Pomiar 1

Profilowany program: Program wyliczający 100000-ny wyraz ciągu Fibonacciego metodą iteracyjną. W obliczeniach zostały wykorzystane wyłącznie zmienne typu SignedMangitude, będące częścią implementowanej biblioteki.

Cel pomiaru: Sprofilowanie operacji dodawania zmiennych SignedMagnitude.

Metoda pomiarowa:
Pomiary przeprowadzono za pomocą narzędzia profilującego Intel VTune. Użyto przy tym opcji Michroarchitecture Exploration, korzystającej ze wsparcia sterowników sprzętowych przy zbieraniu danych. Z uzyskanych dla programu wyników wyfiltrowano wyłącznie moduły, wykonujące kod z implementowanej biblioteki.

Wynik pomiaru:

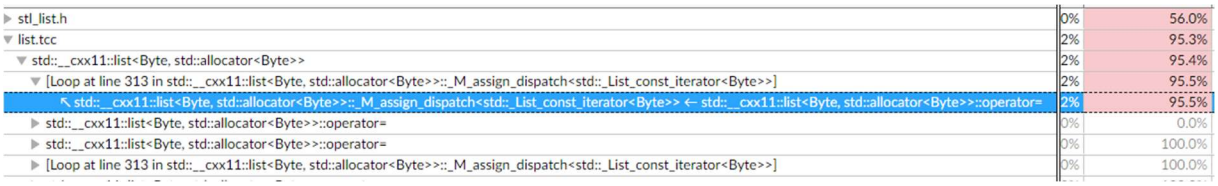
- Microarchitecture Usage: 45.2% of Pipeline Slots
  - Retiring: 45.2%
    - Light Operations: 26.5%
    - Heavy Operations: 18.7%
      - Few Uops Instructions: 18.6%
      - Microcode Sequencer: 0.1%
  - Front-End Bound: 1.6%
  - Bad Speculation: 0.0%
  - Back-End Bound: 63.2%
    - Memory Bound: 46.1%
      - L1 Bound: 7.3%
      - L2 Bound: 15.9%
      - L3 Bound: 31.3%
        - Contested Accesses: 0.1%
        - Data Sharing: 0.0%
        - L3 Latency: 100.0%
        - SQ Full: 0.0%
      - DRAM Bound: 0.5%
      - Store Bound: 0.0%
    - Core Bound: 17.0%

Analiza wyników metodą top-down:
Zgodnie z metodą analizy wyników profilowania top-down można zauważyć, że większości (64.8%) wypadków potok przetwarzający instrukcje pochodzące z biblioteki jest wstrzymany (stalled). Jedynie w 45.2% wypadków potok działa niewstrzymany (not stalled).

W większości wypadków (63.2%) wstrzymanie potoku nastąpiło przez zdarzenia w module back-end procesora. Zatem nowe instrukcje zostały pobrane, zdekodowane i przygotowane do wykonania, ale nie zostały wykonane ze względu na wyczerpanie się zasobów części wykonawczej.

Schodząc głębiej można zauważyć, że aż 46.1% wypadków wstrzymania potoku wynikło z powodów związanych z użyciem systemu pamięci. Najwięcej wąskich gardeł powoduje użycie pamięci podręcznej poziomu 3, gdzie problemem jest jej opóźnienie (udało się uzyskać trafienie w L3).

Wyszukanie wąskiego gardła wydedukowanego podczas analizy:
Za pomocą widoku bottom-up narzędzia Intel VTune wyszukano tę funkcję wywoływaną w trakcie działania biblioteki, która powoduje najwięcej przestojów back-end bound. Okazała się być to metoda alokująca listę obiektów *Byte*, wywoływana w momencie użycia operatora przypisania '='.



Metoda ta znajduje się w pliku list.tcc, za którego kod źródłowy nie byliśmy odpowiedzialni. W celu zoptymalizowania tego wąskiego gardła należy zatem zlokalizować wystąpienia operatora przypisania list obiektów typy *Byte* w kodzie oraz przemyśleć, czy jego użycie w danym miejscu jest rzeczywiście niezbędne.