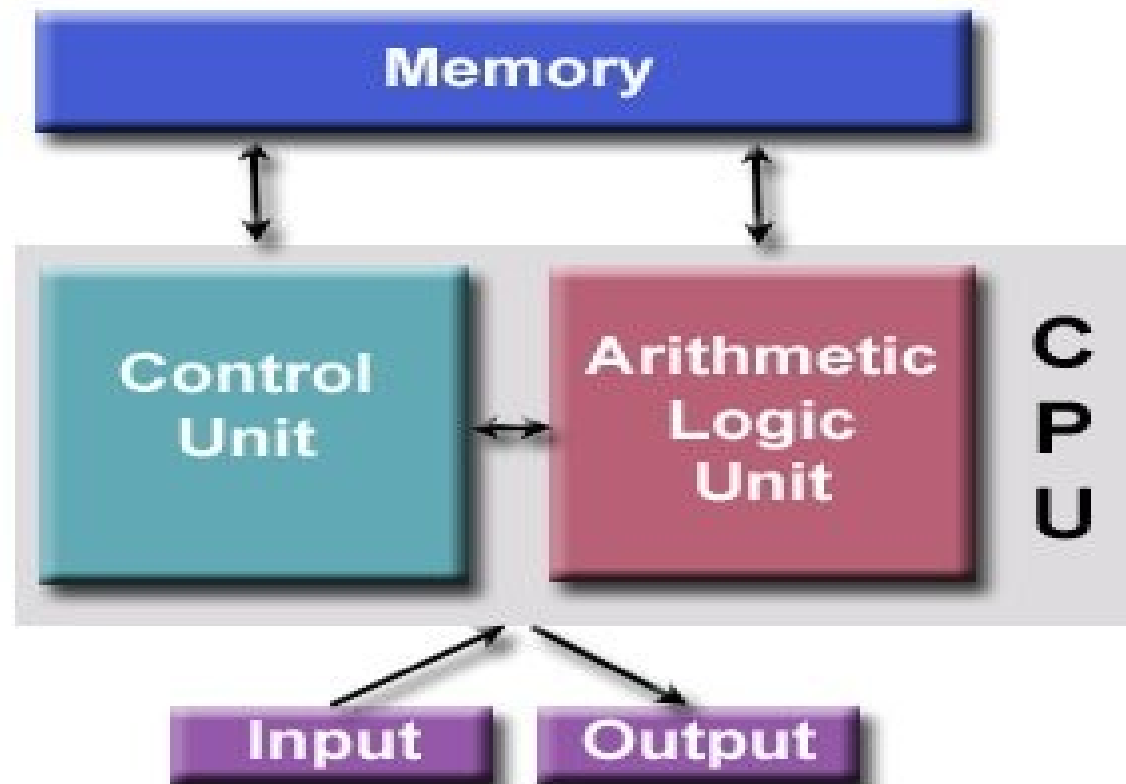


Komputasi Paralel

Slide 2
Komputer Paralel

Arsitektur Von Neumann



Arsitektur Von Neumann

- Selama lebih dari 40 tahun, komputer mengikuti model yang diambil dari nama matematikawan Hungaria, John Von Neumann
- Arsitektur Von Neumann menggunakan konsep stored-program. CPU mengeksekusi program yang tersimpan yang berisi barisan operasi baca tulis pada memori

Arsitektur Von Neumann

- Desain :
 - memori digunakan untuk menyimpan instruksi program dan data
 - program adalah coded-data yang memerintahkan komputer melakukan sesuatu
 - data adalah informasi yang digunakan oleh program
 - CPU mengambil data dari memori, mendekodekannya, dan menjalankannya secara sekuensial

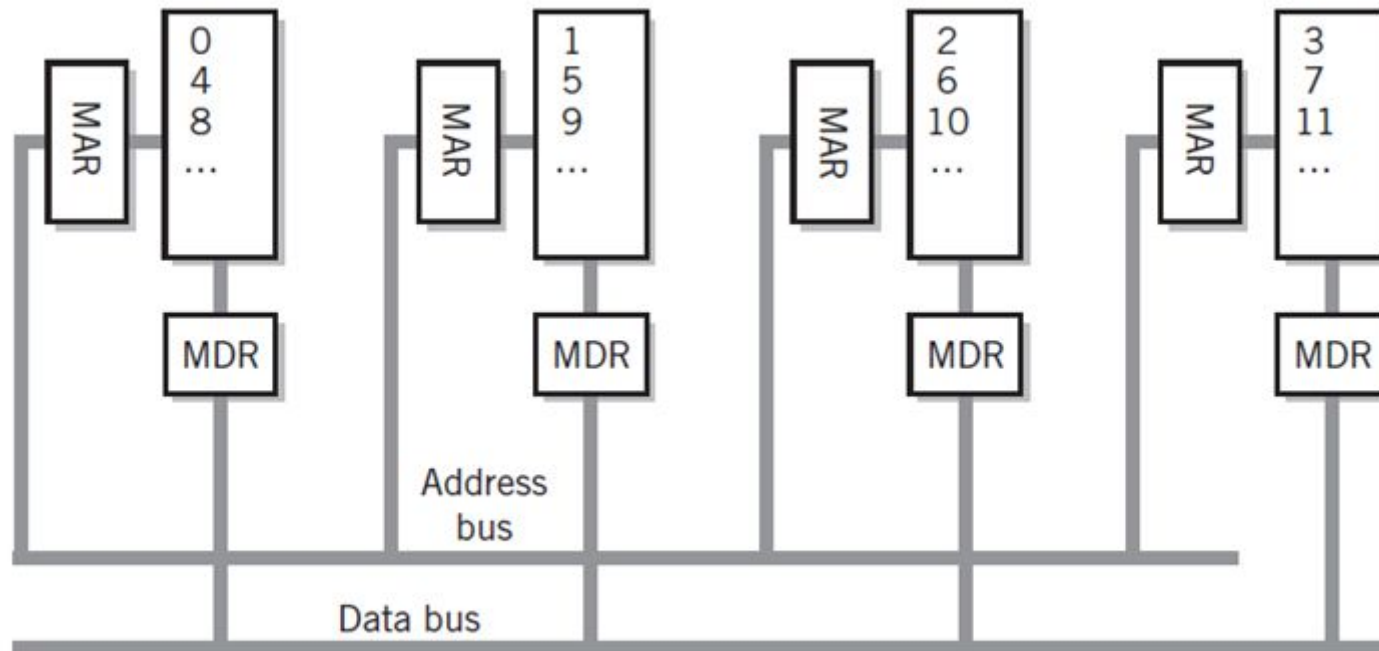
Arsitektur Von Neumann

- Komputer sekuensial bekerja berdasarkan model tersebut
- Performanya dibatasi oleh :
 - Kecepatan pertukaran informasi antara memori dan CPU
 - Laju eksekusi dari instruksinya
- Untuk meningkatkan performa, kedua aspek tersebut yang ditingkatkan

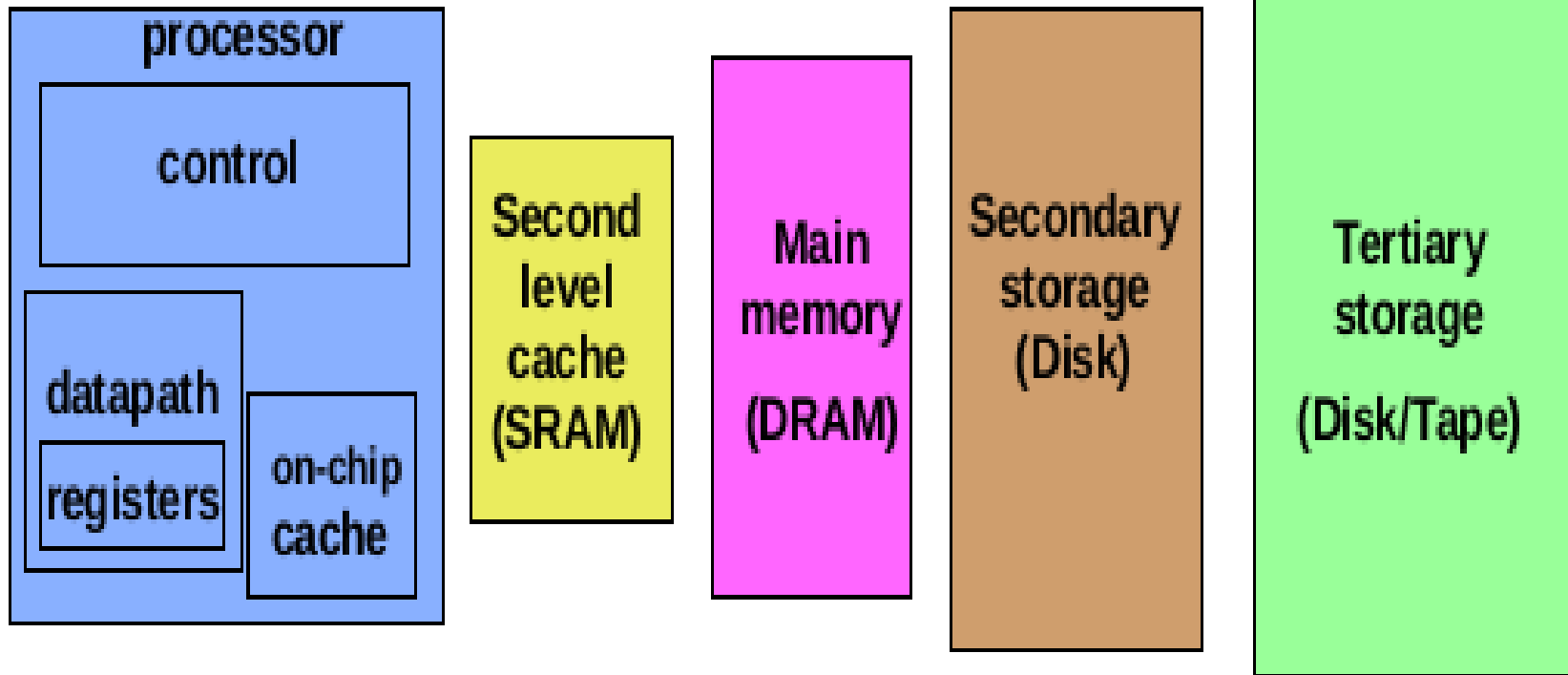
Memory Bank & Cache

- Kecepatan **pertukaran informasi antara memori dan CPU** dapat ditingkatkan dengan :
 - menggunakan *memory interleaving* (mengakses memori bersamaan dengan menggunakan beberapa memory bank)
 - menggunakan memori cepat (*cache*)

Memory Interleaving



Hierarki Memori



Hierarki Memori

Kecepatan akses dan banyak memori yang dapat disimpan dalam tiap bagian seperti berikut :

- Register (1 clock cycle, sekitar 100 byte)
- Cache tingkat 1 (3-5 clock cycle, sekitar 100 KByte)
- Cache tingkat 2 (sekitar 10 clock cycle, MByte)
- Memori utama (sekitar 100 clock cycle, GBytes)
- Disk (milisecond, 100GB ke atas)

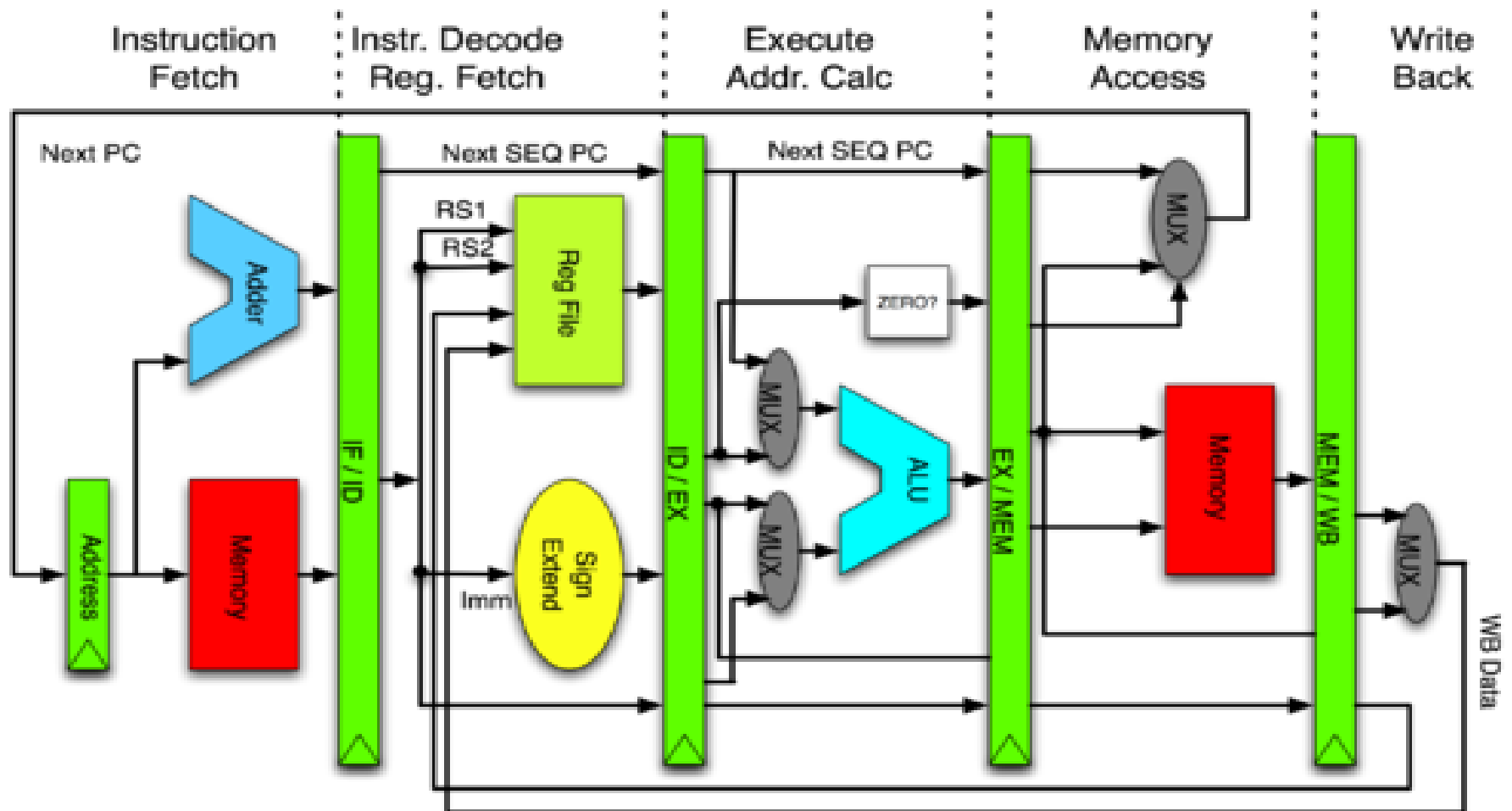
Pipeline

- **Laju eksekusi instruksi program** dapat ditingkatkan dengan membuat eksekusi beberapa instruksi tersebut *overlapping*
 - Eksekusi instruksi biasanya terbagi jadi 4 bagian :
 - fetch
 - decode
 - execute
 - writeback
- * (ada yang menuliskan 5 bagian dengan memisahkan memori access dan writeback)

Pipeline


- Keempat bagian eksekusi tersebut biasanya dijalankan seperti *assembly line* pada industri.
- Dengan membuat beberapa instruksi saling overlapping, eksekusi program lebih cepat

Pipeline



Pipeline

- Jika ada 5 instruksi yang akan dijalankan :

Clock Time 

	1	2	3	4	5	6	7	8
Inst 1	F	D	E	W				
Inst 2		F	D	E	W			
Inst 3			F	D	E	W		
Inst 4				F	D	E	W	
Inst 5					F	D	E	W

Komputer Sekuensial

- Pipelining, memory interleaving, dan caching biasa digunakan pada komputer sekuensial
- Batasannya :
 - kecepatan cache dibatasi teknologi (cahaya 30 cm/ns, tembaga 9 cm/ns)
 - memory interleaving dan pipelining hanya berguna di beberapa kasus saja
- Alternatifnya, menggunakan arsitektur paralel
- Komputer modern masih menggunakan desain ini. Bedanya, dalam jumlah unit yang digunakan.

Komputer Paralel

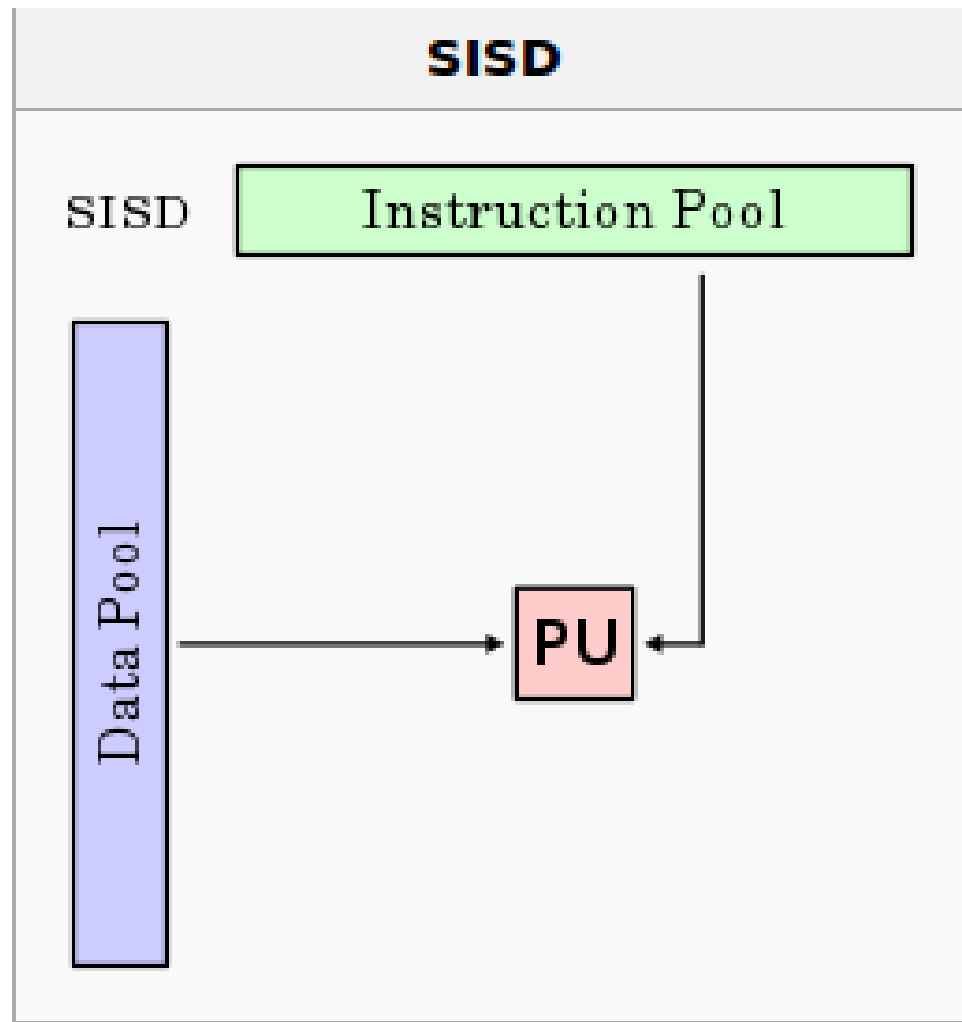
- Komputer paralel bisa dibagi berdasarkan :
 - taksonomi Flynn
 - mekanisme kendali
 - organisasi ruang memori
 - jaringan terinterkoneksi
 - granularitas prosesor

Taksonomi Flynn

- Usaha meningkatkan performa komputasi dengan cara menggunakan banyak sumber bukan hal baru
- Tahun 1966 Michael Flynn memperkenalkan taksonomi arsitektur komputer berdasarkan banyak instruksi dan data yang bisa diproses secara bersamaan

Taksonomi Flynn

	Single Data	Multiple Data
Single Instruction	SISD (prosesor tunggal tradisional)	SIMD (GPU core, <i>array processor</i>)
Multiple Instruction	MISD (<i>flight control computer</i>)	MIMD (GPU, <i>multicore computer</i>)



SISD

- Komputer sekuensial, tidak mengandung paralelisme baik pada instruksi atau aliran data.
- Unit kontrol mengambil satu per satu instruksi dari memori, lalu mengarahkan *processing element (PE)/processing unit (PU)* untuk mengoperasikan aliran data. Satu operasi tiap waktu.

SIMD

SIMD

Instruction Pool

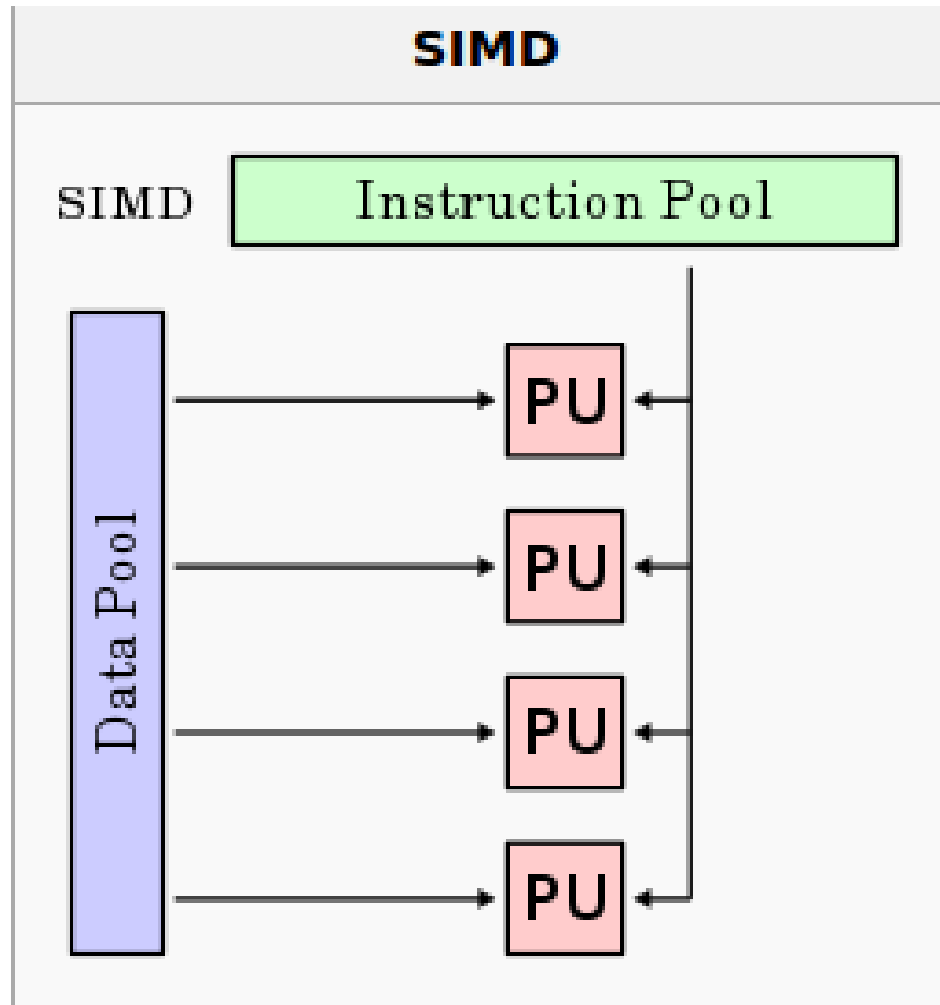
Data Pool

PU

PU

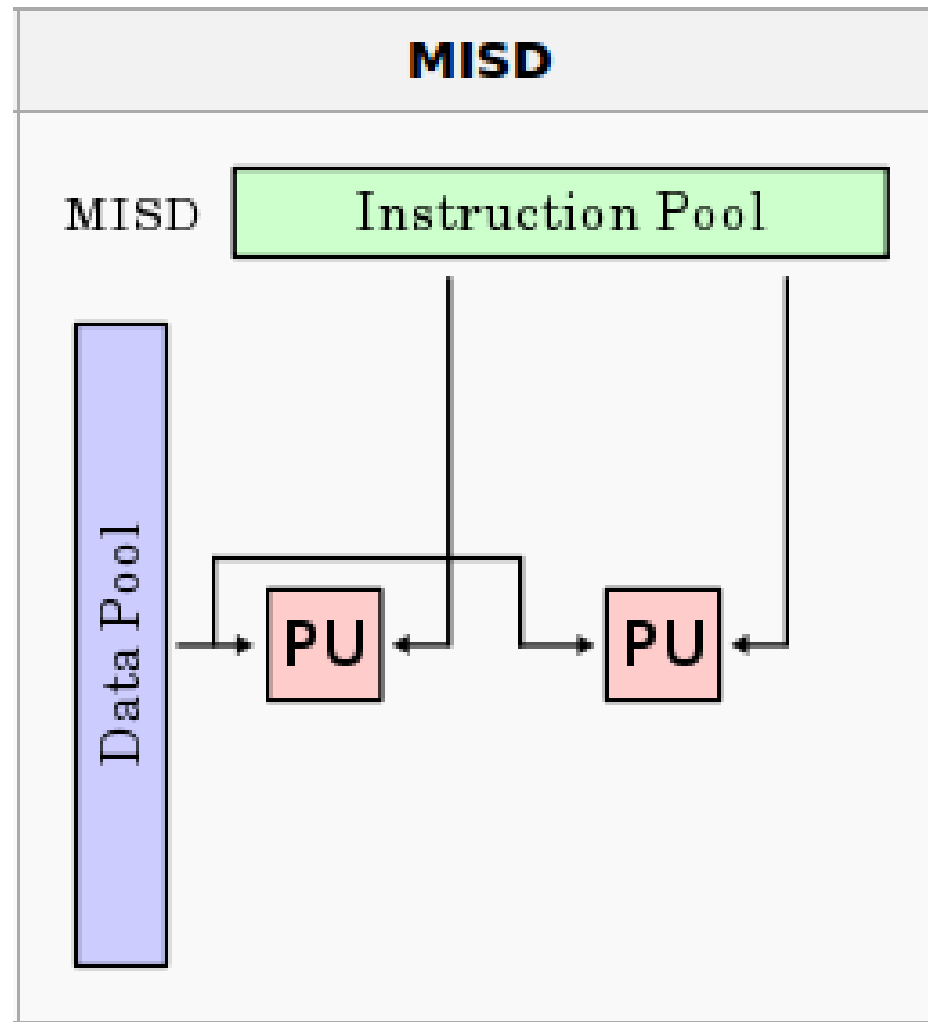
PU

PU



SIMD

- Semua unit proses melakukan instruksi yang sama dalam satu waktu, tapi data yang diproses satu unit dan unit lainnya bisa berbeda.
- Tipe ini biasanya memiliki instruction dispatcher, bandwidth jaringan internal yang sangat tinggi, dan unit instruksi berkapasitas kecil yang berjumlah banyak
- Cocok untuk pekerjaan yang memiliki regularitas tinggi.
Contoh : *image processing*
- Core GPU mengikuti desain ini :
 - (SIMD unit, AMD)
 - streaming multiprocessor (SM, NVIDIA)



MISD

- Sebuah data stream dijalankan ke beberapa PE. Setiap PE mengoperasikan data secara mandiri melalui stream instruksi masing masing.
- Biasanya digunakan untuk proses yang toleransi kesalahannya rendah. Memperbaiki toleransi kesalahan.
- Data diproses oleh beberapa mesin berbeda dan keputusannya diambil dari hasil mayoritas.
- Contoh :
 - komputer kontrol pesawat ulang-alik, aplikasi militer
 - beberapa filter frekuensi dioperasikan pada sebuah aliran sinyal
 - beberapa algoritma dijalankan untuk memecahkan sebuah pesan terenkripsi

MIMD

MIMD

Instruction Pool

Data Pool

PU

PU

PU

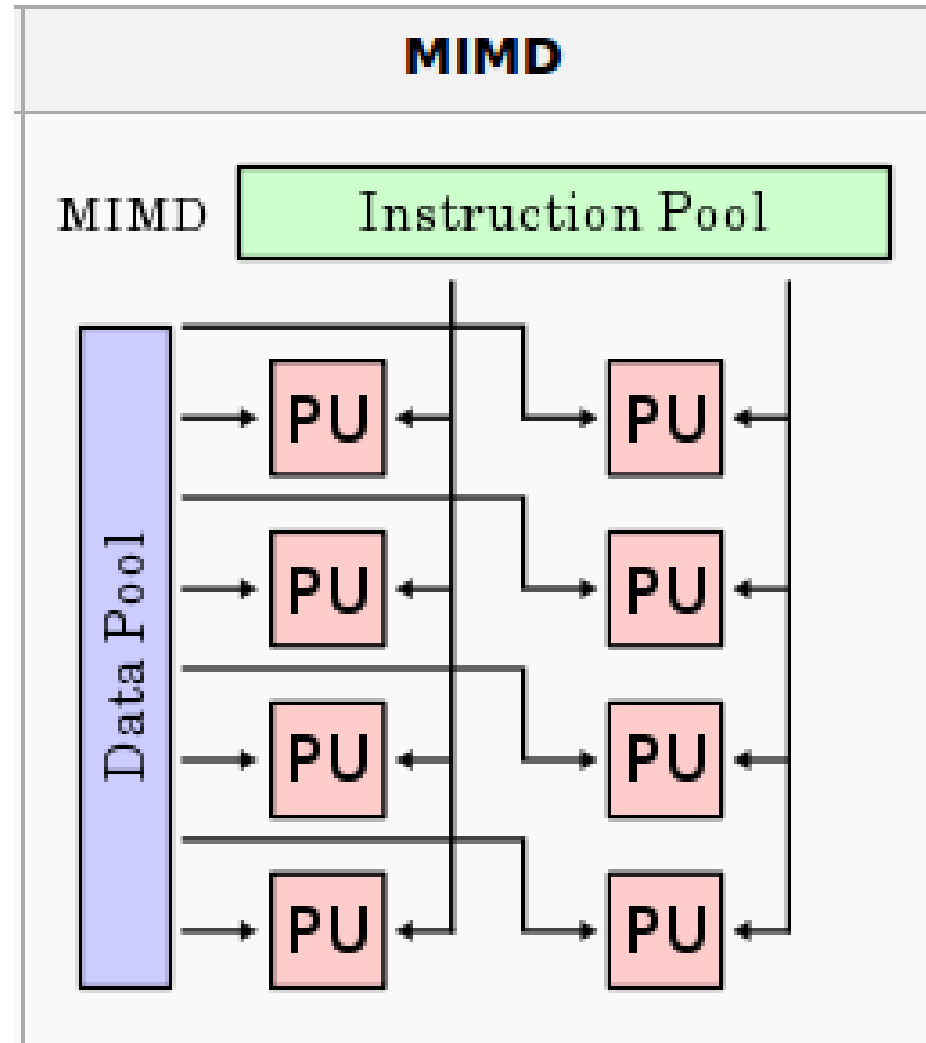
PU

PU

PU

PU

PU



MIMD

- Kategori mesin paling serbaguna dan tipe komputer paralel paling umum.
- Banyak prosesor, dapat melakukan beberapa instruksi sekaligus. Setiap prosesor dapat bekerja pada data yang berbeda.
- Eksekusinya bisa synchronous/asynchronous, deterministik/non-deterministik
- Contoh : GPU, CPU multicore

Sub Kategori MIMD

- Shared Memory MIMD
 - arsitekturnya memiliki sebuah ruang memori bersama yang dapat diakses secara universal
 - menyederhanakan transaksi antar CPU dengan overhead minimum
 - bottle neck dalam skalabilitas (solusi : partisi memori)
- Distributed Memory MIMD (shared nothing MIMD)
 - prosesor saling berkomunikasi mengirimkan pesan
 - overhead komunikasi besar
 - bisa diskalakan tanpa batas (selain batas ruang dan energi)

Mekanisme Kendali

- Komputer paralel memiliki beberapa elemen proses (PE)
- PE bisa beroperasi :
 - dalam kendali terpusat (SIMD)
 - independen (MIMD)

SIMD dan MIMD

- SIMD
 - control unit memberikan instruksi ke PE
 - eksekusi synchronous
 - PE on/off
 - CM-2, Illiac IV, MP-1, MP-2
- MIMD
 - setiap PE dapat mengeksekusi program masing masing
 - iPSC, Symmetry, nCube 2, CM-5

SIMD dan MIMD

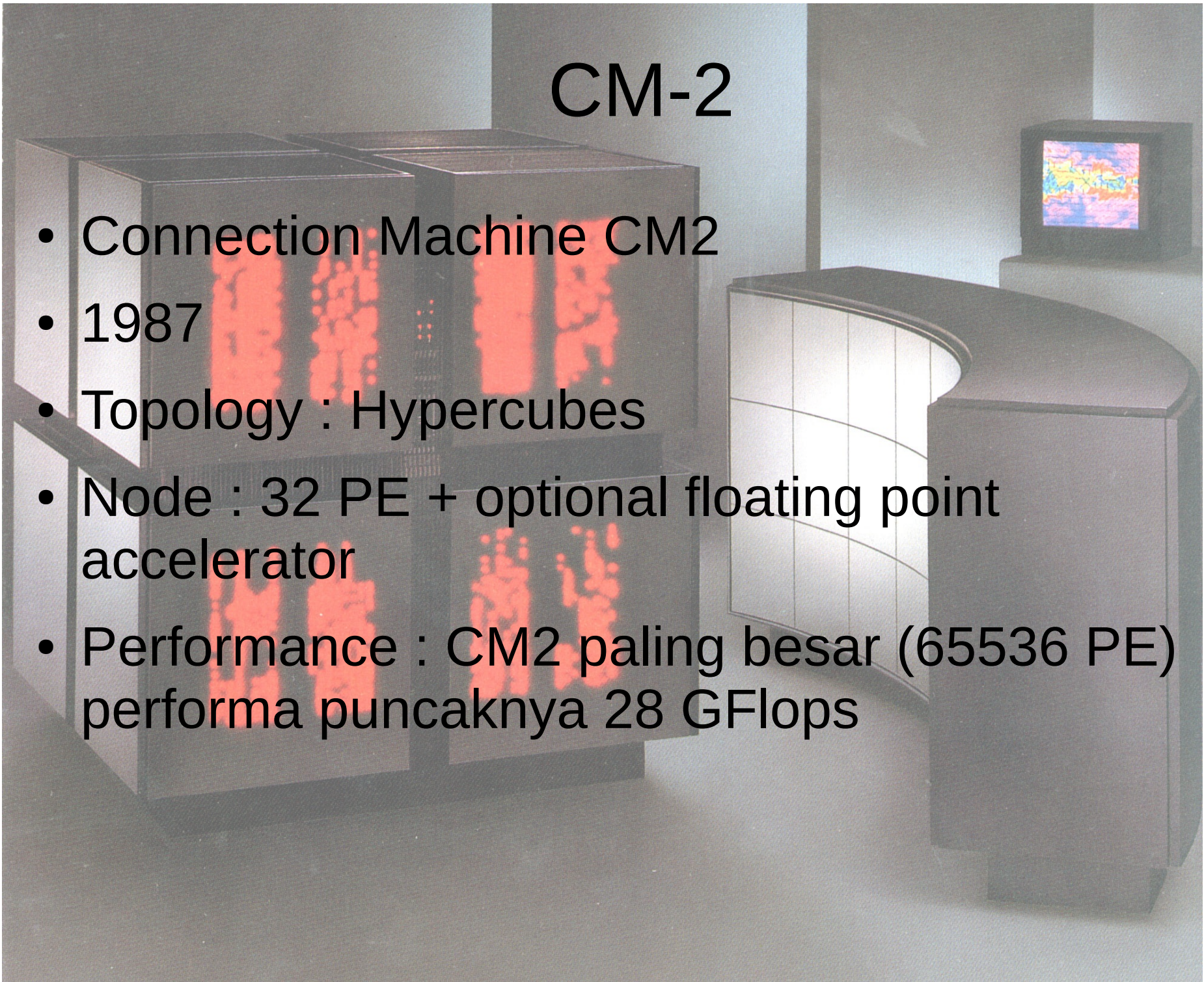
- SIMD membutuhkan hardware lebih sedikit (sebuah control unit)
- prosesor MIMD lebih kompleks karena control unit terpisah
- SIMD membutuhkan memori lebih kecil karena hanya menjalankan sebuah salinan program
- SIMD cocok untuk program yang data-paralel
- SIMD bisa menjalankan satu instruksi tiap satu clock cycle

SIMD dan MIMD

- MIMD cocok untuk program yang dibatasi oleh condition statement
- SIMD memberikan automatic synchronization
- prosesor MIMD lebih umum
- prosesor SIMD adalah desain khusus

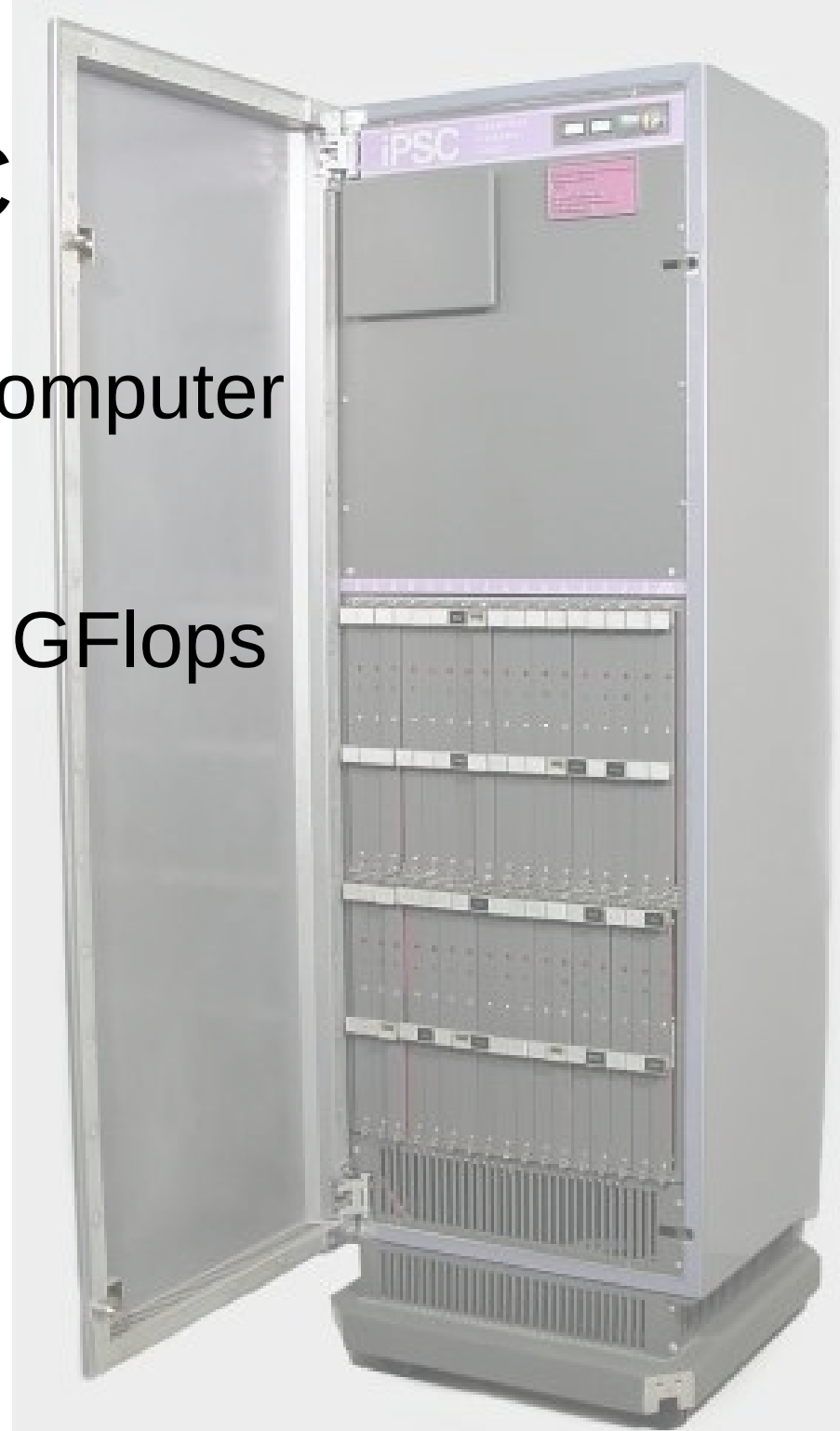
CM-2

- Connection Machine CM2
- 1987
- Topology : Hypercubes
- Node : 32 PE + optional floating point accelerator
- Performance : CM2 paling besar (65536 PE) performa puncaknya 28 GFlops



iPSC

- Intel Personal Scientific Computer
- 1985
- Peak Performance over 7 GFlops



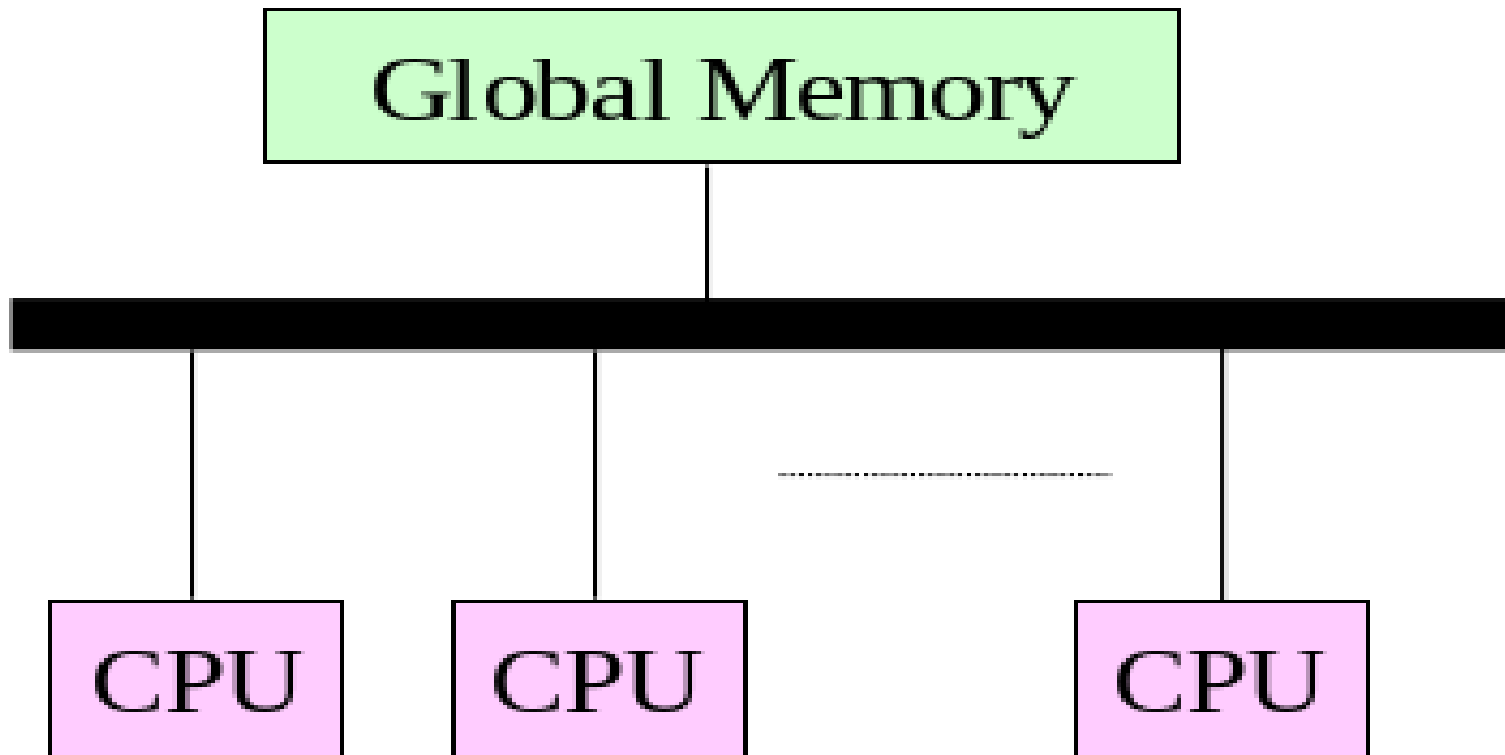
Organisasi Ruang Memori

- Pengaturan memori dalam komputer paralel bisa berupa :
 - Shared Memory
 - Distributed Memory
 - (Distributed Shared Memory)
 - Hybrid

Shared Memory

- Banyak prosesor berbagi memori yang sama (memori global)
- Mudah diprogram
- Memprogram dengan threading
- Sulit dibuat (tight coupling dengan hardware)
- Performa CPU turun jika memori terletak jauh
- Skalabilitas terbatas :
 - menambah CPU mengakibatkan traffic pada jalur CPU-memory meningkat

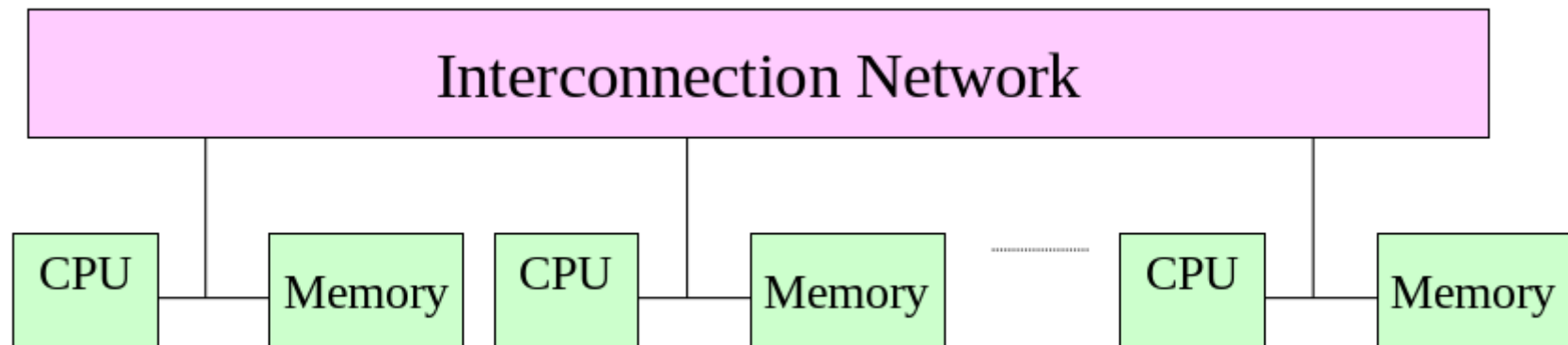
(Single) Shared Memory



Distributed Memory

- Memprogram berdasarkan proses message passing (membutuhkan jaringan komunikasi)
- Dapat diskalakan dengan baik
- Kontrol dan pengiriman data lebih ketat
- Sulit diprogram
 - tidak ada konsep memori global untuk seluruh prosesor.
 - saat prosesor membutuhkan data pada prosesor lain, tugas programmer untuk menentukan bagaimana dan kapan data dikomunikasikan.
 - sinkronisasi antar tasks menjadi tanggung jawab programmer.

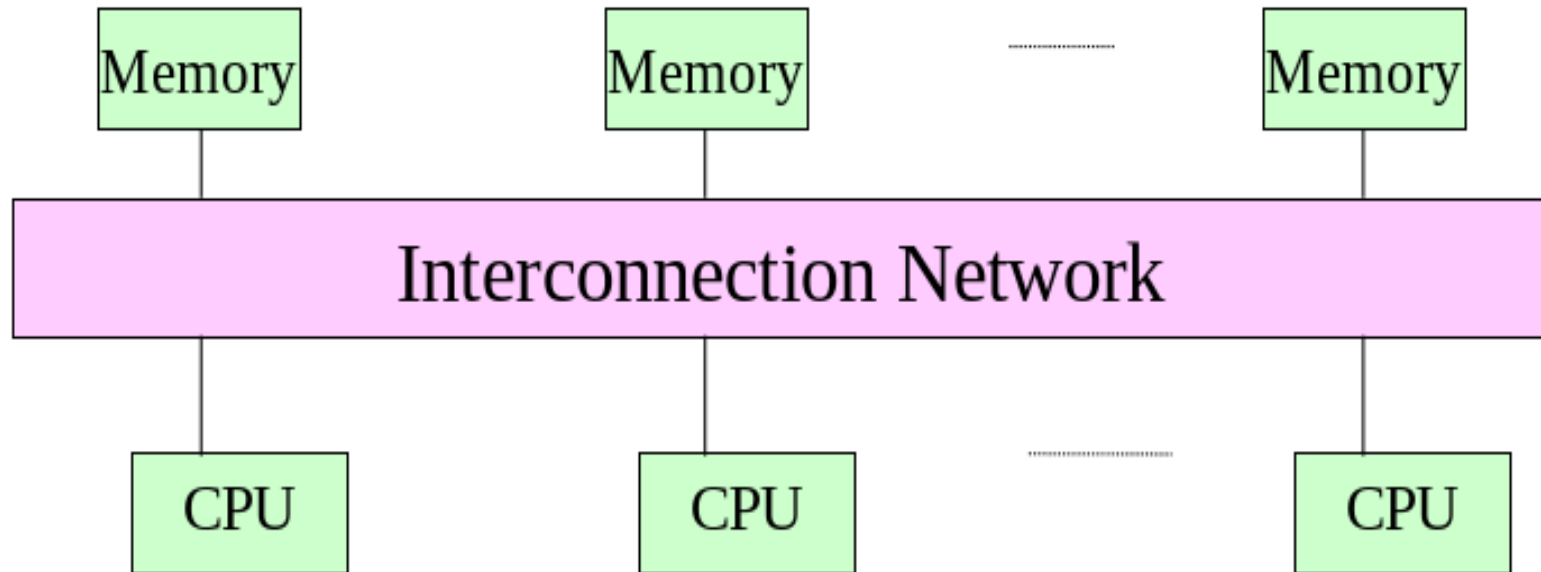
Distributed Memory



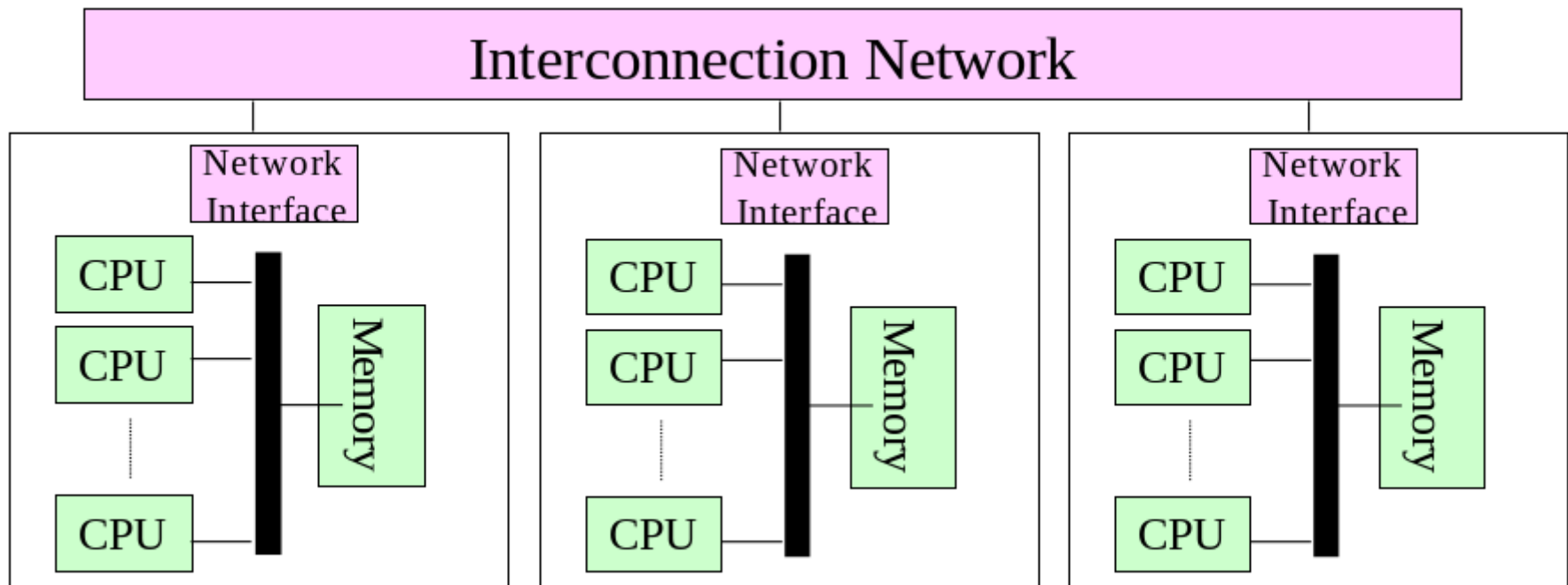
Distributed Shared Memory

- secara logic adalah shared memory
- secara fisik berupa lokal memori yang terdistribusi
- bisa berbasis pada page, variabel, atau object

Distributed Shared Memory



Hybrid



Organisasi Ruang Memori

- Penyelesaian masalah menggunakan arsitektur paralel membutuhkan komunikasi antar prosesor
- Komunikasi bisa dilakukan dengan cara
 - message passing
 - shared memory

Message Passing

- Multi komputer
- tiap prosesor memiliki memori sendiri
- prosesor terhubung melalui jaringan terinterkoneksi
- prosesor berkomunikasi dengan mengirimkan pesan satu sama lain
- contoh komputer : CM-5, iPSC

Shared Memory

- Multi prosesor
- Arsitektur shared memory berbagi memori untuk komunikasi dengan mengubah variabel (lokal/global)

Shared Memory

- Mesin Shared Memory dibagi menjadi 2 kelas utama berdasarkan waktu akses prosesornya :
 - UMA (Uniform Memory Access)
 - NUMA (Non Uniform Memory Access)
- Ada beberapa varian lain, diantaranya :
 - COMA (Cache Only Memory Access)
 - NORMA (No Remote Memory Access)

UMA

- Akses yang sama (hak dan waktu akses) ke memori bersama
- Contoh : SMP (symmetric multiprocessor)
- Masalah kapasitas jaringan :
 - semua prosesor mungkin membutuhkan sesuatu dari memori pada waktu yang bersamaan
 - akses memori melalui jaringan bisa lambat karena strukturnya

NUMA

- Waktu aksesnya bergantung pada lokasi fisik prosesor dan memori
- Menghindari memory reference menggunakan memori lokal
- Konsep memori lokal bisa dikembangkan hingga menjadikan global memori
- Memory reference dipetakan oleh hardware

COMA

- Arsitektur NUMA mirip dengan message passing
 - Arsitektur message passing membutuhkan pengiriman pesan secara eksplisit
- Pada kasus tertentu, prosesor hanya memiliki cache saja, tanpa memiliki memori lokal (COMA)

Koherensi Cache

- Cache dapat dipasang pada prosesor untuk mempercepat memory reference
- Masalah muncul saat prosesor dapat mengubah variabel pada cachanya sendiri
 - nilai pada memori dan nilai pada cache bisa berbeda

Koherensi Cache

- Write Through (WT) Cache
 - memori mengikuti cachanya
- Write Back (WB) Cache
 - memori tidak diupdate hingga ada perintah untuk mengubahnya
- Masalah koherensi cache bisa muncul saat :
 - memodifikasi data bersama
 - migrasi proses
 - proses I/O bypassing cache

Koherensi Cache

- Koherensi cache bisa didapat menggunakan :
 - write invalidate
 - membatalkan semua salinan dari memori
 - write update
 - membroadcast nilai cache yang diperbaharui

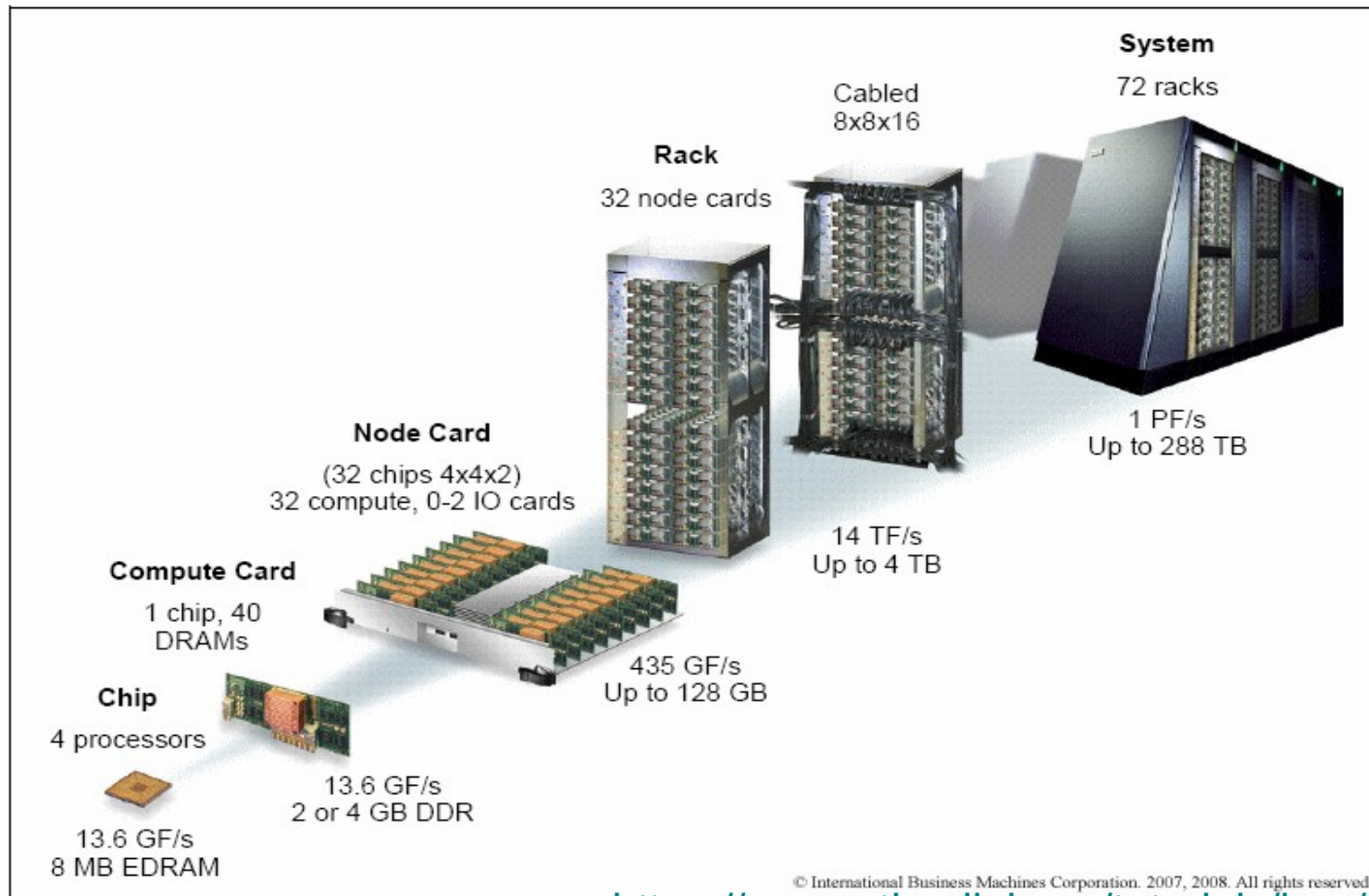
Jaringan Terinterkoneksi

- Baik arsitektur shared memory ataupun message passing bisa menggunakan
 - static interconnection network
 - dynamic interconnection network
- Jaringan static biasanya digunakan pada komputer message passing
- Jaringan dynamic biasanya digunakan pada komputer shared memory

Granularitas Prosesor

- Arsitektur paralel bisa diimplementasikan menggunakan :
 - prosesor yang kuat berjumlah sedikit
 - banyak prosesor dengan kemampuan yang kurang kuat
- Klasifikasinya :
 - coarse-grain
 - medium-grain
 - fine-grain

Blue Gene Architecture



<https://computing.llnl.gov/tutorials/bgp/>