

Урок 3.1: Демонстрація наявності трансляції сторінок із використанням таймера

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 організовується 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 повторює процес із першого уроку із потрібними змінами у Makefile.

ПІД ЧАС ПОБУДОВИ ДОКЕР-ЗОБРАЖЕННЯ МОЖЕ ВИНИКНУТИ ТАКА ПОМИЛКА:

```
1 docker build buildenv -t myos-buildenv
2 [+] Building 0.4s (3/3) FINISHED
   docker: default
3 => [internal] load build definition from Dockerfile
   0.0s
4 => => transferring dockerfile: 294B
   0.0s
5 => [internal] load .dockerignore
   0.0s
6 => => transferring context: 2B
   0.0s
7 => ERROR [internal] load metadata for
   docker.io/randomdude/gcc-cross-x86_64-elf:latest
   0.4s
8 -----
9 > [internal] load metadata for
   docker.io/randomdude/gcc-cross-x86_64-elf:latest:
10 -----
11 Dockerfile:1
12 -----
13 1 | >>> FROM randomdude/gcc-cross-x86_64-elf
14 2 |
15 3 |     RUN apt-get update
16 -----
17 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 записується такою ж, як у другому уроці.

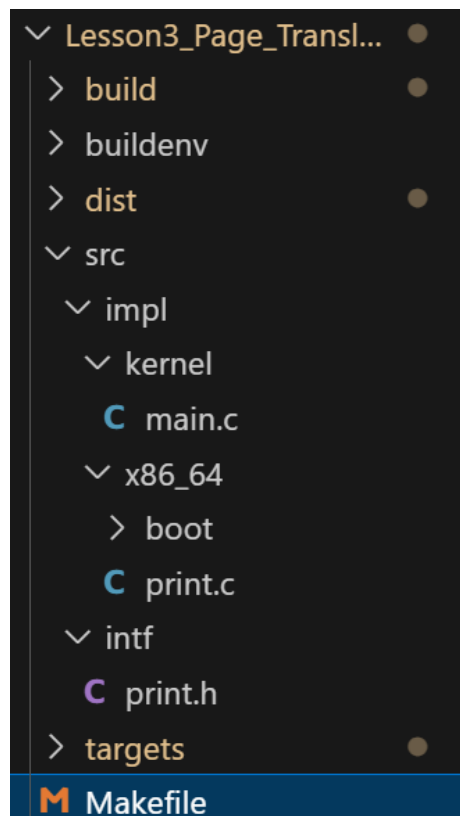


Рис. 1: Структура нашої директорії

4. Код та теорія

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

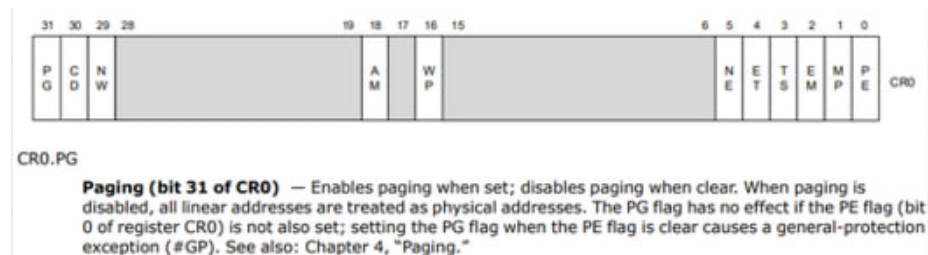


Рис. 2: Структура CR0

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

```
1 #include "print.h"
2
3 #define VGA_BUFFER 0xB8000
4 #define TIMER_DELAY 300000
5
6 void print_full_row(unsigned char character) {
7     for (size_t col = 0; col <= NUM_COLS; col++) {
8         print_char(character);
9     }
10 }
11
12 void delay(int milliseconds) {
13     for (int i = 0; i < milliseconds * 1000; i++) {
14         asm volatile("nop");
15     }
16 }
17
18 void kernel_main() {
19     print_clear();
20     uint64_t cr0;
21     asm volatile("mov %%cr0, %0" : "=r"(cr0));
22
23     cr0 &= ~(1 << 31); // 0 - disable
24     for (unsigned char c = 'a'; c <= 'z'; c++) {
25         print_set_color(PRINT_COLOR_RED, PRINT_COLOR_BLACK);
26         print_full_row(c);
27     }
28
29     delay(TIMER_DELAY);
30
31     cr0 |= (1 << 31); // 1 - enable paging
```

```

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

```

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 її вимкнути, записавши в 31 біт регістра cr0 нульове значення, тоді почекаати і увімкнути її знову.

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));

```

Для затримки нам потрібна функція, яка її виконає за допомогою команди `por`. `por` - No OPeration — інструкція процесора на мові асемблера, або команда протоколу, яка наказує нічого не робити.

```

1  #define TIMER_DELAY 300000
2  void delay(int milliseconds) {
3      for (int i = 0; i < milliseconds * 1000; i++) {
4          asm volatile("nop");
5      }
6  }

```

Тоді ми виводимо червоні літери на екран, тому що трансляція не увімкнена, чекаємо, і виводимо зелені літери.

```

1  for (unsigned char c = 'a'; c <= 'z'; c++) {
2      print_set_color(PRINT_COLOR_RED, PRINT_COLOR_BLACK);
3      print_full_row(c);
4  }
5
6  delay(TIMER_DELAY);
7
8  cr0 |= (1 << 31); // 1 - enable paging
9  asm volatile("mov %0, %%cr0" :: "r"(cr0));
10 print_clear();
11
12 for (unsigned char c = 'a'; c <= 'z'; c++) {

```

```

13     print_set_color(PRINT_COLOR_GREEN, PRINT_COLOR_BLACK);
14     print_full_row(c);
15 }

```



Рис. 3: Трансляція вимкнена

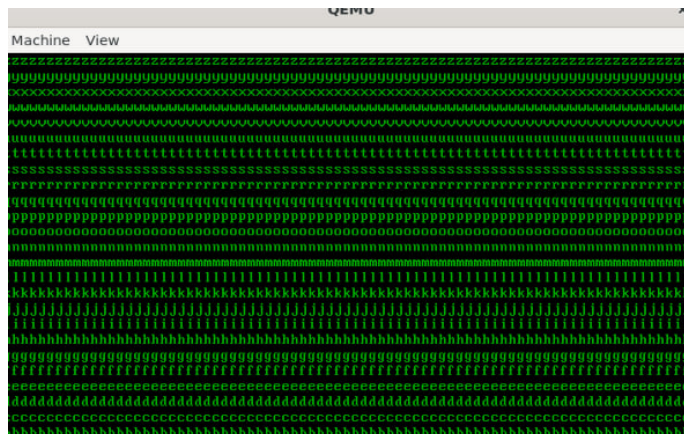


Рис. 4: Дозволена трансляція

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](file:///C:/Users/katja/Downloads/325462-sdm-vol-1-2abcd-3abcd-4%20(2).pdf). Accessed: 2024-12-07.
- [3] Wikipedia contributors. (n.d.). *NOP*. Retrieved December 7, 2024, from <https://uk.wikipedia.org/wiki/NOP>