Politechnika Wrocławska

Podstawy Techniki Mikroprocesorowej

wykład 11: MIMD-koprocesor-CRAY

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

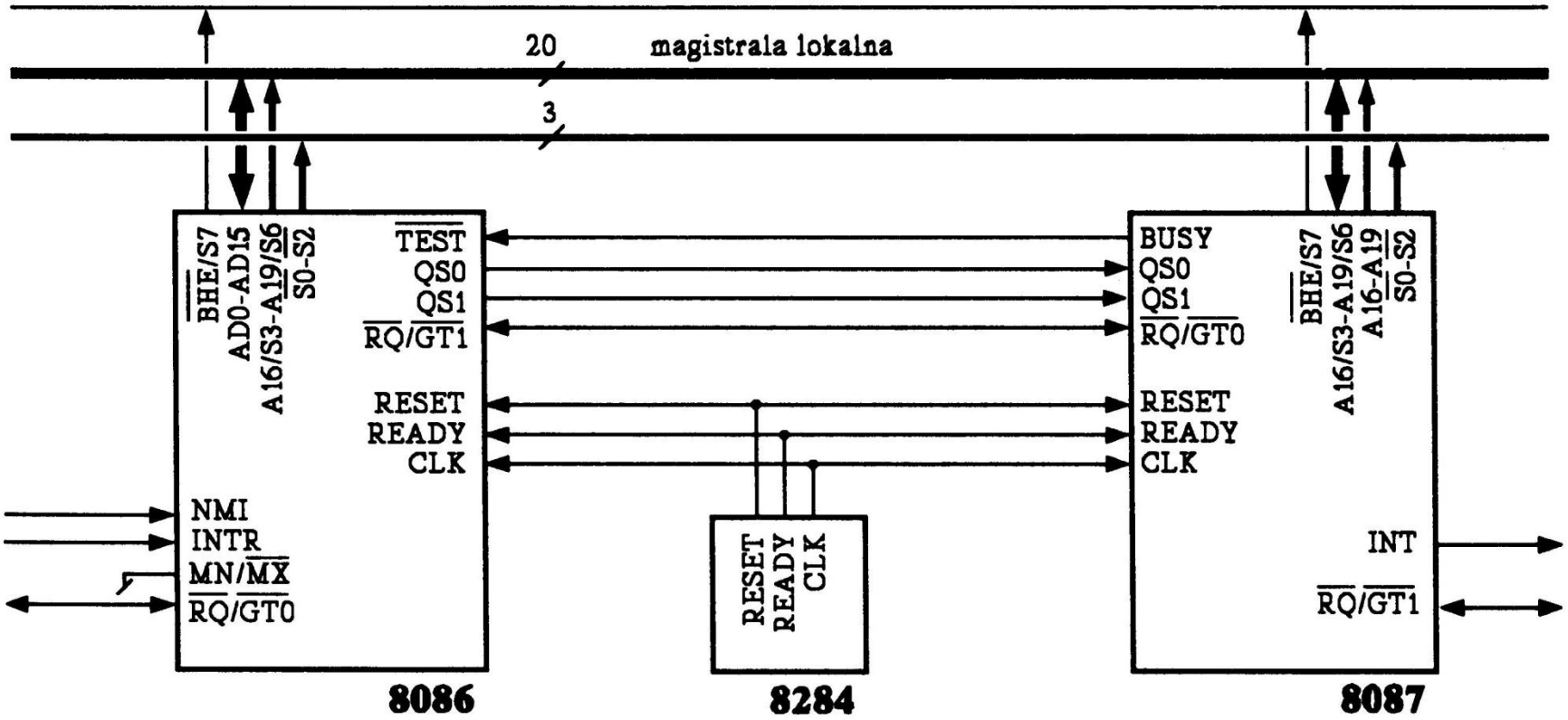
Procesor i koprocesor – 2 w 1

- kiedyś odrębny układ – wersja LUX
- teraz w jednej obudowie
- dwoistość architektury nadal widoczna
- każdy ma swoje rejestry
- koprocesor sam „nie dźwignie”
- nie obsługuje I/O
- nie obsługuje przerwań
- z pamięcią „daje radę”
  ale wymaga wsparcia
- procesor podstawowy „matkuje” koprocesorowi

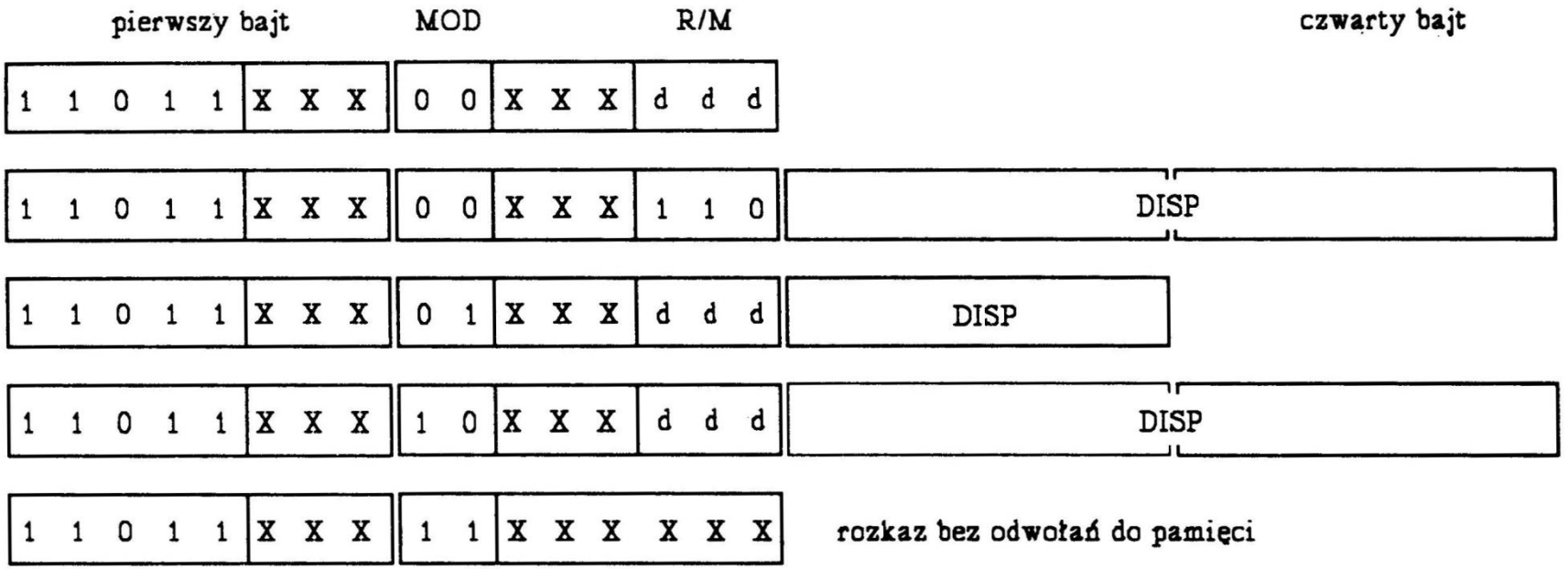
Współpraca procesora i koprocesora (1)

- procesor podstawowy pobiera rozkazy
- rozkazy i swoje, i dla koprocesora
- rozkazy dla koprocesora mają preambułę ESC
- w S2 – dekodowaniu łatwo rozpoznać
- procesor odcina preambułę
- przekazuje kod rozkazu do koprocesora
- procesor wykona jeszcze S3
- wyznaczy adresy efektywne argumentów dla koprocesora
- bez koprocesora też program da się zrobić
- koprocesor śledzi kolejkę rozkazów i stan ich wykonania

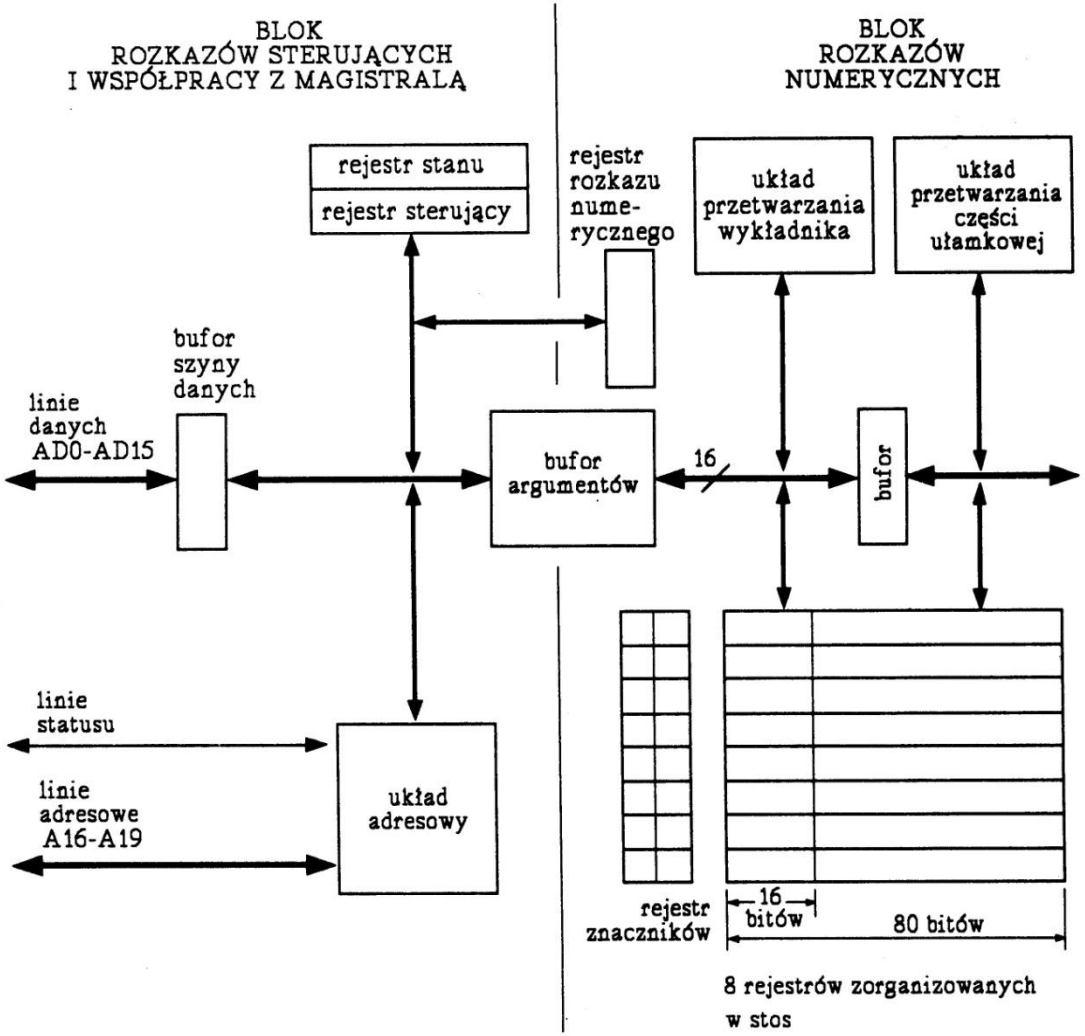
Współpraca procesora i koprocesora (2)



Współpraca procesora i koprocesora (3)



Architektura wewnętrzna



Formaty danych (1)

S	E	F	Obiekt
0	MAX	≠0	plus nieliczba
0	MAX	0	+∞
0	0 < E < MAX	≠0	liczba dodatnia
0	0	≠0	liczba dodatnia w postaci nieznormalizowanej
0	0	0	+0
1	0	0	−0
1	0	≠0	liczba ujemna w postaci nieznormalizowanej
1	0 < E < MAX	≠0	liczba ujemna
1	MAX	0	−∞
1	MAX	≠0	minus nieliczba

S — bit znaku, E — część wykładnicza (bity E_0, E_1, ...), F — część ułamkowa (bity F_0, F_1, ...).

Formaty danych (2)

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

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

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*)

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

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 (3)

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)

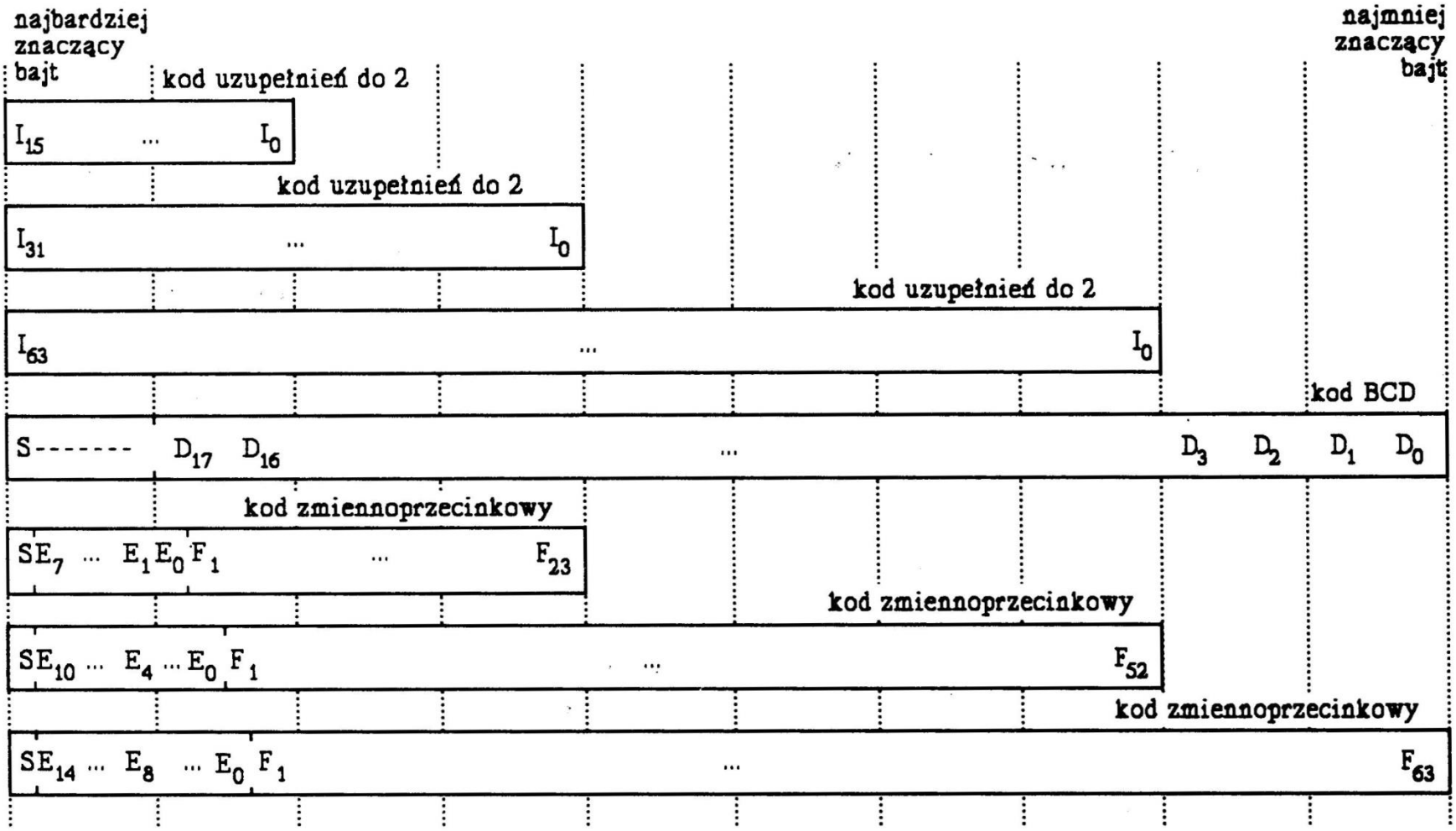
$$X = (-1)^S \sum_{j=0}^{17} D_j 10^j,$$

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 (4)



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;
C_3,C_2,C_1,C_0 (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), jedynkę (FLD1), liczbę π (FLDPI), $\log_2 10$ (FLDL2T), $\log_2 e$ (FLDL2E), $\log_{10} 2$ (FLDLG2) oraz $\log_e 2$ (FLDLN2).

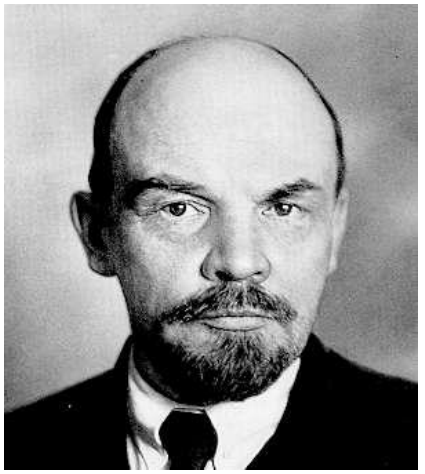
FDECSTP	— dekrementacja wskaźnika stosu;
FFREE	— opróżnienie rejestru;
FNOP	— nic nie rób;
FWAIT	— oczekiwanie procesora głównego na zakończenie rozkazu numerycznego; rozkaz ten nie jest wykonywany przez koprocesor, lecz zamieniany jest przez kompilator na rozkaz WAIT dla układu 8086.

Lista rozkazów (3)

FINIT, FNINIT	—	inicjacja koprocesora — powoduje opróżnienie rejestrów ustawienie rejestru sterującego;
FDISI, FNDISI	—	zablokowanie przerwań z koprocesora;
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 koprocesora
FLDENV	—	ładowanie rejestrów środowiskowych;
FRSTDR	—	odtworzenie stanu koprocesora;
FINCSTP	—	inkrementacja wskaźnika stosu;

„Kompjuter – eta jest’” i klasyfikacja

- jednostka centralna – procesor
- pamięć operacyjna
- urządzenia wejścia-wyjścia
- magistrale
  - von Neumann (!)



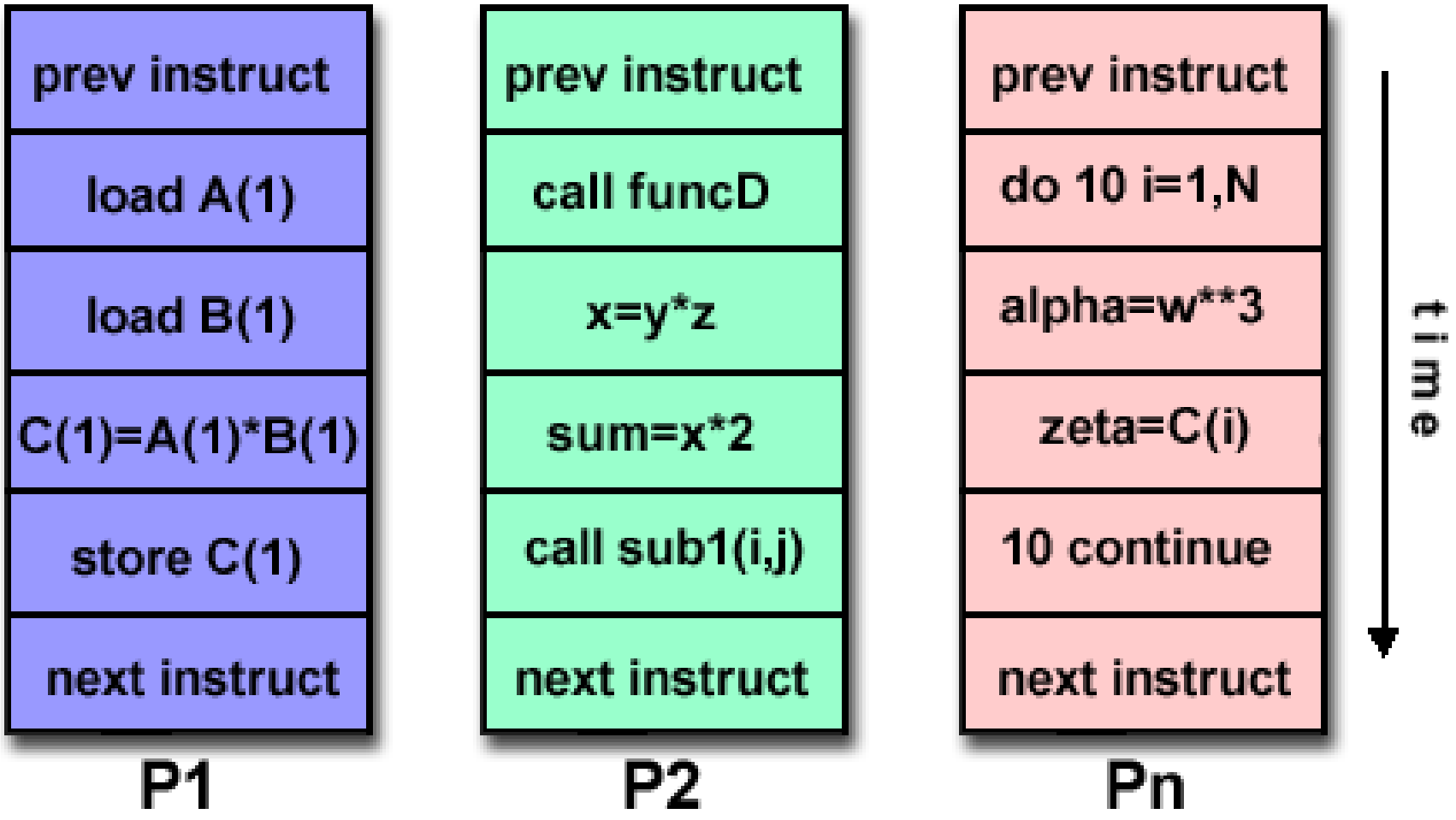
- SISD - Single Instruction Single Data
- SIMD - Single Instruction Multiple Data
- MISD - Multiple Instruction Single Data
- MIMD – Multiple Instruction Multiple Data

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)


prev instruct
load A(1)
load B(1)
C(1)=A(1)*B(1)
store C(1)
next instruct
P1
prev instruct
call funcD
x=y*z
sum=x*2
call sub1(i,j)
next instruct
P2
prev instruct
do 10 i=1,N
alpha=w**3
zeta=C(i)
10 continue
next instruct
Pn
time

MIMD - topologie (1)

Topologia:

- Topologia systemu wieloprocesorowego 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)

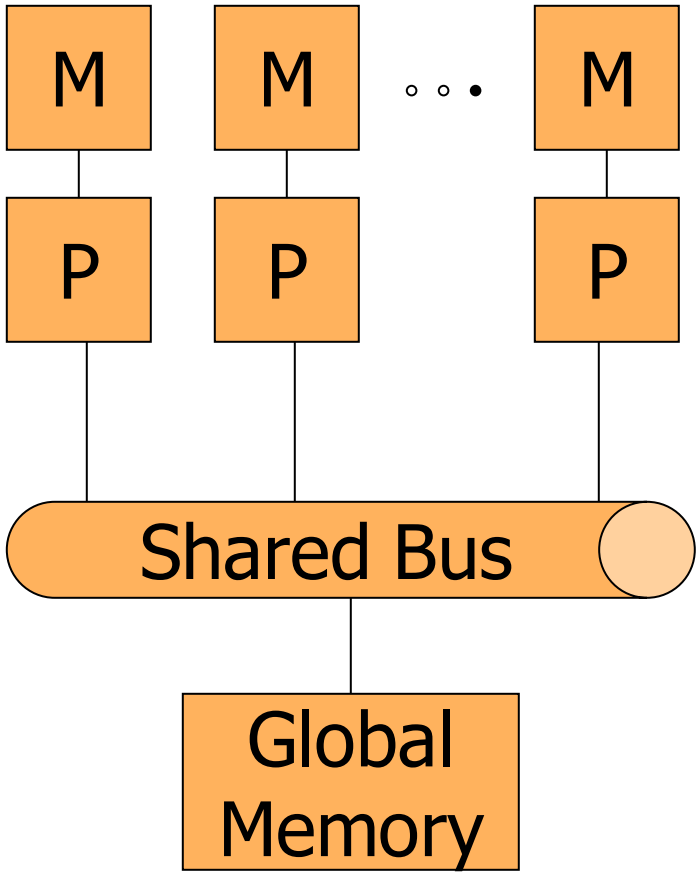
6 podstawowych kategorii topologicznych:

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

MIMD - topologie (3)

Shared Bus:

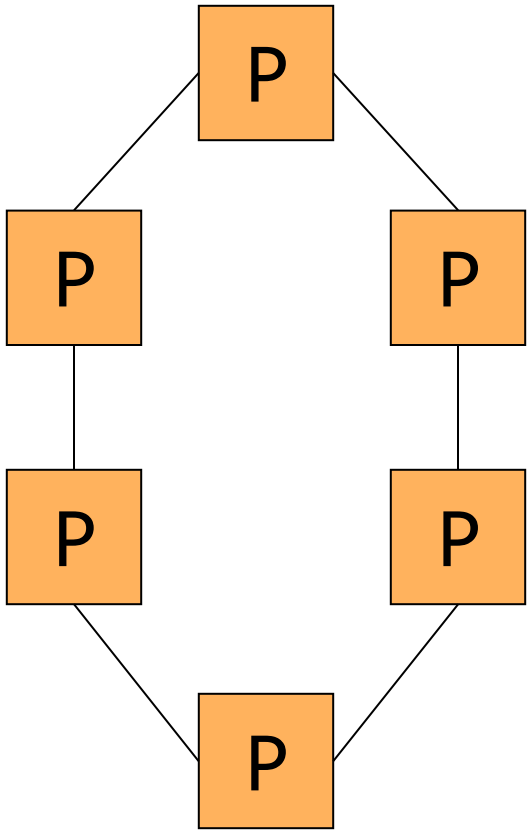
- Prostota konstrukcji
- Procesory komunikują się poprzez magistrale
- Warunki transmisji dyktuje obsługa magistrali
- Łatwość doczepienia nowych procesorów – jeśli zagwarantujemy właściwe sterowanie



MIMD - topologie (4)

Ring:

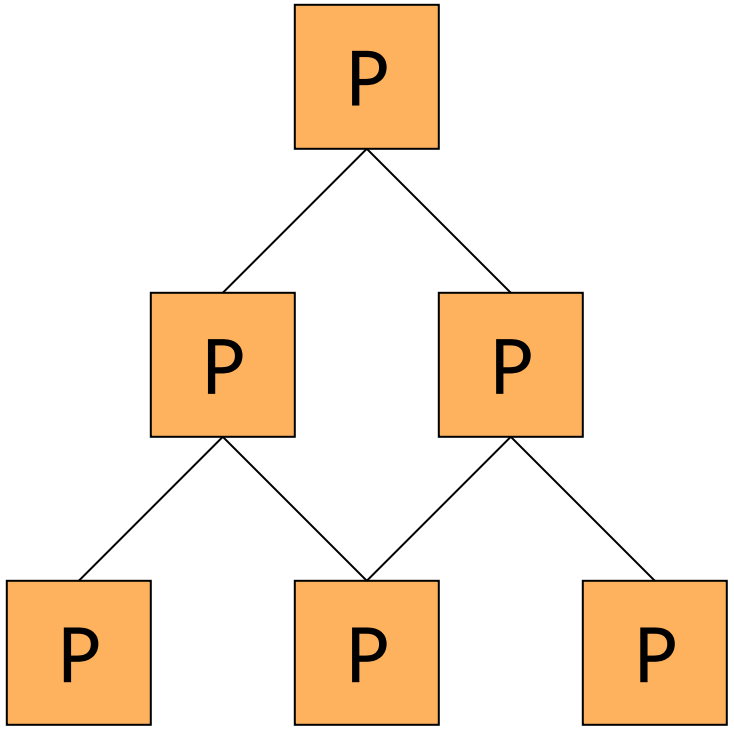
- Dedykowane połączenia miedzy procesorami
- Równoczesna komunikacja jest w zasięgu
- Wędrówka danych przez pośredników
- Każdy procesor ma 2 kanały komunikacyjne



MIMD - topologie (5)

Tree topology:

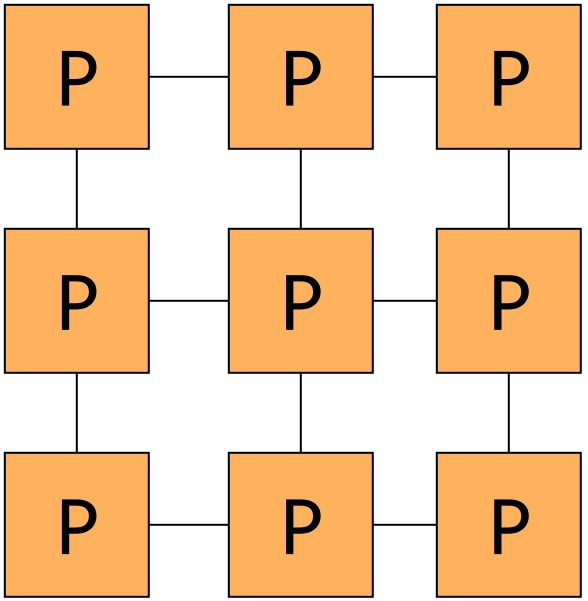
- Bezpośrednie połączenia między procesorami
- Liczba kanałów komunikacyjnych rośnie
- Potencjalna kompaktowość konstrukcji



MIMD - topologie (6)

Mesh topology:

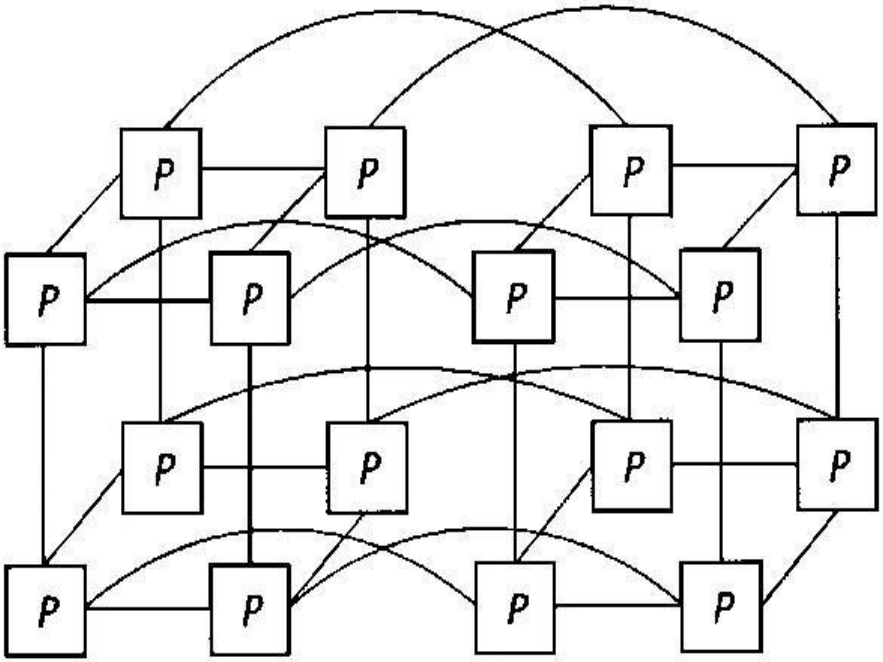
- Każdy procesor ma sąsiadów uporządkowanych
- Zapętlenia nie muszą byś realizowane
  - ani wertykalnie
  - ani horyzontalnie



MIMD - topologie (7)

Hypercube:

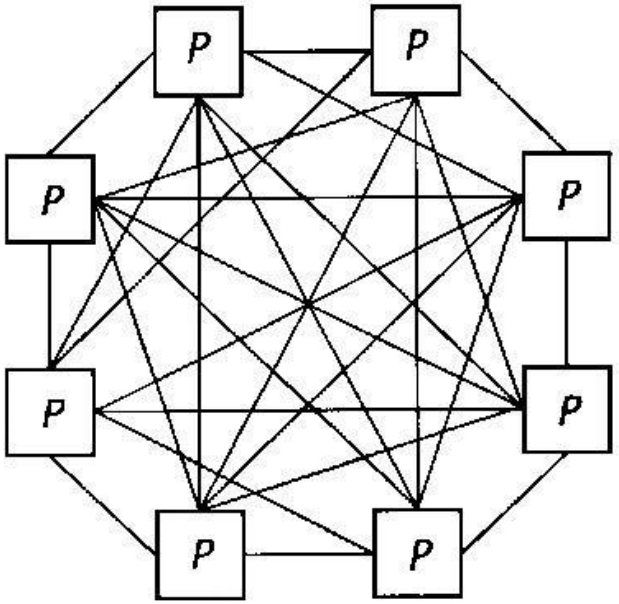
- Wielowymiarowość
- n procesorów
- Każdy ma log n połączeń



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

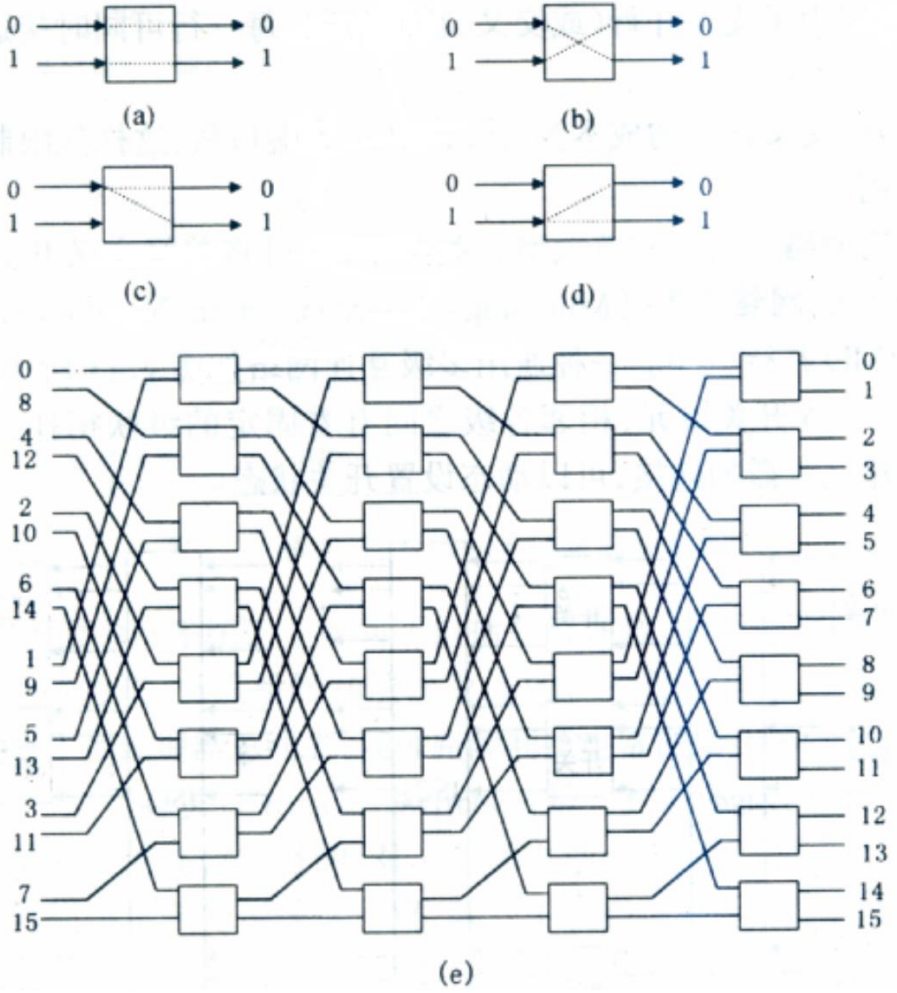
* bez „zawinięcia”
** z „zawinięciem”

l = przepustowość magistrali
n = liczba procesorów

MIMD - topologie (10)

Dynamiczne:

- przełączanie na żądanie



Parallelno - czyli jak?

SIMD — Early MPP System

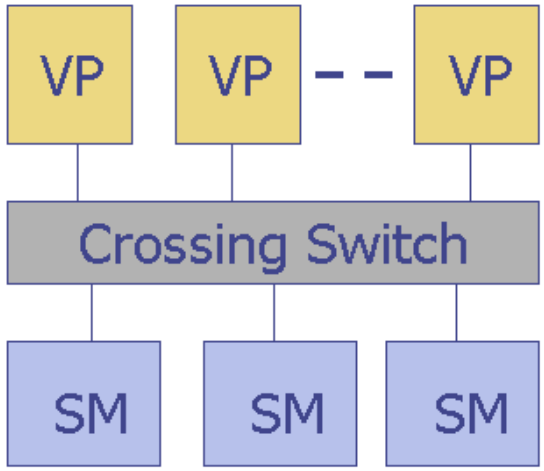
MIMD

SPMD/MPMD

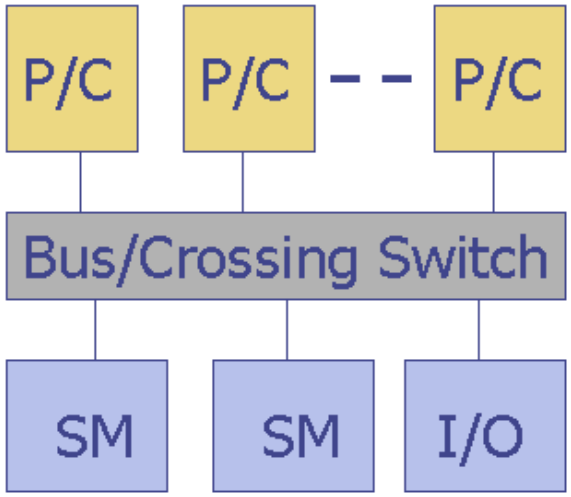
- PVP (Parallel Vector Processor)
- SMP (Symmetric Multiprocessor)
- MPP (Massively Parallel Processor)
- DSM (Distributed Shared Memory)
- COW (Cluster of Workstation)
- Griding Computation
- Multi-Core CPU

MIMD – typy architektur (1)

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



- System ma minimum 2 lub dużo więcej procesorów o porównywalnych możliwościach

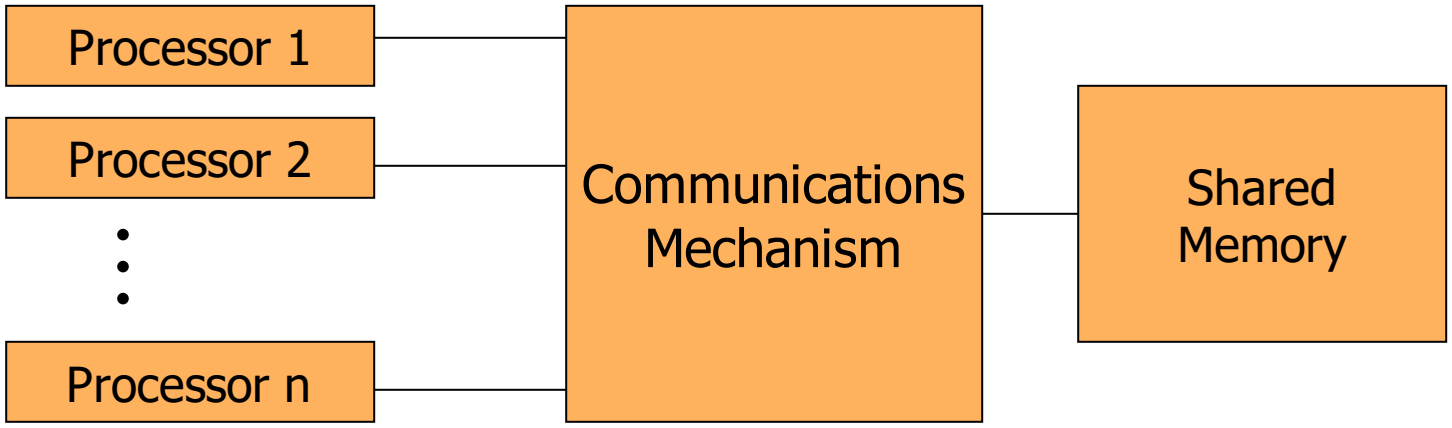


- 4 typy:
  - Uniform memory access (UMA)
  - Nonuniform memory access (NUMA)
  - Cache coherent NUMA (CC-NUMA)
  - Cache only memory access (COMA)

MIMD – typy architektur (2)

Uniform memory access (UMA):

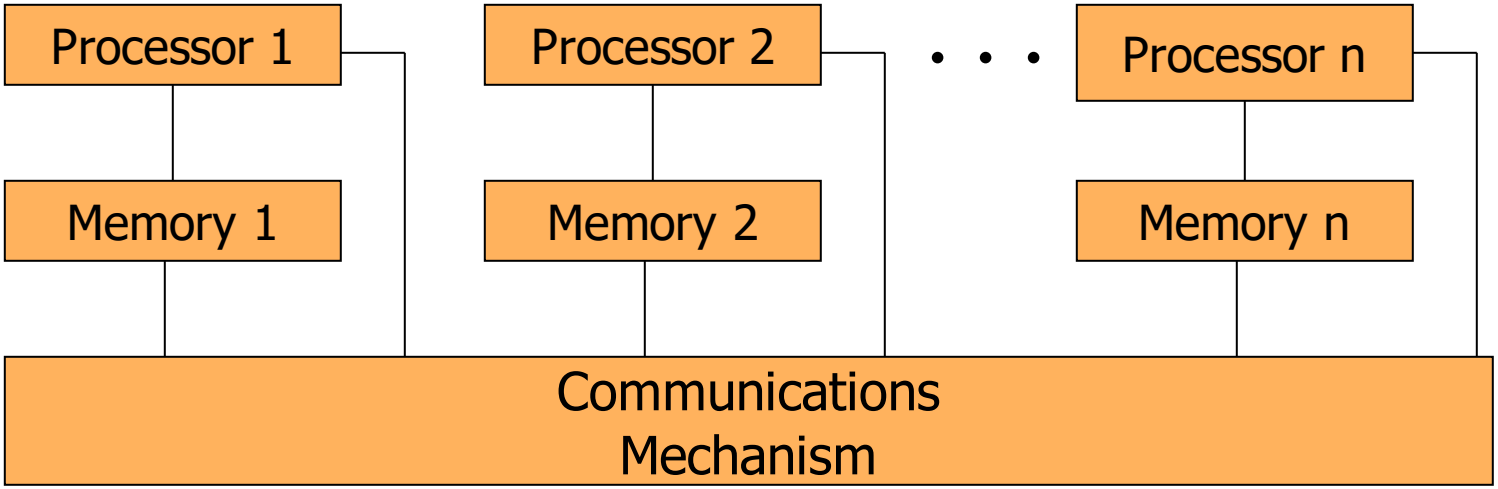
- Zunifikowany dostęp do dzielonych zasobów pamięci
- Każdy procesor może mieć swój cache – który nie jest wprost dostępny innym procesorom



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)

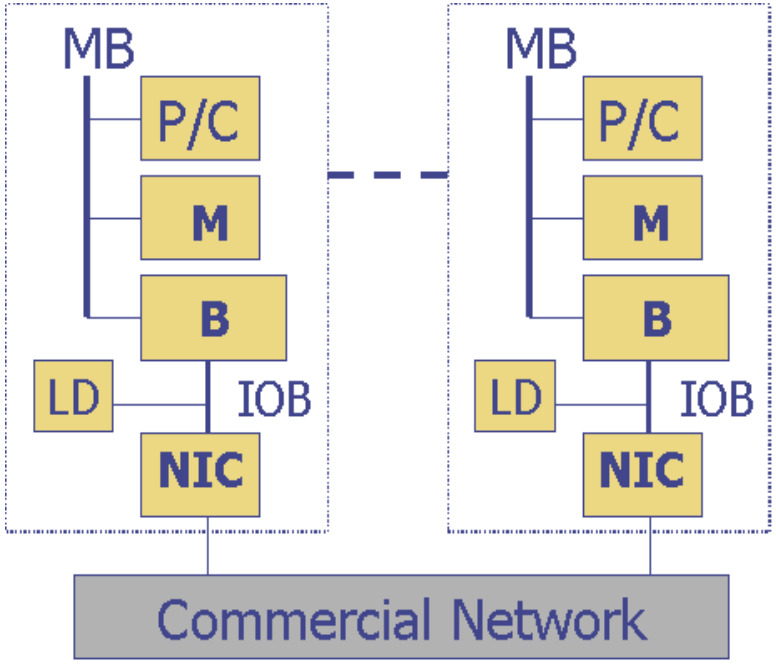
Multicomputer:

- 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):

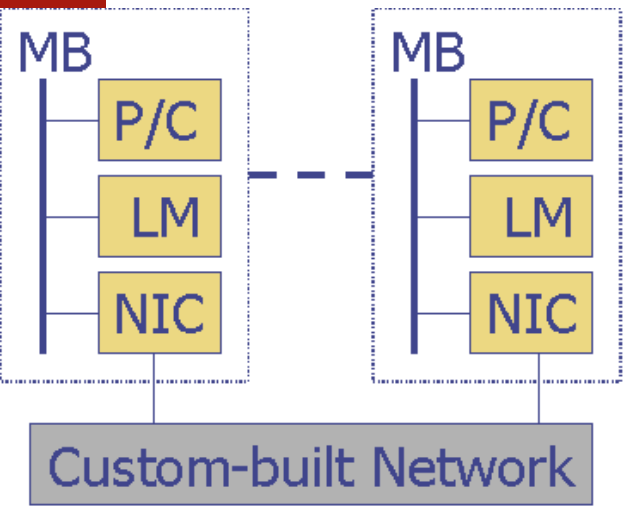
- Grupa pracuje spięta siecią LAN
- Jest scheduler, który rządzi i zarządza pracą

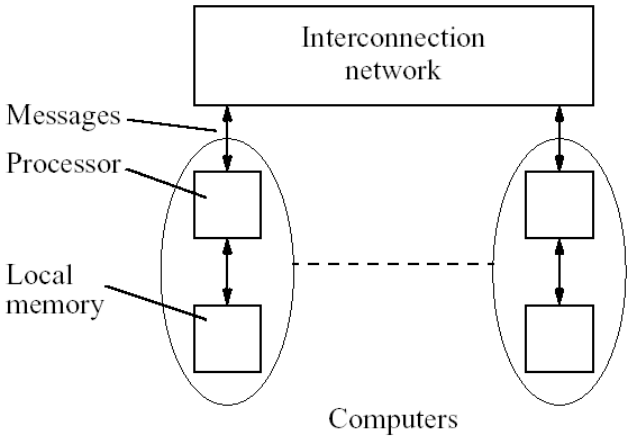


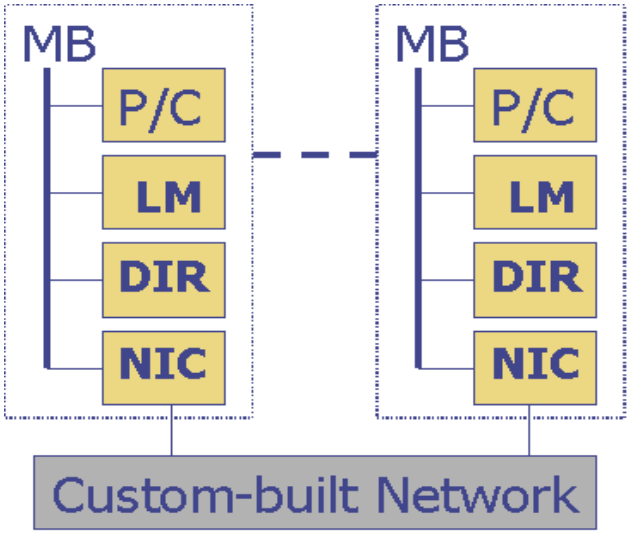
MIMD - typy architektur (8)

Massively Parallel Processor (MPP):

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







Miary i parametry (1)

- Computation/Communication Ratio

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

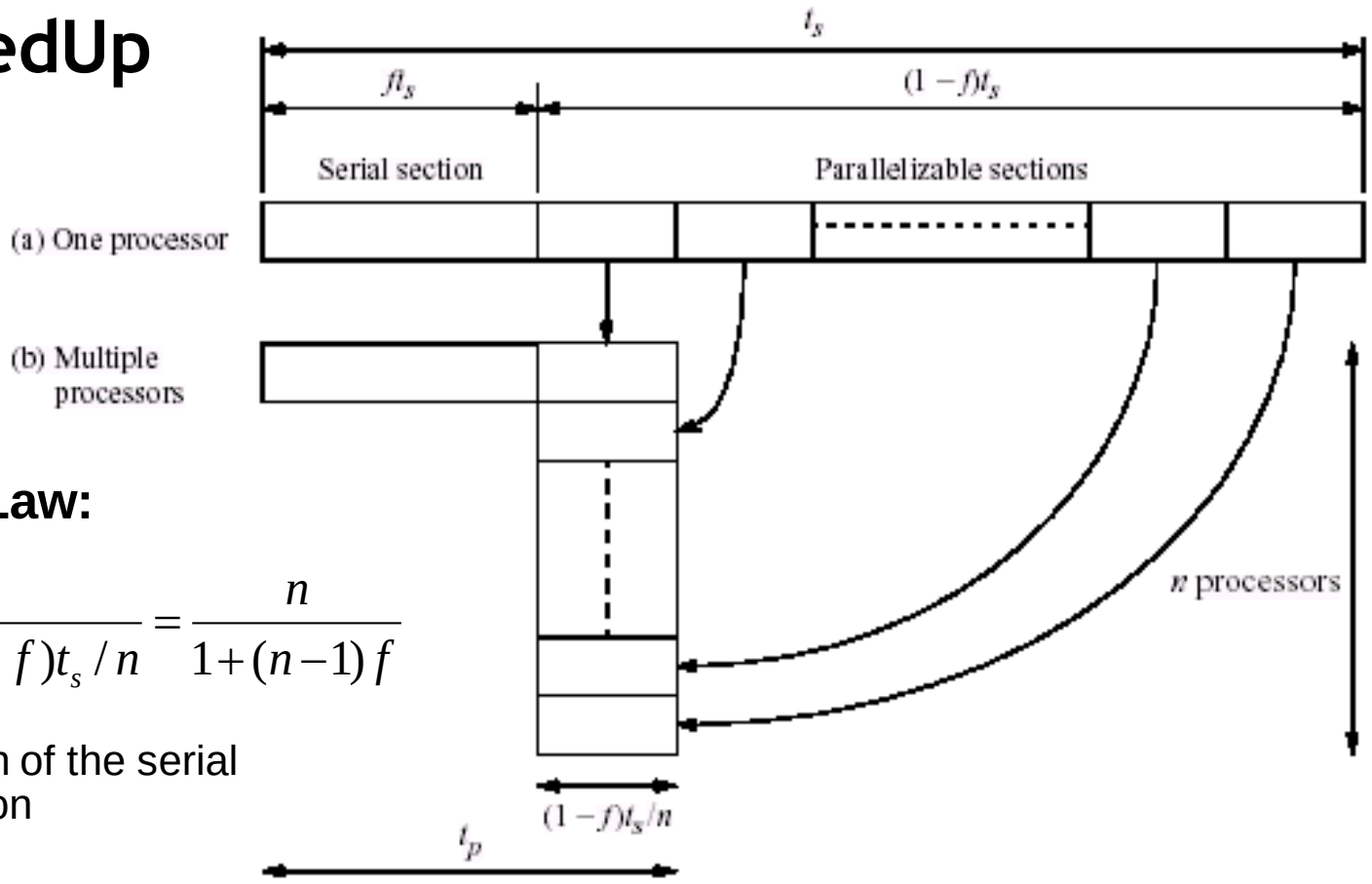
- Speedup Factor

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

- 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

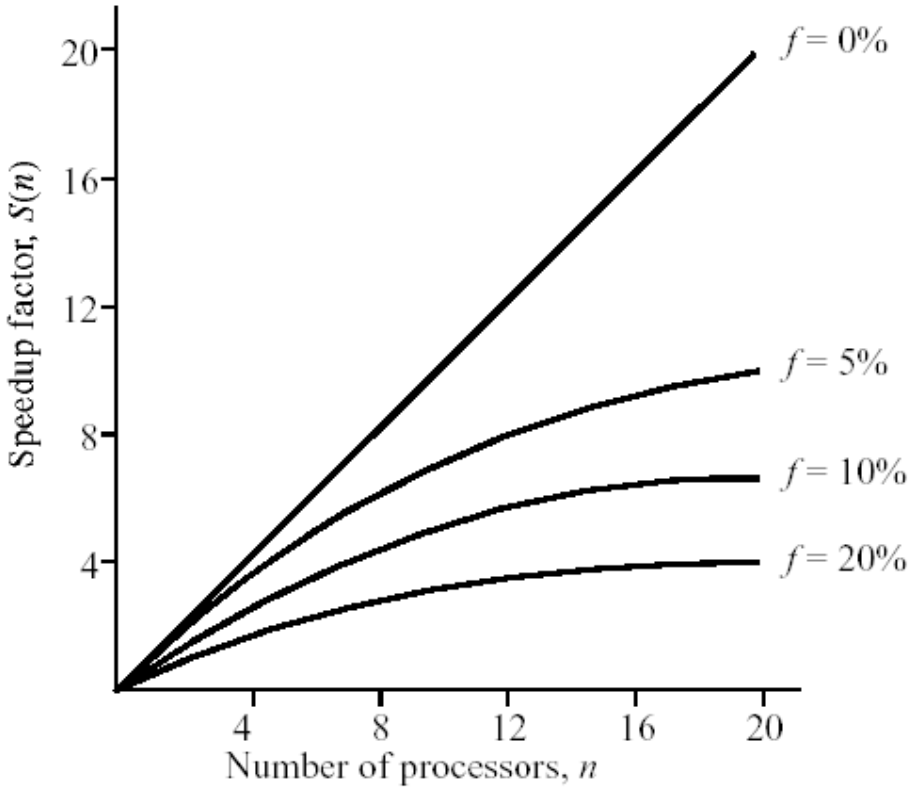


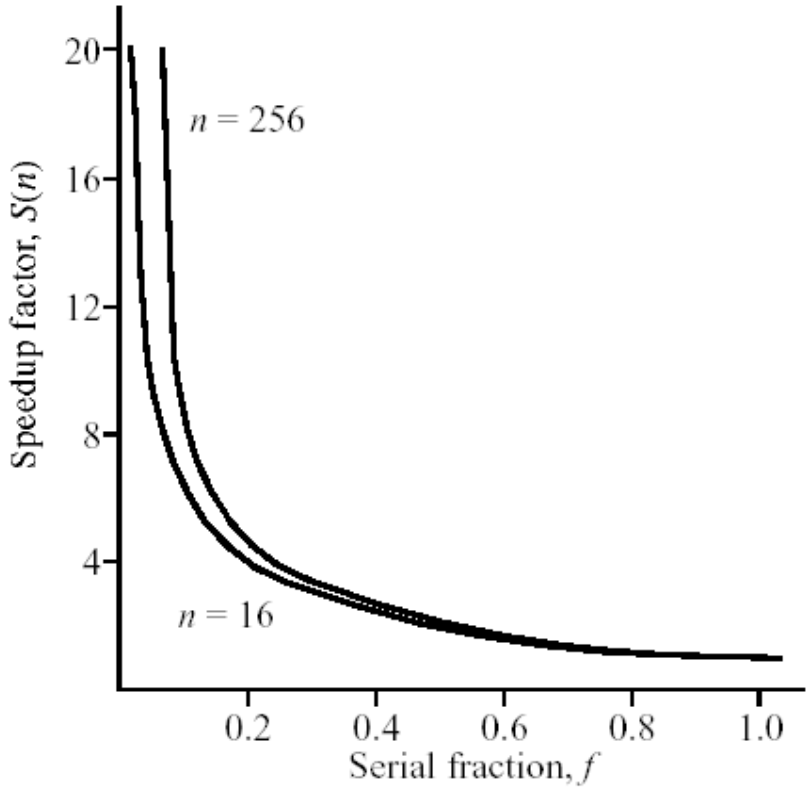
Amdahl's Law:

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

f: The fraction of the serial computation

Miary i parametry (3)





Miary i parametry (4)

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

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

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

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

CRAY - (1)

- Cray-1A ważył – wraz z freonowym systemem chłodzenia – 5.5 tony,
- skracanie do minimum połączeń kablowych. sekcji połączonych w kształt podkowy; najdłuższy przewód w systemie miał 122 cm,
- wykorzystywał procesor wektorowy i zawierał 200000 specjalizowanych układów ECL,
- 12.5 ns okres zegara (80 MHz), 8 rejestrów wektorowych zawierających po 64 słowa, oraz 1 milion 64-bitowych słów szybkiej pamięci (8MB RAM),
- ponad 80 milionów operacji zmiennopozycyjnych na sekundę (80 MFLOPS), późniejszym okresie ustanowił rekord szybkości na poziomie 133 MFLOPS.
- skonfigurowany z 1 milionem słów RAM, maszyna i jej systemy zasilające pobierały około 115 KW mocy; systemy chłodzące oraz pamięć dyskowa podwajały tą liczbę.

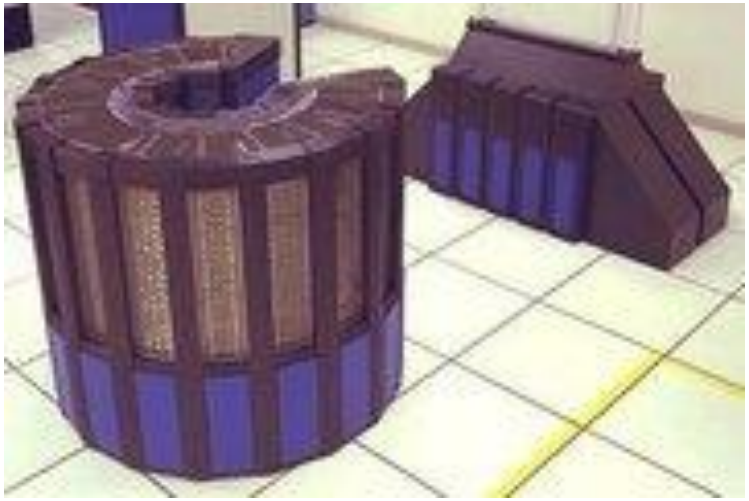
Cray Operating System (COS)
(potem UniCOS, odmiana UNIXa firmy Cray),
Cray Assembler Language (CAL),
Cray FORTRAN (CFT), pierwszy automatycznie wektoryzujący kompilator języka FORTRAN

CRAY - (2)

- **Cray X-MP** był pierwszą maszyną wieloprocesorową Cray-a i najszybszym komputerem na świecie w latach 1983–1985,
- odziedziczył obudowę w kształcie podkowy po swoim poprzedniku,
- zastosowano bardziej upakowane układy (8 krotnie w porównaniu do poprzedniego komputera),
- okres zegara wynosił tylko 8,5 ns dostarczając około 55 MFLOPS na procesor i 235 MFLOPS dla całej maszyny,
- procesory posiadały wsparcie dla łańcuchowania potoków, równoległe potoki artymetryczne i dostęp do pamięci współdzielonej poprzez kilka portów na procesor,
- był sprzedawany z 1 do 4 procesorów i z 1 do 16 megasłów (8–128 MB) głównej pamięci RAM zaprojektowano upgrade architektury adresacji pamięci który powiększał zakres adresacji pamięci do 2 GB.

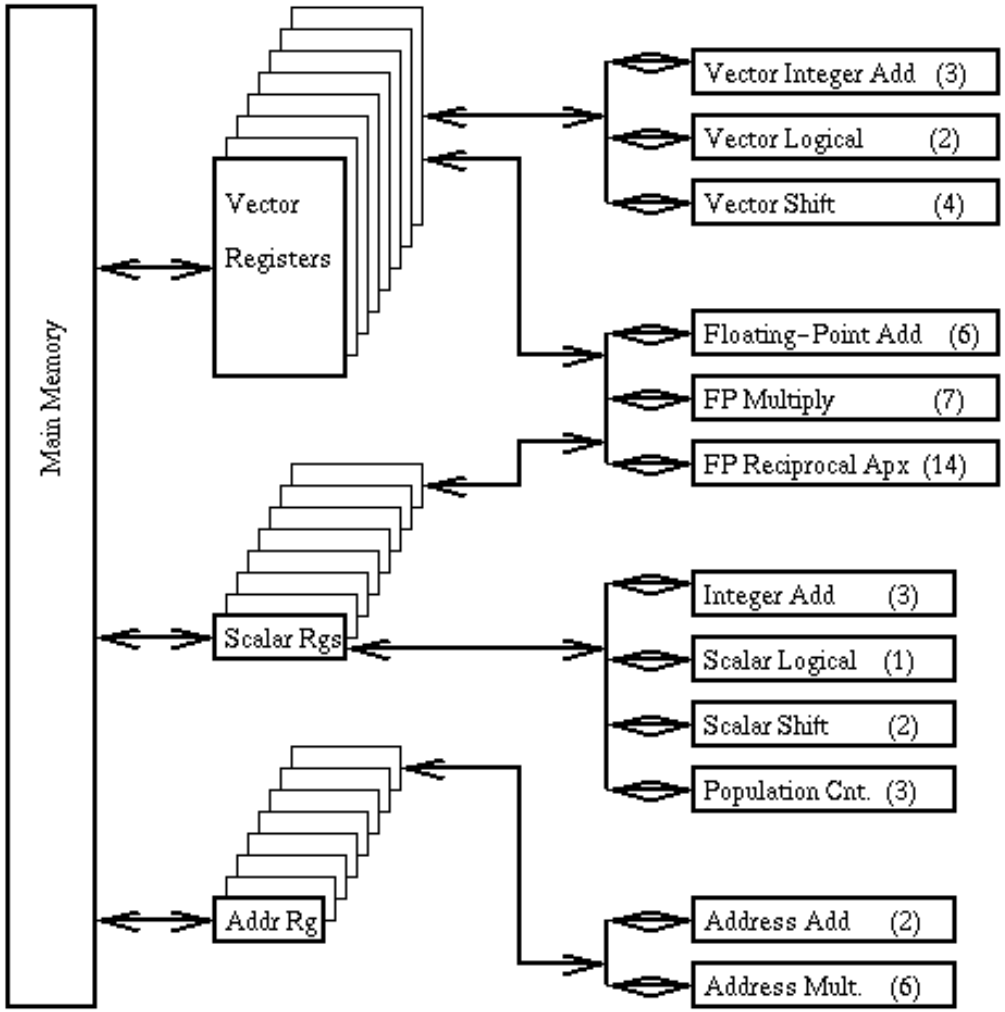
CRAY - (3)

- **Cray-2** był superkomputerem z procesorem wektorowym zaprojektowanym i oferowanym przez od poczatku 1985. Zastąpił Cray X-MP, który do tego czasu był najszybszym komputerem na świecie. Cray-2 został pokonany przez ETA-10G w 1990.
- Cray-2 był systemem wieloprocesorowym z pamięcią współdzieloną wykonanym w technologii ECL i układami z arsenku galu (GaAs).
- składał się z prostego procesora pierwszoplanowego który wykonywał zadania systemowe, obsługiwał zadania I/O oraz synchronizował pracę pozostałych elementów systemu, od 1 do 4 wektorowych procesorów drugoplanowych oraz bardzo szybkiej pamięci współdzielonej
- procesory drugoplanowe wykonywały właściwe obliczenia. Okres zegara wynosił 4,1 ns (244 MHz)

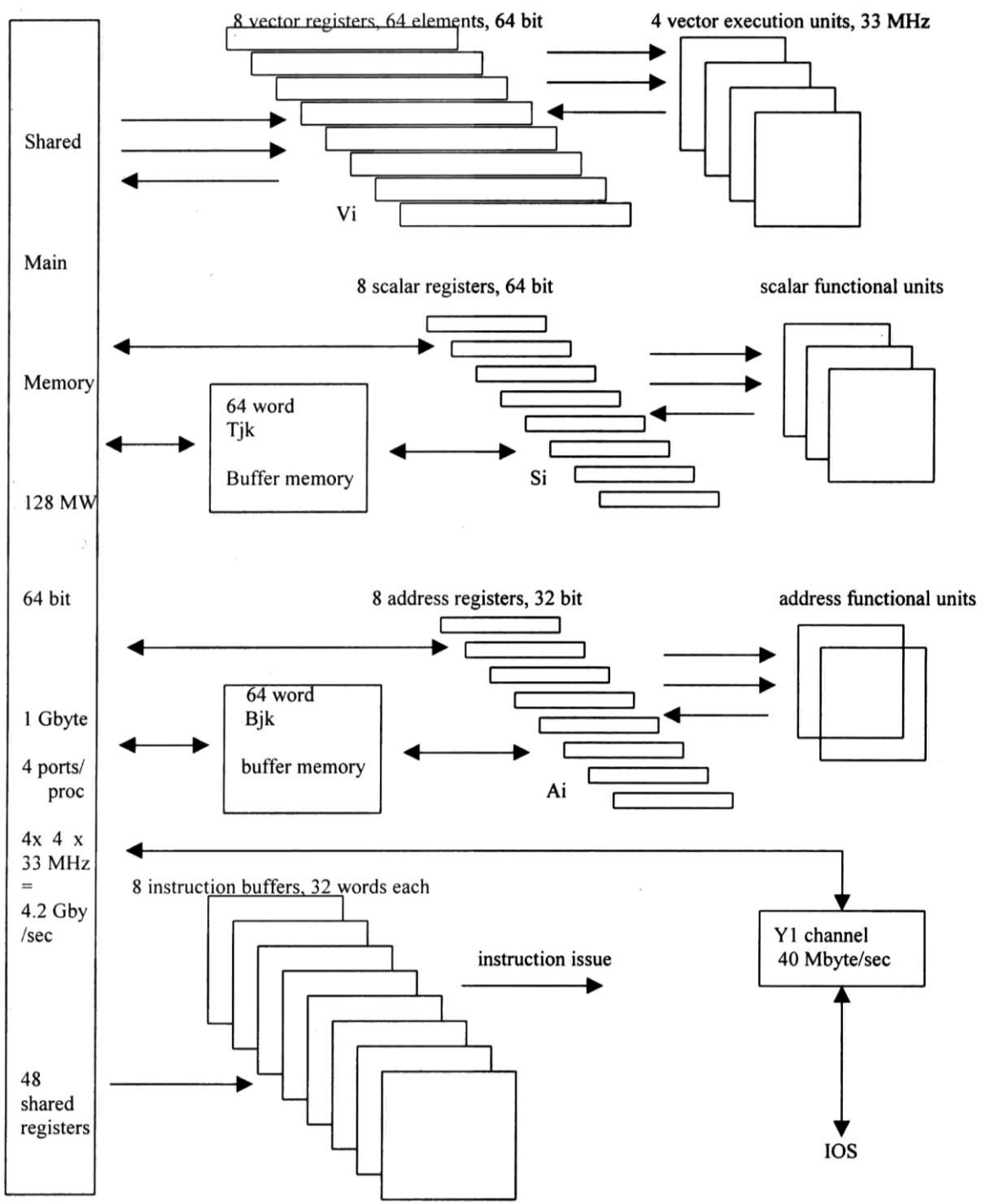


produkowany dla Amerykańskich Departamentów Obrony oraz Energii,
wykorzystywane były głównie do badań nad bronią nuklearną lub w badania oceanograficznych (za pomocą sonaru).

CRAY – architektura (4)



CRAY - (5)


8 vector registers, 64 elements, 64 bit
4 vector execution units, 33 MHz
Shared
Vi
Main
8 scalar registers, 64 bit
scalar functional units
Memory
64 word
Tjk
Buffer memory
Si
128 MW
64 bit
8 address registers, 32 bit
address functional units
1 Gbyte
64 word
Bjk
buffer memory
4 ports/
proc
Ai
4x 4 x
33 MHz
=
4.2 Gby
/sec
8 instruction buffers, 32 words each
instruction issue
Y1 channel
40 Mbyte/sec
48
shared
registers
IOS