Wstęp.

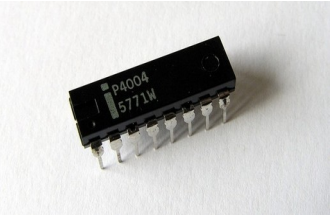
Procesy i wątki.

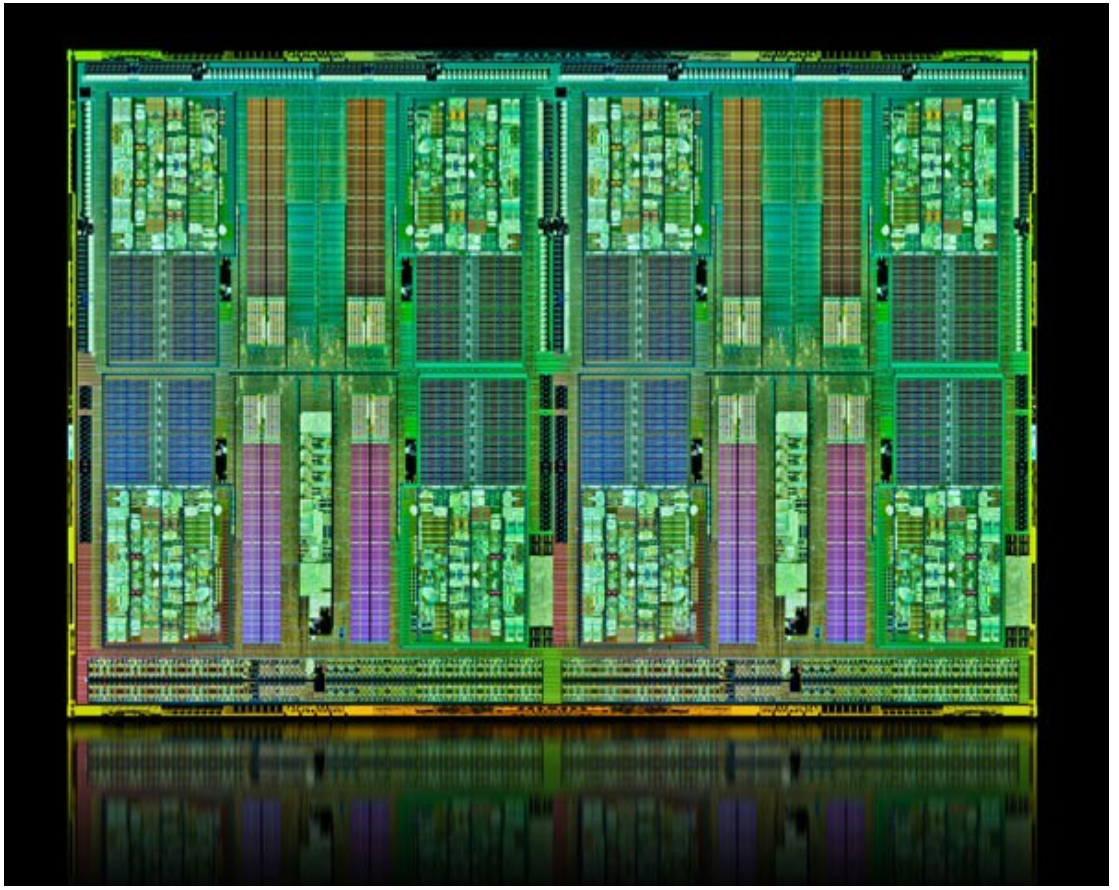
Cel zajęć

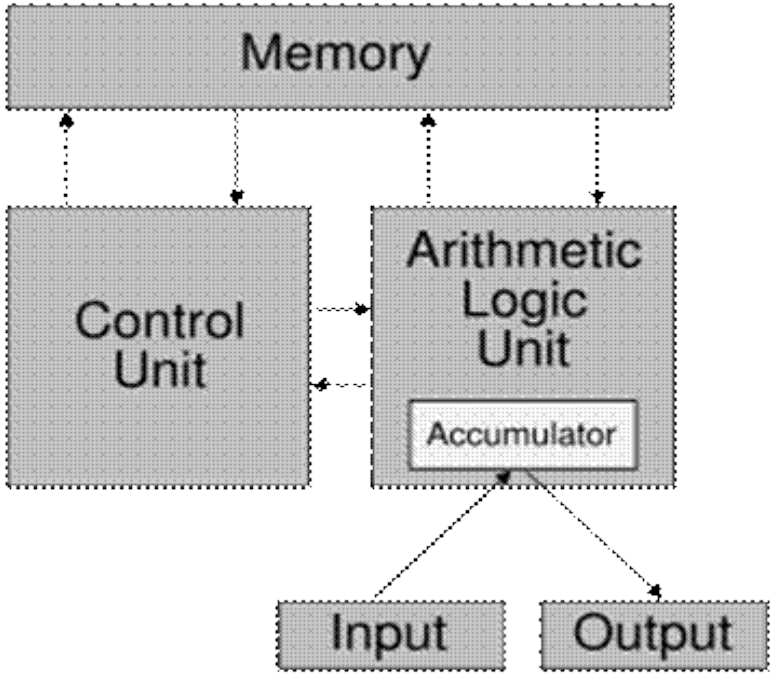
- Zapoznanie z technikami i narzędziami programistycznymi służącymi do tworzenia programów równoległych
- Przedstawienie sprzętu wykorzystywanego do obliczeń równoległych
- Nauczenie sposobów tworzenia i implementacji algorytmów równoległych
- Zapoznanie z technikami analizy programów równoległych oraz rozwiązywania pojawiających się w nich problemów programistycznych
- Przedstawienie wybranych dziedzin zastosowań przetwarzania równoległego oraz występujących tam algorytmów

Po co obliczenia równoległe ?

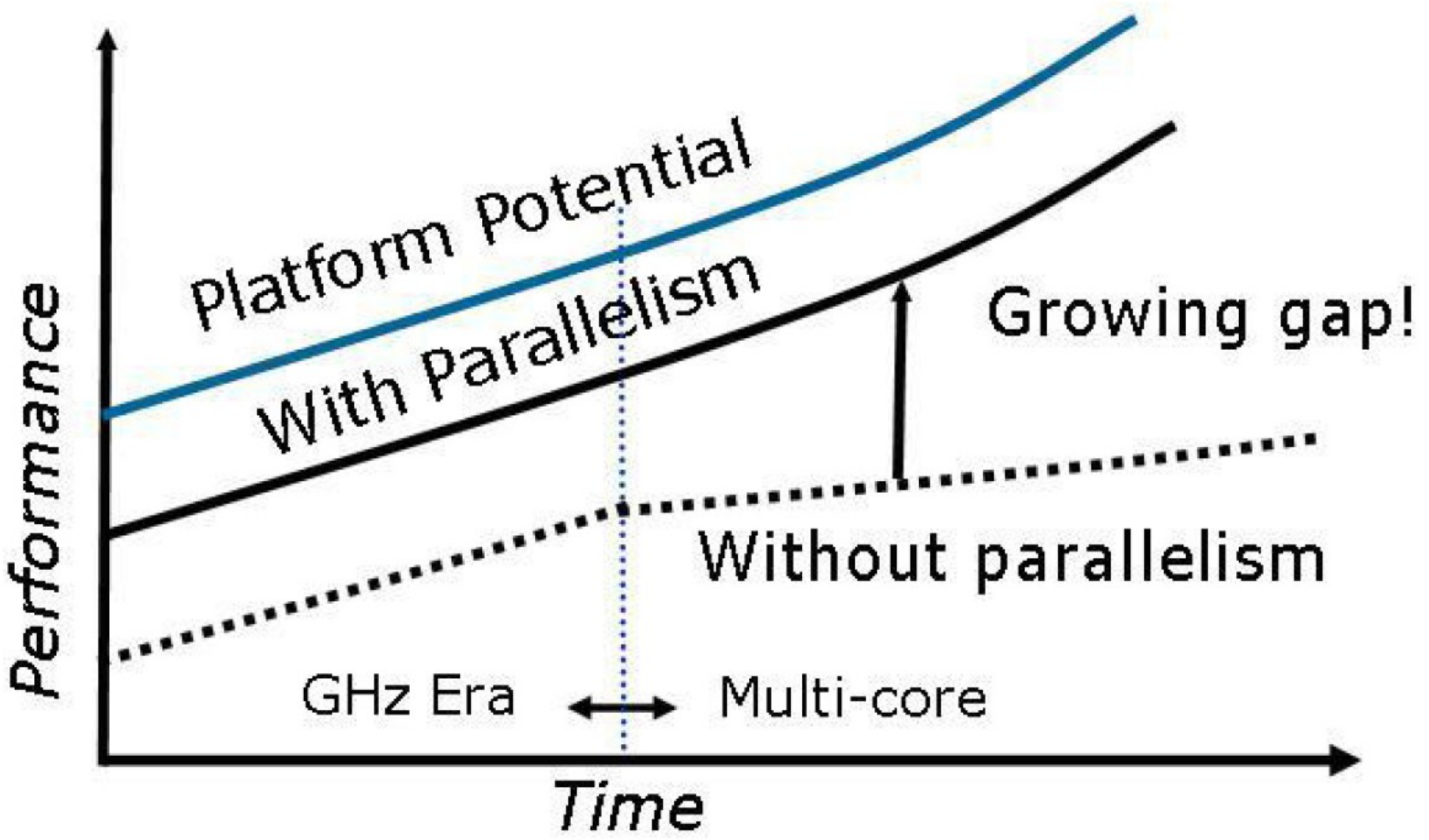
- Lepsze wykorzystanie dostępnego sprzętu





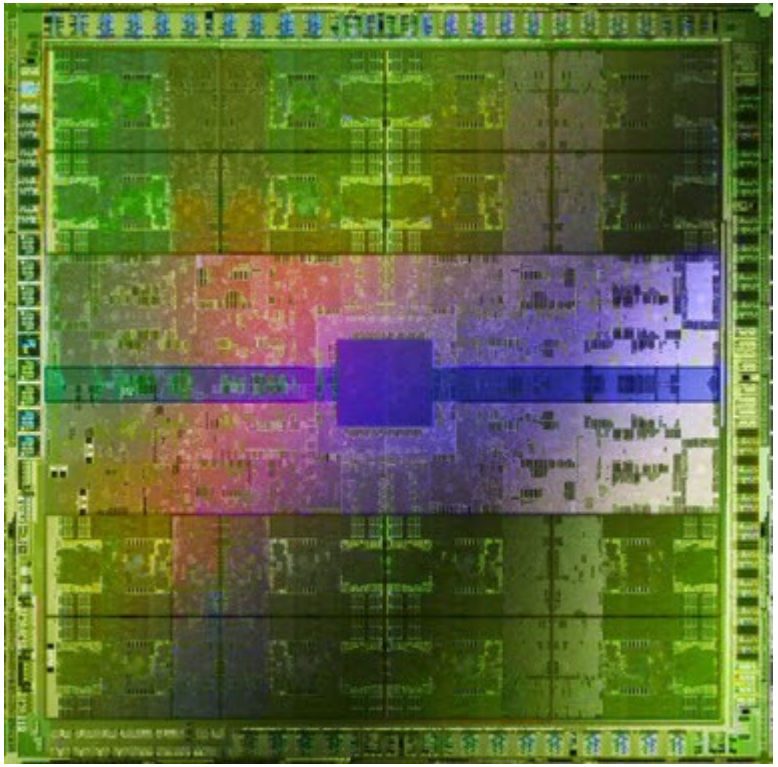


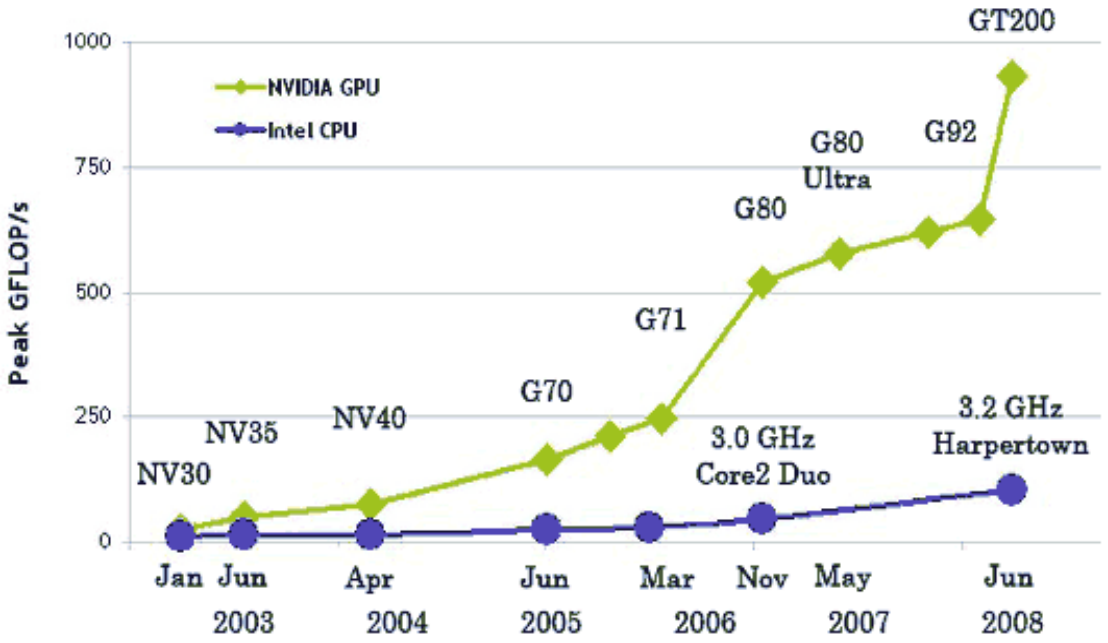
Motywacja


Platform Potential
With Parallelism
Growing gap!
Without parallelism
GHz Era
Multi-core
Performance
Time

Po co obliczenia równoległe ?

- Zwiększenie maksymalnej mocy obliczeniowej





Historia i pojęcia wstępne

- Obliczenia równoległe:
  - dwa lub więcej procesów (wątków) jednocześnie współpracuje (komunikując się wzajemnie) w celu rozwiązania pojedynczego zadania (najczęściej z określonej dziedziny zastosowań)
  - rozwój związany z popularyzacją (szczególnie od lat osiemdziesiątych XX-wieku) komputerów równoległych
  - problemy obliczeń równoległych (poza klasycznymi zagadnieniami współbieżności) są najczęściej związane z konkretnymi algorytmami
  - obliczenia równoległe były silnie związane z dziedziną obliczeń wysokiej wydajności (i obliczeniami naukowo-technicznymi)
  - dziś, w czasach procesorów wielordzeniowych,

    programowanie równoległe jest koniecznością - przetwarzanie równoległe jest jedynym sposobem wykorzystania pełnej mocy współczesnego sprzętu

Klasyfikacja Flynna

- Zwielokrotnienie strumieni przetwarzania rozkazów i danych
  - klasyfikacja Flynna
    - SISD (oryginalna architektura von Neumanna)
    - SIMD – jeden strumień rozkazów i wiele strumieni danych
    - MISD – wiele strumieni rozkazów i jeden strumień danych
      - wykonywanie wielu rozkazów w tej samej chwili na jednym egzemplarzu danych jest trudne do wyobrażenia
      - najbliższe modelowi MISD jest przetwarzanie potokowe, gdzie różne rozkazy są wykonywane przez różne procesory/rdzenie na tym samym egzemplarzu danych, ale w kolejnych chwilach czasu
    - MIMD – wiele strumieni danych i wiele strumieni rozkazów
  - współczesne systemy komputerowe są złożone, realizują zazwyczaj różne typy przetwarzania, z pojedynczymi elementami odpowiadającymi architekturom SISD, SIMD oraz całością odpowiadającą architekturze MIMD – z pamięcią wspólna lub rozproszoną

Rodzaje przetwarzania równoległego

- Równoległość na poziomie:
  - wykonania pojedynczego wątku
    - użycie rozkazów wektorowych – najczęściej realizowane przez kompilatory
    - wykorzystanie możliwości sprzętowych – tzw. równoległość na poziomie pojedynczego rozkazu (*instruction level parallelism - ILP)* – potokowość, superskalarność
  - zadań obliczeniowych (*task level parallelism)*, pętli *(loop level parallelism)* – wykorzystanie wątków (*thread level parallelism)* i procesów
    - rozdział wykonywanych rozkazów (grup rozkazów – zadań lub iteracji pętli) pomiędzy wątki/ procesy, inaczej: podział zadania obliczeniowego na podzadania i przydział poszczególnych podzadań procesom/wątkom
    - przydział procesów/wątków procesorom/rdzeniom
  - programów (*job level parallelism)* – wykonywanie niezależnych programów przez system operacyjny (*OS level parallelism)*, zarządzanie wykonywaniem wielu programów równoległych przez systemy równoważenia obciążenia w klastrach

Narzędzia programowania

- Praktyczne warianty implementacji równoległej:
  - programowanie sekwencyjne ze zrównolegleniem niejawnym:
    - poprzez układ procesora superskalarnego
    - poprzez automatyczny kompilator zrównoleglający
  - programowanie w językach równoległości danych
  - programowanie w modelu z pamięcią wspólną
  - programowanie w modelu z przesyłaniem komunikatów
  - programowanie w modelu przerzucenia (*offload*) części obliczeń na układ wspomagający (koprocesor, akcelerator)
  - programowanie w rozmaitych modelach hybrydowych (łączących cechy modeli powyższych)

Historia i pojęcia wstępne

- Przetwarzanie współbieżne
  - realizacja wielu programów (procesów) w taki sposób, że ich trwanie od momentu rozpoczęcia do momentu zakończenia może się na siebie nakładać
  - współbieżność pojawiła się wraz z wielozadaniowymi systemami operacyjnymi (lata 60-te, Multics – 1965) i nie wymusza równoległości
  - współbieżność związana jest z szeregiem problemów teoretycznych wynikłych z prób realizacji wielozadaniowych systemów operacyjnych
  - istnieje wiele mechanizmów niskiego poziomu (systemowych) do rozwiązywania problemów współbieżności

Procesy

- Proces (jednowątkowy / wielowątkowy):
  - ciąg rozkazów (wątek główny) / ciągi rozkazów (wątek główny i inne wątki)
  - stos (wątku głównego) / odrębne stosy dla każdego wątku
  - przestrzeń adresowa
  - dodatkowe elementy tworzące m.in. kontekst procesu
- Tworzenie procesu:
  - pid_t fork(void)
  - int execv(const char *filename, char *const argv [])
  - pid_t wait(pid_t pid, int *status)

Tworzenie procesów

```
main(){
    pid = fork();
    if(pid==0){
        wyn = execl(.....);        // np. execl(„/usr/bin/ls”, (char *) 0);
        if(wyn==....) ....;
    } else {
        wyn = wait(&stan);
        (if stan==....)....;
    }
}
```

- uwaga: zachowanie zmiennych (w tym globalnych)

Wątki

- Wątek
  - ciąg rozkazów
  - stos
  - niektóre z elementów tworzących kontekst procesu
- Tworzenie wątku (Linux):
  - int clone(int (*fn)(void *), void *child_stack, int flags, void *arg)
- Przyczyna wprowadzenia wątków: łatwość komunikacji przy użyciu pamięci wspólnej i wydajność przełączania kontekstu