



Politechnika Wrocławska

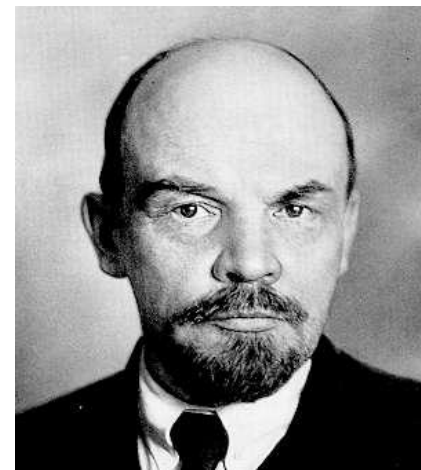
Podstawy Techniki Mikroprocesorowej

wykład 11: SIMD, transputery, koprocessor

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

„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





SIMD - wprowadzenie

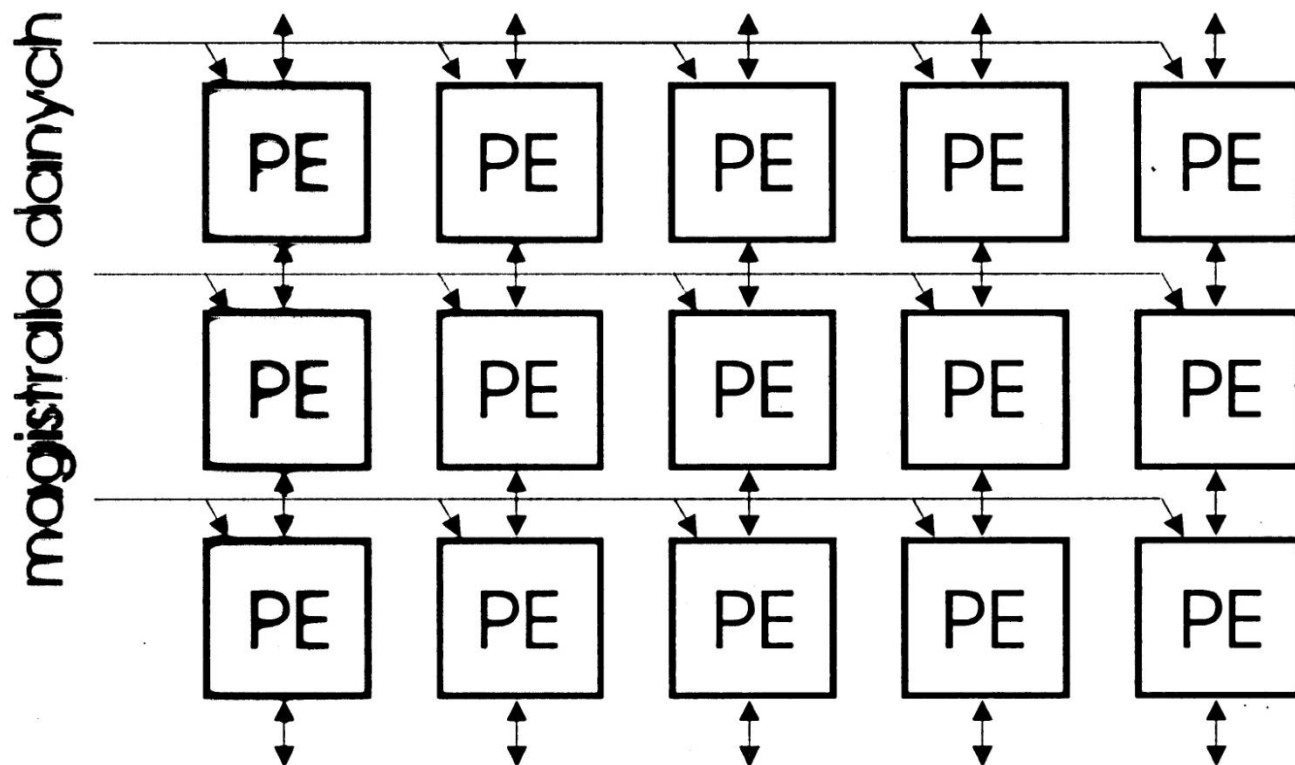
Architektury systoliczne (Kung, Leiserson)

- **architektury specjalizowane do implementacji operacji macierzowych, przetwarzania sygnałów i obrazów w czasie rzeczywistym**
- **zbiór globalnie synchronizowanych procesorów elementarnych połączonych w regularną siatkę**
- **każdy procesor jest połączony tylko z najbliższymi sąsiadami**
- **struktura wewnętrzna procesora, w zależności od postawionego przed nim zadania, może być bardzo prosta (sumator, kilka rejestrów przesuwnych) lub dążyć do stopnia ukomplikowania współczesnych mikroprocesorów**
- **dane i częściowe wyniki są przesyłane między procesorami również w takt wspólnego sygnału zegarowego**
- **wyniki przetwarzania danych uzyskiwane są stopniowo**

Podstawowe struktury SIMD (1)

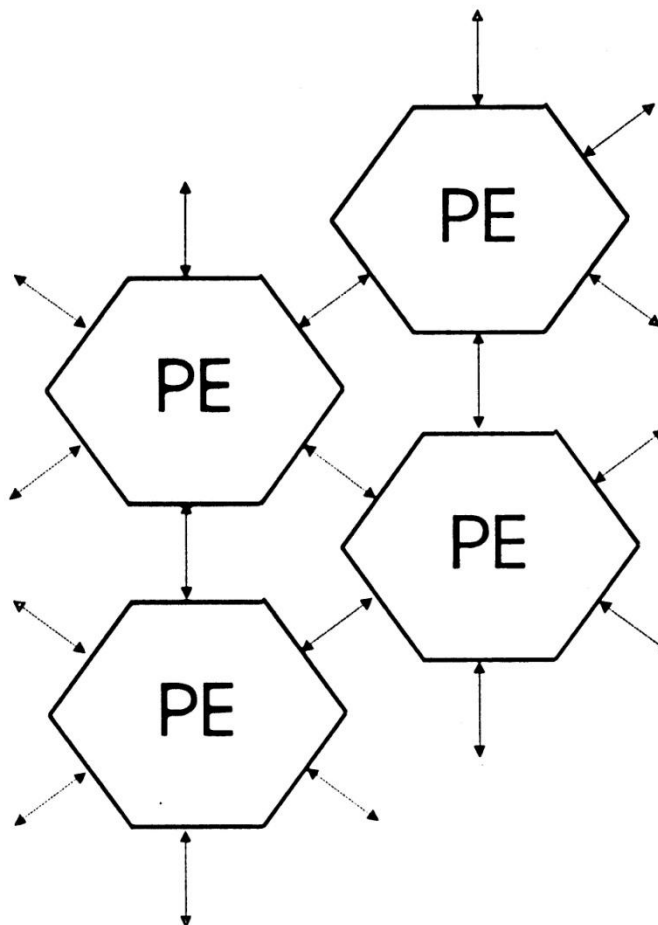
- tablica z częściowym rozpowszechnianiem danych

- Huang, Abraham



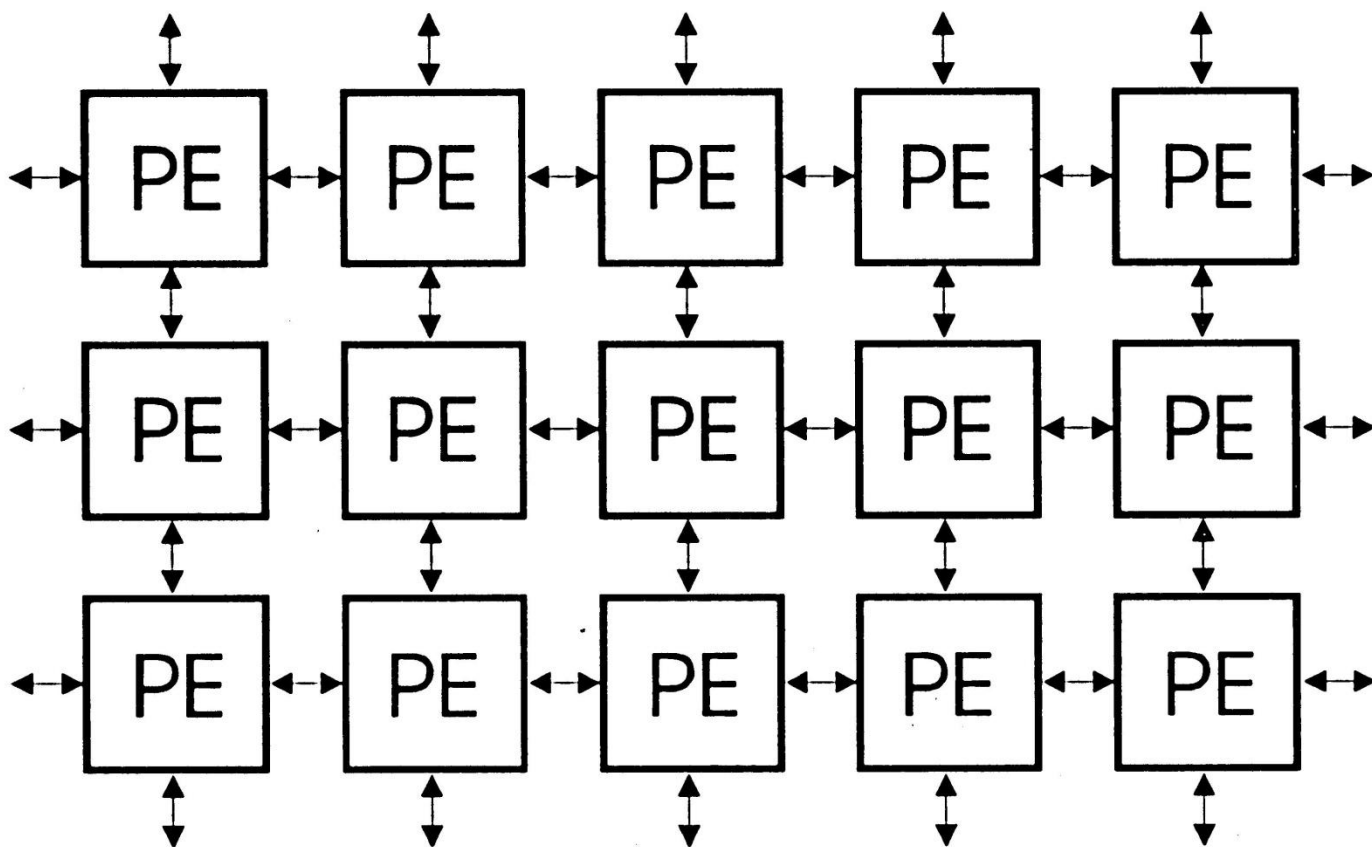
Podstawowe struktury SIMD (2)

- tablica hexagonalna - Kung, Leiserson



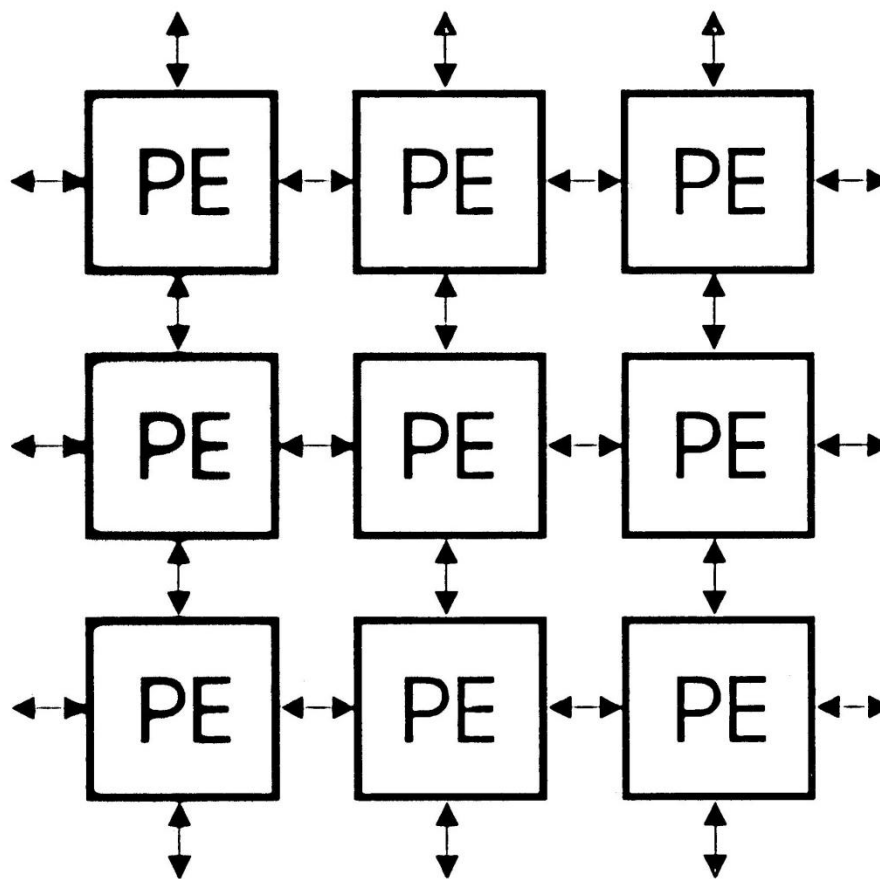
Podstawowe struktury SIMD (3)

- tablica przetwarzania potokowego - Hwang, Cheng



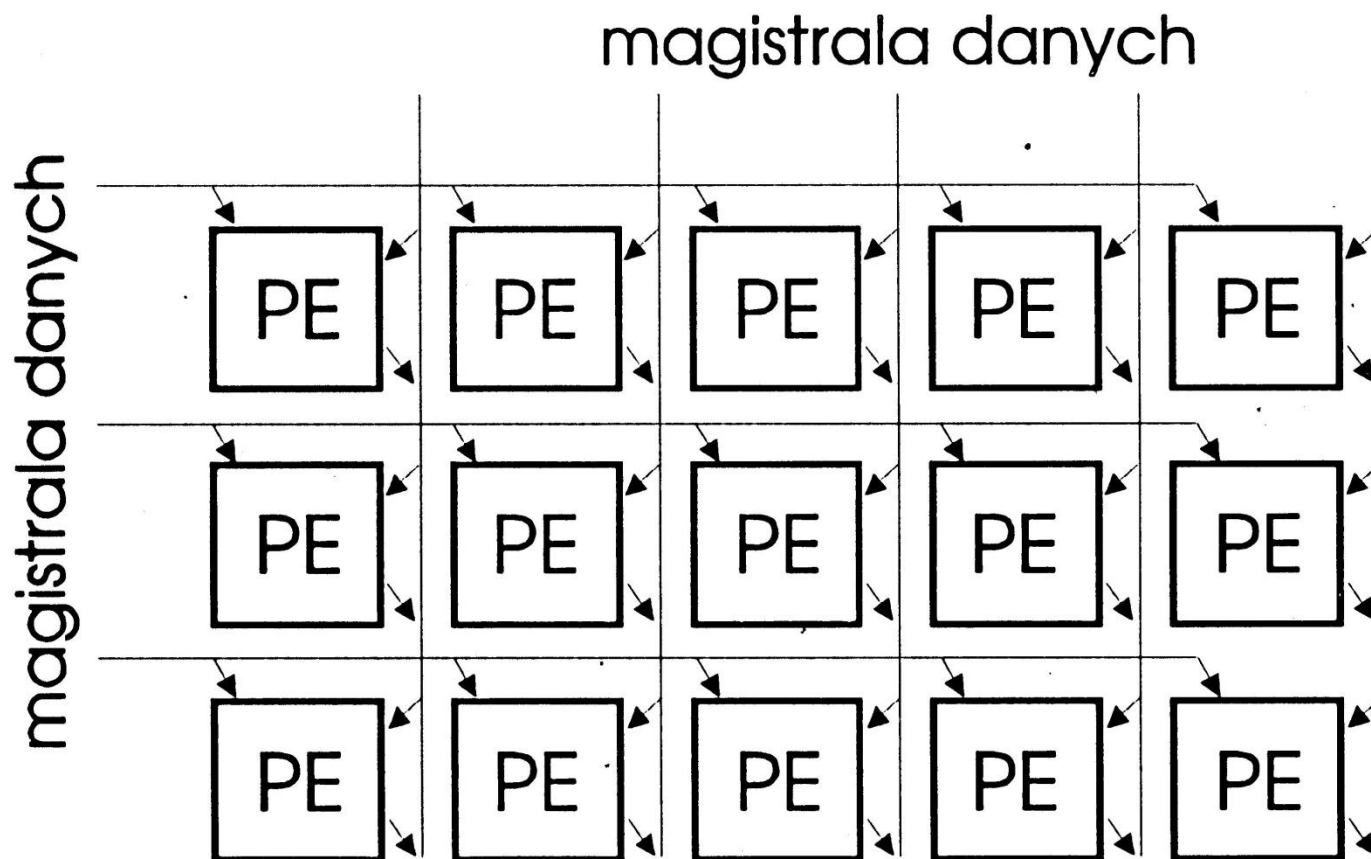
Podstawowe struktury SIMD (4)

- tablica typu wavefront - Kung, Arun



Podstawowe struktury SIMD (5)

- tablica z rozpowszechnianiem danych - Chern, Murata



Efektywność (1)

- tablica hexagonalna - Kung, Leiserson

- komunikacja między PE: punkt - punkt
- komunikacja zewnętrzna: tylko procesory brzegowe
- zadanie mnożenia macierzy o wymiarach $n \times n$:
 - liczba PE: $P = (2n-1)^2$
 - czas realizacji zadania: $T = 4n-2$ $T = 5n-3$
- prosta konstrukcja
- duża liczba stosowanych PE
- niski stopień wykorzystania PE - zazwyczaj $< 50\%$

Efektywność (2)

- tablica przetwarzania potokowego - Hwang, Cheng
 - komunikacja między PE: punkt - punkt
 - komunikacja zewnętrzna: tylko procesory brzegowe
 - zadanie mnożenia macierzy o wymiarach $n \times n$:
 - liczba PE: $P = n(2n-1)$
 - czas realizacji zadania: $T = 4n-2$
 - prosta konstrukcja
 - liczba PE mniejsza niż w tablicy hexagonalnej
 - stopień wykorzystania PE - przynajmniej 50%

Efektywność (3)

- tablica z częściowym rozpowszechnianiem danych
 - Huang, Abraham
- komunikacja między PE: punkt - punkt, magistrala
- komunikacja zewnętrzna: poprzez magistralę
- zadanie mnożenia macierzy o wymiarach $n \times n$:
 - liczba PE: $P = n^2$
 - czas realizacji zadania: $T = 2n$ $T = 3n$
- rozbudowane układy sterujące
- stopień wykorzystania PE - ponad 50%

Efektywność (4)

- tablica typu wavefront - Kung, Arun

- **komunikacja między PE:** punkt - punkt asynchronicznie
- **komunikacja zewnętrzna:** tylko procesory brzegowe
- **zadanie** mnożenia macierzy o wymiarach $n \times n$:

liczba PE: $P = n^2$

czas realizacji zadania: $T = 3n-2$ $T = 4n-2$

- **dobra skalowalność**
- **łatwe przeprogramowywanie**
- **dobre parametry FTC**
- **stopień wykorzystania PE** - zazwyczaj 50%

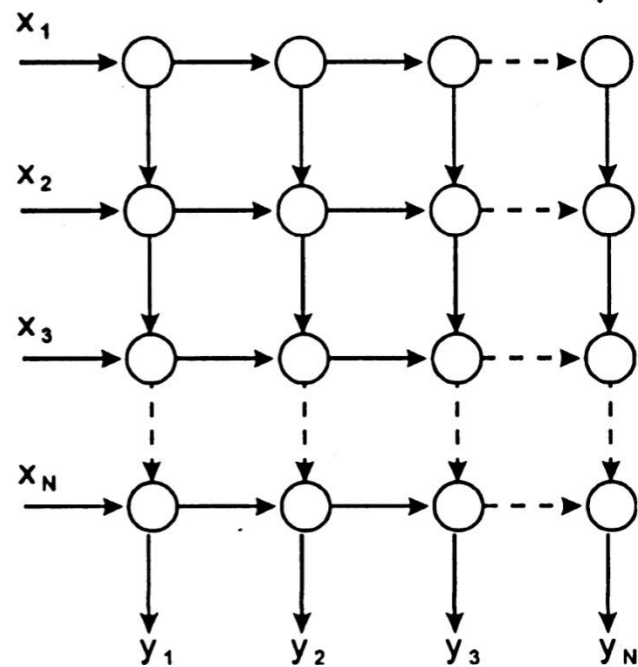
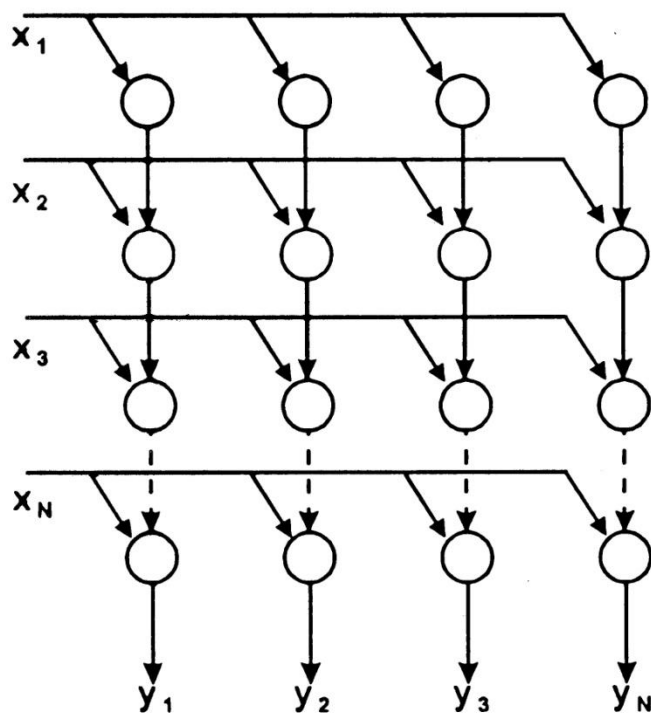
Efektywność (5)

- tablica z rozpowszechnianiem danych - Chern, Murata
 - komunikacja między PE: magistrale
 - komunikacja zewnętrzna: magistrale
 - zadanie mnożenia macierzy o wymiarach $n \times n$:
 - liczba PE: $P = n^2$
 - czas realizacji zadania: $T = n$ $T = 2n$
 - rozbudowane układy sterujące
 - łatwa implementacja algorytmów
 - wysoki stopień wykorzystania PE

Graf zależności

- zlokalizowany graf zależności

- kompletny graf zależności



Graf przepływu (1)

**Założenia o wektorze liniowej projekcji $\vec{d}$
Grafu Zależności na przestrzeń procesorów elementarnych i
wektorze następstwa czasowego $\vec{s}$:**

$$\vec{s}^T \vec{e} > 0$$

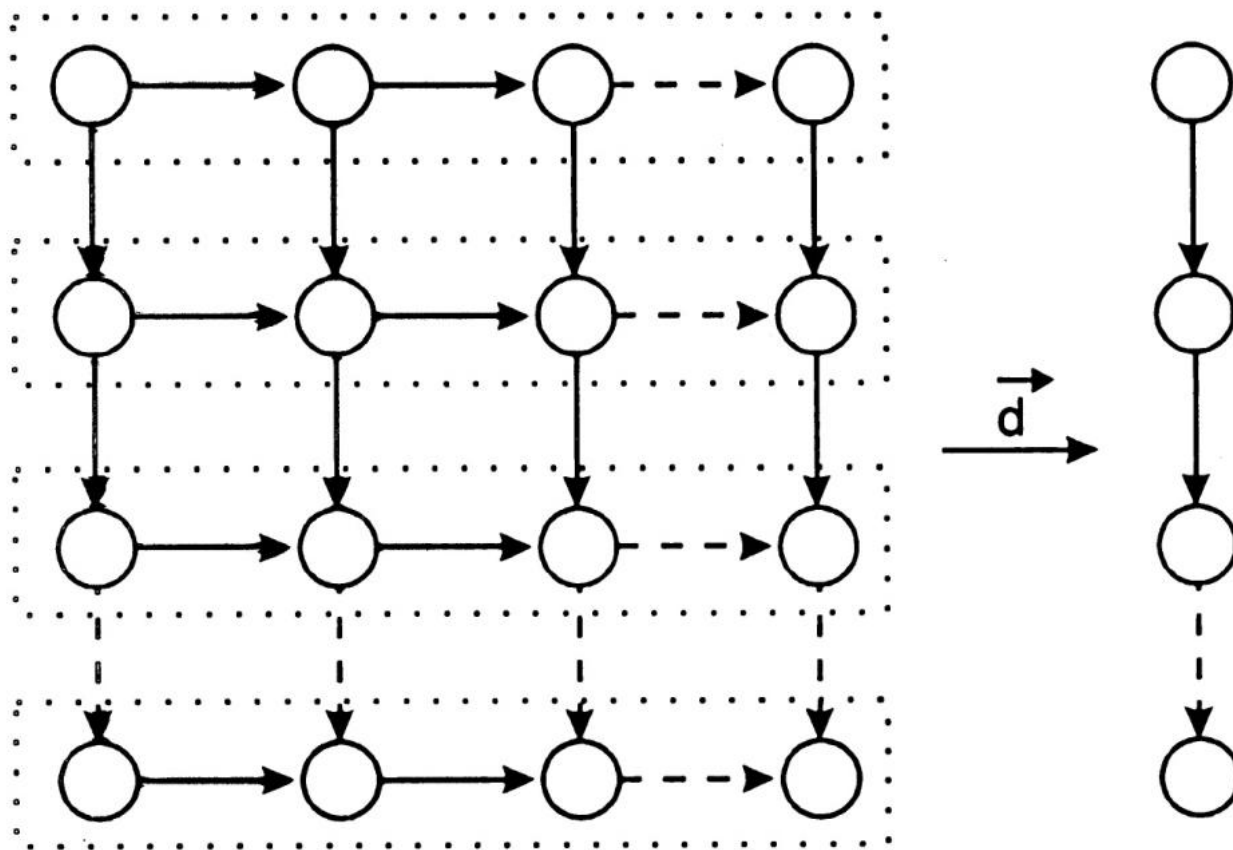
$\vec{e}$ - wektor reprezentujący dowolny łuk w Grafie Zależności
Elementy przetwarzające są oddzielone przynajmniej jednym łukiem

$$\vec{s}^T \vec{d} > 0$$

Zachowanie następstwa czasowego w Grafie Zależności

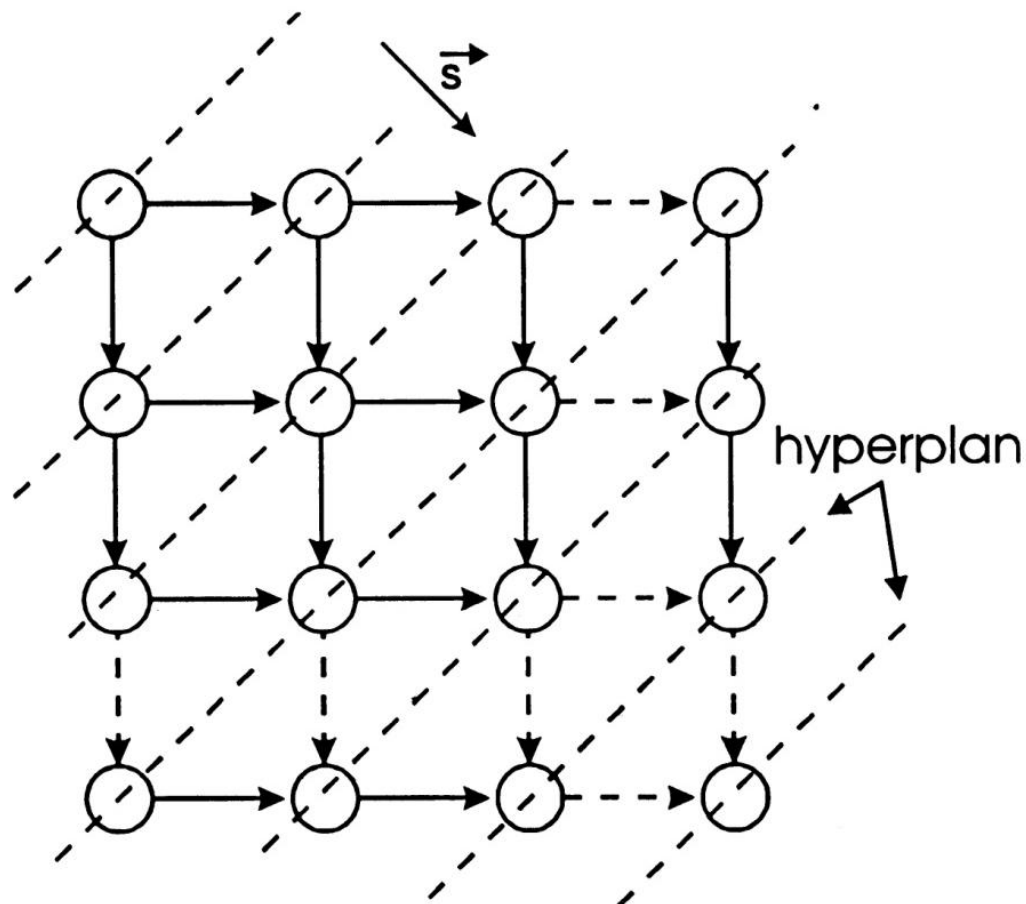
Graf przepływu (2)

- przyporządkowanie operacji do PE



Graf przepływu (3)

- szeregowanie zadań w przestrzeni czasu





Jakość (1)

Computation Time:

This is the time interval between starting the first computation and finishing the last computation of a problem.

Pipelining Period

This is the time interval between two successive computations in a processor.

Block Period:

This is the time interval between the initiation of two successive blocks.

Jakość (2)

Processor Utilization Rate:

$$\text{speed} - \text{up} = \frac{\text{sequential computation time}}{\text{array computation time}}$$

$$\text{utilization rate} = \frac{\text{speed} - \text{up}}{\text{number of processors}}$$



Jakość (3)

Array Processor Size:

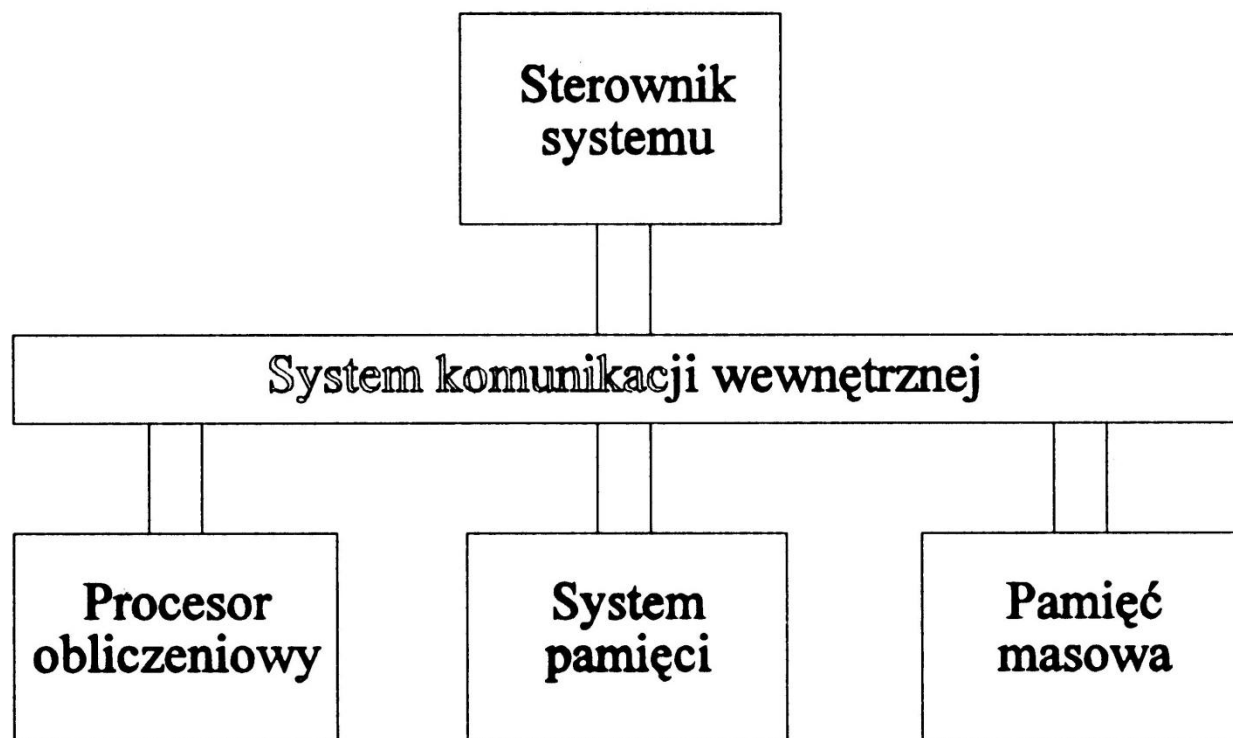
The array size cannot be unlimited. Sometimes, the size of the array is much smaller than the full problem size. In another situation, a one-dimensional array is used in place of an idealistic two-dimensional array. This incurs the so-called partitioning problem.

Local Memory:

Local memory can be used to expand the physical array to a much larger virtual array size.

Komercyjne struktury SIMD (1)

System Saxpy Matrix-1 - Fousler, Schreiber

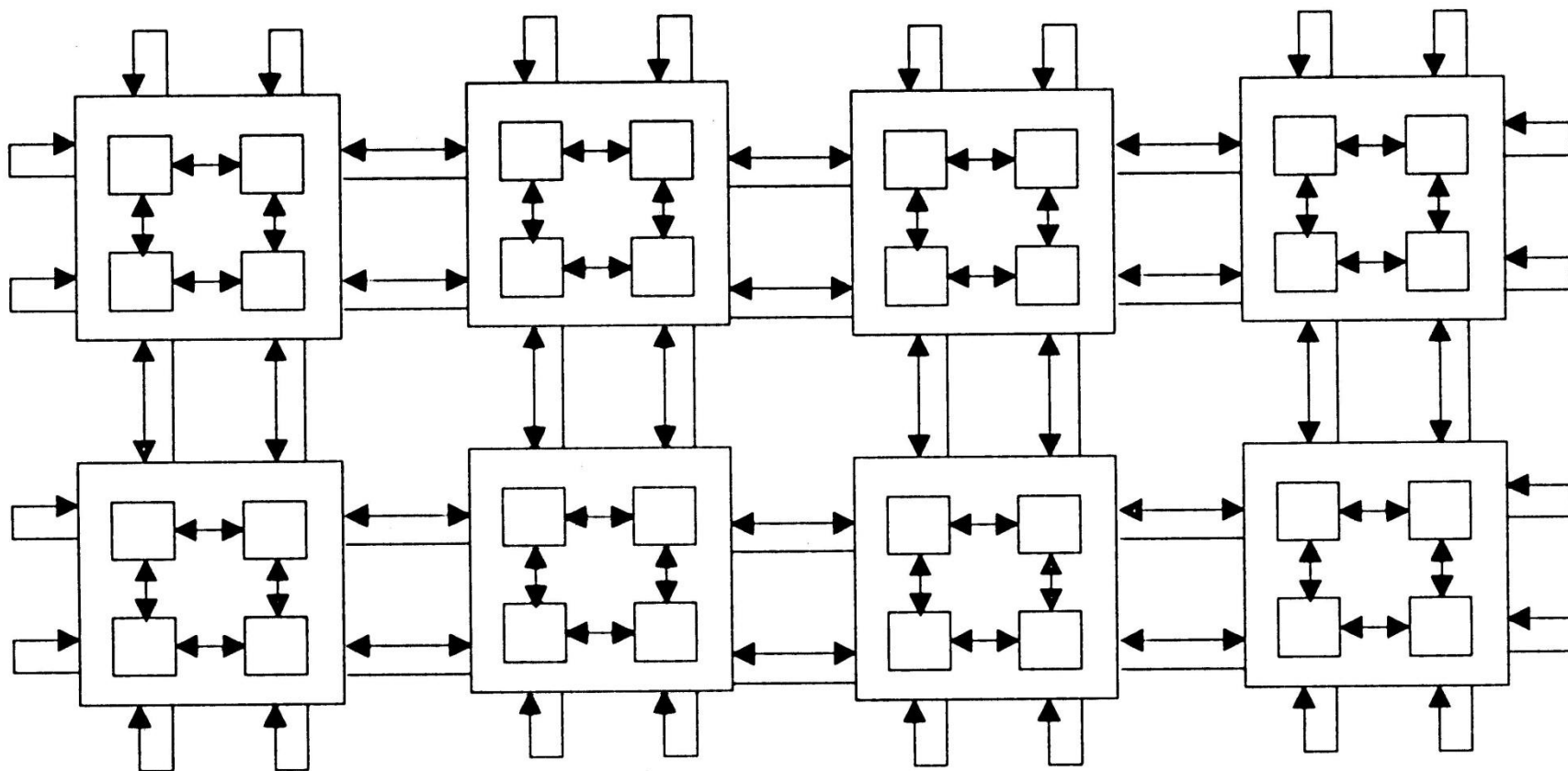


Komercyjne struktury SIMD (2)

- sterownik systemu - komputer, który podejmuje realizację zadania i odpowiednio alokuje zasoby,
- procesor obliczeniowy zbudowany w postaci łańcucha o maksymalnie 32 potokowych, zmiennoprzecinkowych jednostkach z zapewnioną zarówno lokalną (punkt-punkt), jak i globalną komunikacją między nimi,
- system pamięci gromadzący wszystkie dane używane przez procesor opisany uprzednio,
- system masowej pamięci z łączem pozwalającym na szybką wymianę danych z urządzeniami peryferyjnymi,
- system nadzorujący prawidłową współpracę wszystkich elementów.

Komercyjne struktury SIMD (3)

System iWARP - Carnegie Mellon University, INTEL





Komercyjne struktury SIMD (4)

Pojedynczy element:

procesor obliczeniowy i procesor komunikacyjny
4 kanały komunikacyjne przepustowość każdego: 40MB/s
20 MFLOP -32 bitowa precyzja,
10 MFLOP - 64-bitowa precyzja obliczeń

Procesor obliczeniowy:

15 rejestrów - gromadzenie danych,
współdzielona pamięć: 32-bitowe komórki
całkowitoliczbowa 32-bitowa ALU, 20 MIPS
zmiennoprzecinkowa ALU

Procesor komunikacyjny:

właściwa komunikacja systoliczna: punkt - punkt
przekazanie danych z pominięciem
procesora obliczeniowego

Transputery (1)

- transistor + computer
- INMOS - Bristol (UK)
- pracują „stadem”
- komunikacja punkt-punkt
- każdy ma 4 sąsiadów
- 4 kanały: 5, 10, 20 Mb/s
- CPU stały przecinek
- koprocesor
- pamięć lokalna





Transputery (2)

- komputer komunikacyjny
- procesory RISC
- scheduler
- booting
- OCCAM
- C
- 16-bit
- 32-bit
- 64-bit

Mnemonic	Description
J	Jump — add immediate operand to instruction pointer.
LDLP	Load Local Pointer — load a Workspace-relative pointer onto the top of the register stack
PFIX	Prefix — general way to increase lower nibble of following primary instruction
LDNL	Load non-local — load a value offset from address at top of stack
LDC	Load constant — load constant operand onto the top of the register stack
LDNLP	Load Non-local pointer — Load address, offset from top of stack
NFIX	Negative prefix — general way to negate (and possibly increase) lower nibble
LDL	Load Local — load value offset from Workspace
ADC	Add Constant — add constant operand to top of register stack
CALL	Subroutine call — push instruction pointer and jump
CJ	Conditional jump — depending on value at top of register stack
AJW	Adjust workspace — add operand to workspace pointer
EQC	Equals constant — test if top of register stack equals constant operand
STL	Store local - store at constant offset from workspace
STNL	Store non-local - store at address offset from top of stack
OPR	Operate - general way to extend instruction set

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

 **Wartburg 1.53W**

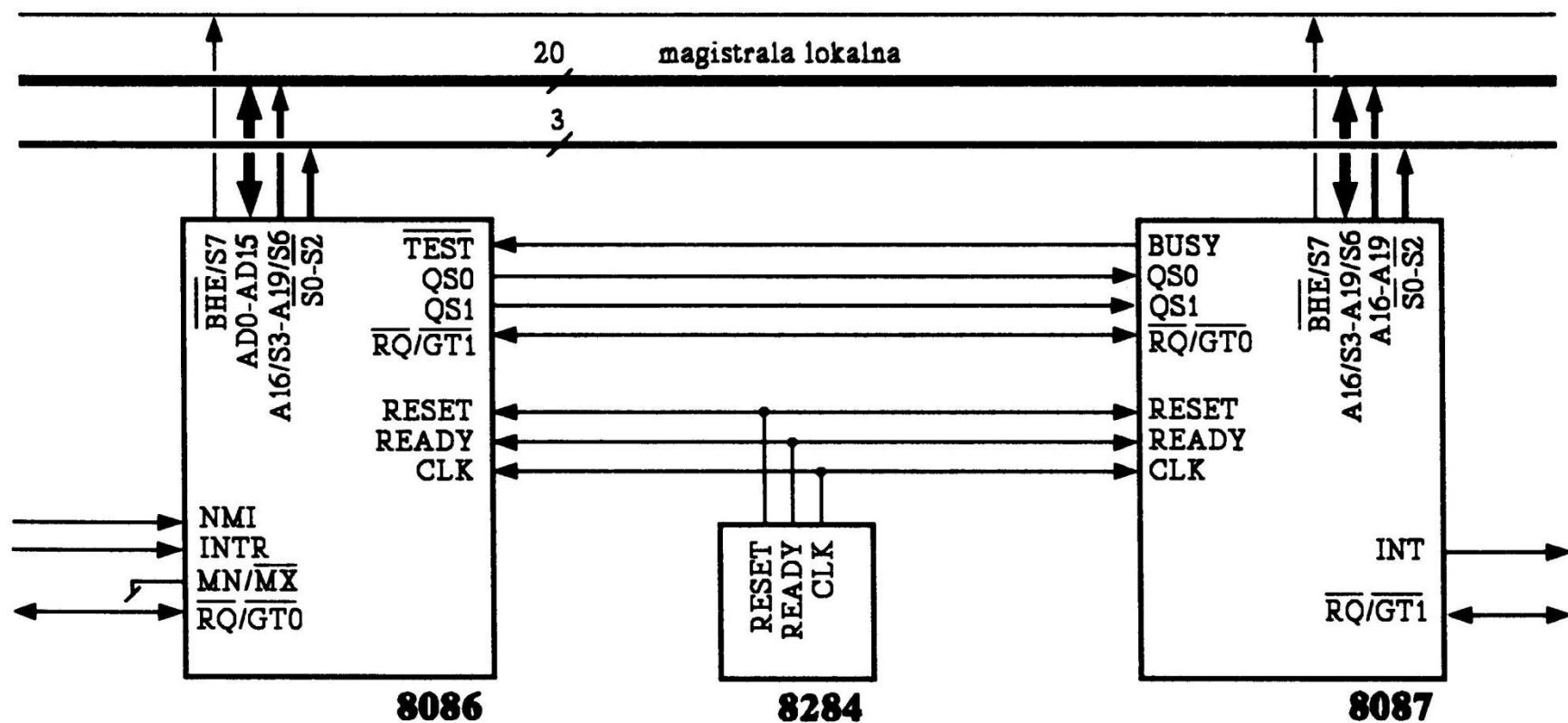




Współpraca procesora i koprocesora (1)

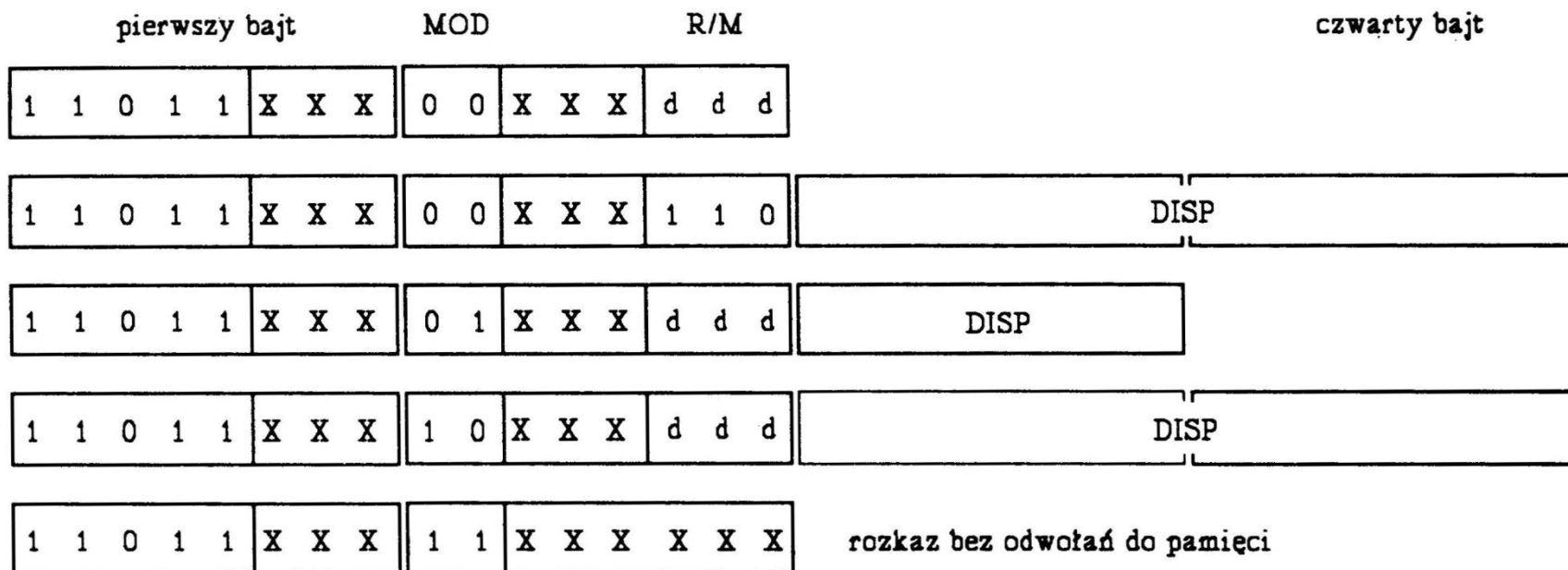
- procesor podstawowy pobiera rozkazy
- rozkazy i swoje, i dla koprocesora
- rozkazy dla koprocesora mają preambułę ESC
- w S2 - dekodowaniu łatwo rozpoznać
- procesor odcina preambułę
- przekazuje kod rozkazu do koprocesora
- procesor wykona jeszcze S3
- wyznaczy adresy efektywne argumentów dla koprocesora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

Współpraca procesora i koprocatora (2)



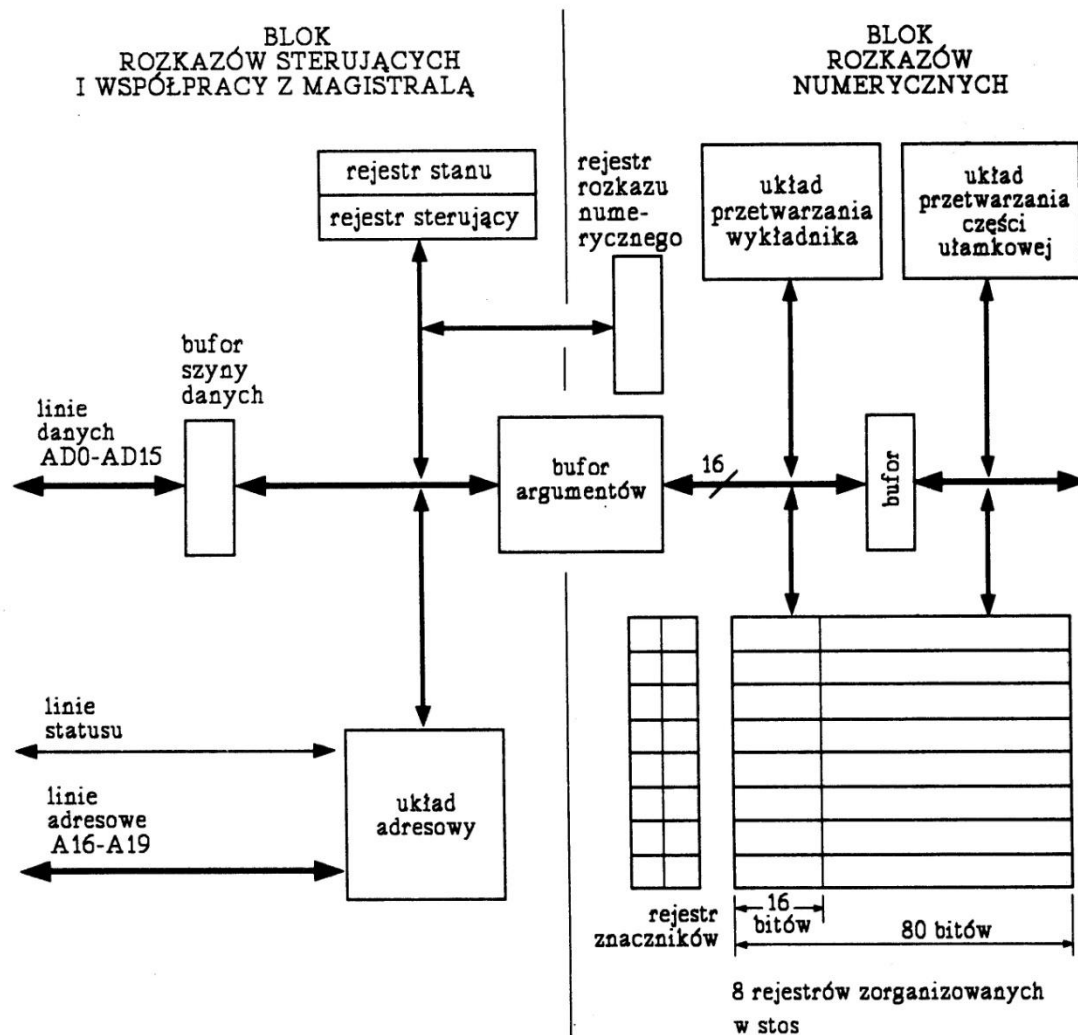


Współpraca procesora i koprocatora (3)





Architektura wewnętrzna





Formaty danych (1)

S	E	F	Obiekt
0	MAX	$\neq 0$	plus nieliczba
0	MAX	0	$+\infty$
0	$0 < E < \text{MAX}$	$\neq 0$	liczba dodatnia
0	0	$\neq 0$	liczba dodatnia w postaci nieznormalizowanej
0	0	0	+0
1	0	0	-0
1	0	$\neq 0$	liczba ujemna w postaci nieznormalizowanej
1	$0 < E < \text{MAX}$	$\neq 0$	liczba ujemna
1	MAX	0	$-\infty$
1	MAX	$\neq 0$	minus nieliczba

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

Formaty danych (2)

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

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

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 (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;

