Politechnika Wrocławska

Architektura Komputerów 2 - laboratorium

sprawozdanie

Autor:
Jarosław Piszczała 209983

Prowadzący:
Prof. Janusz Biernat

Wydział Elektroniki
Informatyka
IV rok

30 listopada 2016

Spis treści

Laboratorium 6

1.1 Tematyka

Tematem laboratorium było zapoznanie się z procesorem: wyłuskiwaniem informacji na jego temat, wykorzystywaniem cykli procesora oraz weryfikowania działania pamięci cache.

1.2 Zakres prac

Zadaniem było zapoznanie się z:

1. CPUID - Wydobycie informacji na temat procesora i pamięci podręcznej.
2. RDTSC - Odczyt aktualnego licznika cykli procesora.
3. CACHE - Instrukcje z rozszerzeniem *"FENCE" a także wykonanie pętli z cyklicznym dostępem do kolejnych komórek tablicy.

1.3 Rozwiązanie

1.3.1 CPUID

CPUID pozwala na odczytanie wielu informacji na temat procesora, są to przede wszystkim informacje na temat tego jakie instrukcje on obsługuje czy jakiego rozmiaru posiada pamięć cache i tym podobne. Dostać się do tych informacji można na dwa sposoby które w sumie sprowadzają się do jednego. Pierwszym sposobem jest wprowadzenie „ID" do rejestru „EAX" i wywołanie funkcji CPUID która zwróci odpowiednie dane do rejestrów EAX-EDX.

Listing 1.1: Funkcja CPUID

```
mov $2, %eax
cpuid
```

Drugim sposobem jest wykorzystanie w C funkcji „__get_cpuid" znajdującej się w bibliotece „cpuid.h". Należy do niej wprowadzić „ID" oraz adresy do czterech zmiennych int które będą emulować nasze cztery rejestry.

Listing 1.2: Funkcja odczytująca i weryfikująca cpuid

```
#include <cpuid.h>
int main()
```

```
{
 int a,b,c,d,lvl;
 lvl = 1;
 __get_cpuid(lvl,&a,&b,&c,&d);
 printf("Floating-point unit on-Chip: %d\n", d & 0x1);
 printf("Intel Architecture MMX technology supported: %d\n", d>>23 & 0x1);
 printf("Streaming SIMD Extensions supported: %d\n", d>>25 & 0x1);
 printf("Streaming SIMD 2 Extensions supported: %d\n", d>>26 & 0x1);
 printf("CLFLUSH line size: %d\n", b>>8 & 0xF);
```

Tak jak wspomniano wcześniej, zasada działania jest tak naprawdę taka sama. W wyniku otrzymujemy ciągi 32 bitów które dla odpowiednich „ID" mają odpowiednie wartości. Dla przykładu w powyższym kodzie widać, iż dla jedynki możemy wyciągnąć informacje na temat wspieranych technologii, używanych na poprzednich laboratoriach.

1.3.2 RDTSC

Rdtsc jest funkcją która zwraca ilość cykli procesora. W ten sposób najefektywniej można porównywać czas pracy funkcji, algorytmów itp. rozwiązań. Wystarczy odczytać stan procesora przed funkcją, po funkcji i odjąć wartości aby otrzymać ilość cykli procesora potrzebną na wykonanie danego zadania. Na te potrzeby został stworzony poniższy kod asemblerowy aby otrzymywać informacje z RDTSC.

Listing 1.3: Funkcja RDTSC

```
.global _rdtsc
_rdtsc:
 pushq %rbp
 movq %rsp, %rbp

 rdtsc

 movq %rbp, %rsp
 pop %rbp
ret
```

1.3.3 Cache

Pamięć cache pozwala na szybszy dostęp do danych. Gdy tylko procesor widzi, że pracujemy na tablicy, stara się przetrzymywać ją całą w pamięci aby umożliwić jak najszybsze przeprowadzanie operacji. Aby sprawdzić czy to prawda przeprowadzono eksperyment. Do testu stworzono kilka tablic o rozmiarach od dużo mniejszych do dużo większych od rozmiaru cache a także tablice która jest minimalnie od niego większa. Następnie puszczono operacje odczytu z tablicy taką samą ilość razy dla każdej z nich

Listing 1.4: Przykładowa funkcja obliczająca ilość cykli dla odczytu danych z tablicy 4096 intów

```
int tablica[4096];
unsigned long start, stop;
```

```
int loop=100000;
start = _rdtsc();
for(i=0;i<loop;i++){
        tablica[i%4096];}
stop = _rdtsc();
printf("Time for 4096 bytes:    %lu\n",stop-start);
```

Z tego eksperymentu wynikło, że dla tablic które mieściły się w pamięci cache, czas odczytu wahał się w okolicach podobnej ilości cykli. Dla tablicy która była niewiele większa odnotowano najdłuższy czas dostępu, a dla kolejnych były to ilości pomiędzy poprzednimi wartościami.

1.4 Wnioski

Dzięki laboratorium zapoznano się z działaniem pamięci cache, sposobem testowania czasu działania algorytmów oraz odczytywania danych z procesora. Jest to przydatna wiedza, gdyż pozwoli usprawnić sposób pisania aplikacji, weryfikowania czasu ich działania. Najfajniejsza okazała się funkcja CPUID dzięki której możemy tworzyć aplikacje które same odczytają istotniejsze informacje o naszym procesorze i wykorzystają jego maksymalne możliwości.

Listings