Математичка гимназија

МАТУРСКИ РАД

- из рачунарства и информатике -

Израда X86 32bit i686 језгра оперативног система

Ученик:
Алекса Вучковић IVд

Ментор:
Милош Арсић

Београд, јун 2021.

Садржај

1

Увод

Идеја за овај рад прозишла је из екстензивног коришћења GNU/Linux система, као и жеља за разумевањем рада рачунара на најнижем нивоу.
Цео код је писан у GNU Asembler-у и C-у и може се наћи на GitHub-u на страници `https://github.com/aleksav013/mykernel`. Сав код је доступан под GPLv3 лиценцом.

2

Теорија

2.1 X86 архитектура

X86 архитектура је пробитно била 8битна (садржала је регистре дужине 8 битова), 16битна, затим 32битна и на крају 64битна. Данас 64битну X86 архитектуру знамо као и AMD64, X86-64 или X86_64.
Заједно са ARM-ом једна од најкоришћенијих архитектура.

2.1.1 Регистри процесора

Постоји више врста регистара процесора: 16битни регистри опште намене: ax,bx,cx,dx.
`https://wiki.osdev.org/CPU_Registers_x86`

2.1.2 32bit i686

32битни регистри опште намене: eax,ebx,ecx,edx. 64битни регистри опште намене: rax,rbx,rcx,rdx.

2.1.3 Real mode

Реални мод је стање процесора у којем нам је дозвољено адресирање само првих 20мб меморије. Прелазак из реалног у застицени мод постиже се далеким скоком "far jump".
`https://wiki.osdev.org/Real_Mode`

2.1.4 Сегментација

Сегментација је решење којим се омогућава адресирање више меморије него што је то хардверски предвиђено.
`https://wiki.osdev.org/Segmentation`

2.1.5 Protected mode

Заштићен мод је стање процесора у којем процесор има пун приступ целом опсегу меморије за разлику од реалног мода.
`https://wiki.osdev.org/Protected_Mode`

2.2 Редослед покретања

Од притиска дугмета за паљење рачунара, па до учитавања оперативног ситема постоји цео један процес. Након притиска дугмета рачунар прво извршава POST (Power On Self Test) који је једна од почетних фаза BIOS (Basic Input Output System). У POST-у рачунар покушава да инцијализује компоненте рачунарског система и проверава да ли оне испуњавају све услове за стартовање рачунара. Уколико је цео процес прошао без грешака наставља се даље извршавање BIOS-а. BIOS сада има улогу да пронађе медијум који садржи програм који ће учитати језгро оперативног система у рам меморију рачунара. Тај програм се назива Bootloader.
`https://wiki.osdev.org/Boot_Sequence`

2.2.1 Bootloader

Bootloader је програм који се налази у првих 512битова медијума, и његов задатак је да учита језгро оперативног система у рам меморију и преда му даље управљање.
`https://wiki.osdev.org/Bootloader`

2.3 ELF

ELF је формат бинарни фајл који се састоји од тачно одређених секција и који може да се покрене.
`https://wiki.osdev.org/ELF`

3

Коришћени алати

У даљем тексту се могу видети неки од алата коришћених у креирању овог рада.
Сви коришћени алати поседују GPLv2 или GPLv3 лиценцу. GNU Public Licence је лиценца отвореног кода која дозвољава модификовање и дистрибуирање кода све док тај је тај код јавно доступан.
Једини програм са листе који није GNU-ов је QEMU виртуална машина.
Оперативни систем коришћен у изради овог пројекта је Artix Linux. Artix Linux је GNU/Linux дистрибуција базирана на Arch Linux-у. Већина коришћених програма је већ компајлована и спремна за употребу и налази се у официјалним репозиторијима.
За програме који су морали бити мануелно компајловани дате су инструкције у даљем тексту.

3.1 Binutils

Изворни код софтвера се може наћи на страници `https://www.gnu.org/software/binutils/`, заједно са упутством за компајловање и коришћење.
Овај пакет садржи програме неопходне за компајловање као што су асемблер и линкер.

3.1.1 Пре додавања LIBC

Из разлога што се не користи стандардна библиотека већ самостално написана специфицно за овај пројекат, потребно је мануелно компајловати GNU Binutils. Међутим, постоји могућност коришћења већ спремног пакета који се за дистрибуције базиране на Arch Linux-у може наћи на станици `https://aur.`

archlinux.org/packages/i686-elf-binutils/. Поједине дистрибуције већ имају овај пакет компајлован, али је препорука мануелно компајловати да би се избегла некомпатибилност, а и просто из разлога што ће након формирања наше C библиотеке бити неопходно компајловати овај програм за сваки систем посебно. За оне које желе сами да компајлују дат је део инструкција који се разликује од упутства датог на званичном сајту а тиче се конфигурисања пре компилације.

```
mkdir build
cd build

../configure \
    --target=i686-elf \
    --with-sysroot \
    --prefix=/usr \
    --bindir=/usr/bin \
    --libdir=/usr/lib/i686-elf \
    --disable-nls \
    --disable-werror

make
make install
```

3.1.2 После додавања LIBC

Након додавања наше C библиотеке потребно је компајловати GNU Binutils тако да ту библиотеку и користи приликом компајловања нашег оперативног система. **Напомена:** Потребно је поставити $SYSROOT на локацију где се библиотека налази. То је могуће урадити на следећи начин:

```
export SYSROOT=/put/do/biblioteke
```

Инстукције за компајловање дате су у наставку:

```
../configure \
    --target=i686-elf \
    --with-sysroot=$SYSROOT \
    --prefix=/usr \
    --bindir=/usr/bin \
    --libdir=/usr/lib/i686-elf \
    --disable-nls \
    --disable-werror
```

3.1.3 GNU Asembler

Иако тренутно постоје много популарније алтернативе попут NASM(Netwide Assembler) и MASM(Microsoft Assembler) који користе новију Интелову синтаксу, ипак сам изабрао GASM због компатибилности са GCC компајлером.

GASM користи старију AT&T синтаксу коју карактерише: обрнут поредак параметара, префикс пре имена регистара и вредности константи, а и величина параметара мора бити дефинисана. Због тога ће можда неким читаоцима бити користан програм "intel2gas" који се за Arch Linux може наћи на станици `https://aur.archlinux.org/packages/intel2gas/`.
Овај програм је коришћен за компајловање дела кода написаног у асемблеру.

3.1.4 GNU Linker

Овај програм је коришћен за линковање("спајање") свог комапјлованог кода у једну бинарну датотеку која представља кернел.

3.2 GCC

Изворни код софтвера се може наћи на страници `https://gcc.gnu.org/`, заједно са упутством за компајловање и коришћење.
`https://aur.archlinux.org/packages/i686-elf-gcc/`
GCC је GNU-ов сет компајлера.

3.2.1 Пре додавања LIBC

```
mkdir build
cd build

../configure \
    --target=i686-elf \
    --prefix=/usr \
    --disable-nls \
    --disable-plugin \
    --enable-languages=c,c++ \
    --without-headers

make all-gcc
make all-target-libgcc

make -k check || true

make install-gcc
make install-target-libgcc
```

3.2.2 После додавања LIBC

```
../configure \
    --target=i686-elf \
    --prefix=/usr \
    --with-sysroot=$SYSROOT \
    --disable-nls \
    --disable-plugin \
    --enable-languages=c,c++
```

3.3 GRUB

Изворни код софтвера се може наћи на страници `https://www.gnu.org/software/grub/`, заједно са упутством за компајловање и коришћење.

GRUB је bootloader који је коришћен на овом пројекту. План је да у будућности GRUB буде замењен са мојим bootloader-ом, и да се комплетан код буде мој.

3.4 QEMU

Изворни код софтвера се може наћи на страници `https://www.qemu.org/`, заједно са упутством за компајловање и коришћење.

QEMU је виртуална машина у којој ће језгро бити тестирано и приказано зарад практичних разлога. QEMU је одабран за овај пројекат јер за разлику од других виртуалних машина поседује cli (command line interface) из кога се лако може позивати из скрипти као што су Makefile-ови.

3.5 Make

Изворни код софтвера се може наћи на страници `https://www.gnu.org/software/make/` заједно са упутством за компајловање и коришћење.

Make нам омогућава да са лакоћом одржавамо и манипулишемо изворним фајловима. Могуће је све компајловати, обрисати, креирати iso фајл као и покренути QEMU виртуелну машину са само једном кључном речи у терминалу. Креирани Makefile за потребе овог пројекта биће детаљно објашњен у даљем тексту.

3.6 Мање битни алати

3.6.1 git

Изворни код софтвера се може наћи на страници `https://git.kernel.org/pub/scm/git/git.git`.
Git је програм који нам помаже да одржавамо изводне фајлове и

3.6.2 xorriso(libisoburn)

`https://www.gnu.org/software/xorriso/`
Служи за креирање ISO фајлова који се могу "нарезати" на CD или USB флеш са којих се касније диже систем.

3.6.3 GDB

`https://www.sourceware.org/gdb/`

4

Језгро оперативног система

4.1 Почетак

https://wiki.osdev.org/Bare_Bones

4.1.1 Теорија

4.1.2 Имплементација

4.2 Испис на екран - VGA

4.2.1 Теорија

4.2.2 Имплементација

4.3 LIBC почетак

4.3.1 Теорија

4.3.2 Имплементација

4.4 Глобални конструктори, заштита стека

https://wiki.osdev.org/Calling_Global_Constructors
https://wiki.osdev.org/Stack_Smashing_Protector

4.4.1 Теорија

4.4.2 Имплементација

4.5 Global Desctiptor Table

https://wiki.osdev.org/GDT

4.5.1 Теорија

4.5.2 Имплементација

4.6 Interrupt Desctiptor Table

https://wiki.osdev.org/IDT

4.6.1 Теорија

4.6.2 Имплементација

4.7 IRQ и PIC

https://wiki.osdev.org/IRQ https://wiki.osdev.org/PIC

4.7.1 Теорија

4.7.2 Имплементација

4.8 Тастатура

4.8.1 Теорија

4.8.2 Имплементација

4.9 PIT - Programmable Interval Timer

https://wiki.osdev.org/PIT

4.9.1 Теорија

4.9.2 Имплементација

4.10 Heap

https://wiki.osdev.org/Heap

4.10.1 Теорија

4.10.2 Имплементација

4.11 Paging

https://wiki.osdev.org/Paging

4.11.1 Теорија

4.11.2 Имплементација

4.12 Moj LIBC

https://wiki.osdev.org/Creating_a_C_Library

4.12.1 Теорија

4.12.2 Имплементација

5

Закључак

Овај пројекат је био сјајна прилика да тестирам границе свог знања.

Литература

[1] https://wiki.osdev.org/Expanded_Main_Page

[2] http://jamesmolloy.co.uk/tutorial_html/