Lista 13 (pamięć podręczna - ćwiczenia programistyczne)

$x_{13} = 4$

Wszystkie zadania należy zaprogramować w języku C i testować na komputerze klasy PC z procesorem `x86-64` lub `x86-32`. W środowisku Uniksowym informacje na temat organizacji pamięci podręcznej procesora można uzyskać wykonując w konsoli polecenie "`x86info -c`" z uprawnieniami administratora. Podobne narzędzie dla Windows można bez problemu znaleźć wpisując w wyszukiwarce "x86info".

Właściwy sposób mierzenia czasu wykonania programu polega na wielokrotnym jego uruchomieniu, wyrzuceniu skrajnych pomiarów i uśrednieniu reszty wyników.

Do listy dołączony jest szkic programu, z którego fragmentów można korzystać. Uwaga program kompiluje się wyłączenie w środowisku Uniksowym (Linux, MacOSX). Do analizy swoich programów można (opcjonalnie) wykorzystać symulator pamięci podręcznej Cachegrind.

Zadanie 1 [1pkt.]

Utwórz dwuwymiarową tablicę elementów typu `int32_t`, a następnie przejrzyj wszystkie jej elementy osobno wierszami i kolumnami. Wysokość i szerokość tablicy oraz kierunek przechodzenia należy wczytać z linii poleceń (np.: "h 1024 4096"). Rozmiary mogą być wyłącznie potęgą dwójki. Parametry należy dobrać tak by wielkość tablicy była taka sama jak pamięci podręcznej Twojego procesora lub większa. Porównaj czasy wykonania programu dla różnych parametrów. Czy jesteś w stanie uzasadnić te wyniki. Zaobserwuj, że kierunek przechodzenia ma znaczenie. Dlaczego?

Zadanie 2 [3/2/2/2 pkt.]

Wprowadźmy następujące oznaczenia dla pamięci podręcznych w procesorach Intela:

- L1d → pamięć podręczną danych poziomu pierwszego,
- L2 → zunifikowana pamięć podręczną drugiego poziomu,
- TLB1d → bufor translacji adresów pierwszego poziomu dla stron z danymi,
- TLB2 → zunifikowany bufor translacji adresów drugiego poziomu.

Znajdź metodę, która na podstawie czasu wykonania pewnych programów wykazuje jaka jest:

- [3 pkt.] długość linii pamięci podręcznej,
- [2 pkt.] wielkość pamięci L1d i L2,
- [2 pkt.] wielodrożność pamięci L1d i L2,
- [2 pkt.] wielkość TLB1D i TLB2

… dla Twojego procesora.

Oczekujemy, że studenci w trakcie prezentacji rozwiązania będą w stanie sprawnie uzasadnić w jaki sposób na podstawie wyników doszli do wniosków.

Do napisania programów skorzystaj z niżej podanych narzędzi (są zaimplementowane).

Wędrowanie po tablicy (`array_walker`)

Tworzymy jednowymiarową tablicę "`array`" 32-bitowych słów (`int32_t`) o `N` elementach. Tablica zaczyna się pod adresami podzielnymi przez rozmiar strony i ma rozmiar wielokrotności

rozmiaru strony. Mamy zbiór S wszystkich indeksów tej tablicy. Chcemy wygenerować pewne szczególne permutacje podzbioru S, tj. ciągi niepowtarzających się indeksów ($i_1, i_2, \ldots, i_l$). Ciągi te będziemy reprezentować w naszej tablicy w następujący sposób: $T[0] = i_1$, $T[i_k] = i_{k+1}$, $T[i_l] < 0$. Po zakodowaniu permutacji odpalamy procedurę `array_walker`. Będzie ona przechodziła po kolejnych elementach ciągu generując odczyty pamięci (i potencjalnie chybienia). Procedura zakończy się po osiągnięciu ostatniego elementu ciągu lub po przejrzeniu N elementów, gdyby permutacja była cyklem.

Idea jest taka, że odpowiednia permutacja będzie nam generować chybienia, które będą zaburzać czas wykonania programu. Pamiętaj, że czas odczytu z pamięci niższego poziomu dominuje nad czasem dostępu do poszczególnych słów bloku.

Zatruwanie pamięci podręcznej (`cache_poison`)

W niektórych przypadkach należy zapełnić pamięć podręczną pewnymi danymi w taki sposób, aby każdy kolejny odczyt innych danych generował chybienie. Dzięki temu, w pewnym sensie, "opróżniamy" pamięć podręczną. Określa się to mianem "zatruwania pamięci podręcznej". Zadanie to realizuje procedura `poison_cache`.

Pobieranie czasu (timer_*)

Należy mierzyć czas z dokładnością do mikrosekund. Służy do tego zestaw procedur `timer_init`, `timer_start`, `timer_stop`, `timer_print`.