KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt "Nowoczesne metody i techniki kształcenia w UMCS. Wzmocnienie potencjału dydaktycznego Wydziału MFiI" współfinansowany jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego Program Operacyjny Kapitał Ludzki 2007-2013 Priorytet IV, Poddziałanie 4.1.1 "Wzmocnienie potencjału dydaktycznego uczelni"

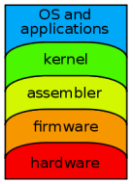
Człowiek - najlepsza inwestycja

Programowanie współbieżne... (1)

Andrzej Baran **2010/11**

LINK: http://kft.umcs.lublin.pl/baran/prir/index.html

Uwagi wstępne

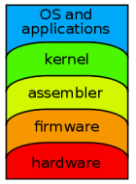


- OR = obliczenia równoległe
- OW = obliczenia współbieżne

W czasie wykładu zajmiemy się następującymi zagadnieniami:

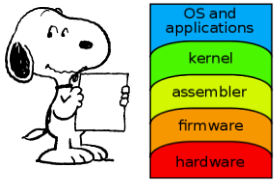
- Programowanie równoległe (współbieżne) na klastry komputerowe - zespoły połączonych komputerów, pracujące wspólnie nad jednym zadaniem obliczeniowym.
- Języki programowania: FORTRAN, C++ (Java – wątki)
- Algorytmy równoległe (wybrane)
- Sposoby realizacji obliczeń
  - MPI (Message Passing Interface)
  - OpenMP
  - Wątki (Java)

Literatura



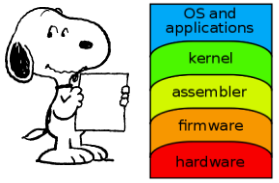
- Karniadakis, Kirby. Parallel scientific computing in C++ and MPI. Cambridge, 2003.
- RS/6000 SP: Practical MPI Programming (sieć)
- Introduction to MPI – PACS Training Group (sieć)
- MPICH http://www-unix.mcs.anl.gov/mpi/mpich/
- C, C++ oraz FORTRAN z MPI-1.2 http://www.lam-mpi.org/tutorials/bindings/

Literatura (cd)



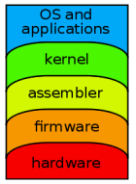
- MPIhttp://www.netlib.org/utk/papers/mpi-book/mpi-book.html
- MPI Home
  http://www.mpi-forum.org/
  http://www-unix.mcs.anl.gov/mpi/

Pierwsze OR...?



Stanley Phillips "Stan" Frankel (1919 – 1978) was an American computer scientist. He was born in Los Angeles, received his PhD in physics from the University of Rochester, and began his career as a post-doc student under J. Robert Oppenheimer at University of California, Berkeley in 1942. Frankel helped develop computational techniques used in the nuclear research taking place at the time. He joined the theoretical division of the Manhattan Project at Los Alamos in 1943. While there, he organized teams of persons (known as "computers") using electromechanical calculators to divide the massive calculations required for the project into manageable assembly line groups. Even that proved too slow, and Frankel turned to IBM tabulating machines to help process the numbers. This research led to his interest in the then-dawning field of digital computers. In August 1945, Frankel and Nick Metropolis traveled to the Moore School of Engineering in Pennsylvania to learn how to program the ENIAC computer. That fall they helped design a calculation that would determine the likelihood of being able to develop a fusion weapon. Edward Teller used the ENIAC results to prepare a report in the spring of 1946 that answered this question in the affirmative. (Wikipedia)

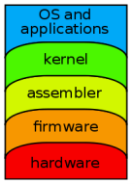
Feynman



Feynman, w książce „Pan raczy żartować, panie Feynman” (Znak, Kraków, 1996) pisze,

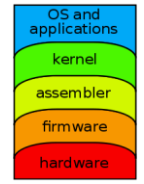
„...uznaliśmy, że główne zagadnienie – czyli dokładne określenie, co się dzieje podczas implozji bomby, żeby można było ustalić, ile powstaje energii i tak dalej – wymaga znacznie większej ilości obliczeń, niż my byliśmy w stanie przeprowadzić. Pewien niegłupi facet, Stanley Frenkel wpadł na pomysł, że można by użyć maszyn IBM. Firma IBM robiła maszyny liczące do celów rachunkowości, na przykład tzw. tabulatory do sumowania czy mnożarki, do których wkładalo się kartę, a one brały dwie liczby z karty i przemnażały je przez siebie. Były też kolatory, sortery itd. Frenkel wykoncypował sprytny program. Gdyby zgromadzić wystarczającą liczbę tych maszyn w jednym pomieszczeniu, można by przepuszczać karty przez cały cykl operacji. (...) Przedtem próbowaliśmy czegoś takiego z samymi maszynami dodającymi, ale każdy sam wykonywał wszystkie kroki. Frenkel opracował cały system i zamówił maszyny u IBM, ponieważ zdaliśmy sobie sprawę, że to dobry sposób na rozwiązanie naszych problemów. (...) opracowaliśmy wszystkie numeryczne kroki, które miały wykonać maszyny – pomnożyć, dodać, odjąć – i mieliśmy cały program gotowy, ale nie było jeszcze maszyn. Posadziliśmy więc w jednym pomieszczeniu dziewczyny z Marchantem (maszyny liczące, [ab]), każda od innego działania. Jedna na przykład tylko podnosiła do sześcianu – dostawała na karcie liczbę, podnosiła ją do sześcianu i podawałą następnej dziewczynie. (...) Okazało się, że liczy się w ten sposób dużo szybciej niż dawnym systemem, w którym każdy sam wykonywał wszystkie działania.”

Do czego potrzebne są OR?



- Projektowanie obwodów VLSI
- Aplikacje CAD/CAM
- Rozwiązywanie problemów polowych (dynamika strukturalna, badania materiałowe i jądrowe, układy cząstek)
- Przewidywanie pogody
- Systemy inteligentne
- Modelowanie (ekonomia, planowanie itd)
- Opracowywanie danych satelitarnych (...)
- Problemy energetyki jądrowej
- Reakcje chemiczne i jądrowe
- Aktywność sejsmiczna Ziemi
- Problemy genetyki
- inne

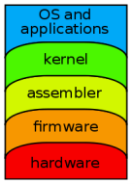
Najszybsze komputery świata



- Top500: http://www.top500.org/
- Jaguar: http://computing.ornl.gov/

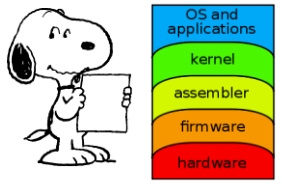
- SciDAC: http://www.scidacreview.org/

Architektura komputerów

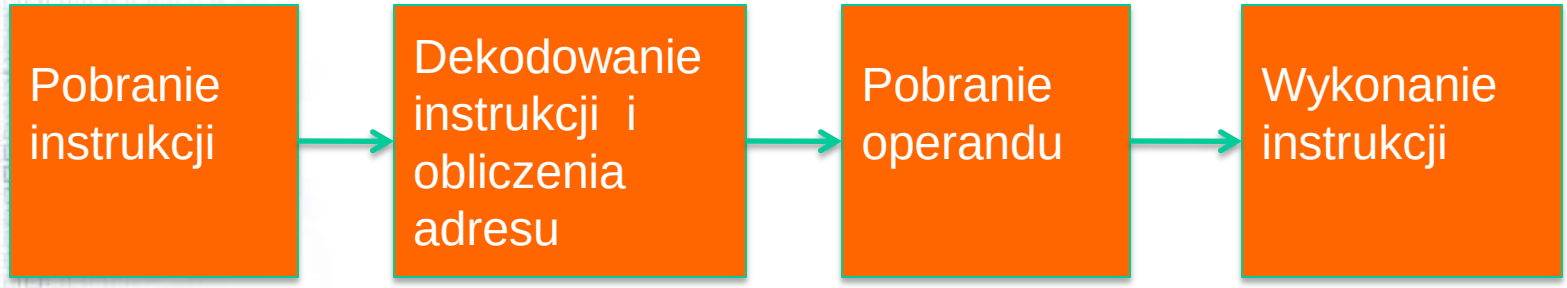


- Istnieją dwa główne typy architektury komputerów równoległych: z pamięcią rozproszoną (**distributed memory**) i z pamięcią dzieloną, wspólną (**shared memory**)
  - Komputery z pamięcią rozproszoną są w zasadzie komputerami seryjnymi (węzły), które wspólnie rozwiązują dany problem. Każdy węzeł posiada bardzo szybki dostęp do własnej pamięci oraz dostęp do pamięci innych węzłów poprzez specjalną, szybką sieć, poprzez wymienę informacji, komunikatów (messages) z innymi węzłami.
  - W przypadku architektury z pamięcią dzieloną węzły mają dostęp do bardzo szybkiej pamięci wspólnej przez specjalne łącze (szyna danych; **bus**). Tego rodzaju dostęp do pamięci pozwala procesorom na wymianę i dostęp do danych. W grupie węzłów, która korzysta ze wspólnej pamięci znajduje się typowo kilka – kilkanaście (2-16) procesorów. Jest to związane z szerokością pasma przesyłowego szyny danych łączącej procesory.

Architektura SISD



- Obliczenia sekwencyjne (potoki)



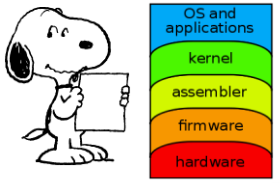
Operacje skalarne SISD
LOAD R1, A LOAD R2, B

MUL R1, R2
LOAD R3, C
ADD R1, R3

STORE R1, D

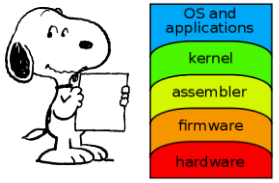
SISD = single instruction stream
single data stream

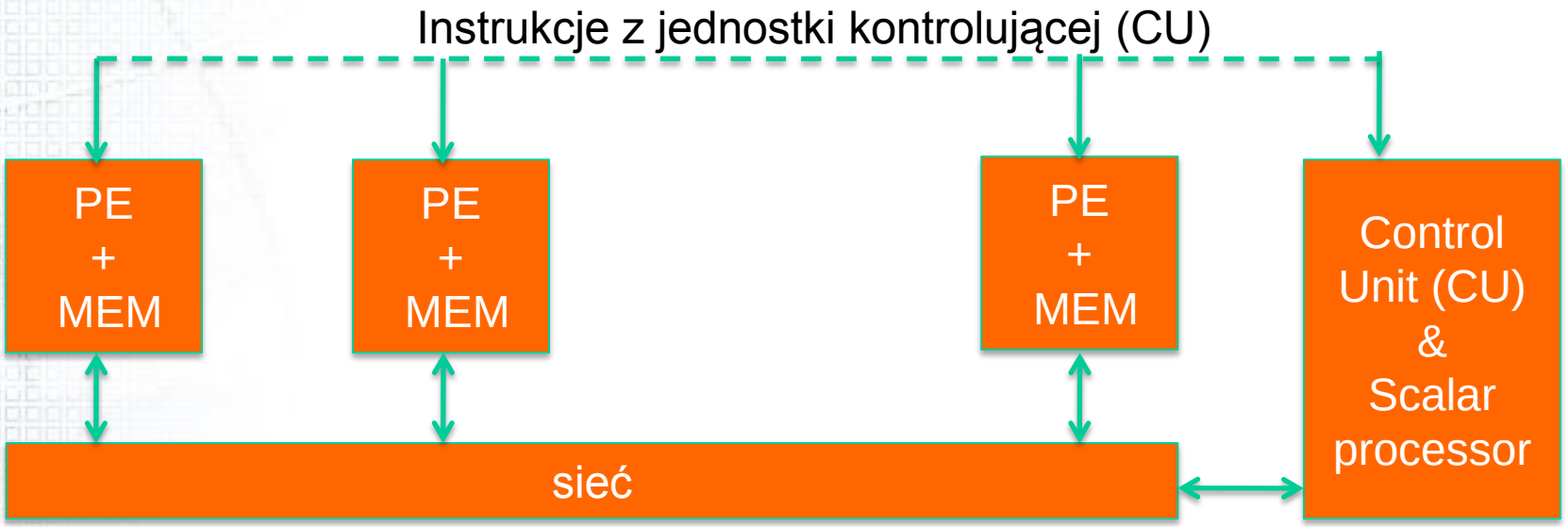
Architektura SIMD



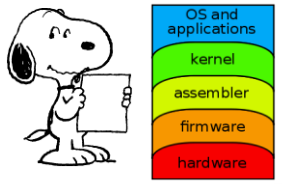
- Multiprocesory synchroniczne (Array Processors) – wszystkie wykonują te same instrukcje na różnych danych; sterowane przez jednostkę centralną; SIMD (single instruction stream with multiple data streams); synchronizacja sprzętowa;

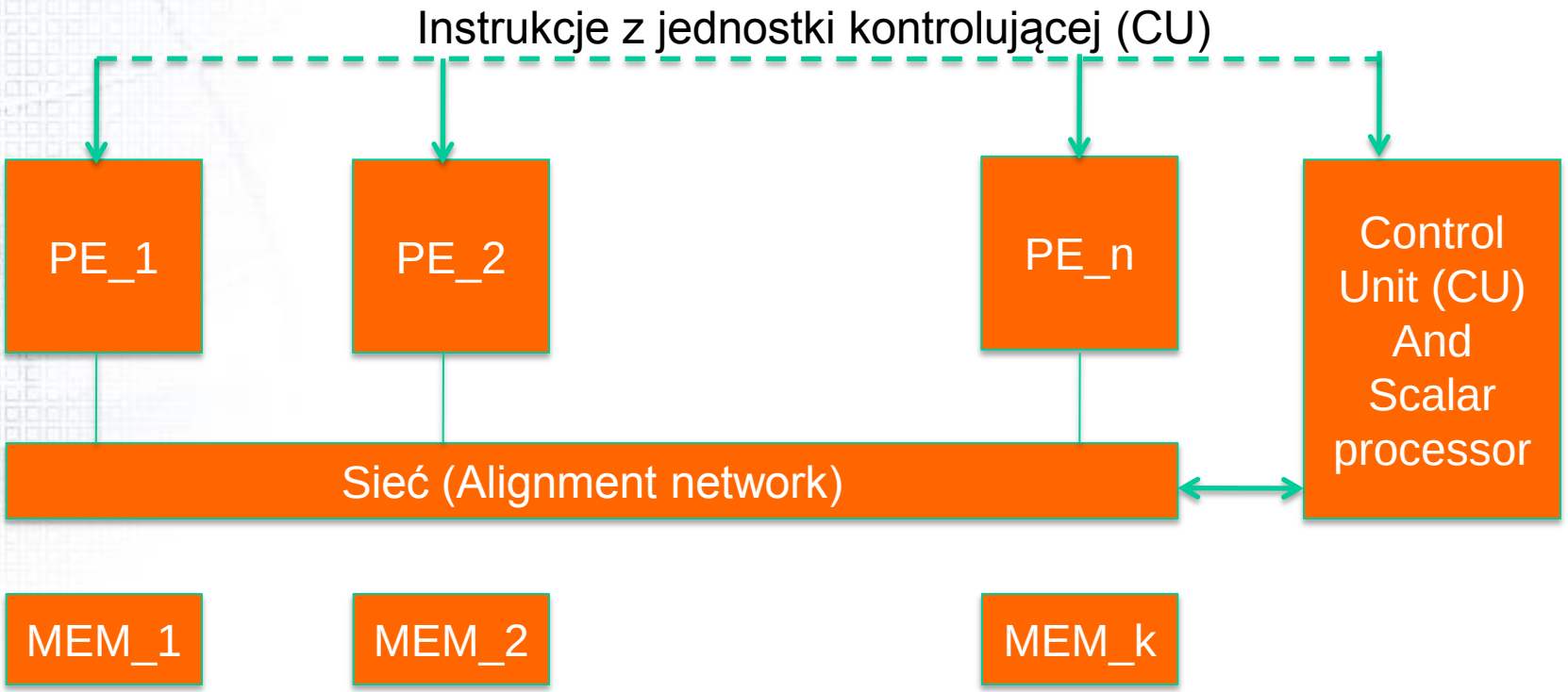
Architektura SIMD cd.



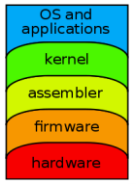


Architektura SIMD cd.





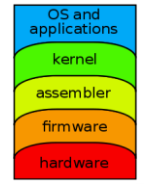
Obliczenia SIMD



Program SIMD (Operacje wektorowe)

PE_1	**PE_2**	**PE_3**
LOAD R1, A(1)	LOAD R1, A(2)	LOAD R1, A(3)
LOAD R2, B(1)	LOAD R2, B(2)	LOAD R2, B(3)
MUL R1, R2	MUL R1, R2	MUL R1, R2
LOAD R3, C(1)	LOAD R3, C(2)	LOAD R3, C(3)
ADD R1, R3	ADD R1, R3	ADD R1, R3
MULT R1, D(1)	MULT R1, D(2)	MULT R1, D(3)

Przykład obliczeń SISD i SIMD

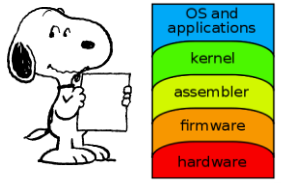


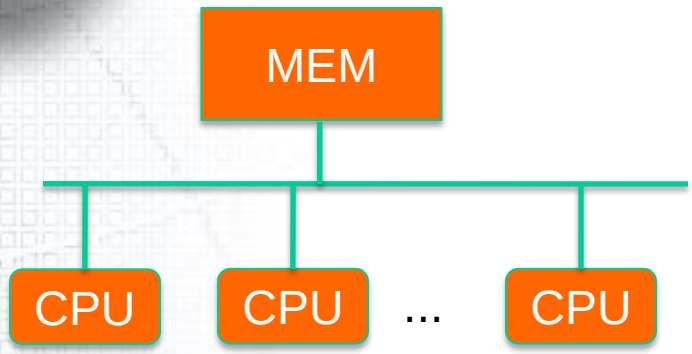
	PE_1	PE_2	PE_n
LOAD R1, A	LOAD R1, A(1)	LOAD R1, A(2)	LOAD R1, A(n)
LOAD R2, B	LOAD R2, B(1)	LOAD R2, B(2)	LOAD R2, B(n)
MUL R1, R2	MUL R1, R2	MUL R1, R2	MUL R1, R2
LOAD R3, C	LOAD R3, C(1)	LOAD R3, C(2)	LOAD R3, C(n)
ADD R1, R3	ADD R1, R3	ADD R1, R3	ADD R1, R3
STORE R1, D	MULT R1, D(1)	MULT R1, D(2)	MULT R1, D(n)

(a) SISD Program

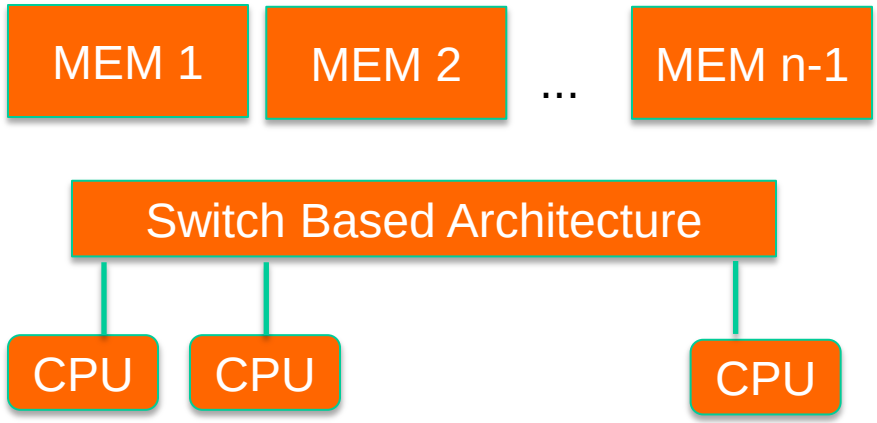
(b) SIMD Program

Architektury multiprocesorowe

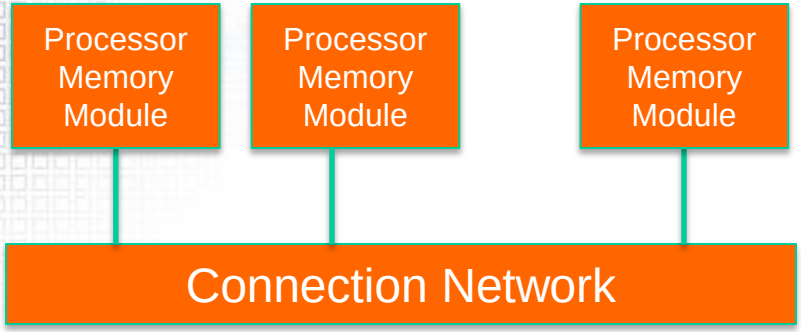




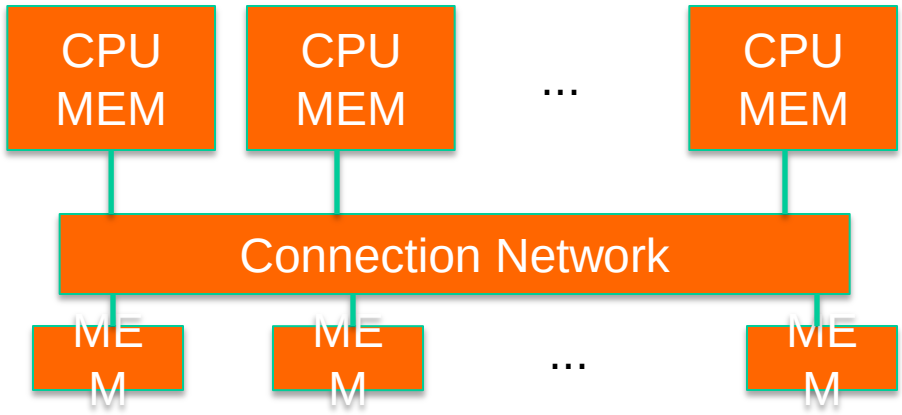
Common Bus Architecture



Switch Based Architecture

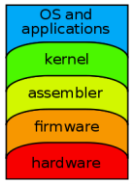


Message Based Architecture



Shared Memory & Message Based Architecture

Efektywność, wydajność obliczeń



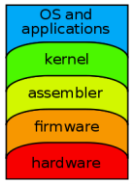
Szybkość obliczeń współbieżnych:

$$S=T1/Tn$$

T1 – czas wykonania na jednym procesorze
Tn – cas wykonania na n procesorach
n - liczba procesorów

TWIERDZENIE: 1 <= S <= n.

Efektywność, wydajność obliczeń



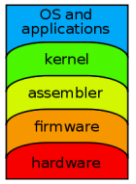
Efektywność (dowolne urządzenie):

E = (Energia wykorzystana)/(Energia dostarczona)

Maksymalna wartość E=1.
Maksymalna wartość S=n.
W praktyce:

$$E << 1, \qquad S << n$$

Prawo Amdahla



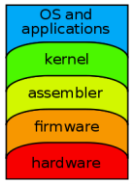
(Gene Amdahl, 1967)

Przyspieszenie obliczeń współbieżnych

$$S \leq \frac{1}{f + \frac{(1-f)}{n}}$$

gdzie *f* = część (<1) odpowiadająca obliczeniom sekwencyjn (*1-f* odpowiada tej części programu, ktora daje sie uwspółbieżnić).

Prawo Amdahla (cd)



Dowód. (P = procesor, n = liczba procesorów)

T – całkowity czas obliczeń na 1P

*f*T* – czas obl. sekwencyjnych na 1P

*(1-f)*T* – czas obl. równoległych na 1P

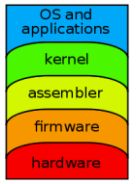
*(1-f)*T/n* – czas obl. równoległych na nP

Przyspieszenie:

$$S \leq \frac{T}{fT + \frac{(1-f)T}{n}}$$

Stąd prawo Amdahla.

Prawo Amdahla (cd)



Wydajność E = S/n (na procesor). Stąd

$$S = En \leq \frac{1}{f + \frac{1-f}{n}}$$

$$E \leq \frac{1}{fn + 1 - f}$$

Przy ustalonej wydajności E ułamek f obliczeń sekwencyjnych algorytmu powinien być odwrotnie proporcjonalny do liczby procesorów. (Czas obliczeń sekwencyjnych T*f ~ 1/n)