Урок 3: Демонстрація наявності трансляції сторінок

Катерина Ковальчук

Грудень 2024р

Анотація

У цьому документі описується трансляція сторінок та як її продемонструвати у x86_64

1. Вступ

Перша реалізація трансляції сторінок була введена у процесорі 80386 у 1985 році.

Внутрішня архітектура процесора 80386 поділяється на три секції:

- Центральний процесорний блок (Central processing unit)
- Блок управління пам’яттю (Memory management unit)
- Блок інтерфейсу шини (Bus interface unit)

Центральний процесорний блок додатково розділений на:

- Виконавчий блок (Execution unit) - має 8 регістрів загального призначення та 8 регістрів спеціального призначення, які використовуються для обробки даних або розрахунку зсуву адрес.
- Блок інструкцій (Instruction unit) - декодує байти опкоду, отримані з 16-байтової черги кодів команд, і впорядковує їх у 3-інструкційну декодовану чергу команд. Після декодування він передає їх до секції керування для отримання необхідних сигналів керування. Перемикач стволів збільшує швидкість всіх операцій зсуву і повороту.

Блок керування пам'яттю складається з блоку сегментації та блоку підкачки. Блок сегментації дозволяє використовувати два компоненти адреси, а саме сегмент і зсув, для переміщення і спільного використання коду і даних. Блок сегментації дозволяє створювати сегменти розміром не більше 4 Гбайт. Блок підкачки організовує фізичну пам'ять у вигляді сторінок розміром 4 кбайт кожна. Також блок підкачки працює під контролем блоку сегментації, що дозволяє розбивати кожен сегмент окремо на сторінки. За таким ж принципом оргганізовується і віртуальна пам'ять. Він ж перетворює віртуальні адреси у фізичні.

Завдяки блоку сегментації забезпечується 4-рівневий механізм захисту системного коду та даних від коду та даних прикладної програми.

2. Компіляція

Процес компіляції повністю поторює процес із першого уроку із потрібними змінами у Makefile.

ПІД ЧАС ПОБУДОВИ ДОКЕР-ЗОБРАЖЕННЯ МОЖЕ ВИНИКНУТИ ТАКА ПОМИЛКА:

```
docker build buildenv -t myos-buildenv
[+] Building 0.4s (3/3) FINISHED

    docker:default
 => [internal] load build definition from Dockerfile

     0.0s
 => => transferring dockerfile: 294B

     0.0s
 => [internal] load .dockerignore

     0.0s
 => => transferring context: 2B

     0.0s
 => ERROR [internal] load metadata for
     docker.io/randomdude/gcc-cross-x86_64-elf:latest

     0.4s
------
 > [internal] load metadata for
     docker.io/randomdude/gcc-cross-x86_64-elf:latest:
------
Dockerfile:1
--------------------
   1 | >>> FROM randomdude/gcc-cross-x86_64-elf
   2 |
   3 |     RUN apt-get update
--------------------
ERROR: failed to solve: randomdude/gcc-cross-x86_64-elf: error
    getting credentials - err: exit status 1, out: ``
```

Це може бути пов'язане із правами доступу до цього образу.

Це можна виправити такою командою:

docker pull randomdude/gcc-cross-x86_64-elf

Та повторити спробу побудови Докер-Зображення.

3. Структура директорії

Структура директорії заишається такою ж, як у другому уроці.

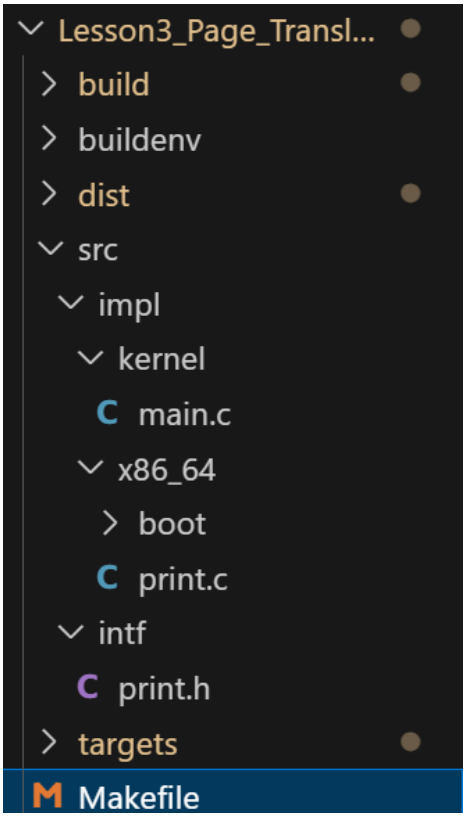

Рис. 1: Структура нашої директорії

4. Код та теорія

Для того, щоб дізнатись, чи дозволена у нас трансляція чи ні, нам потрібно прочитати 31 біт регістра CR0.

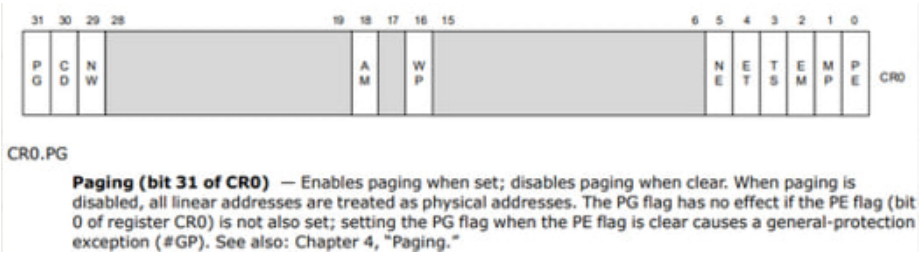

Рис. 2: Структура CR0

Для того, щоб це зробити ми маємо додати такий код до нашого main.c:

```c
#include "print.h"

#define VGA_BUFFER 0xB8000

void print_full_row(unsigned char character) {
  for (size_t col = 0; col <= NUM_COLS; col++) {
    print_char(character);
  }
}

void kernel_main() {
  print_clear();
  uint64_t cr0;
  asm volatile("mov %%cr0, %0" : "=r"(cr0));

  cr0 &= ~(1 << 31); // 0 - disable

  // cr0 |= (1 << 31); // 1 - enable

  // asm volatile("mov %0, %%cr0" ::"r"(cr0));

  int paging_enabled = cr0 & (1 << 31);

  if (paging_enabled) {
    for (unsigned char c = 'z'; c >= 'a'; c--) {
      print_set_color(PRINT_COLOR_GREEN, PRINT_COLOR_BLACK);
      print_full_row(c);
    }
  } else {
    cr0 |= (1 << 31); // 1 - enable

```

```
32      asm volatile("mov %0, %%cr0" ::"r"(cr0));
33
34      for (unsigned char c = 'a'; c <= 'z'; c++) {
35        print_set_color(PRINT_COLOR_RED, PRINT_COLOR_BLACK);
36        print_full_row(c);
37      }
38    }
39
40    asm volatile("hlt");
41  }
```

Для того, щоб пересвідчитись чи увімкнена трансляція чи ні (оскільки ми її вмикали при переході до 64-бітного режиму в уроці 2, то вона завжди буде увімкнена), нам потрібно вручну додавати чи забирати певні рядки коду. Це погана реалізація, тому як буде можливість я спробую додати реалізацію яка реагує на клік чи вмикає трансляцію по таймеру.

Як ми можемо встановити трансляцію? За допомогою використання асемблерного коду в main.c ми створюємо змінну нашого регістру cr0:

```
1  uint64_t cr0;
```

Тоді ми записуємо в неї значення нашого регістра за допомогою інструкції вставки асемблерного коду в C - volatile:

```
1  asm volatile("mov %%cr0, %0" : "=r"(cr0));
```

Далі, залежно від того чи ми хочемо увімкнути чи вимкнути трансляцію ми додаємо чи забираємо потрібні рядки коду:

```
1  cr0 &= ~(1 << 31); // 0 - disable
2  cr0 |= (1 << 31); // 1 - enable
```

Тоді ми дивимось чи увімкнена трансляція і якщо так, то виводимо зелені літери на екран:

```
1  int paging_enabled = cr0 & (1 << 31);
2
3    if (paging_enabled) {
4      for (unsigned char c = 'z'; c >= 'a'; c--) {
5        print_set_color(PRINT_COLOR_GREEN, PRINT_COLOR_BLACK);
6        print_full_row(c);
7      }
8    }
```

Якщо ні - червоні:

```
1  else {
2      cr0 |= (1 << 31); // 1 - enable
3
4      asm volatile("mov %0, %%cr0" ::"r"(cr0));
5
```

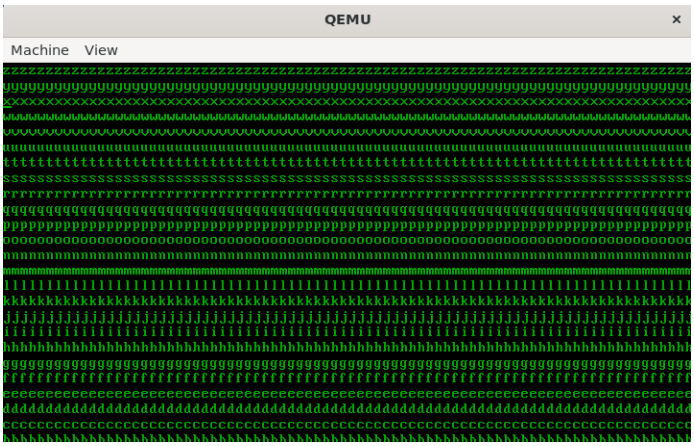

Рис. 3: Дозволена трансляція

```
6       for (unsigned char c = 'a'; c <= 'z'; c++) {
7         print_set_color(PRIM_COLOR_RED, PRINT_COLOR_BLACK);
8         print_full_row(c);
9       }
10    }
```

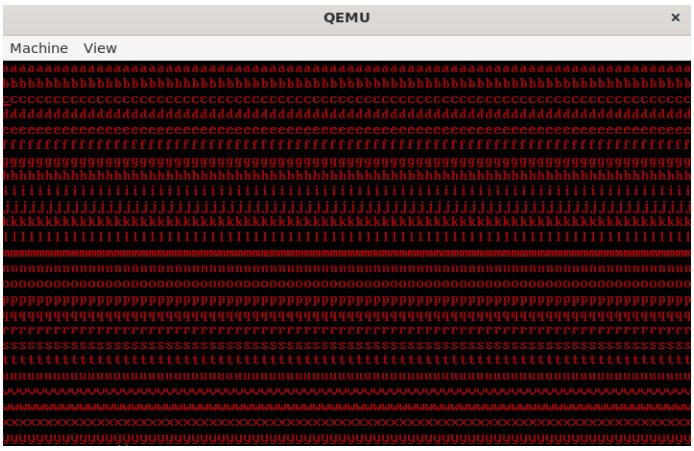

Рис. 4: Трансляція вимкнена

Література

[1] SlideShare. *Introduction to 80386 Microprocessor*, 2014. Available at: https://www.slideshare.net/slideshow/introduction-to-80386-microprocessor/34253668. Accessed: 2024-12-07.

[2] Intel Corporation. *Intel® 80386 Microprocessor Software Development Manual, Vol. 1 and 2*, 1991. Available at: file:///C:/Users/katja/Downloads/325462-sdm-vol-1-2abcd-3abcd-4%20(2).pdf. Accessed: 2024-12-07.