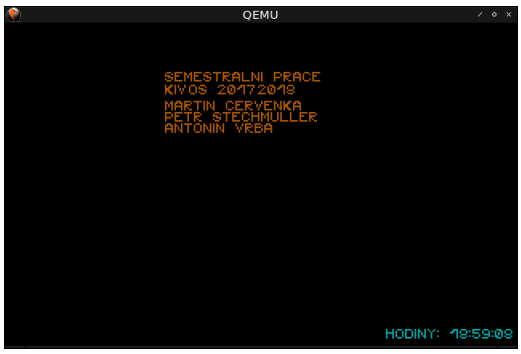
Programové vybavení OS obsahuje také počítačovou hru Loydova patnáctka. Na obrázku je stav těsně před koncem hry, kdy stačí hráči k vítězství stisknout pravou šipku, aby přesunul písmeno *O* na správnou pozici a seřadil tak celé herní pole.



Obrázek 3.5: Hra Loydova patnáctka.

3.5 Informace o systému

Poslední položkou v menu jsou informace o OS na Obrázku 3.6. Zde se také nachází možnost jak naformátovat disk. Z této práce je zřejmé, že byla inspirována hrou DOOM, proto znalci této hry nebudou mít žádný problém tuto funkci najít.



Obrázek 3.6: Uložený text do souboru.

4. Závěr

Zadáním bylo vytvoření minimalistického operačního systému. Tento cíl byl splněn, ve vytvořeném operačním systému bylo řešeno mnoho problémů, které se týkají návrhu a implementace i větších operačních systémů. V rámci práce byl implementován minimalistický souborový systém a jeho základní obsluha pomocí programů pro procházení a editaci souborů. Dále byl řešen problém více vláken ve formě stále běžících hodin v pravém dolním rohu obrazovky a z toho důvodu byl implementován i minimalistický plánovač. Nadstavbou práce je pak grafické rozhraní, které velmi usnadňuje práci s operačním systémem. V neposlední řadě byla implementována i hra Loydova patnáctka, protože správný operační systém se bez her prostě neobejde.

Práci by bylo možné dále rozšiřovat. Pro běžné použití by bylo vhodné rozšířit souborový systém, protože v aktuálním stavu je možné uložit pouze $8kB$ uživatelských dat[1].

[1] Aby se tento OS nedržel nevhodného tvrzení „640K is more memory than anyone will ever need.“