Технології графічного процесінгу & розподілених обчислень

Лекція 4: Моделі паралелізму

Кочура Юрій Петрович
iuriy.kochura@gmail.com
@y_kochura

Сьогодні

Мета — дізнатися, як ядро CUDA використовує ресурси апаратного забезпечення

- Основні причини використання паралельного програмування
- Таксономія Флінна
- Призначення блоків потоків на ресурси виконання
- Обмеження потужності ресурсів виконання
- Планування потоків без накладних витрат

Основні причини використання паралельного програмування

ЕКОНОМІЯ ЧАСУ ТА/АБО ГРОШЕЙ

ВИРІШЕННЯ БІЛЬШИХ / СКЛАДНІШИХ ЗАДАЧ

1. Багато задач настільки великі та/або складні, що їх недоцільно або неможливо розв’язати за допомогою послідовної програми, особливо враховуючи обмежену пам’ять комп’ютера.

- Приклад: Грандіозні завдання - Grand Challenges, які потребують петафлопси і петабайти обчислювальних ресурсів.
- Приклад: веб-пошукові системи/бази даних обробляють мільйони транзакцій щосекунди.

КОНКУРЕНТНІСТЬ

- Один обчислювальний ресурс може виконувати лише одну дію одночасно. Кілька обчислювальних ресурсів можуть виконувати багато дій одночасно.

ВІДДАЛЕНІ РЕСУРСИ

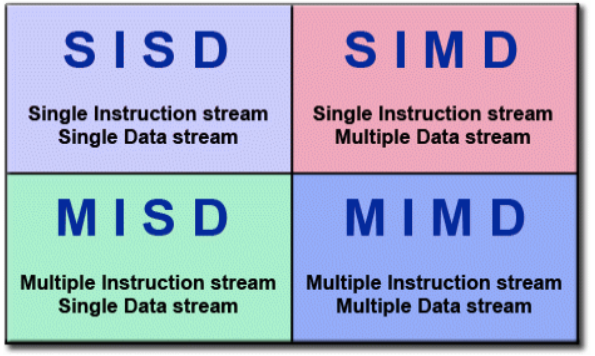
- Використання обчислювальних ресурсів у мережі або навіть Інтернеті, коли локальних обчислювальних ресурсів недостатньо.

ВИКОРИСТАННЯ

- Сучасні комп’ютери, навіть ноутбуки, мають паралельну архітектуру апаратного забезпечення з кількома процесорами/ядрами.
- Паралельне програмне забезпечення спеціально призначене для паралельного апаратного забезпечення з кількома ядрами, потоками тощо.
- У більшості випадків послідовні програми, що виконуються на сучасних комп'ютерах, «марно витрачають» потенційну обчислювальну потужність.

Таксономія Флінна

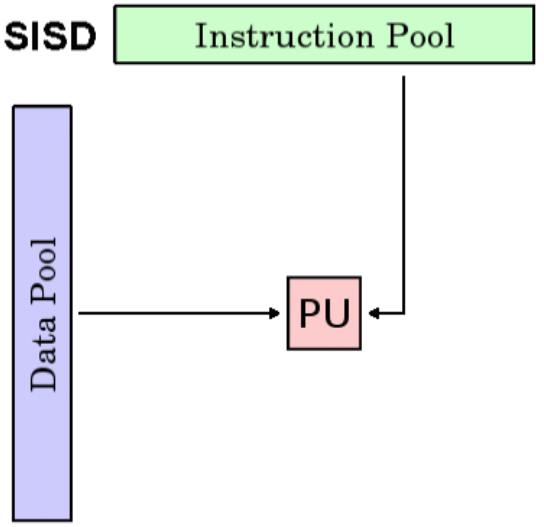
Таксономія Флінна

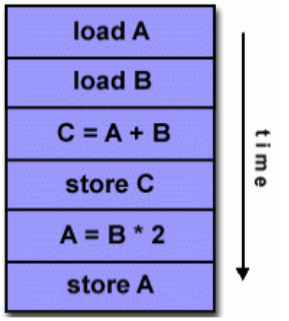


- Існує кілька різних способів класифікації паралельних комп’ютерів
- Одна з найбільш поширених класифікацій, яка використовується з 1966 року, називається таксономією Флінна
- Таксономія Флінна розрізняє багатопроцесорні комп'ютерні архітектури відповідно до того, як їх можна класифікувати за двома незалежними вимірами: потоку команд (Instruction Stream) та потоку даних (Data Stream). Кожен із цих вимірів може мати лише один із двох можливих станів: одиничний (Single) або множинний (Multiple)

—
Джерело слайду: Lawrence Livermore National Laboratory

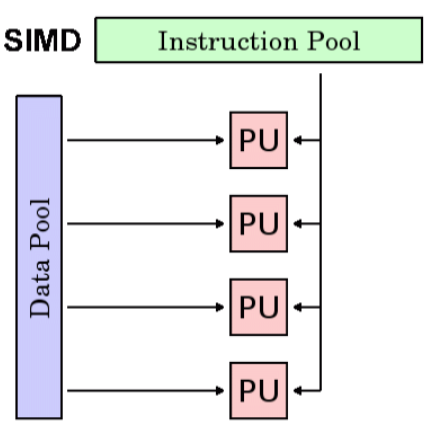
Single Instruction, Single Data (SISD)

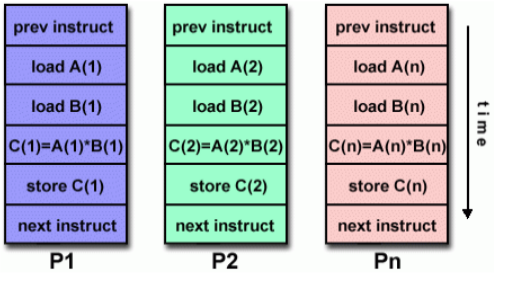


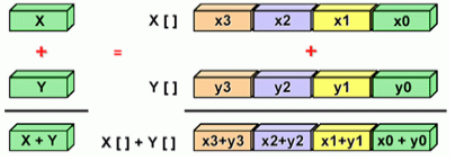


—
Джерело слайду: Lawrence Livermore National Laboratory

Single Instruction, Multiple Data (SIMD)

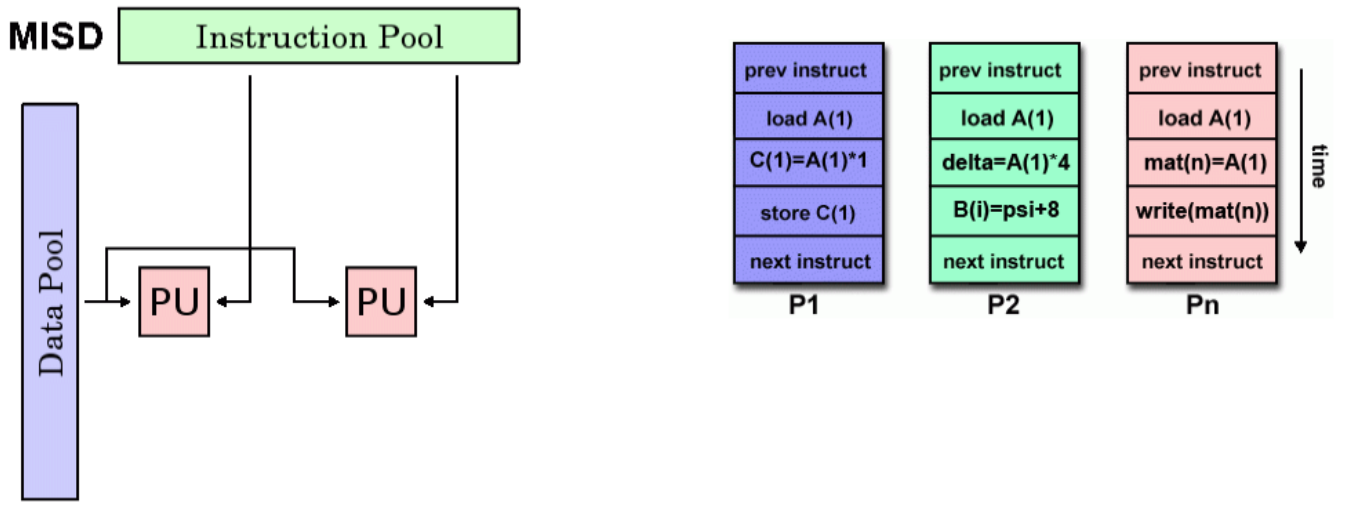






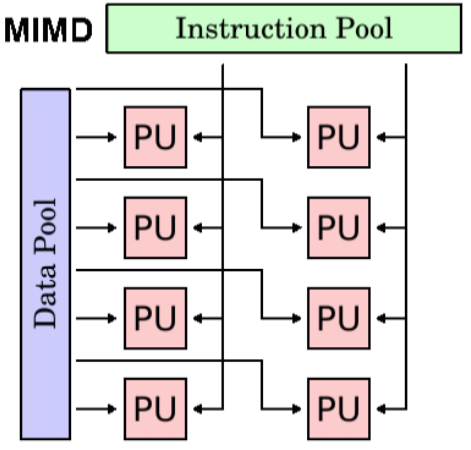
Джерело слайду: Lawrence Livermore National Laboratory

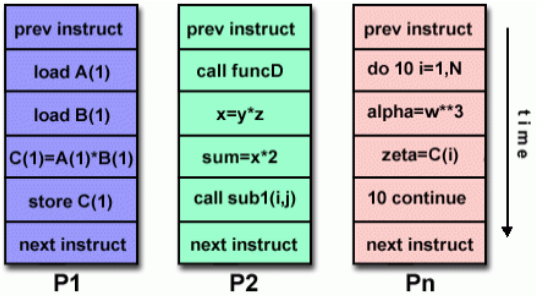
Multiple Instruction, Single Data (MISD)



Джерело слайду: Lawrence Livermore National Laboratory

Multiple Instruction, Multiple Data (MIMD)



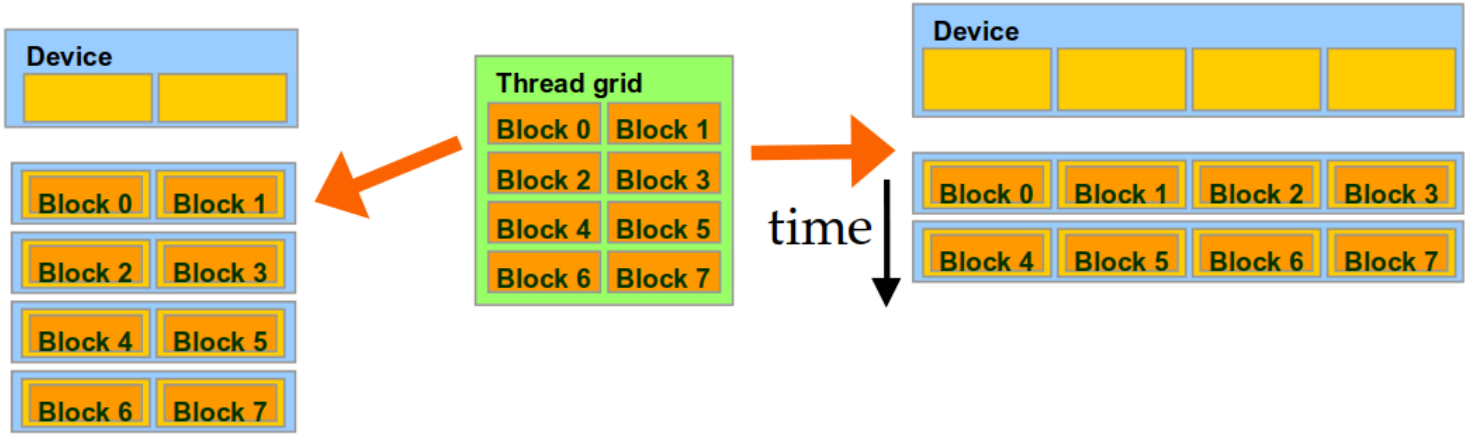


—

Джерело слайду: Lawrence Livermore National Laboratory

Призначення блоків потоків на ресурси виконання

Прозора масштабованість

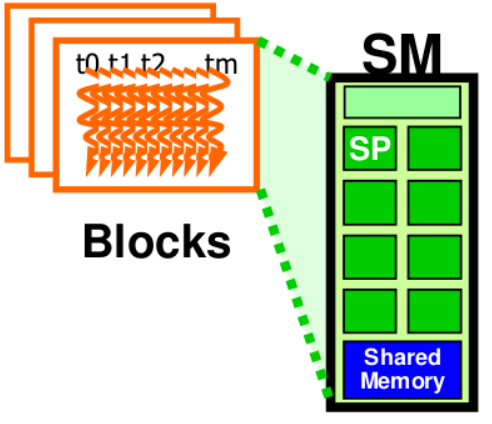


- Блоки виконуються в довільному порядку
- Пристрій може в будь-який час призначити блок будь-якому потоковому процесору
- Ядро масштабується на будь-яку кількість паралельних процесорів

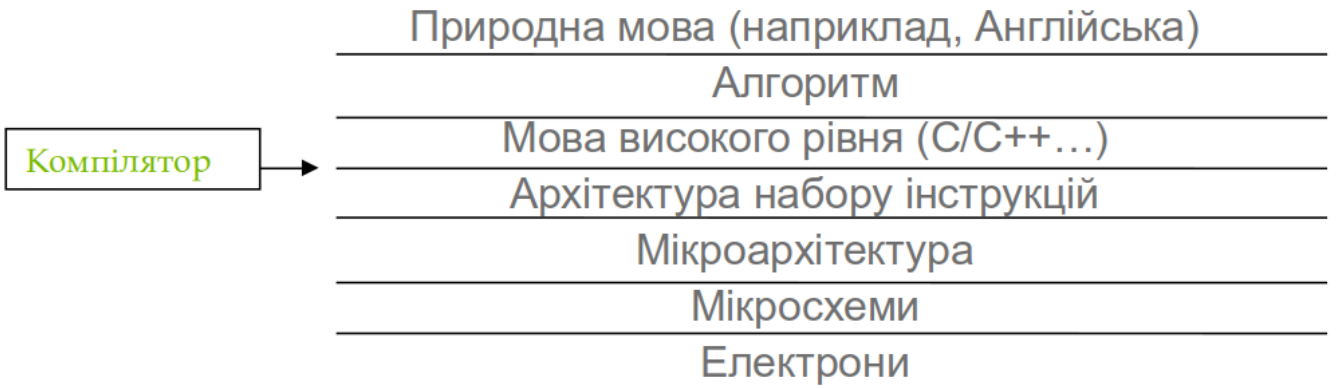
—
Джерело слайду: NVIDIA, DLI

Приклад: Виконання блоку потоків

- Потоки утворюють блоки, які передаються на виконання Потоковим мультипроцесорам - Streaming Multiprocessors (SM)
- Кожен мультипроцесор може обробити до 32 блоків, якщо дозволяє ресурс
- В архітектурі Volta мультипроцесор може взяти до 2048 потоків, або
  - 256 (потоків на блок) * 8 блоків
  - 512 (потоків на блок) * 4 блоки, і т.д.
- Доступ до індексів потоків та блоків забезпечує SM
- SM управляє/планує виконання потоків



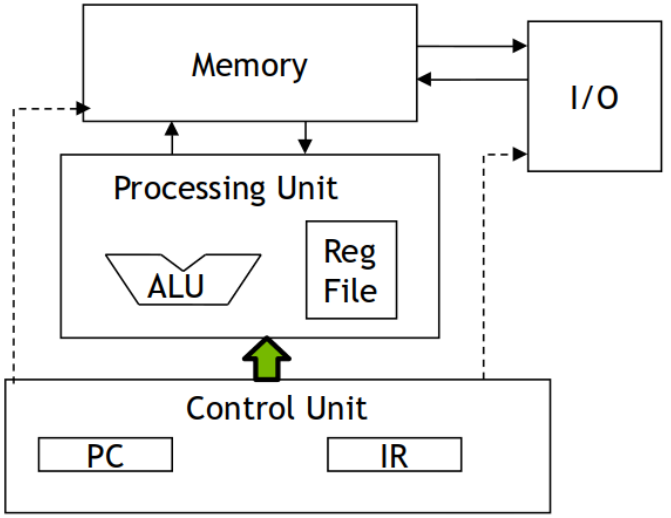
Від природної мови до електронів



©Yale Patt and Sanjay Patel, *From bits and bytes to gates and beyond*

—
Джерело слайду: NVIDIA, DLI

Модель фон Неймана

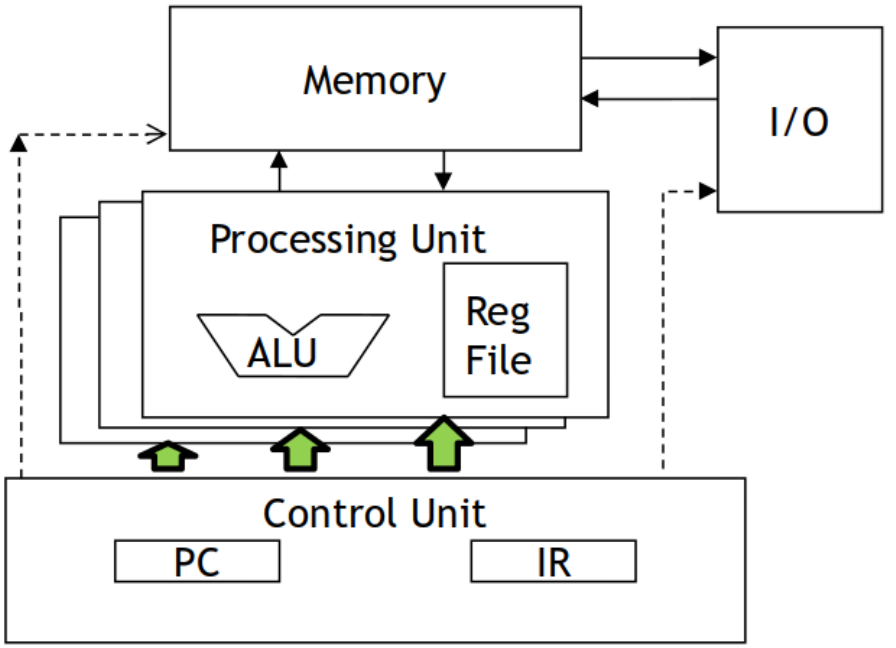


Потік — це "віртуальний" або "абстрактний" процесор фон Неймана

Джон фон Нейман

—
Джерело слайду: NVIDIA, DLI

Модель фон Неймана з **SIMD** елементами



—
Джерело слайду: NVIDIA, DLI

Обмеження потужності ресурсів виконання

Варпи як одиниці виконання

Кожен блок виконується у варпі на 32 потоки

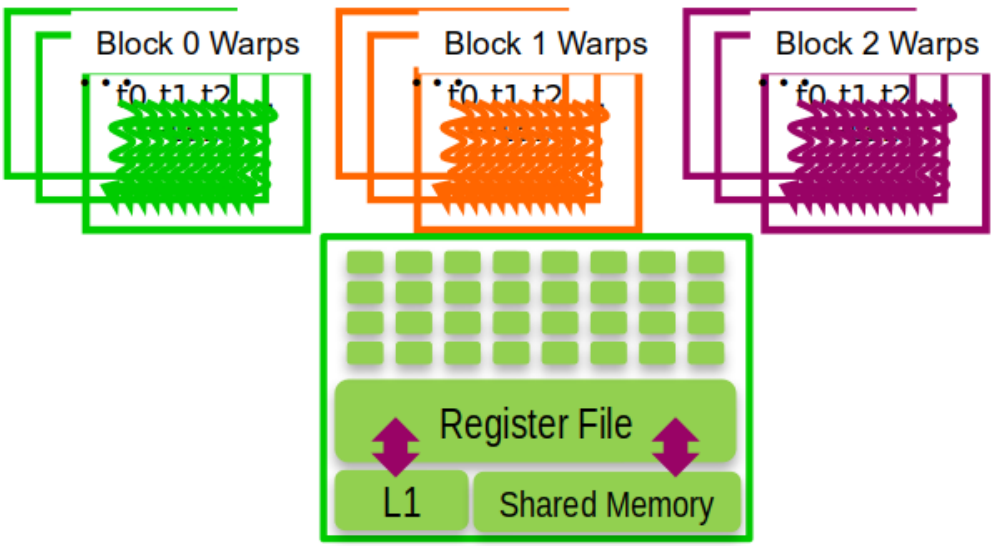
- Реалізація не є частиною програмної моделі CUDA
- Варпи є одиницями виконання в багатопоточних мультипроцесорах
- Потоки у варпі виконуються за моделлю SIMD
- Майбутні версії GPU можуть мати іншу кількість потоків у варпі

—
Джерело слайду: NVIDIA, DLI

Приклад роботи варпа

Якщо 3 блоки по 256 потоки кожен призначено на мультипроцесор, скільки варпів буде відправлено на SM?

- Кількість варпів в одному блоці: $256/32 = 8$
- Загальна кількість варпів: $3 * 8 = 24$



—

Джерело слайду: NVIDIA, DLI

Планування потоків без накладних витрат

Планування потоків

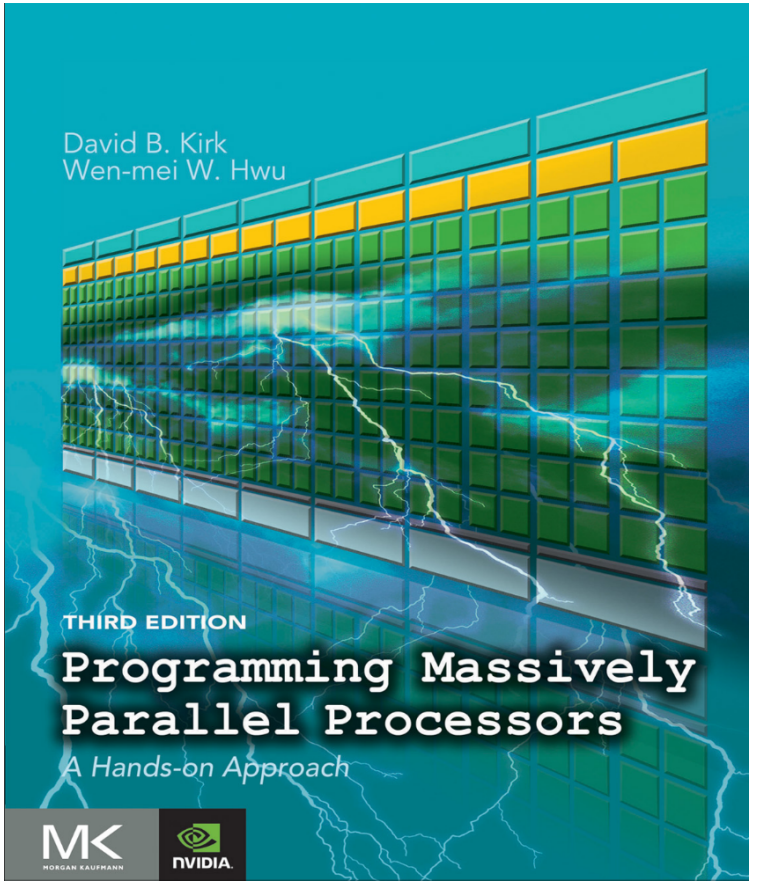
SM виконує планування потоків без накладних витрат

- Варпи, операнди наступних інструкцій яких вже обчислені, стають претендентами на виконання
- Варпи, що претендують на виконання, відбираються на виконання, дотримуючись обраної політики пріоритетів
- Усі потоки у варпі виконують одні й тій ж самі інструкції

—
Джерело слайду: NVIDIA, DLI

Література

PMPP Ch. 3, pp. 43-69



Кінець 🏁