ORGANIZACJA I ARCHITEKTURA KOMPUTERÓW

LAB 5

13 CZERWCA 2019
ŚRODA, TN 17:05
AUTOR: WOJCIECH KUR
PROWADZĄCY: DR INŻ. PIOTR PATRONIK

Spis treści

1. Treść ćwiczenia

1.1. Zakres i program ćwiczenia

1.1.1. Napisanie programu uruchamiającego dwa wątki (w obrębie tej samej przestrzeni adresowej.

1.1.2. Napisanie programu wypełniającego zerami tablicę dwuwymiarową.

a) zmierzenie czasu wypełniania po wierszach i kolumnach

1.1.3. Wczytanie ze standardowego wejścia na stos kod programu i wykonanie go.

1.2. Zrealizowane zadania

1.1.1. Stworzenie programu realizującego wypełnienie tablicy dwuwymiarowej oraz pomiar czasu wykonania wypełnienia po wierszach i kolumnach.

2. Przebieg ćwiczenia

2.1. Konstrukcja pliku źródłowego „table”

2.1.1 Listing 2: kod źródłowy table.s

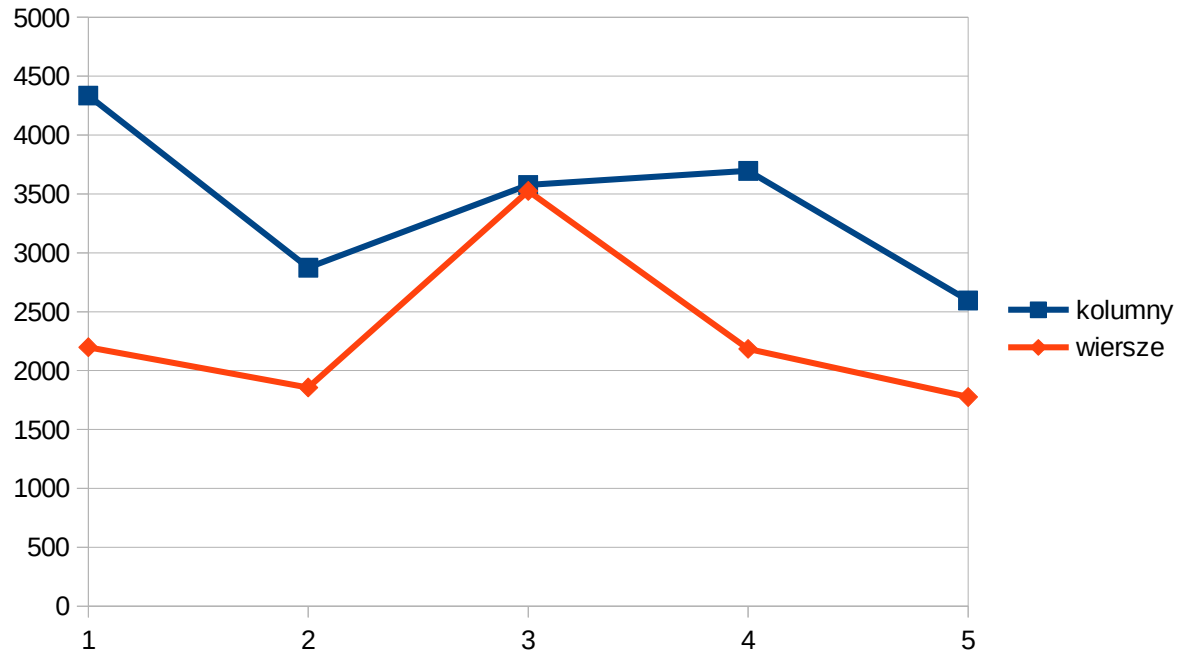
```
.data
EXIT =                  1
READ =                   3
WRITE =                  4
STDIN =                  0
STDOUT =                   1
SYSCALL =                  0x80

row: .int 0
col: .int 0

.bss
.comm array, 20*20*4
.equ j, 20
.equ i, 20

.comm rows_time_counter, 8
.comm cols_time_counter, 8

.text
.global _start
_start:
# Mierzenie wypelnienia kolumn
   rdtsc
   movl %eax,                   cols_time_counter
   mov $cols_time_counter,         %ecx
   movl %edx,                   4(%ecx)

   movl $0,                    col
   movl $0,                    row
next_col_cols:
   movl col,                   %ecx
next_row_cols:
   movl row,                    %eax
   movl $i,                    %edx
   mull %edx
   movl $4,                    %edx
   mull %edx
   movl $1,                    array(%eax, %ecx, 4)
   incl row
   movl row,                    %eax
   cmpl $j,                    %eax
   jne next_row_cols
```

```
    movl $0,                    row
    incl col
    incl                      %ecx
    cmp $i,                     %ecx
    jne next_col_cols

    movl $0,                    col

    rdtsc
    subl cols_time_counter,       %eax
    movl %eax,                  cols_time_counter
    movl $cols_time_counter,      %ecx
    subl 4(%ecx),               %edx
    movl %edx,                  4(%ecx)

# Mierzenie wypelnienia wierszy
    rdtsc
    movl %eax,                  rows_time_counter
    mov $rows_time_counter,       %ecx
    movl %edx,                  4(%ecx)

    movl $0,                    col
    movl $0,                    row
next_row_rows:
    movl row,                    %eax
    movl $i,                    %edx
    mull %edx
    movl $4,                    %edx
    mull %edx
next_col_rows:
    movl col,                   %ecx
    movl $2,                    array(%eax, %ecx, 4)
    incl col
    incl %ecx
    cmpl $i,                    %ecx
    jne next_col_rows

    movl $0,                    col
    incl row
    movl row,                    %eax
    cmp $j,                     %eax
    jne next_row_rows

    movl $0,                    row

    rdtsc
    subl rows_time_counter,       %eax
    movl %eax, rows_time_counter
    movl $rows_time_counter,      %ecx
    subl 4(%ecx),               %edx
    movl %edx,                  4(%ecx)
```

```
end:
   mov $EXIT,                %eax
   mov $0,                   %ebx
   int $SYSCALL
```

2.1.2. Opis programu

Program rozpoczyna się z zdefiniowaną wcześniej wielkością tablicy array, dla porównania czasu posiadającą tyle samo kolumn i wierszy. Najpierw zostaje zmierzony czas wypełnienia zerami kolumn. Aby to zrobić wymagane jest przesunięcie o stałą wartość (wielkość wiersza) aby przejść do następnego elementu tej samej kolumny. Iteracja przebiega aż do wypełnienia w ten sposób całej tablicy zerami. Następnie zostaje zmierzony czas wypełniania wierszy. W tym celu iterujemy po kolei, co 4 bajty.



Ilustracja 1: Wykres próby od czasu dla poszczególnych iteracji

W każdym z 5 pomiarów iteracje po wierszach okazują się szybsze. Jest to spowodowane brakiem narzutu w postaci pominięcia pewnych elementów (wybieramy co n-ty element aby iterować po kolumnie).

3. Podsumowanie

Wyniki eksperymentu związanego z wypełnianiem tablicy dwuwymiarowej były proste do przewidzenia. Iteracja element po elemencie jest szybsza niż iteracja z wykluczeniem kilku elementów (i powrót do nich w kolejnych iteracjach). Wymaganych operacji do przeprowadzenia takich iteracji jest więcej, zatem czas jest równomiernie większy.

4. Literatura

4.1. http://zak.ict.pwr.wroc.pl/materials/architektura/laboratorium%20AK2/wzorzec%20sprawozdania.pdf, wzorzec sprawozdania

4.2. http://jedrzej.ulasiewicz.staff.iiar.pwr.wroc.pl/Architektura-Komputerow/lab/Architektura-63.pdf, laboratorium architektury komputerów – materiały dr Jędrzeja Ułasiewicza

4.3. https://pl.wikibooks.org/wiki/Asembler_x86, teoria oraz prosty program krok po kroku

4.4. http://quantum-mirror.hu/mirrors/pub/gnusavannah/pgubook/ProgrammingGroundUp-1-0-booksize.pdf?fbclid=IwAR0b_yzxqe1Ib9ANvA1BX5r4fFWKh6TarAiws4QtwKpikw2nGNYaYwcYwfI, Programming from the Ground Up

4.5. http://www.cs.umd.edu/~meesh/cmsc311/links/handouts/ia32.pdf

4.6. http://zak.ict.pwr.wroc.pl/materials/architektura/laboratorium%20AK2/Dokumentacja/Intel%20Penium%20IV/IA-32%20Intel%20Architecture%20Software%20Developers%20Manual%20vol.%202%20-%20Instruction%20Set%20Reference.pdf, dokumentacja intela (Instruction Reference vol.2)