**1. Opisać cechy rolę i znaczenie w masywnych obliczeniach równoległych: Rejestrów wektorowych w architekturze procesora i obliczeń wektorowych.**

Rejestry procesora – komórki pamięci o niewielkich rozmiarach (najczęściej 4/8/16/32/64/128 bitów) umieszczone wewnątrz procesora i służące do przechowywania tymczasowych wyników obliczeń, adresów lokacji w pamięci operacyjnej itd.

Rejestry procesora stanowią najwyższy szczebel w hierarchii pamięci, będąc najszybszym z rodzajów pamięci komputera, zarazem najdroższą w produkcji, a co za tym idzie – o najmniejszej pojemności.

Rejestry wektorowe – przechowują dane do jednoczesnego przetwarzania wielu danych przez instrukcje typu SIMD.

SIMD (ang. Single Instruction, Multiple Data) – jeden z podstawowych rodzajów architektur komputerowych według taksonomii Flynna, obejmujący systemy, w których przetwarzanych jest wiele strumieni danych w oparciu o pojedynczy strumień rozkazów. Architektura SIMD jest charakterystyczna dla komputerów wektorowych.

Obecnie jednostki realizujące zadania zgodnie z metodologią SIMD obecne są także w stosowanych w domowych komputerach procesorach opartych na architekturze x86 (i386) oraz x64 (amd64). Procesory oferują listę rozkazów poszerzoną o zestawy rozkazów typu SIMD, takie jak: MMX, 3DNow!, SSE, SSE2, SSE3, SSSE3, SSE4, SSE5, AVX, AltiVec.

Główne korzyści wynikające z architektury SIMD osiągają aplikacje, w których te same operacje arytmetyczne lub logiczne, są wykonywane na dużych zbiorach danych. Architektura SIMD na dwa sposoby przyspiesza ten proces:

* Po pierwsze, odczyt i zapis danych do pamięci odbywa się blokowo i określona liczba danych zostaje odczytana/zapisana w jednej operacji
* Po drugie, określona operacja arytmetyczna lub logiczna jest jednocześnie wykonywana na całym bloku danych, podczas wykonania jednej instrukcji

**2. Opisać cechy, rolę i znaczenie w masowych obliczeniach równoległych: hierarchii i organizacji pamięci, w szczególności pamięci podręcznej**

**1. Rejestry procesora, które znajdują się w jego wnętrzu i przechowują argumenty operacji wykonywanych przez procesor.** Jest to pamięć statyczna, złożona z przerzutników bistabilnych, ulotna, gdyż jej zawartość znika po wyłączeniu zasilania procesora. Pamięć ta ma bardzo małą pojemność, typowo obejmuje kilkanaście rejestrów 16, 32 lub 64 bitowych procesora. Natomiast charakteryzuje się bardzo dużą szybkością. Dostęp do niej odbywa się przez bramki sterowane z Jednostki Sterującej (CU) procesora i czas dostępu wynosi ułamki nanosekundy. Jest to najszybsza pamięć w systemie komputerowym.

**2. Pamięć podręczna procesora (**[**ang.**](https://pl.wikipedia.org/wiki/Język_angielski) ***CPU cache*)** - jest pamięcią typu [SRAM](https://pl.wikipedia.org/wiki/SRAM_(pamięci_komputerowe)) (pamięć statyczna) o krótkim czasie dostępu. Zlokalizowana jest często bezpośrednio w jądrze procesora. Zastosowanie wielopoziomowej hierarchii pamięci podręcznej pozwala, korzystając z zasady lokalności przestrzennej i czasowej na zapewnienie złudzenia posiadania szybkiej i pojemnej pamięci głównej, a więc zmniejsza średni czas dostępu do pamięci głównej.

**3. Pamięć operacyjna - Pamięć główna RAM** przechowuje dane i kody programów. Jest to ulotna pamięć dynamiczna o dostępie swobodnym (DRAM). We współczesnych systemach stanowi ona przede wszystkim ogniwo pośrednie pomiędzy informacją zapisaną na dysku a procesorem i pamięciami podręcznymi. Im więcej danych i programów aktualnie wykonywanych przechowywane jest w pamięci RAM, a nie na dysku, tym system działa szybciej. W szczególności w pamięci operacyjnej rezerwuje się specjalny bufor zwany dyskową pamięcią podręczną służący do zbierania danych, które mają być zapisane na dysk lub są często pobierane z dysku.

**4. Pamięć zewnętrzna (en. secondary storage)** – [pamięć komputerowa](https://pl.wikipedia.org/wiki/Pamięć_komputerowa), która nie jest bezpośrednio dostępna przez procesor. Jest dostępna jako [urządzenie zewnętrzne](https://pl.wikipedia.org/wiki/Urządzenie_peryferyjne), które przechowuje dane. Z tego powodu tego typu pamięci komputerowe są wolniejsze od [pamięci operacyjnej](https://pl.wikipedia.org/wiki/Pamięć_operacyjna). (CD,DVD,Pamięć Flash,Dyskietki,HDD,SDD)

**Pamięć podręczna pierwszego poziomu (L1)** (ang. Primary cache memory) ma pojemność kilkadziesiąt KB (np. Pentium III - 32KB, AMD Athlon - 128KB, Pentium 4 - 20KB) i służy do przechowywania pewnej ilości danych z pamięci operacyjnej najczęściej używanych przez procesor. Wspomaga ona komunikację procesora z pamięcią, zapewniając bardzo mały czas dostępu (rzędu nanosekund) do danych, które akurat znajdują się w tej pamięci. W tym przypadku odczyt i zapis możliwy jest z szybkością zegara procesora, bez stanów oczekiwania. Zmniejsza to wydatnie częstość występowania stanów oczekiwania. Pamięć L1 jest asocjacyjną pamięcią SRAM zintegrowaną z procesorem.

**Pamięć podręczna drugiego poziomu (L2)** (ang. Secondary cache memory) jest to pamięć RAM o pojemności 256KB lub 512KB obecnie również zintegrowana z procesorem. Służy do wspomagania pracy pamięci podręcznej pierwszego poziomu, przechowując najczęściej używane fragmenty zawartości pamięci operacyjnej, w tym zawartość pamięci podręcznej pierwszego poziomu. Gdy procesor nie znajduje danych w pamięci podręcznej pierwszego poziomu odwołuje się do pamięci podręcznej drugiego poziomu, w tzw. trybie seryjnym (ang. burst mode) po magistrali lokalnej. Stany oczekiwania są generowane na tej magistrali tylko w przypadkach nie znalezienia danych w pamięci podręcznej i konieczności odwołania się do pamięci głównej. Zmniejsza to jeszcze bardziej częstość występowania stanów oczekiwania i podnosi wydajność o 3 do 5%.

[**Pamięć podręczna**](https://pl.wikipedia.org/wiki/Pamięć_podręczna)[**procesora**](https://pl.wikipedia.org/wiki/Procesor) **trzeciego poziomu (L3)** jest wykorzystywana, kiedy pamięć L2 jest niewystarczająca aby pomieścić potrzebne dane. Najczęściej spotykana jest w procesorach przeznaczonych do zastosowań serwerowych. Obecność cache trzeciego poziomu pozwala na znaczącą poprawę wydajności w stosunku do procesorów o konstrukcji pamięci dwupoziomowej w wielu aplikacjach i programach. Jest "ratunkiem" przed sięgnięciem po dane do powolnej pamięci RAM, aczkolwiek gdy w pamięci L3 brakuje miejsca komputer szuka "pomocy" w powolnej pamięci [RAM](https://pl.wikipedia.org/wiki/RAM).

**3.Opisać cechy, znaczenie i rolę w masywnych obliczeniach równoległych: Optymalizacji programów uwzględniającej różne aspekty architektur.**

Optymalizacja na przykładzie kompilatora GCC:

Kompilator [GCC](http://pl.wikipedia.org/wiki/GNU_Compiler_Collection) przeprowadza najpierw optymalizację niezależną od architektury, a następnie,jeśli użytkownik sobie tego zażyczy, optymalizację kodu pod konkretny procesor lub nawet model procesora.

Optymalizacja ogólna jest wykonywana na zasadach, które są wspólne dla wszystkich maszyn, architektur i procesorów. Użytkownik może samodzielnie wybrać optymalizacje, które mają zostać wykonane, albo wybrać jeden ze zdefiniowanych poziomów optymalizacji (przełączniki -O, -O2, -O3, -Os):

-O – optymalizacja tylko w zakresie podstawowym, skraca to proces kompilacji programu;

-O2 – wszystkie bezpieczne optymalizacje, tzn. nie zostanie włączona żadna optymalizacja, która mogłaby zmienić w istotny sposób działanie programu (np. zmniejszyć precyzję obliczeń zmiennoprzecinkowych);

-O3 – najwyższy poziom optymalizacji, może powodować problemy w działaniu skomplikowanych programów, zmniejszyć precyzję obliczeń i spowodować znaczny wzrost objętości kodu maszynowego programu;

-Os – kod optymalizowany w celu minimalizacji rozmiaru pliku wykonywalnego;

Wbrew pozorom najwyższy poziom optymalizacji (-O3) nie musi przekładać się na szybsze działanie programu. Dłuższy kod wynikowy sprawia, że stosunkowo niewielka jego część da się zmieścić się w szybkiej pamięci podręcznej procesora.

Kompilator może również wykonać wiele optymalizacji, aby program lepiej działał na konkretnej maszynie. Jest to związane z pewnymi charakterystycznymi cechami procesora, takimi jak: ilość i rodzaje rejestrów (np. Procesory PowerPC mają więcej rejestrów ogólnego przeznaczenia niż Pentium), ilość i algorytmy zarządzające pamięcia cache (aby zmaksymalizować jej użycie). Rodzaje instrukcji (z naciskiem na instrukcje wyspecjalizowane do obróbki pewnych typów danych oraz instrukcje SIMD), wielopotokowość, czy superskalarność mają również ogromne znaczenie z punktu widzenia optymalizacji.

GCC na platformie x86 potrafi optymalizować kod pod szereg różnych procesorów. Włączenie optymalizacji powoduje często, że kod wynikowy nie uruchomi się na innym, podobnym procesorze lub będzie na nim działał wyjątkowo wolno.

Optymalizacja przy użyciu wielowątkowości

Wielowątkowość – cecha systemu operacyjnego, dzięki której w ramach jednego procesu może wykonywać kilka zadań lub jednostek wykonawczych. Nowe zadania to kolejne ciągi instrukcji wykonywane oddzielnie. Wszystkie zadania w ramach tego samego procesu współdzielą kod programu i dane.

Wielowątkowość może także odnosić się do samych procesorów. W takim wypadku oznacza możliwość jednoczesnego wykonywania wielu wątków sprzętowych na pojedynczej jednostce wykonawczej – rdzeniu.

OMP wieloplatformowy interfejs programowania aplikacji (API) umożliwiający tworzenie programów komputerowych dla systemów wieloprocesorowych z pamięcią dzieloną. Może być wykorzystywany w językach programowania C, C++ i Fortan na wielu architekturach, m.in. Unix i Microsoft Windows. Składa się ze zbioru dyrektyw kompilatora, bibliotek oraz zmiennych środowiskowych mających wpływ na sposób wykonywania się programu. Celem OpenMP jest implementacja wielowątkowości, czyli metody zrównoleglania programów komputerowych, w której główny wątek (czyli ciąg następujących po sobie instrukcji) „rozgałęzia” się na kilka „wątków potomnych”, które wspólnie wykonują określone zadanie. Wątki pracują współbieżnie i mogą zostać przydzielone przez środowisko uruchomieniowe różnym procesorom.

MPI protokół komunikacyjny będący standardem przesyłania komunikatów pomiędzy procesami programów równoległych działających na jednym lub więcej komputerach. Interfejs ten wraz z protokołem oraz semantyką specyfikuje, jak jego elementy winny się zachowywać w dowolnej implementacji. Celami MPI są wysoka jakość, skalowalność oraz przenośność. Standard MPI implementowany jest najczęściej w postaci bibliotek, z których można korzystać w programach tworzonych w różnych językach programowania, np. C, C++, Ada, Fortan. Zaletami MPI nad starszymi bibliotekami przekazywania wiadomości są przenośność oraz prędkość. Przenośność, ponieważ MPI został zaimplementowany dla każdej architektury opartej na rozproszonej pamięci. Prędkość, ponieważ każda implementacja jest zoptymalizowana pod sprzęt, na którym działa.

**4. Opisać cechy, rolę i znaczenie w masowych obliczeniach równoległych: kart graficznych ogólnego przeznaczenia GPGPU i ich architektury**

GPGPU

Procesor karty graficznej (tzw. GPU) to koprocesor zoptymalizowany do obliczeń w grafice komputerowej[30], z których większość to podlegające łatwemu urównolegleniu algebraiczne operacje macierzowe wykonywane w arytmetyce zmiennopozycyjnej. Dlatego współczesne GPU posiadają architekturę masowo równoległą z setkami rdzeni obliczeniowych, a ich teoretyczna moc obliczeniowa sięga 1 TFLOPS (bilion operacji zmiennoprzecinkowych na sekundę).

Początkowo w programach wykorzystujących technikę GPGPU używano standardowych interfejsów programowania aplikacji graficznych, głównie DirectX i OpenGL, co było niewygodne i nieefektywne. Dlatego producenci kart graficznych udostępnili specjalne języki programowania i środowiska programistyczne dostosowane do tworzenia aplikacji w technologii GPGPU: CUDA dla procesorów firmy Nvidia oraz CTM i ATI Stream dla procesorów firmy AMD. Innymi środowiskami przeznaczonymi do pracy w technologii GPGPU są m.in. BrookGPU, PeakStream oraz RapidMind. Istnieje również niezależne od platformy sprzętowej i wspierane przez wszystkich liczących się producentów sprzętu środowisko OpenCL. Zarówno Nvidia, jak i AMD produkują tzw. procesory obliczeniowe, czyli dedykowane do technologii GPGPU karty graficzne: Tesla i FireStream.

Współczesne procesory graficzne umożliwiają generowanie, w czasie rzeczywistym, realistycznej grafiki dzięki wprowadzeniu programowalnych jednostek realizujących potok graficzny w miejsce ich statycznych odpowiedników.

**5. Opisać cechy, rolę i znaczenie w masowych obliczeniach równoległych: środowiska i języka programowania kart GPGPU**

Modele programowania GPGPU

CUDA – pierwszy naprawdę popularny model programowania GPGPU

OpenCL – wzorowany na CUDA, dla szerszej grupy urządzeń (GPU, CPU, procesory heterogeniczne, akceleratory)

CUDA (ang. Compute Unified Device Architecture) – opracowana przez firmę Nvidia uniwersalna architektura procesorów wielordzeniowych (głównie kart graficznych) umożliwiająca wykorzystanie ich mocy obliczeniowej do rozwiązywania ogólnych problemów numerycznych w sposób wydajniejszy niż w tradycyjnych, sekwencyjnych procesorach ogólnego zastosowania.

Integralną częścią architektury CUDA jest oparte na języku programowania C środowisko programistyczne wysokiego poziomu, w którego skład wchodzą m.in. specjalny kompilator (nvcc), debugger (cuda-gdb, który jest rozszerzoną wersją debuggera gdb umożliwiającą śledzenie zarówno kodu wykonywanego na CPU, jak i na karcie graficznej), profiler oraz interfejs programowania aplikacji.

Bardzo ważnym aspektem architekruty CUDA jest wprowadzenie koncepcji wątku (thread) oraz bloku wątków (thread block). Dane są przetwarzne przez zupełnie niezależne bloki. W zależności od architektury sprzętu bloków naraz wykonuących obliczenia może być więcej lub mnie. Każdy blok składa się z pewnej liczby wątków. Wątki w obrębie jednego bloku można synchronizować i mają dostęp do wspólnych danych dzielonych.

Zalety

Pełna kompatybilność programów – napisany dziś program wykonywalny ma w przyszłości działać bez żadnych zmian na coraz wydajniejszych procesorach graficznych posiadających coraz większą liczbę rdzeni, rejestrów, pamięci operacyjnej i innych zasobów.

Model pamięci procesora ściśle odpowiadający architekturze sprzętowej, co umożliwia świadome, efektywne wykorzystywanie dostępnych zasobów GPU, w tym pamięci współdzielonej. Pamięć ta jest współdzielona przez wszystkie wątki w tzw. bloku (zwykle 128-512 wątków). Można jej używać jako programowalnej pamięci typu cache.

Kod uruchamiany na GPU może odczytywać i zapisywać dane z dowolnego adresu w pamięci GPU.