



Politechnika Wrocławska

Podstawy Techniki Mikroprocesorowej wykład 12: koprocesor, MIMD

Dr inż. Jacek Mazurkiewicz
Katedra Informatyki Technicznej
e-mail: Jacek.Mazurkiewicz@pwr.edu.pl

Formaty danych (1)

e) format krótki rzeczywisty (ang. *short real*)

$$X = (-1)^S \left(2^{\left(\sum_{j=0}^7 E_j 2^j - 127 \right)} \right) \sum_{k=0}^{23} F_k 2^{-k}, \quad F_0 = 1,$$

bit F_0 nie występuje w słowie kodowym i jest we wzorze domyślnie traktowany jako 1 ze względu na posługiwanie się postaciami znormalizowanymi liczb;

f) format długi rzeczywisty (ang. *long real*)

$$X = (-1)^S \left(2^{\left(\sum_{j=0}^{10} E_j 2^j - 1023 \right)} \right) \sum_{k=0}^{52} F_k 2^{-k}, \quad F_0 = 1,$$

bit F_0 nie występuje w słowie kodowym i jest we wzorze domyślnie traktowany jako 1 ze względu na posługiwanie się postaciami znormalizowanymi liczb;

g) format rzeczywisty rozszerzony (ang. *temporary real*)

$$X = (-1)^S \left(2^{\left(\sum_{j=0}^{14} E_j 2^j - 16383 \right)} \right) \sum_{k=0}^{63} F_k 2^{-k},$$



Formaty danych (2)

a) format słowowy całkowity (ang. *word integer*)

$$X = -I_{15}2^{15} + \sum_{j=0}^{14} I_j 2^j,$$

b) format krótki całkowity (ang. *short integer*)

$$X = -I_{31}2^{31} + \sum_{j=0}^{30} I_j 2^j,$$

c) format długi całkowity (ang. *long integer*)

$$X = -I_{63}2^{63} + \sum_{j=0}^{62} I_j 2^j,$$

d) format BCD, upakowany (ang. *packed BCD*)

$$X = (-1)^s \sum_{j=0}^{17} D_j 10^j,$$

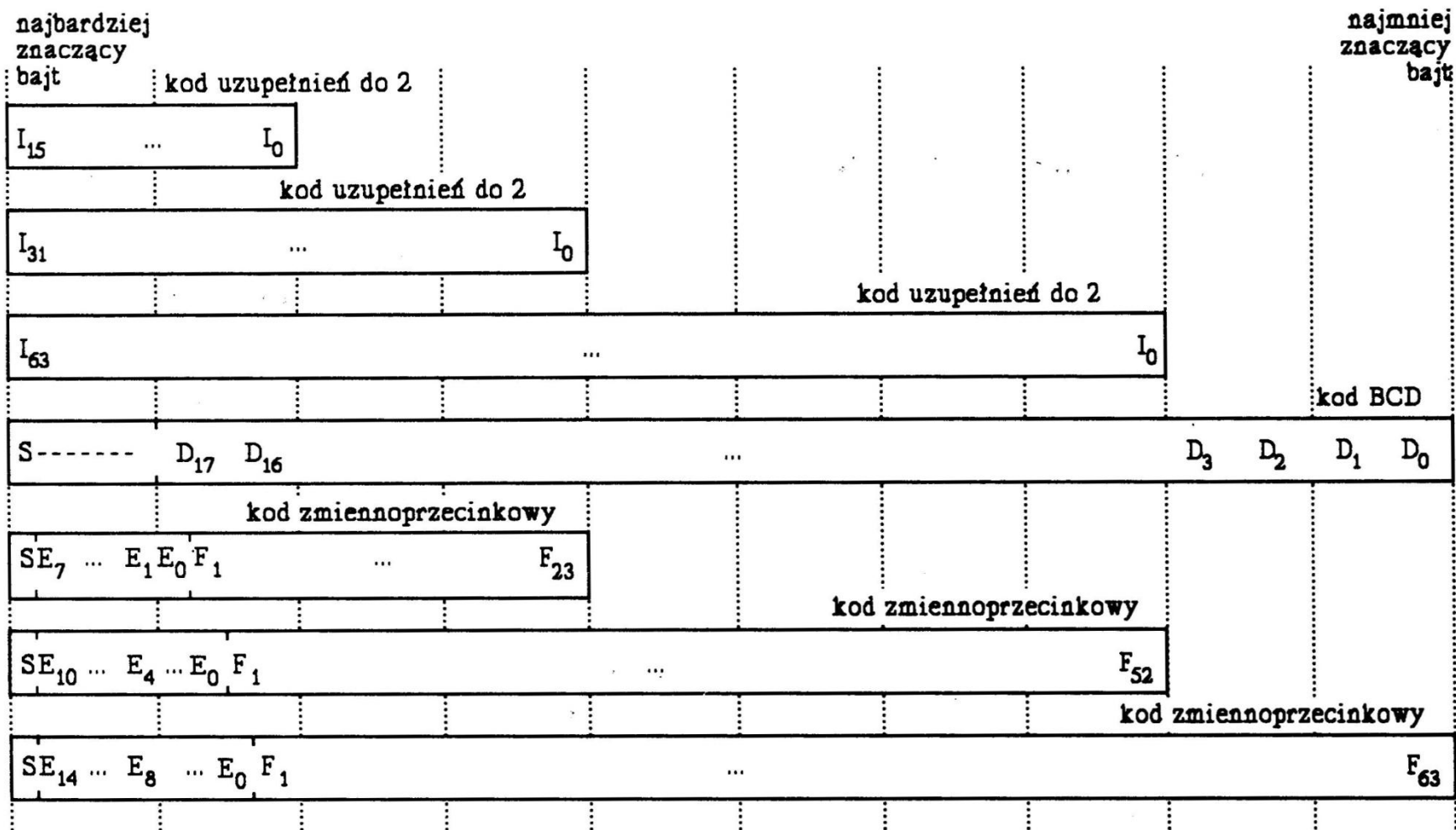
gdzie D_j — cyfra dziesiętna kodowana wtórnie za pomocą czterech bitów według wzoru:

$$D_j = \sum_{k=0}^3 d_k 2^k$$

(d_3 – d_0 — bity kodujące cyfrę D_j), bity b_6 – b_0 w najstarszym bajcie nie są wykorzystane;



Formaty danych (3)





Regulacje i rejestry (1)

Rejestr znaczników składa się z ośmiu dwubitowych pól TAG0–TAG7 zawierających syntetyczne dane o zawartości rejestrów na stosie. Znaczenie bitów każdego z pól jest następujące:

- 00 — liczba zwykła;
- 01 — zero;
- 10 — zawartość specjalna (nieskończoność, nieliczba);
- 11 — rejestr pusty.

- a) rejestr sterujący (ang. *control register*) (16-bitowy),
- b) rejestr stanu (ang. *status register*) (16-bitowy),
- c) rejestr znaczników (ang. *tag register*) (16-bitowy),
- d) rejestr adresu rozkazu (ang. *instruction pointer*) (20-bitowy),
- e) rejestr kodu rozkazu (ang. *instruction register*) (11 mniej znaczących bitów rozkazu ESC),
- f) rejestr adresu danej (ang. *data pointer*) (20 bitów).

Regulacje i rejestry (2)

IC (ang. *infinity control*) — określenie modelu nieskończoności (0 — tryb rzutowy, 1 — afiniczny); w trybie afinicznym rozróżnia się $+\infty$ i $-\infty$, przyjmując, że $-\infty$ jest obiektem leżącym na osi liczbowej na lewo względem każdej liczby, natomiast w trybie rzutowym nieskończoność jest nieskończonością bez znaku; przyjęcie jednego bądź drugiego modelu nieskończoności wpływa na sposób generowania wyników przy porównaniach;

RC (ang. *rounding control*) — sposób zaokrąglania:

00 — zaokrąglanie do liczby najbliższej,

01 — zaokrąglanie w dół (w kierunku $-\infty$),

10 — zaokrąglanie w górę (w kierunku $+\infty$),

11 — zaokrąglanie w kierunku 0 (tzn. w dół dla liczb dodatnich i w górę dla ujemnych);

PC (ang. *precision control*) precyzja obliczeń:

00 — 24 bity,

10 — 53 bity,

11 — 64 bity,

01 — zarezerwowane.



Regulacje i rejestry (3)

Znaczenie pól rejestru stanu jest następujące:

B (ang. *busy*) — jedynka oznacza, że rozkaz numeryczny nie jest zakończony;

TOP (ang. *top of stack pointer*) — numer rejestru na szczycie stosu;

C₃, C₂, C₁, C₀ (ang. *condition code*) — kod warunku

IR (ang. *interrupt request*) zgłoszenie przerwania.

Przyczynę przerwania, czyli sytuację wyjątkową, precyzują następujące bity:

PE (ang. *precision*) — utrata dokładności;

UE (ang. *underflow*) — niedomiar;

OE (ang. *overflow*) — nadmiar;

ZE (ang. *zero divide*) — dzielenie przez zero;

DE (ang. *denormalized operand*) — argument w postaci nieunormowanej;

IE (ang. *invalid operation*) — operacja błędna.

Lista rozkazów (1)

Rozkazy przesłań FLD, FST, FSTP, FXCH, FILD, FIST, FBLD, FBSTB są przeznaczone do ładowania liczb z pamięci lub odsyłania ich do pamięci. W celu przesyłania liczb rzeczywistych, całkowitych lub dziesiętnych stosuje się oddzielne rozkazy. Rozkazy, których nazwy mają literę P na końcu, po wysłaniu liczby do pamięci usuwają ją ze stosu. Rozkaz FXCH pozwala wymieniać liczby między rejestrami.

Rozkazy arytmetyczne pozwalają wykonywać operacje dodawania, odejmowania, mnożenia, dzielenia oraz inne operacje: pierwiastek kwadratowy (FSQRT), skalowanie (FSCALE), częściowa reszta z dzielenia (FPREM), zaokrąglanie do liczby całkowitej (FRINDINT), rozdzielenia na część ułamkową i wykładnik — ekstrakcja (FXTRACT), moduł (FABS) i zmiana znaku (FCHS). W przypadku operacji odejmowania i dzielenia, oprócz rozkazów realizujących operacje w zwykłej postaci, istnieją także rozkazy wykonujące te operacje dla argumentów zamienionych miejscami na stosie, co upraszcza manipulowanie argumentami. Można wyróżnić rozkazy usuwające argument ze stosu i pozostawiające go.

Rozkazy porównań i testowania pozwalają dokonać porównania dwóch liczb albo uzyskać charakterystyczną informację o liczbie w celu wykonania skoku warunkowego.



Lista rozkazów (2)

Rozkazy realizujące funkcje przestępne pozwalają obliczać tangens (FPTAN) arcus tangens (FPATAN), $2^x - 1$ (F2XM1), $y \log_2 x$ (FYL2X), $y \log_2(x+1)$ (FYL2X).

Rozkazy ładowania stałych pozwalają wstawić do rejestru zero (FLDZ), jedynekę (FLD1), liczbę π (FLDPI), $\log_2 10$ (FLDL2T), $\log_2 e$ (FLDL2E), $\log_{10} 2$ (FLDLG2) oraz $\log_e 2$ (FLDLN2).

FDECSTP

— dekrementacja wskaźnika stosu;

FFREE

— opróżnienie rejestru;

FNOP

— nic nie rób;

FWAIT

— oczekiwanie procesora głównego na zakończenie rozkazu numerycznego; rozkaz ten nie jest wykonywany przez koprocesor, lecz zamieniany jest przez kompilator na rozkaz WAIT dla układu 8086.



Lista rozkazów (3)

FINIT, FNINIT

— inicjacja koprocatora — powoduje opróżnienie rejestrów
ustawienie rejestru sterującego;

FDISI, FNDISI

— zablokowanie przerwań z koprocatora;

FENI, FNENI

— odblokowanie przerwań;

FLDCW

— ładowanie słowa sterującego z pamięci;

FSTCW, FNSTCW

— zapamiętywanie słowa sterującego;

FSTSW, FNSTSW

— zapamiętywanie stanu;

FCLEX, FNCLEX

— zerowanie bitów wyjątków;

FSTENV, FNSTENV

— zapamiętywanie rejestrów środowiskowych

FSAVE, FNSAVE

— zapamiętywanie stanu koprocatora

FLDENV

— ładowanie rejestrów środowiskowych;

FRSTDR

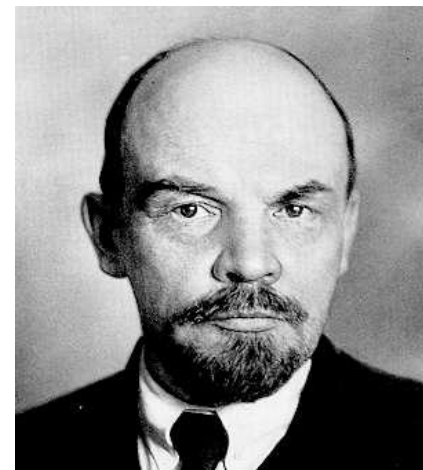
— odtworzenie stanu koprocatora;

FINCSTP

— inkrementacja wskaźnika stosu;

„Kompjuter - eta jest’” i klasyfikacja

- jednostka centralna - procesor
- pamięć operacyjna
- urządzenia wejścia-wyjścia
- magistrale
 - von Neumann (!)
- SISD - Single Instruction Single Data
- SIMD - Single Instruction Multiple Data
- MISD - Multiple Instruction Single Data
- MIMD - Multiple Instruction Multiple Data





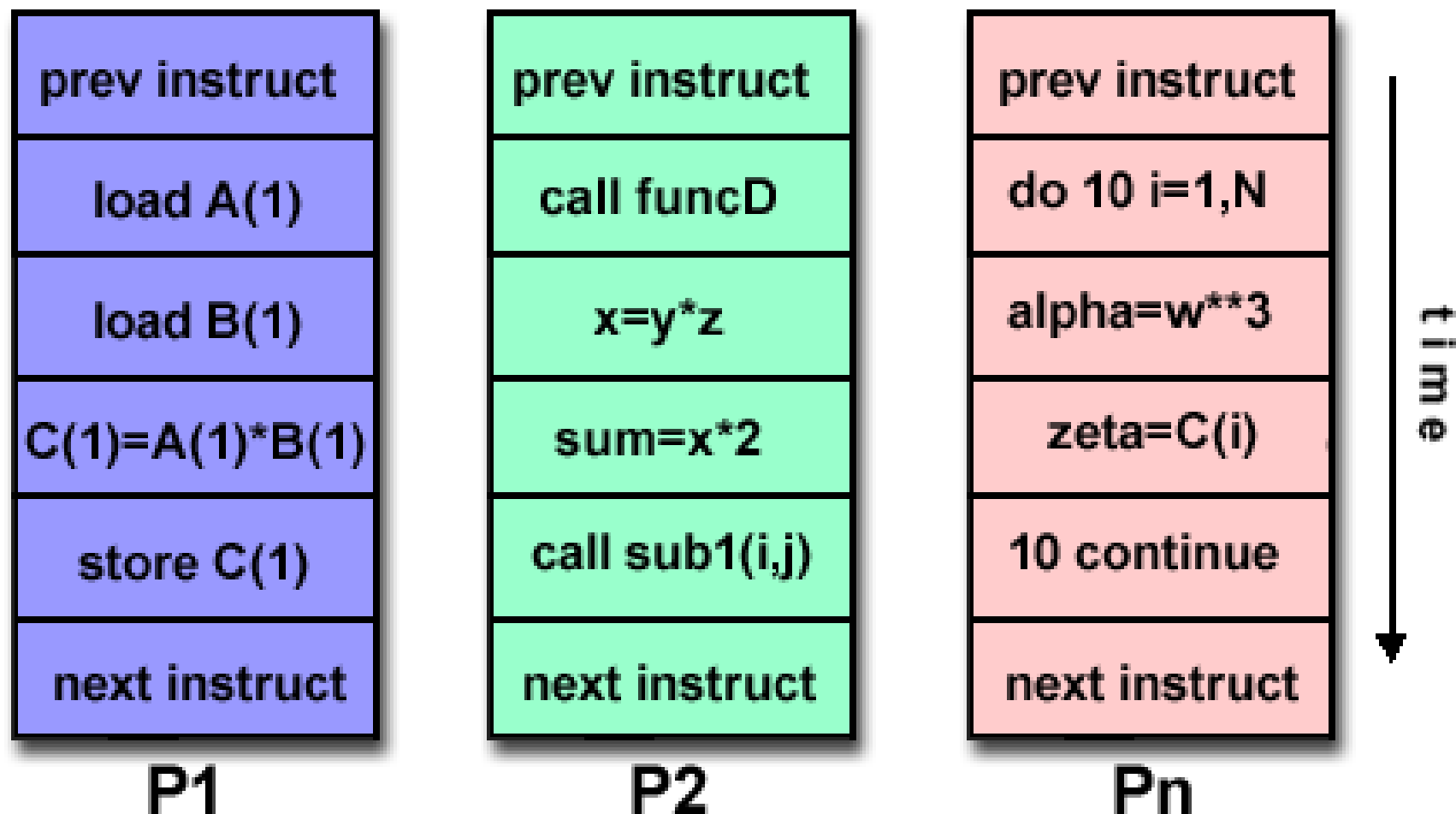
MIMD - wstęp (1)

Multiple Instruction Multiple Data (MIMD):

- Realizacja różnych instrukcji w jednym czasie
- Każdy procesor zaopatrzony w swe sterowanie
- Procesory mogą być dedykowane do jednego zadania lub zupełnie różnych
- Systemy multiprocesorowe, multikomputerowe



MIMD -wstęp (2)



MIMD - topologie (1)

Topologia:

- Topologia systemu wieloprocessorowego wynika ze struktury połączeń jednostek
- Typowe metryki:
 - ✓ *Diameter* maksymalna odległość między dwoma procesorami w systemie
 - ✓ *Bandwidth* przepustowość pojedynczego linka przemnożona przez liczbę połączeń
 - ✓ *Bisectional Bandwidth* przepustowość związana z komunikacją między fragmentami systemu



MIMD - topologie (2)

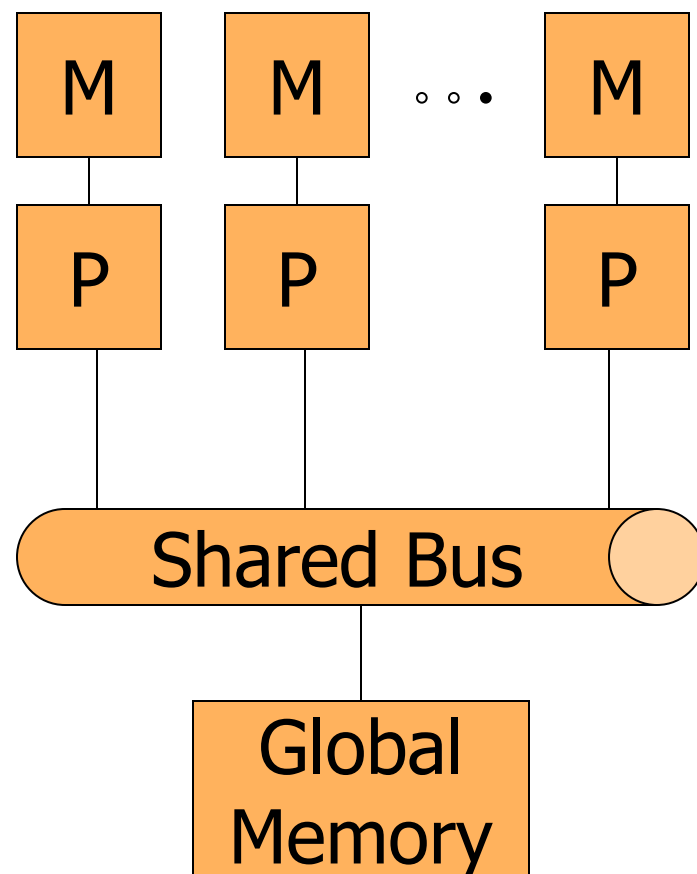
6 podstawowych kategorii topologicznych:

- Shared Bus
- Ring
- Tree
- Mesh
- Hypercube
- Completely Connected

MIMD - topologie (3)

Shared Bus:

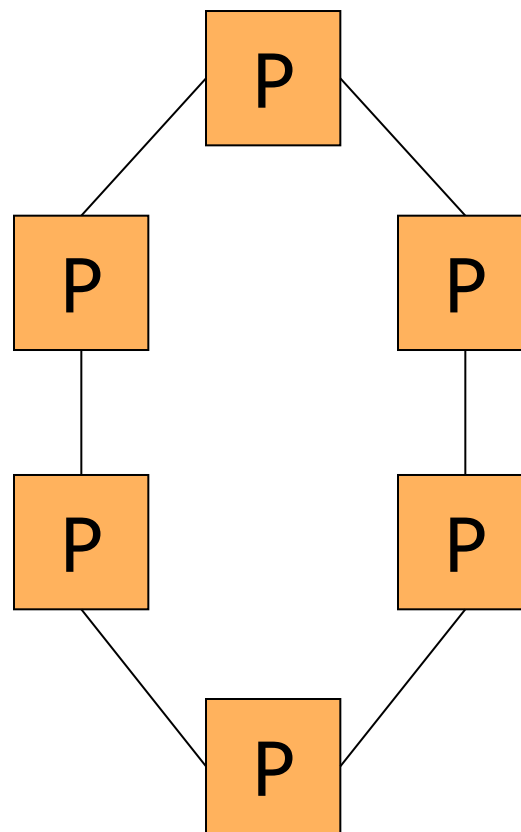
- Prostota konstrukcji
- Procesory komunikują się poprzez magistrale
- Warunki transmisji dyktuje obsługa magistrali
- Łatwość doczepienia nowych procesorów – jeśli zagwarantujemy właściwe sterowanie



MIMD - topologie (4)

Ring:

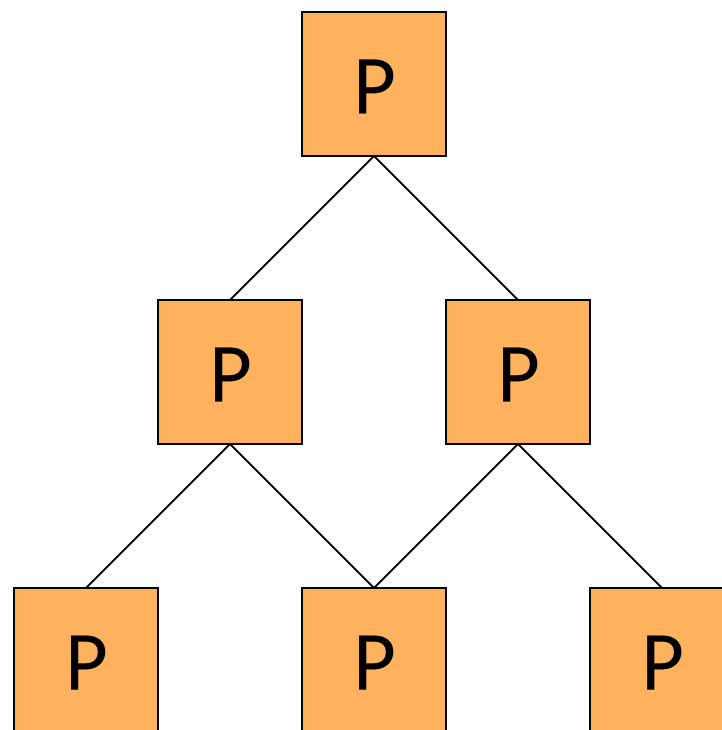
- Dedykowane połączenia między procesorami
- Równoczesna komunikacja jest w zasięgu
- Wędrówka danych przez pośredników
- Każdy procesor ma 2 kanały komunikacyjne



MIMD - topologie (5)

Tree topology:

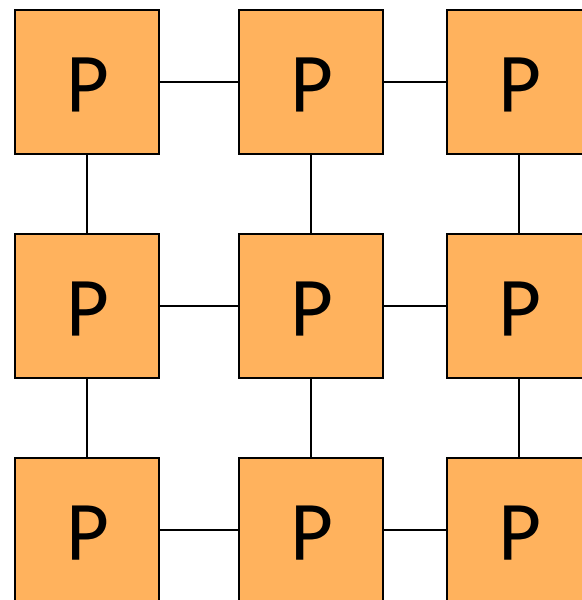
- Bezpośrednie połączenia między procesorami
- Liczba kanałów komunikacyjnych rośnie
- Potencjalna kompaktowość konstrukcji



MIMD - topologie (6)

Mesh topology:

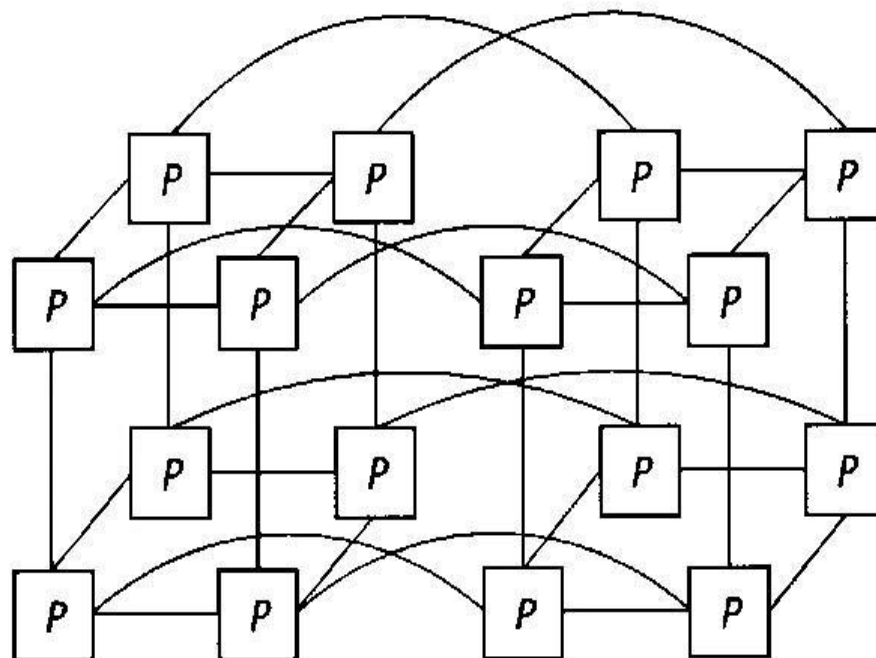
- Każdy procesor ma sąsiadów uporządkowanych
- Zapętlenia nie muszą być realizowane
 - ani wertykalnie
 - ani horyzontalnie



MIMD - topologie (7)

Hypercube:

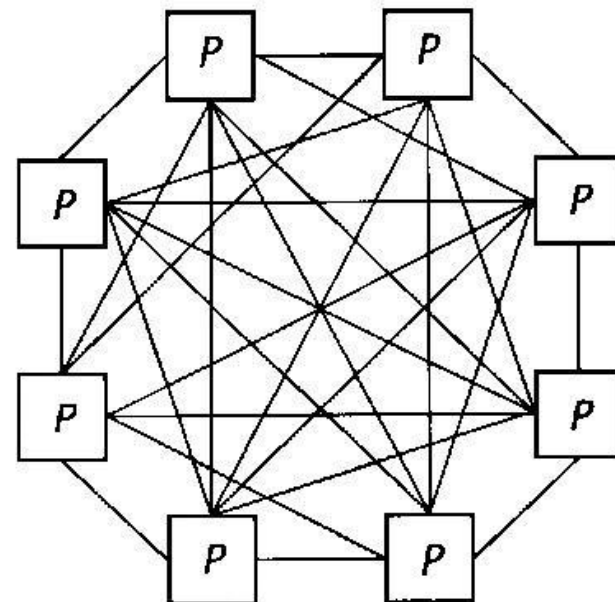
- Wielowymiarowość
- n procesorów
- Każdy ma $\log n$ połączeń



MIMD - topologie (8)

Completely Connected:

- Każdy procesor ma $n-1$ połączeń, jeden do wszystkich pozostałych
- Ukomplikowanie rośnie wraz z systemem
- Możliwości komunikacyjne!



MIMD - topologie (9)

TOPOLOGY	DIAMETER	BANDWIDTH	BISECTION BANDWIDTH
Shared	l	$1 * l$	$1 * l$
Ring	$\lfloor n / 2 \rfloor$	$n * l$	$2 * l$
Tree	$2 \lfloor \lg n \rfloor$	$(n - 1) * l$	$1 * l$
Mesh *	$2 \sqrt{n}$	$2n - 2 \sqrt{n}$	$2 \lceil \sqrt{n} / 2 \rceil * l$
Mesh **	$\sqrt{n}$	$2n * l$	$2 \sqrt{n} * l$
Hypercube	$\lg n$	$(n/2) * \lg n * l$	$(n/2) * l$
Comp. Con.	1	$(n/2)*(n-1) * l$	$(\lfloor n/2 \rfloor * \lceil n/2 \rceil) * l$

* bez „zawinięcia”

** z „zawinięciem”

l = przepustowość magistrali

n = liczba procesorów

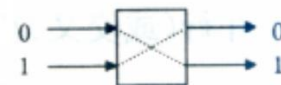
MIMD - topologie (10)

Dynamiczne:

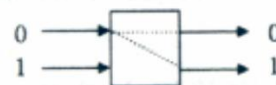
- przełączanie na żądanie



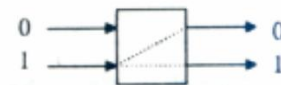
(a)



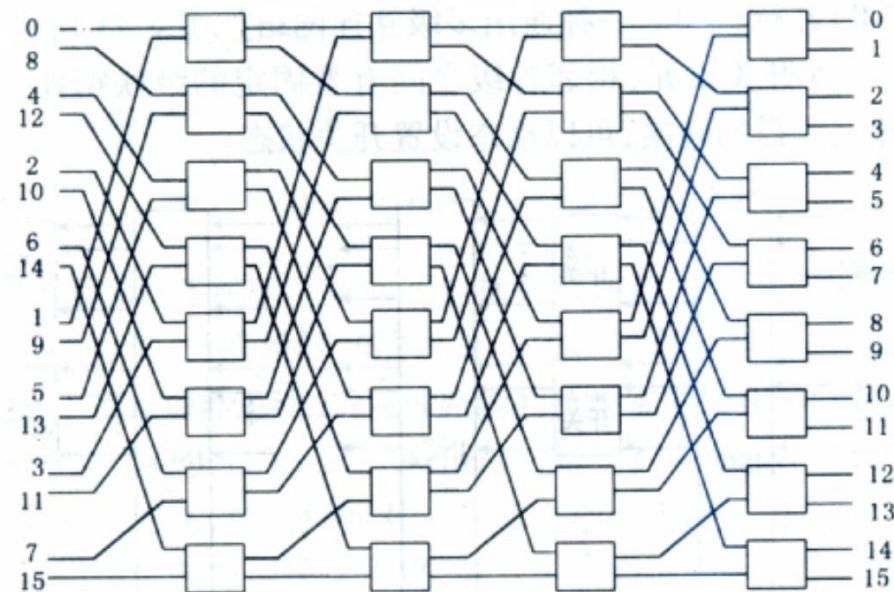
(b)



(c)



(d)



(e)



Paralelno - czyli jak?

SIMD

Early MPP System

MIMD

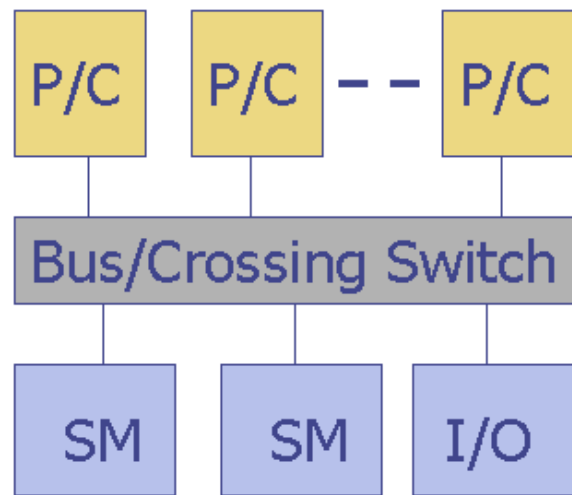
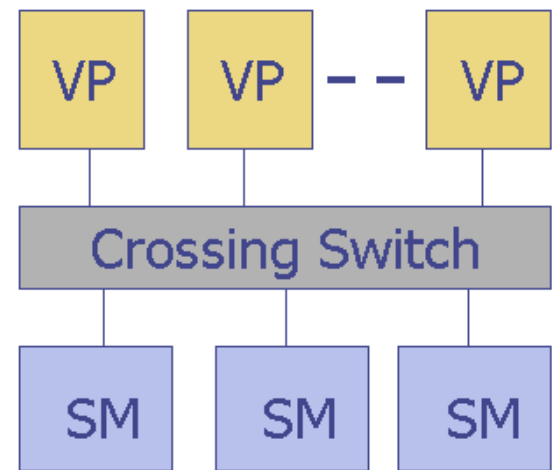
SPMD/MPMD

PVP (Parallel Vector Processor)
SMP (Symmetric Multiprocessor)
MPP (Massively Parallel Processor)
DSM (Distributed Shared Memory)
COW (Cluster of Workstation)
Griding Computation
Multi-Core CPU

MIMD - typy architektur (1)

Symmetric multiprocessor (SMP): Parallel Vector Processor (PVP)

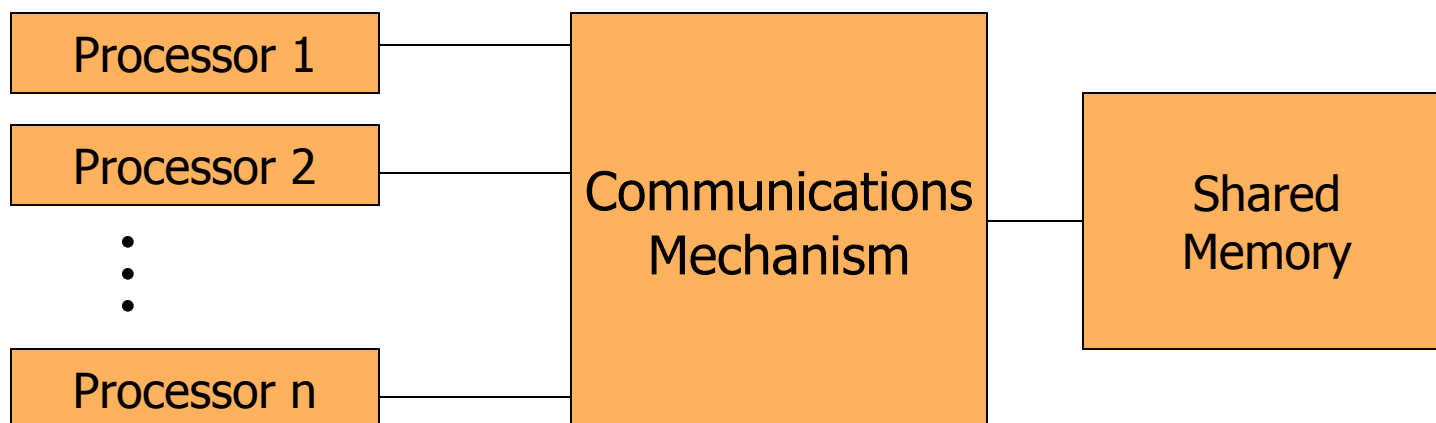
- System ma minimum 2 lub dużo więcej procesorów o porównywalnych możliwościach
- 4 typy:
 - Uniform memory access (UMA)
 - Nonuniform memory access (NUMA)
 - Cache coherent NUMA (CC-NUMA)
 - Cache only memory access (COMA)



MIMD - typy architektur (2)

Uniform memory access (UMA):

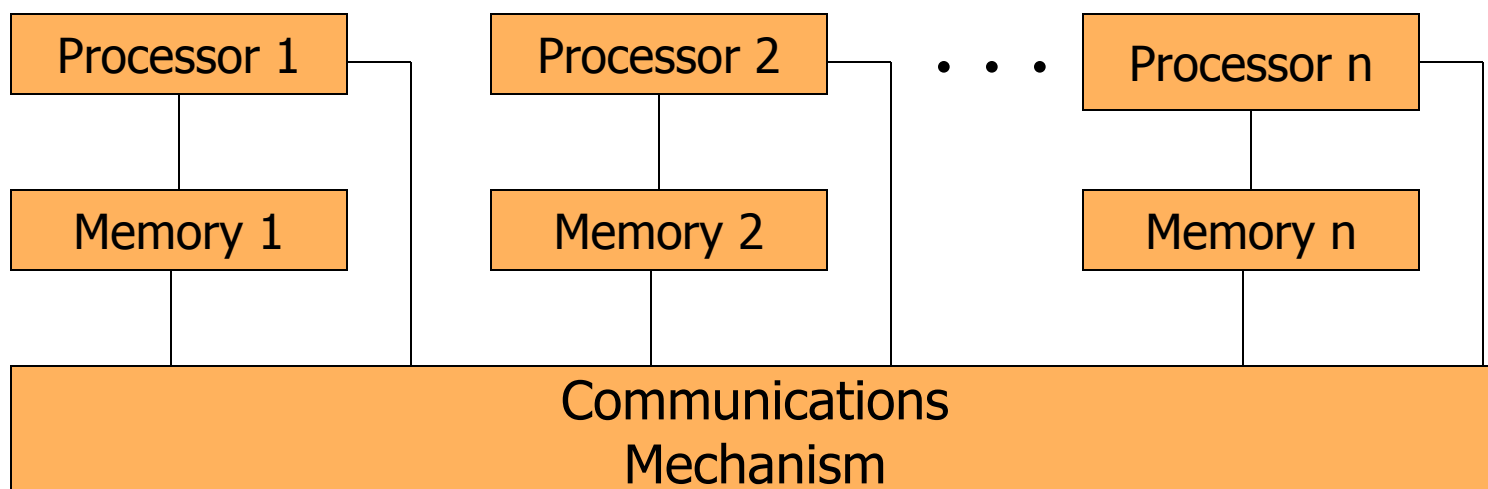
- Zunifikowany dostęp do dzielonych zasobów pamięci
- Każdy procesor może mieć swój cache - który nie jest wprost dostępny innym procesorom



MIMD - typy architektur (3)

Nonuniform memory access (NUMA):

- Dostęp do wszystkich lokacji pamięci nie jest jednakowy
- Dostęp do niektórych lokacji pamięci jest szybszy niż do innych, aczkolwiek każdy fragment pamięci jest osiągalny



MIMD - typy architektur (4)

Cache Coherent NUMA (CC-NUMA):

- Podobnie jak NUMA - każdy procesor ma cache
- Cache może gromadzić dane z innych
 - nie tych lokalnych dla procesora - miejsc pamięci
 - bonus!
- Może być, że te same dane będą zgromadzone w kilku cache-ach
- Rozwiązanie: Cache Only Memory Access (COMA)

MIMD - typy architektur (5)

Cache Only Memory Access (COMA):

- Pamięć lokalna procesora
 - jak cache
- Kiedy procesor potrzebuje danych, których w lokalnej pamięci nie ma
 - są one tam ładowane z pamięci podstawowej

MIMD - typy architektur (6)

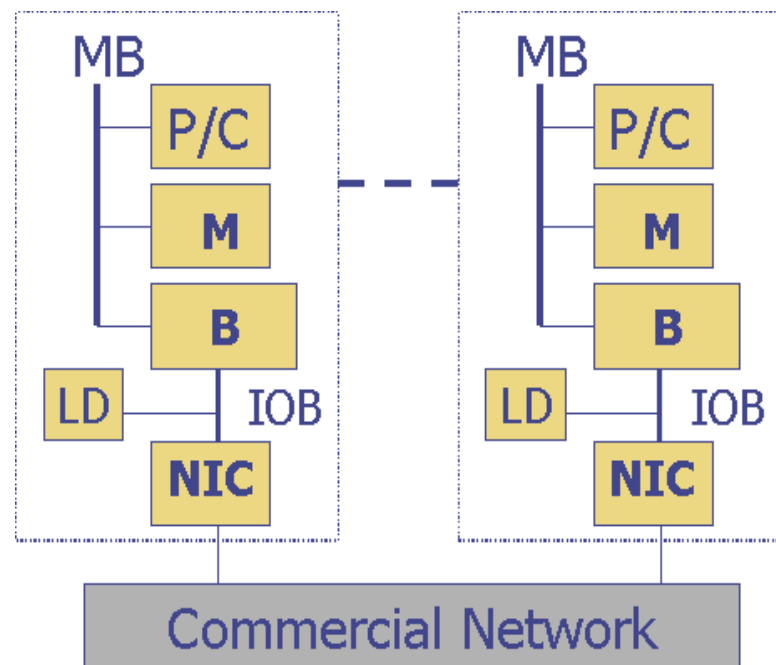
Multicomputer:

- MIMD – nie wszystkie procesory są kontrolowane jednym systemem operacyjnym
- Grupy procesorów / każdy procesor zarządzany odrębnymi systemami
- Dwa typy:
 - Network or Cluster of Workstations (NOW or COW)
 - Massively Parallel Processor (MPP)

MIMD - typy architektur (7)

Network of Workstation (NOW) lub Cluster of Workstation (COW):

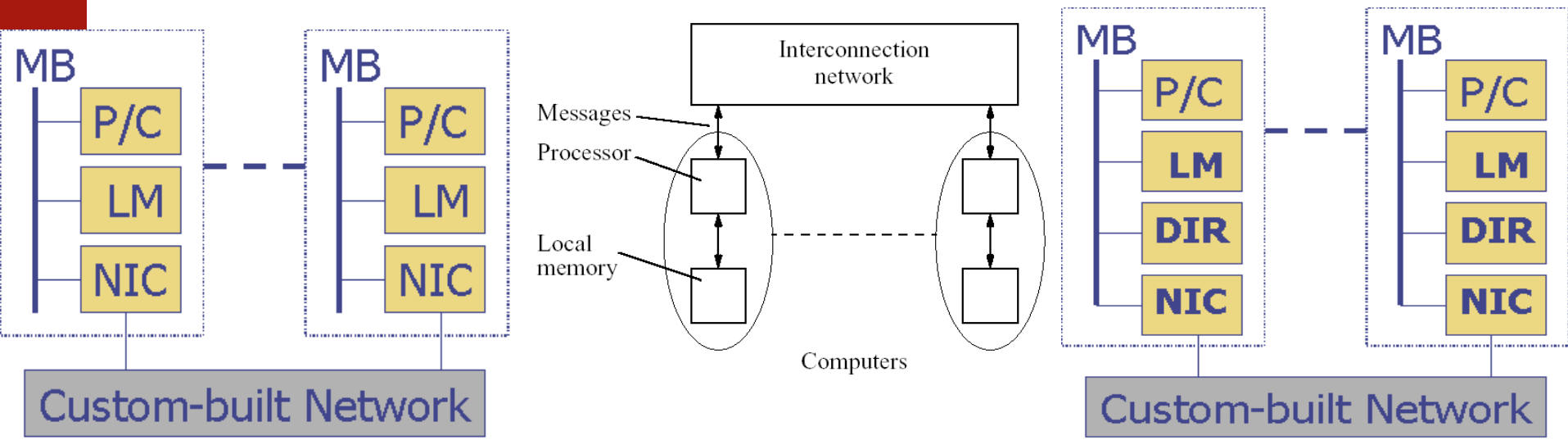
- Grupa pracuje spięta siecią LAN
- Jest scheduler, który rządzi i zarządza pracą



MIMD - typy architektur (8)

Massively Parallel Processor (MPP):

- Mnogość węzłów - każdy ma procesor, pamięć, sprzęt do funkcjonowania autonomicznego
- Dzielona pamięć daje szansę komunikacji
- Przykład: IBM's Blue Gene Computer



Miary i parametry (1)

- Computation/Communication Ratio

$$\frac{\text{Computation Time}}{\text{Communication Time}} = \frac{t_{\text{comp}}}{t_{\text{comm}}}$$

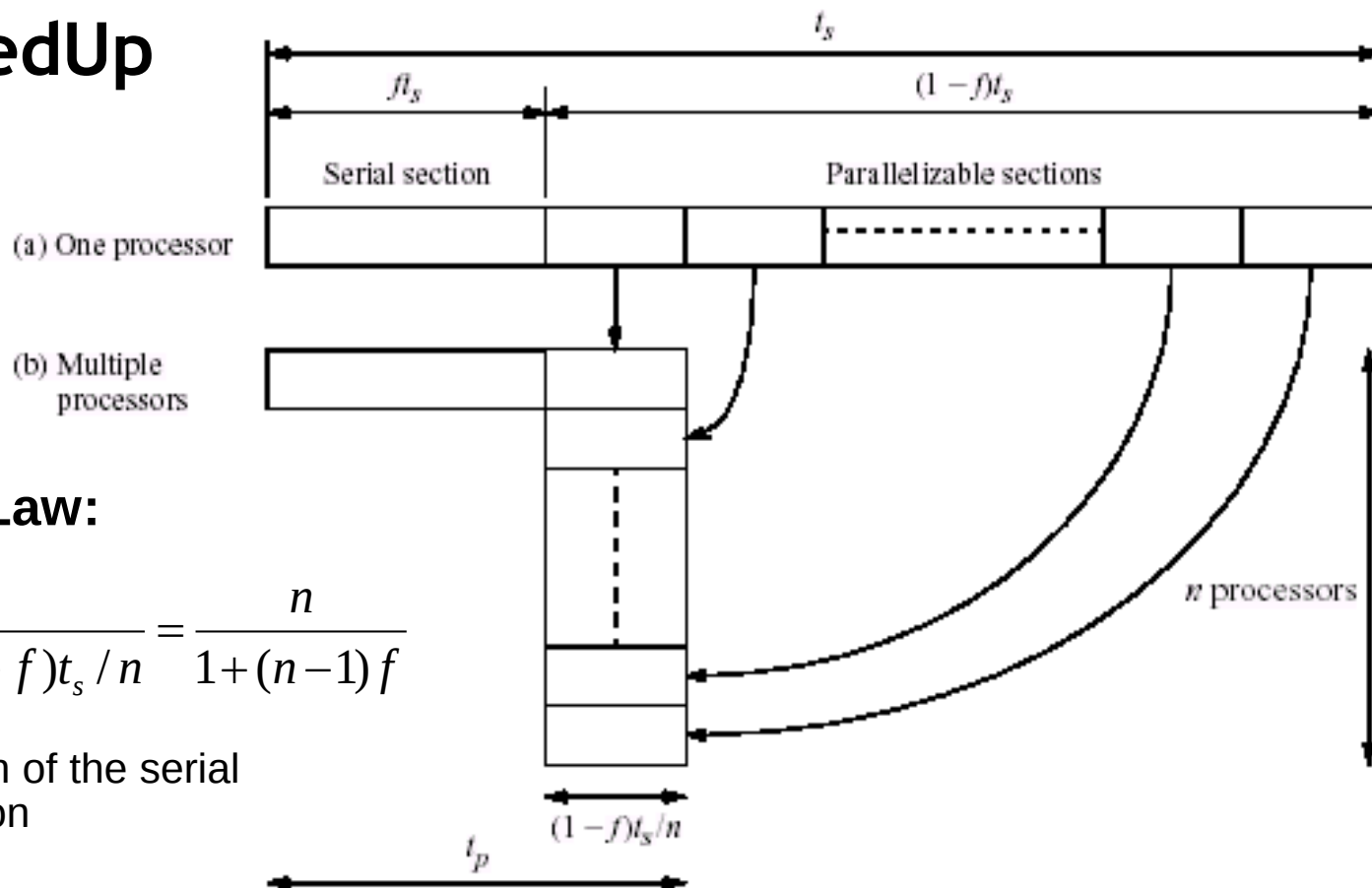
- Speedup Factor

$$S(n) = \frac{\text{Execution time (one processor system)}}{\text{Execution time (multiprocessor system)}} = \frac{t_s}{t_p}$$

- Ziarnistość zadania
- Miara R-to-C
 - duża: mało komunikacji, poważne bloki rachunkowe
 - mała: dużo komunikacji, bloki obliczeniowe małe
- FTC, skalowalność

Miary i parametry (2)

Max SpeedUp



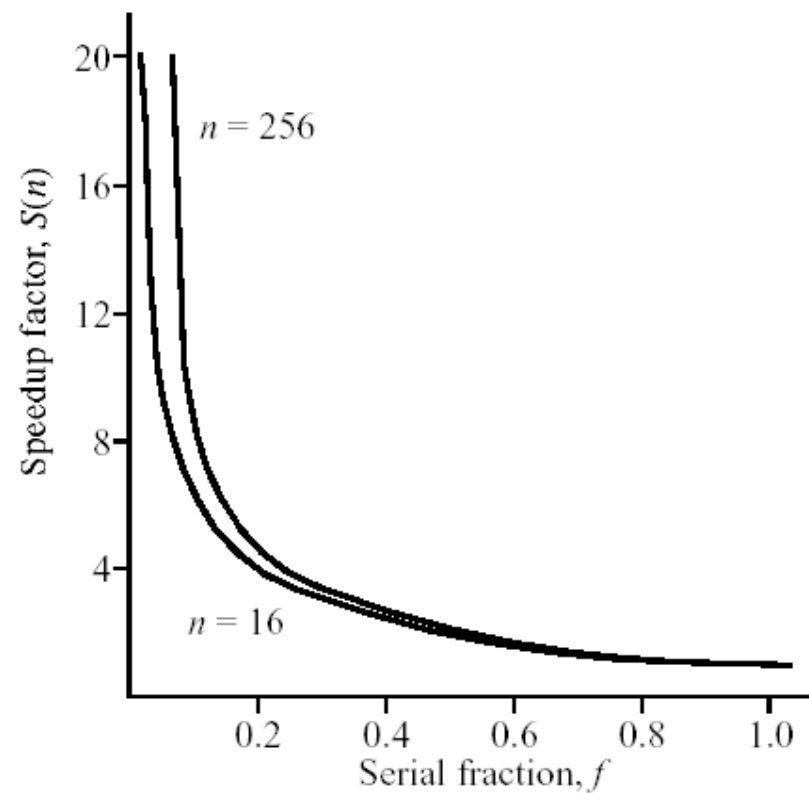
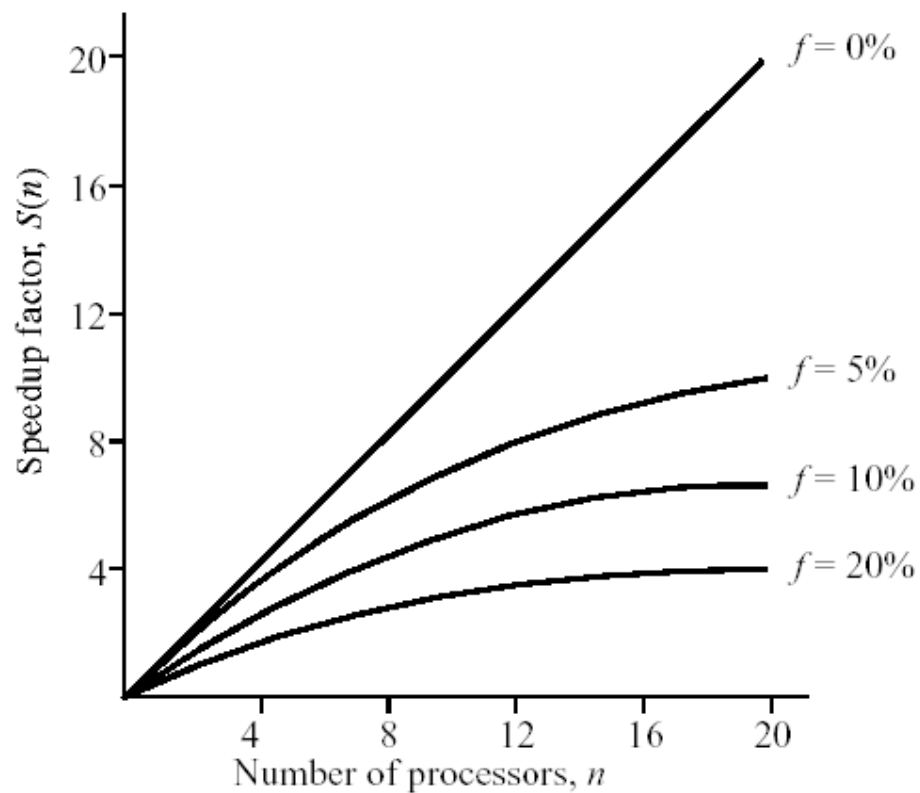
Amdahl's Law:

$$S(n) = \frac{t_s}{ft_s + (1-f)t_s/n} = \frac{n}{1 + (n-1)f}$$

f: The fraction of the serial computation



Miary i parametry (3)



Miary i parametry (4)

$$E = \frac{\text{Execution time using one processor}}{\text{Execution time using a multiprocessor} \times \text{number of processors}}$$

$$\textbf{Efektywność} = \frac{t_s}{t_p \times n} \qquad E = \frac{S(n)}{n} \times 100\%$$

The *processor-time* product or *cost* (or *work*) of a computation defined as

$$\text{Cost} = (\text{execution time}) \times (\text{total number of processors used})$$

The cost of a sequential computation is simply its execution time, t_s . The cost of a parallel computation is $t_p \times n$. The parallel execution time, t_p , is given by $t_s/S(n)$.

Hence, the cost of a parallel computation is given by

$$\text{Cost} = \frac{t_s n}{S(n)} = \frac{t_s}{E}$$