Електротехнички факултет Универзитет у Београду
Катедра за рачунарску технику и информатику

Архитектура и организација рачунара 2
– пројектни задатак –

Циљ пројекта је имплементација хипервизора коришћењем *Kernel-based Virtual Machine (KVM) API*. Хипервизор је потребно имплементирати у језику C/C++. Пројекат је подељен у три целине (верзије), А, Б и В. Препоручено је да студент прочита цео пројекат у целости пре него што крене у реализацију истог.

Структура пројекта је следећа:

```
project/
├── guest/
│   ├── inc/
│   │   ├── descriptors.h
│   │   ├── interrupts.h
│   │   └── io.h
│   ├── src/
│   │   ├── interrupts.c
│   │   └── main.c
│   ├── guest.ld
│   └── Makefile
└── host/
    ├── inc/
    │   └── vm.h
    ├── src/
    │   ├── main.c
    │   └── vm.c
    └──Makefile
```

Пројекат се састоји од два директоријума. Први садржи целокупан код који одговара систему госта (`guest`), а други домаћина (`host`). Оба директоријума садрже под директоријуме за заглавља (`inc`) и за код (`src`) фајлове. Код који се тренутно налази у пројекту је из задатка `KVM_Zadatak_4` са сајта предмета. `Makefile` је написан тако да је могуће додавати произвољан број фајлова са изворним кодом у `inc` и `src` под директоријуме. `make` алат ће направити `build` директоријум преведеним кодом, конкретно извршни фајл хипервозора (`host`) и преведени код госта (`guest`) који је потребно покренути у виртуелној машини. Извршни фајл хипервиозра је `hypervisor` (`host/ build/hypervisor`), а преведен код госта `guest.img` (`guest/build/guest.img`).

А. [15 поена] Основне особине хипервозора

Обезбедити следеће основне особине хипервизора:

- Величина физичке меморије госта је 2MB, 4MB или 8MB. Одговарајућа величина се задаје као параметар командне линије хипервизора преко опције **`-m`** или **`--memory`**.
- Виртуелна машина (ВМ) раде у 64-битном моду (*long mode*).
- Величина странице је 4KB или 2MB. Одговарајућа величина се задаје као параметар командне линије хипервизора преко опције **`-p`** или **`--page`**.
- ВМ са само једним виртуелним процесором.

- Подржава серијски испис и читање на *IO* порт `0xE9`. Величина података који може да се пише/прочита на/са порт је 1 бајт.
- Подржава само ВМ које завршавају извршавање инструкцијом `hlt`.
- Учитавање и покретање једног или више госта који је дат као параметар командне линије хипервизора преко опције **`-g`** или **`--guest`**. Број ВМова који се покреће се задаје низ фајлова који представљају извршни код госта. За сваки фајл је потребно покренути једну ВМ.
- За сваку ВМ је потребно покренути једну *POSIX* нит (искористити `pthread.h` у коду хипервозора/домаћина) која ће да обезбеди успешно извршавање гост кода.
- У случају да ВМ падне или се деси неочекивани ВМ излаз, потребно је завршити извршавање те ВМ односно нити и исписати на стандардном излазу код грешке која је нагло зауставила извршавање ВМ.

Пример позива можете да видите у наставку:
```
./hypervisor --memory 4 --page 2 --guest guest1.img guest2.img
```

Б. [15 поена] Подршка за рад са фајловима

Потребно је проширити верзију А хипервизора додавањем функционалности рада са фајловима. Обезбедити да гост може да отвори, затвори, чита, уписује и позиционира у фајлу. Потпис наведених функција су:

```
int open(const char *path, int flags);
int close(int fd);
int read(int fd, char *buf, int count);
int write(int fd, const char *buf, int count);
int lseek(int fd, const int offset, int off_flag);
```

Детаљи функција у функцији за отварање фајла су:

Функција:	`open`
Параметри:	`path` – назив фајла. `flags` – индикатори за начин отварања фајла.
Повратна вредност:	Јединствени индикатор за отворени фајл или -1 у случају да је дошло до грешке.

Функција:	`close`
Параметри:	`fd` – јединствени индикатор за фајл.
Повратна вредност:	0 у случају да је операција успешно извршена или -1 ако је дошло до грешке.

Функција:	`read`
Параметри:	`fd` – јединствени индикатор за фајл из кога се чита. `buf` – показивач на бафер који се пуни са садржајем из фајла. `count` – величина бафера.
Повратна вредност:	Број прочитаних бајтова или -1 ако је дошло до грешке.

Функција:	`write`
Параметри:	`fd` – јединствени индикатор за фајл у који се уписује. `buf` – показивач на бафер из кога се уписује. `count` – величина бафера.
Повратна вредност:	Број уписаних бајтова или -1 ако је дошло до грешке.

Функција:	`lseek`
Параметри:	`fd` – јединствени индикатор за фајл у који се уписује. `offset` – померај од почетка фајла где жели корисник да се позиционира. `off_flag` – индикатор помераја.
Повратна вредност:	Померај у односу на почетак фајла.

Индикатори за отварање фајла су:

Индикатор	**Вредност**	**Значење**
`O_RD`	1	Фајл се отвара само за читање.
`O_WR`	2	Фајл се отвара само за уписивање.
`O_RDWR`	4	Фајл се отвара за читање и уписивање.
`O_CREATE`	8	Фајл се креира у случају да не постоји.

Индикатори за `lseek` функцију су:

Индикатор	**Вредност**	**Значење**
`SEEK_SET`	1	Курсор у фајлу се помера на `offset` вредност.
`SEEK_END`	2	Курсор у фајлу се помера на крај фајла. Вредност `offset` се игнорише у случају овог индикатора.

Потребно је обезбедити:

- Имплементирати наведене функције заједно са индикаторима за отварање фајла. Функције реализовати преко *IN*/*OUT* инструкција користећи *IO* порт `0x0278`.
- Померај се поставља на 0 приликом отварања фајла.
- Имена фајлова могу да се састоје од малих и великих слова абецеде, цифара 0 до 9 и тачке. Име фајла мора да започне словом. У случају покушаја креирања фајла са именом који не испуњава овај услов, вратити госту грешку.
- Локални и дељени фајлови се чувају на систему домаћина.
- Дељење фајлова између ВМ. Дељени фајлови између ВМ се задају хипервозору као низ вредности преко параметра командне линије **`-f`** или **`--file`**. Хипервизору се додељују низ путања до дељених фајлова. Задати фајлови преко ове опције су дељени између виртуелних машина. У случају да нека ВМ покуша да упише у ове дељене фајлове, хипервизор мора да направи локалну копију фајла за ту ВМ (*copy on write*) и надаље да ради са том локалном копијом. Копија се прави приликом првог уписа у фајл, односно приликом првог позива функције за упис у фајл.

Обезбедити да ВМ може да ради са дељеним фајловима као и са локалним фајловима (НЕ дељеним фајловима). Хипервизор мора да обезбеди правилно руковање фајловима, односно хипервизор мора да води рачуна да сваки гост приступа само фајловима које је тај гост направио или којима има приступ за читање.

Пример позива можете да видите у наставку:

```
./hypervisor -m 4 -p 2 -g guest1.img guest2.img --file a.txt b.txt
```

<u>В. [15 поена] Подршка за рад са прекидима</u>

Потребно је проширити верзију Б хипервизора додавањем подршке за прекиде. Потребно је обезбедити комуникацију између више виртуелних машна користећи прекиде и подршку хипервизора за убацивање прекида у госта. Комуникација између виртуелних машина се обавља

преко дељеног бафера који контролише хипервизор. Гост може да чита или да уписује у дељени бафер.

Обезбедити да гост има две могуће улоге (мод рада):

- Читање – читање из дељеног бафера који контролише хипервизор. Користити порт `0x510` за читање.
- Уписивање – уписивање у дељени бафер који контролише хипервизор. Користити порт `0x510` за уписивање.

Хипервизор првом приликом убацује захтев за прекид са улазом 32 у госта и тако задаје мод рада (са улазом 32). Приликом прве обраде овог прекида гост добија преко *IO* порта `0x510` мод рада (0 – читање, 1 – уписивање). Хипервизор само једној виртуелној машини додељује мод рада уписивање, док свим осталим додељује мод рада читање.

У оквиру прекидне рутине за улаз 32 обезбедити:

- Приликом прве обраде прекида потебно је сачувати мод рада виртуелне машине.
- Приликом сваке наредне обраде, ВМ са улогом читања чита из дељеног бафера преко порта `0x510`. ВМ након читања шаље хипервизору број прочитаних бајтова преко порта `0x520` као индикатор да је операција читања завршена. У случају да нису сви бајтови прочитани потребно је зауставити извршавање ВМ.
- Приликом сваке наредне обраде, ВМ са улогом уписивања у дељени бафер уписује преко порта `0x510`. Након уписивања, ВМ од хипервизора чита количину уписаних бајтова са порта `0x520`.

Уписивање у дељени бафер се врши слањем прво броја бајтова (неозначена променљива ширине 32 бита) који ВМ жели да упише у бафер, а затим и низ бајтова који се уписује. Приликом читања, ВМ са улогом читања прво добија од хипервизора број бајтова који чита, а затим и низа бајтова из дељеног бафера. Направити систем у ком ВМ са улогом уписивања, наставља са уписивањем у дељени бафер тек кад све остале ВМ заврше са читањем. Величину дељеног бафера дефинисати преко макроа `BUFFER_SIZE`. У случају да је број бајтова који се уписује у дељени бафер већи од вредности `BUFFER_SIZE`, хипервизор игнорише вишак послатих бајтова.

Тестирати ову фазу покретањем хипервизора са две или више виртуелних машина, где једна виртуелна машина врши читање из локалног или дељеног фајла и уписивање у дељени бафер, а остале виртуелне машина преко прекида врше читање из дељеног бафера и уписивање прочитане вредности у локални фајл.

У свакој фази пројекта обезбедити да у случају да је хипервизор покренут са некоректним вредностима за опције хипервизора (нпр. вредност 3 за **`--memory`** опцију, јер су дозвољене вредности 2, 4 или 8), потребно је обавестити корисника од грешци и стопирати извршавање.

Одбрана и тестирање

На одбрани се очекује од студента да:

- Зна да одговори на питања о детаљима имплементације пројекта.
- Одради успешно модификацију за фазу пројекта коју брани.
- Покаже неколико (2 или 3) примера који тестирају функционалност сваке фазе пројекта.