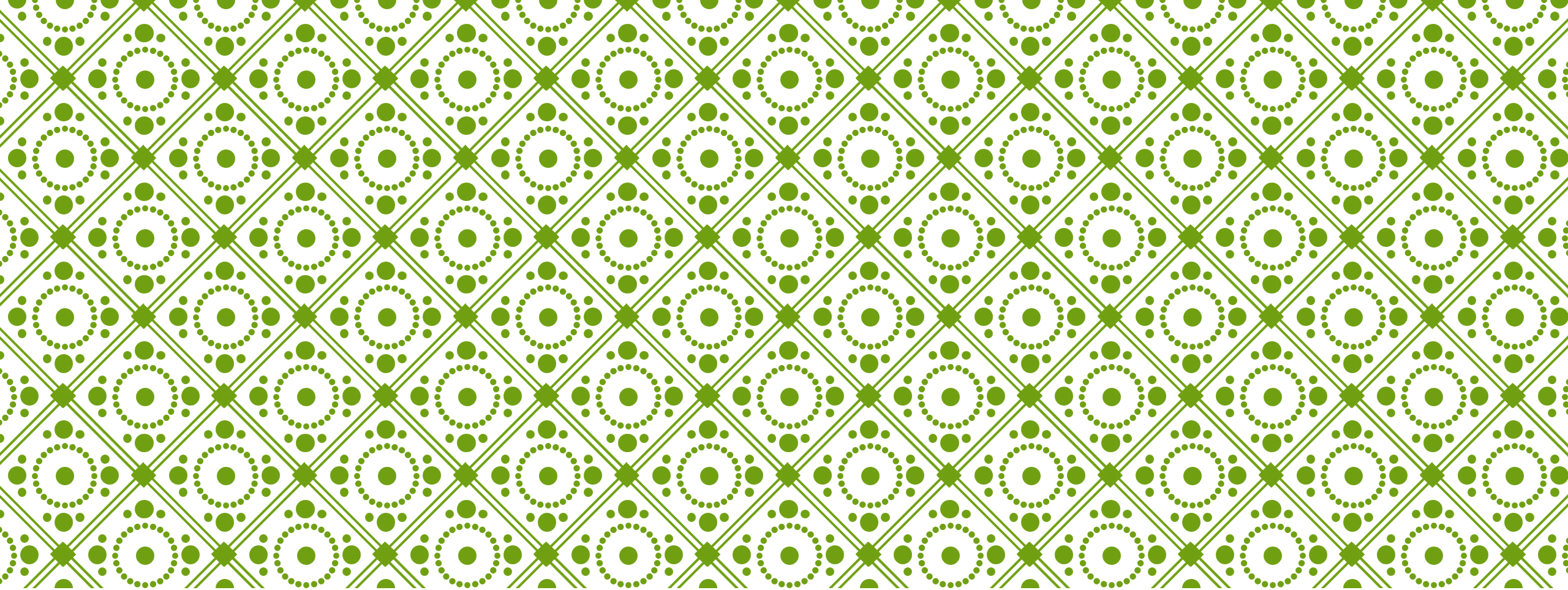


МЕМОРИЈА, ДИНАМИЧКА МЕМОРИЈА

Програмирање у реалном времену
Вјежба 2.0

УВОД

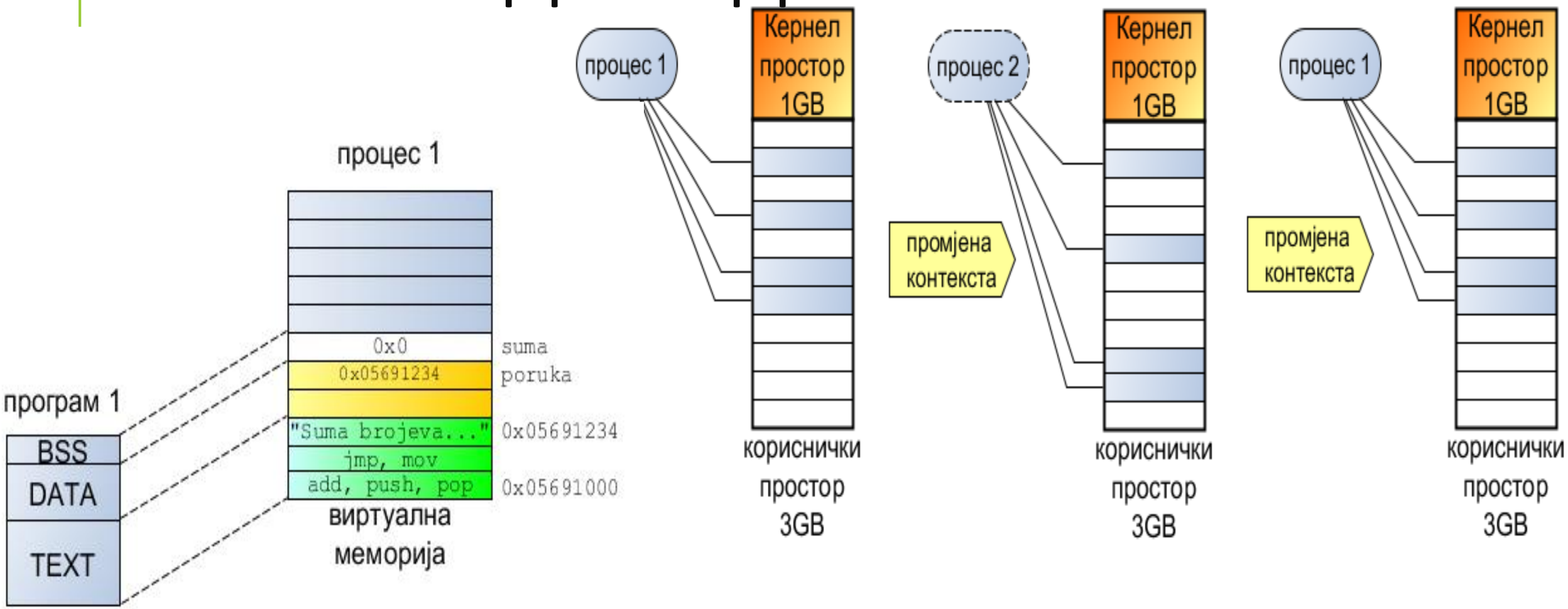
Потребна знања

- Це програмски језик, компајлирање
- Основе Линукса
- Одслушана прва вјежба
- Урађена задаћа [ово смо се договорили да не треба]

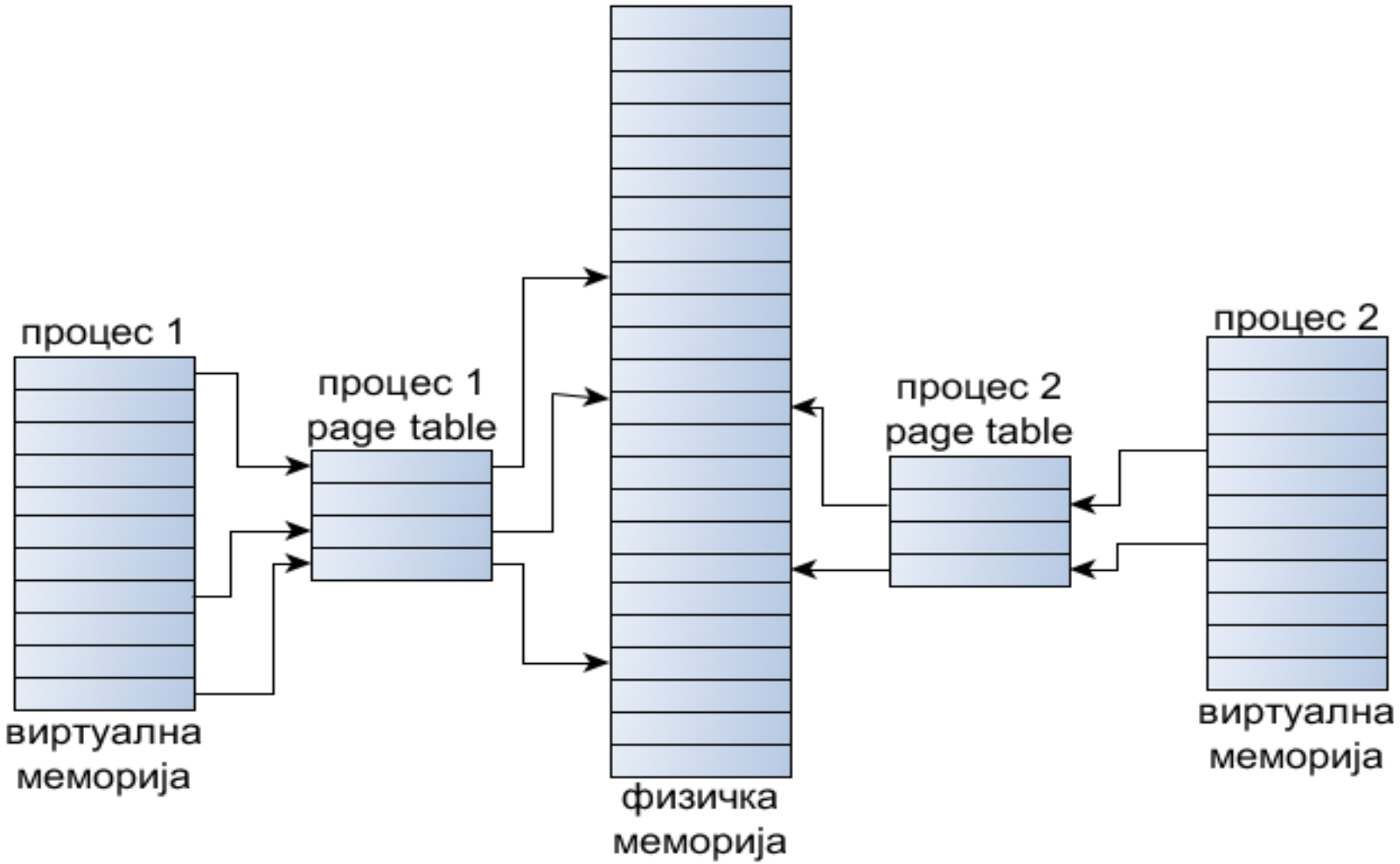
Циљ вјежби

- Упознати се са динамичким аспектима меморије (ХИП и MMS) и утицаја изворног кода на меморију
- Упознати се и оспособити се за коришћење разних Линукс команди помоћу којих се добија боља слика о меморији процеса и/или програма

ШТА СМО ДО САДА НАУЧИЛИ



КАКО ТО ИЗГЛЕДА ФИЗИЧКИ



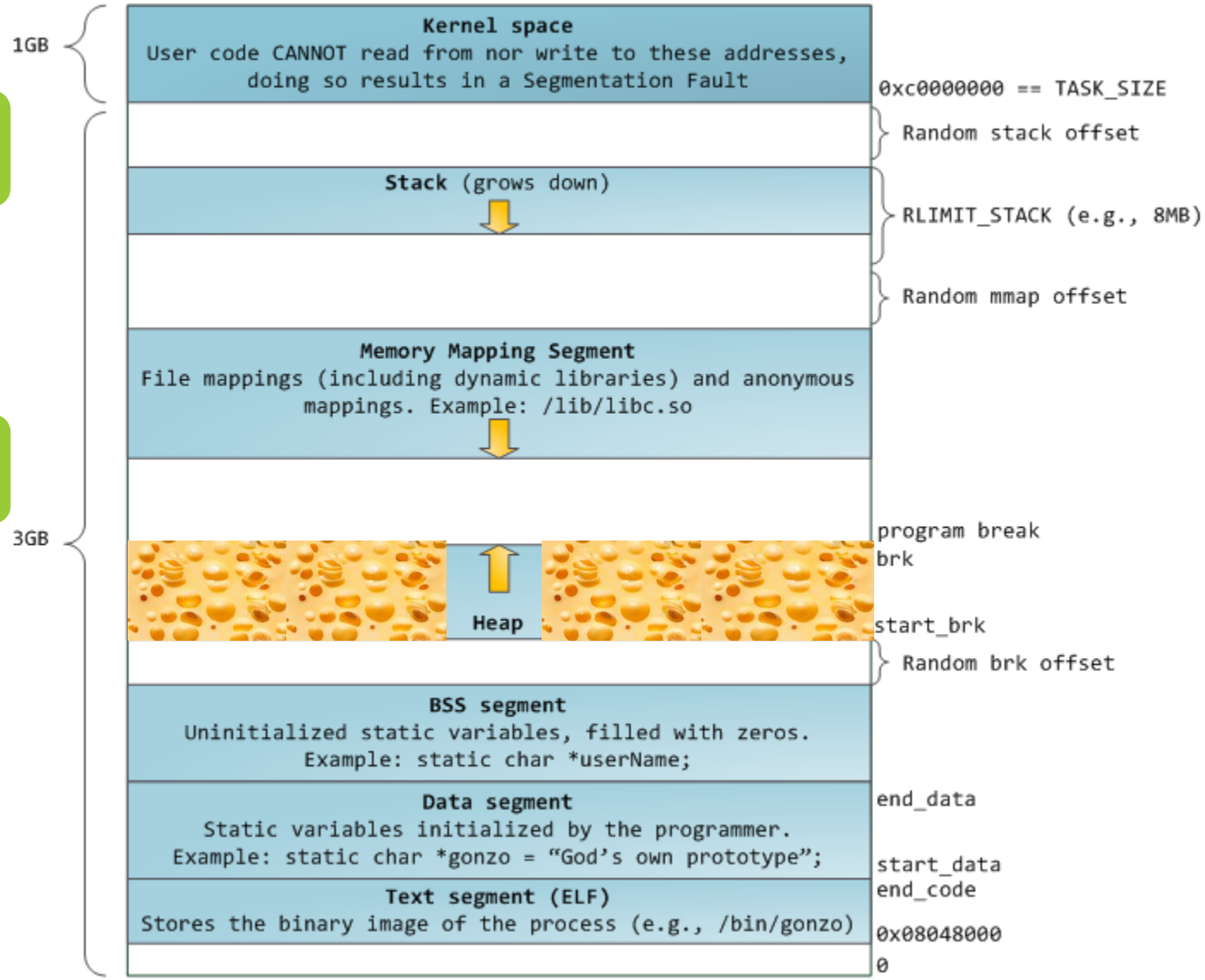
АЛОЦИРАЊЕ МЕМОРИЈЕ

Меморија се алоцира на три начина:

- **`expand_stack()`** – када понестане стеку
- **`mmap()`** у MMS – за велике меморијске блокове
- **`brk()`** у ХИПу – за мале меморијске алокације

Меморија се алоцира у блоковима исте величине:

- M_TOP_PAD + PAGESIZE
  - M_TOP_PAD – промјењива из malloc.h, уобичајена вриједност је 128 * 1024.
    - Може бити промијењена вриједност коришћењем функције mallopt()
  - PAGESIZE – Линукс конфигурациони параметар, који се може прочитати са
    - getconf PAGESIZE



ИЗВОРНИ КОД ПРОГРАМА MEMORY_LAYOUT.C

Користећи научену технику ишчитавања покренутог процеса прегледати стање сегмената сваки пут када се пређе један корак (од једне до друге наредбе `getchar()`)

```
cat /proc/PID/maps
gcc memory_layout.c -lpthread -o memory_layout
```

Разлог због којег се алокација меморије у програмској нити ради помоћу `mmap()` лежи у конкурентности програмских нити која може да дође у колизију позивом `brk()` због начина на који ова функција ради (само помјера `brk` показивач)

ИЗМЈЕНЕ У ИЗВОРНОМ КОДУ MMAP.C

Промијенити коментаре. Повећати алоцирану меморију у main() функцији са 1000 бајтова на:

1. нешто мало испод 128 килобајта,
2. нешто мало изнад 128 килобајта али мање од 132 килобајта и
3. Изнад 132 килобајта

Са првобитним кодом, пронаћи границу када се позива mmap() а када brk()

КОРИСНЕ ЛИНУКС КОМАНДЕ И ОСТАЛО

```
readelf --file-header ./memory_layout
ldd ./memory_layout
strace ./memory_layout
objdump --disassemble-all./memory_layout | less
sudo gcore <procID>
hexdump -C core.procID
stat memory_layout
cat /proc/<procID>/maps - bolje tac
```

https://en.wikisource.org/wiki/The_Paging_Game#The_Crating_Game

ДОМАЋА ЗАДАЋА

Направити програм у програмском језику Це који ради исто што и команда:

1. `readelf --file-header`
2. `readelf --symbols`
3. `readelf --section-headers`
4. `readelf --program-headers`

Сваки од студената ће добити САМО ЈЕДНУ од ове четири команде да уради за задаћу.

При изради, користити документацију:

- http://www.ouah.org/RevEng/x430.htm
- http://www.skyfree.org/linux/references/ELF_Format.pdf

МОГУЋЕ ЈЕ ДА ПОСТОЈЕ РАЗЛИКЕ У 64 И 32 БИТНИМ АПЛИКАЦИЈАМА

Послије одгледаног остатка предавања, ријешену задаћу, са форматом имена `file_header_broj_indeksa.c` (уколико је –file-header) оставити на локацију

`rt-programming-2020/labs/lab2/`

РОК: 04.04.2020