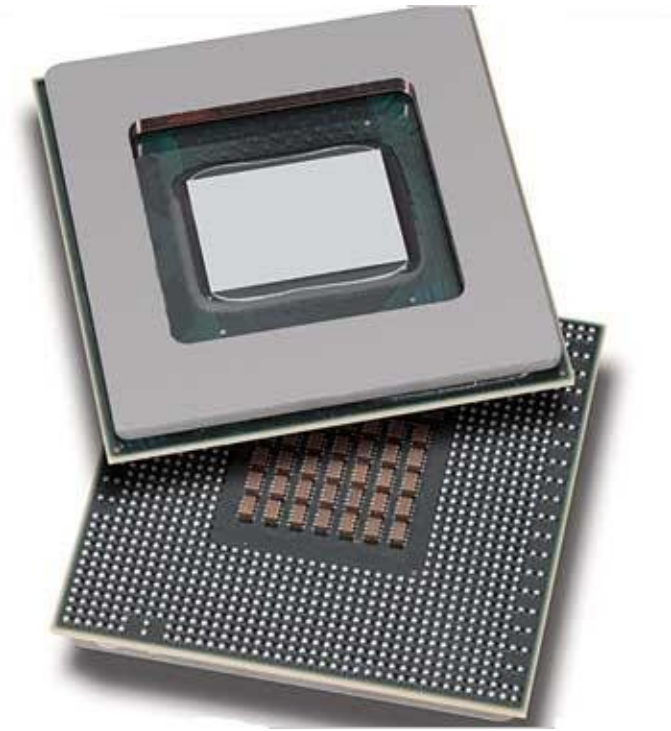
Wpływ architektury procesora na system operacyjny

Sławomir Kierat
Paweł Tryfon
Tomasz Świerczek

6 listopada 2008



Powtórka z architektury

CISC	RISC
Duża liczba skomplikowanych instrukcji	Liczba instrukcji ograniczona do niezbędnego minimum
Mała liczba uniwersalnych rejestrów	Duża liczba uniwersalnych rejestrów
Instrukcje arytmetyczno-logiczne mogą pobierać argumenty z pamięci i umieszczać wynik w pamięci	Instrukcje arytmetyczno-logiczne wykonywane są na rejestrach

- Wraz z rozwojem technologicznych granica staje się mniej wyraźna:
  - W CISC również stosuje się pipeline
  - RISC są wzbogacane w bardziej skąplikowane instrukcje

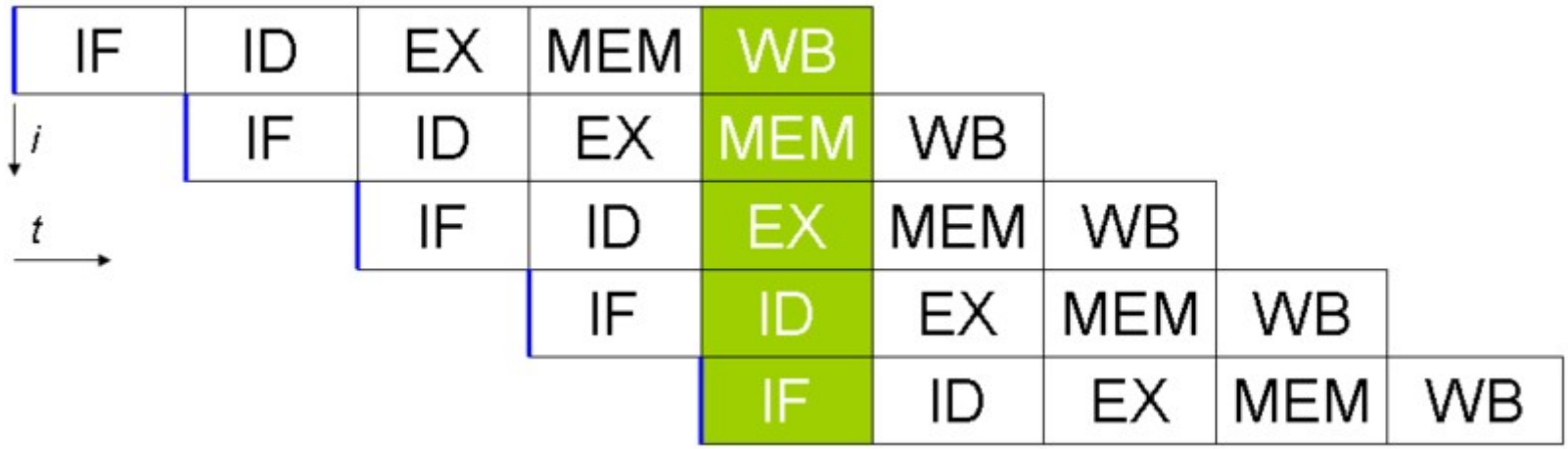
CISC

- Przykłady rodzin procesów o architekturze CISC
- System/360
- VAX
- PDP-11
- x86
- Wpływ oprogramowania na architekture
  - Wsteczna kompatybilność
- Obecnie procesory x86 przetwarzaja rozkazy procesora na proste mikropolecenia wg idei RISC

RISC

- Początki architektury RISC – koniec lat 70-tych
- MIPS – jeden z pierwszych procesorów typu RISC
  - Założenia co do długości instrukcji
  - Pierwsza implementacja potokowości
- Architektury RISC znajdują zastosowanie na rynkach:
  - Wysokowydajnych serwerów
  - Systemów wbudowanych
- Duża liczba rejestrów może zwiększyć kontekst procesu, co stanowi narzut na obsługę przerwań, ale również może umożliwić systemowi operacyjnemu trzymanie pewnych danych w rejestrach

Klasyczny RISCowy pipeline



- Pobranie instrukcji z pamięci - ang. instruction fetch (IF)
- Zdekodowanie instrukcji - ang. instruction decode (ID)
- Wykonanie instrukcji - ang. execute (EX)
- Dostęp do pamięci - ang. memory access (MEM)
- Zapisanie wyników działania instrukcji - ang. store; write back (WB)

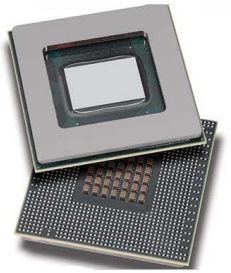
Problemy potokowości

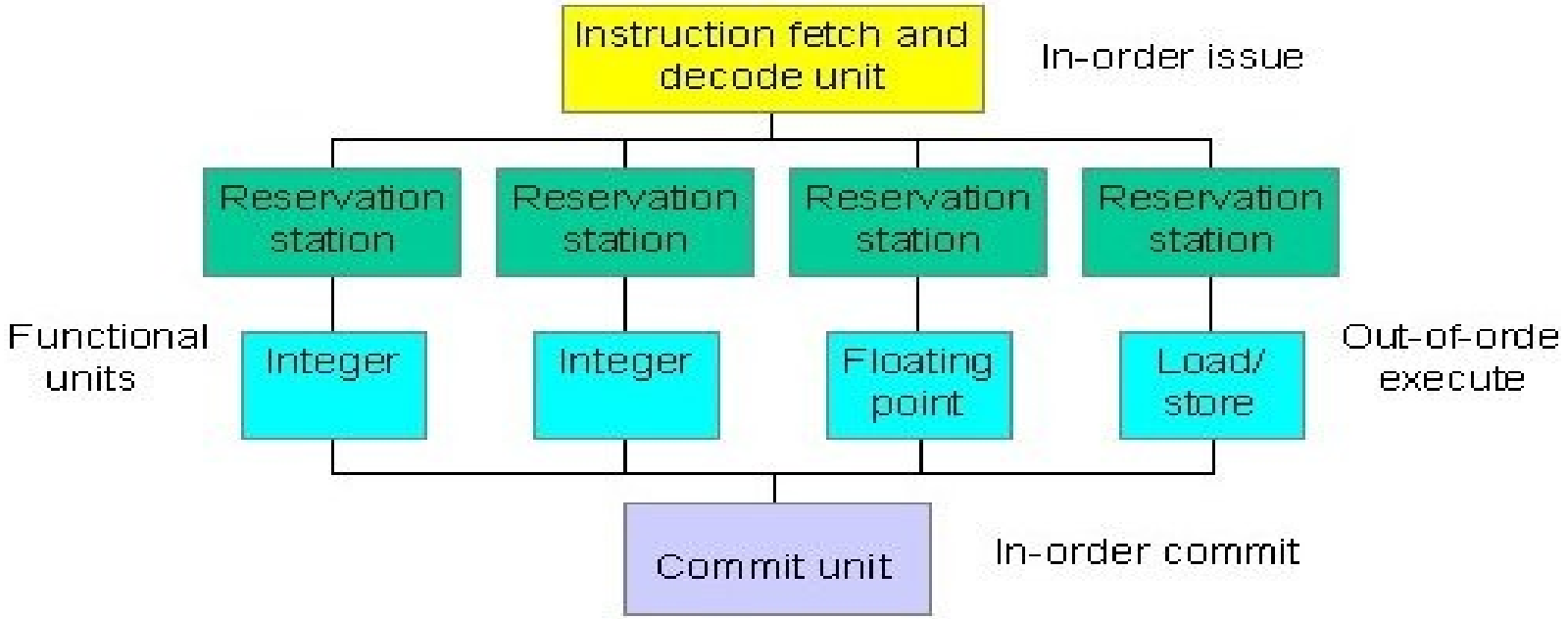
- Zależności zasobu
- Zależności danych – rozwiązywane przez kompilatory
    - RAW
    - WAR
    - WAW
- Zależności sterowania
- Równoległe przetwarzanie obu gałęzi programu
- Predykcja skoków

Metody przyspieszania pipeline

- Podział potoku na więcej kroków np.
  - 1. Instruction Fetch (First Half)
  - 2. Instruction Fetch (Second Half)
  - 3. Register Fetch
  - 4. Instruction Execute
  - 5. Data Cache Access (First Half)
  - 6. Data Cache Access (Second Half)
  - 7. Tag Check
  - 8. Write Back
- Superskalarność – kilka równoległych potoków

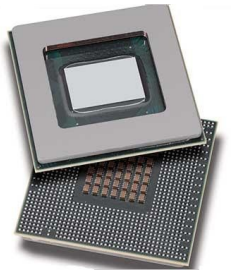
Potok dynamiczny





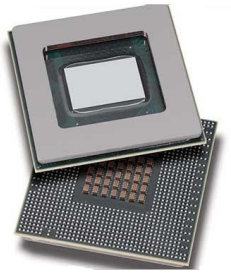
- Budowa potoku dynamicznego
- Jednostka pobierania rozkazów
- 5-10 jednostek funkcjonalnych z buforami
- Jednostka łącząca

Advanced RISC Machine



- Architektura ARM jest 32-bitową architekturą typu RISC
- Stanowi 75% rynku 32-bitowych CPU dla systemów wbudowanych
- Używany między innymi w dyskach twardych, telefonach komórkowych, routerach, kalkulatorach
- Cechy ARM:
  - Energooszczędna budowa
  - Umożliwia instalacje systemu operacyjnego z zaimplementowanymi mechanizmami wielowątkowości, stosem TCP/IP oraz systemem plików

Systemy operacyjne pod ARM

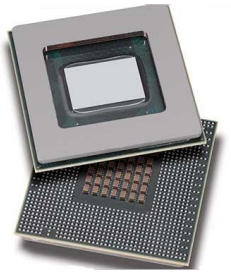


- Systemy operacyjne instalowane na ARM
  - Windows CE
  - NUTOS
  - Embedded Debian, Embedded Ubuntu
- Cechy Windows CE
  - spełnia podstawowe kryteria systemów operacyjnych czasu rzeczywistego (przewidywalny system przerwań)
  - Jądro może być uruchomione w 1MB pamięci

Początki wieloprocesorowości

- Wieloprocesorowość – skoordynowane wykonanie programu przez więcej niż jeden procesor
- Szukanie przyspieszenia
- Podnoszenie częstotliwości taktowania ograniczone barierami technologicznymi
- Wykonywanie równoległych operacji – większa przepustowość
- Na niezależnych jednostkach obliczeniowych odprowadzanie ciepła jest łatwiejsze niż dla pojedynczej jednostki o podobnej mocy obliczeniowej

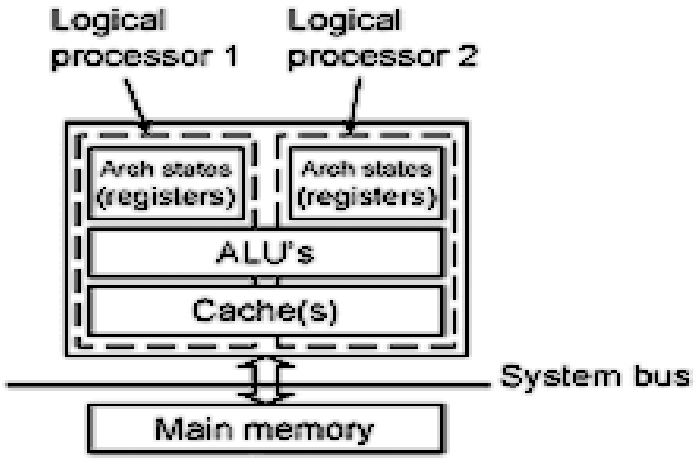
Hiperwątkowość

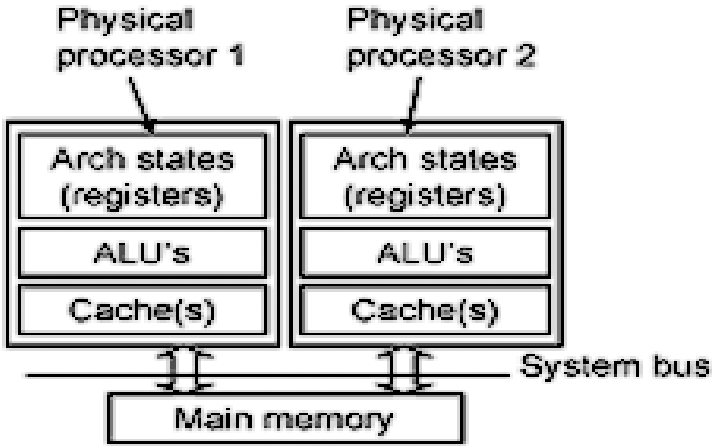


- Idea: korzystanie dwóch niezależnych wątków z tych samych jednostek wykonowczych jeżeli jeden z nich ulega przeczekaniu (np. przy złej predykcji skoku)
- Technologia zastosowana przez firmę Intel w:
  - Pentium Extreme Edition
  - Pentium 4
- System operacyjny widzi dwa procesory logiczne
- Zysk 20-30%
- Technologia zarzucona na rzecz dwurdzeniowości...

Hiperwątkowość - reaktywacja

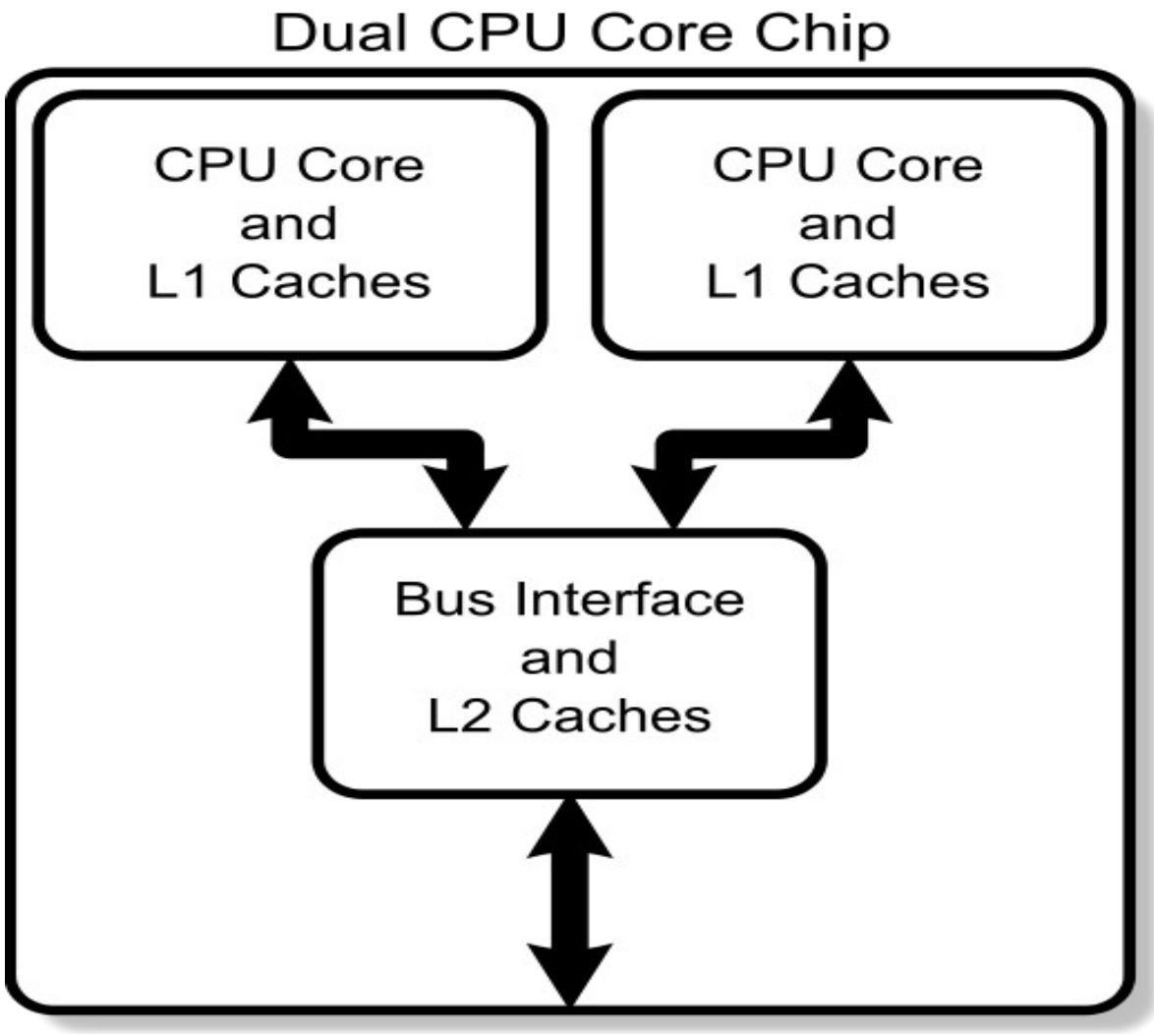
- Pomysł Intela na połączenie technologii hiperwątkowości oraz wielordzeniowości w najnowszych procesorach:
- Intel Core i7
- 8-rdzeniowy Nehalem może przetwarzać jednocześnie 16 wątków



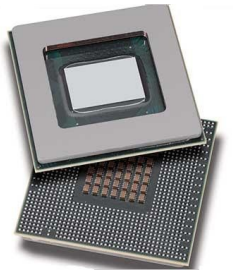


Procesor wielordzeniowy

- Składa się z kilku jednostek obliczeniowych na jednym układzie scalonym
- Dwupoziomowy cache:
- Szybki mały (rzędu 32kB) cache dostępny dla każdego rdzenia
- Cache wspólny dla wszystkich rdzeni (rzędu kilku MB)



Sposoby wspierania wielordzeniowości

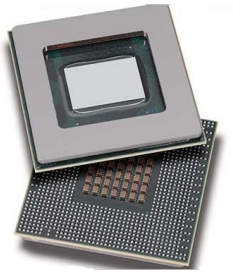


- Sposoby przydzielania zadań poszczególnym rdzeniom
  - AMP/ASMP - Asymmetric multiprocessing
  - SMP - Symmetric multiprocessing
  - BMP - Bound multiprocessing

AMP vs SMP

- AMP
  - Różne role rdzeni – często jeden rządzi
  - Nierówny dostęp do zasobów
  - Balansowanie obciążenia procesorów na barkach programisty
  - Mało popularne ze względu na stopień skomplikowania tej architektury. Używane do wysoce specjalistycznych celów.
- SMP
  - Rdzenie symetryczne
  - Identyczny dostęp do zasobów
  - Przydziałem zadań zarządza system operacyjny
  - Rosnąca popularność. Powszechne zastosowanie w obecnych systemach operacyjnych

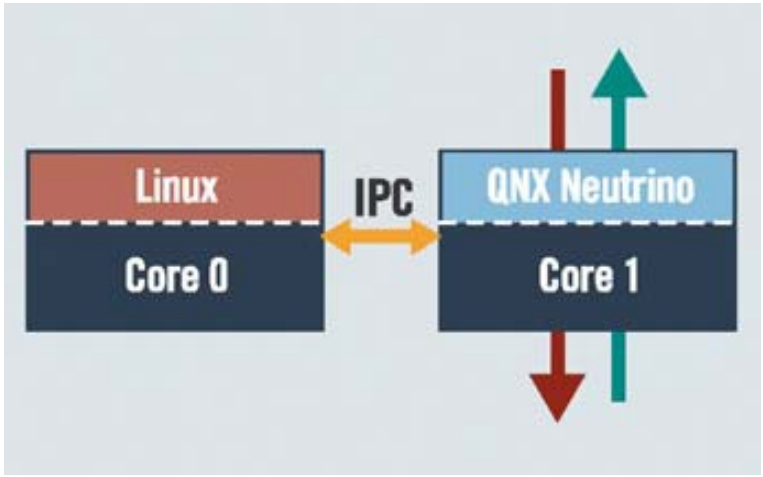
Inne oblicze AMP



- Różne instancje systemu operacyjnego (systemy też mogą być różne) działają na każdym z rdzeni
- Cechy:
  - Środowisko wykonania kodu podobne do systemu jednoprocesorowego
    - Łatwa migracja kodu
  - Trudności z komunikacją między aplikacjami działającymi na różnych rdzeniach
  - Problemy związane z dostępem do wspólnych urządzeń zewnętrznych
  - Ograniczona możliwość przełączania aplikacji pomiędzy rdzeniami

AMP - przykłady

- 2 rdzenie – 2 systemy:
  - Linux – zadania zwykłe
  - QNX Neutrino – zadania czasu rzeczywistego

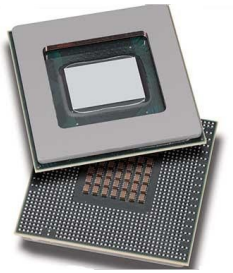


- Giga wydajna konsola:
  - Procesor Cell
    - 9 rdzeni
    - Master-Slave, jeden rządzi pozostałe liczą
    - ok. 250 GFlopów

AMP - przykłady

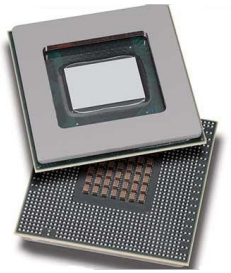
- Asymmetric MultiProcessing Linux
  - Zaimplementowany jako patch do zwykłego jądra Linuxa
  - Podział systemu operacyjnego na partycje:
    - Systemowa – tylko jedna
    - Czasu rzeczywistego – co najmniej jedna
  - W ramach partycji przydzielane są pamięc, procesory, urządzenia wejścia/wyjścia.

Sposoby wspierania wielordzeniowości - BMP



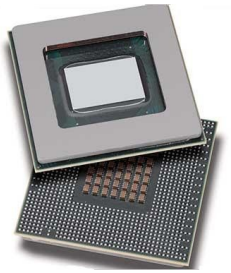
- Ponownie jeden system operacyjny, który wszystkim rządzi
- Podstawowa różnica: Przypisania każdej aplikacji do jednego rdzenia
- Proces działa niby w środowisku jednoprocesorowym
- Łatwe przełączani aplikacji pomiędzy rdzeniami
- Nie ma problemu z migotaniem cache'u jak w SMP

Architektura ccNUMA



- CcNUMA – cache coherent Non-Uniform Memory Access
- Procesory mają pewien przydzielony im obszar pamięci RAM, do którego mają szybszy dostęp (ok. 3 razy)
- Nie tylko superkomputery: AMD Opteron i Intel Nehalem
- Konieczne zapewnienie lokalności wykonania procesu:
  - Cała pamięć i wykonanie procesu w jednym węźle

SMP - przykłady



- Powszechnie stosowane rozwiązanie we współczesnych PC
- Przykładowe systemy operacyjne wspierające SMP
  - Windows od 2000
  - Linux od wersji jądra 2.0

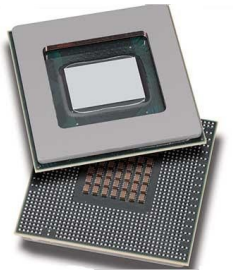
Obsługa SMP w linuxie

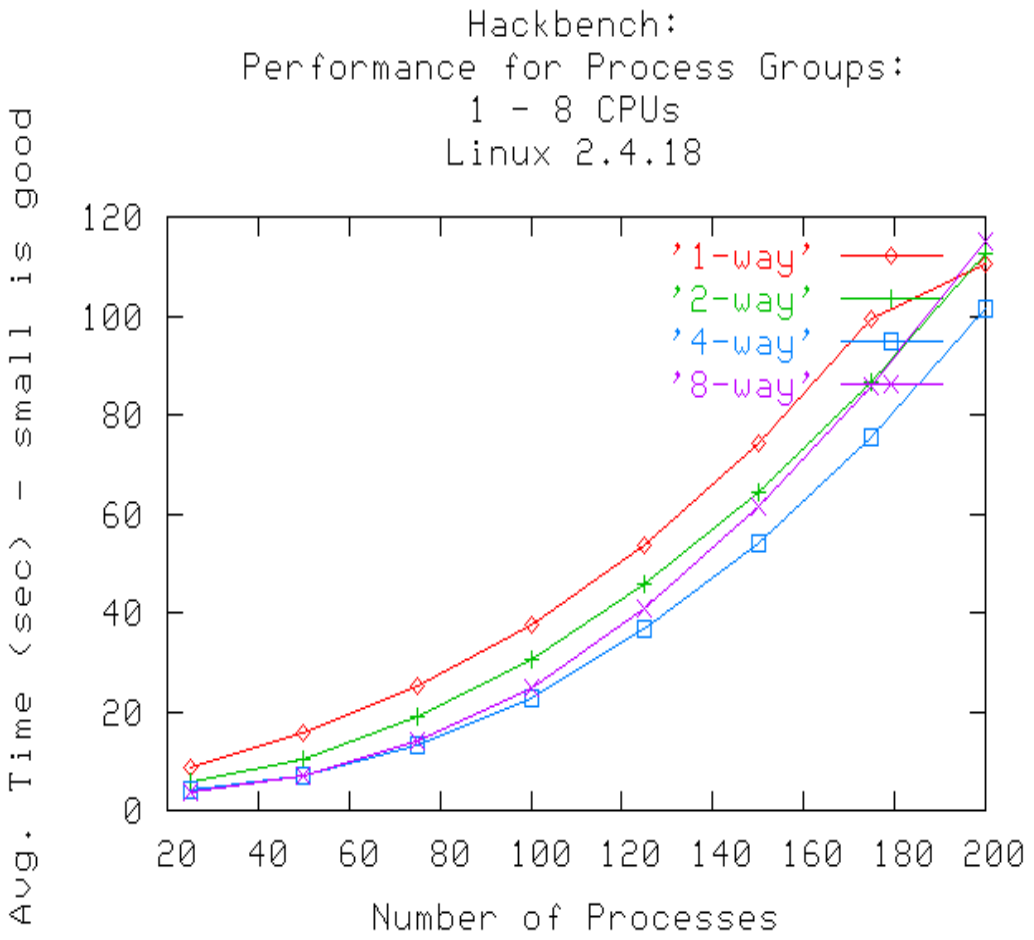
- wsparcie SMP po raz pierwszy pojawia się w jądrze v2.0
- Aby włączyć obsługę SMP wystarczy skąpilować jądro z odpowiednią opcja (zakładka Processor type and features)
- Pierwsze wersje obsługi SMP były dość mało skomplikowane i mało skuteczne (scheduler – 1400 linii kodu)m.in:
  - Nie sprzyjały procesom zorientowanym na wejście-wyjście
  - Nie wywłaszczały procesów
- Prawdziwy skok w rozwoju SMP nastąpił w jądrze v2.6, wraz z nowym algorytmem szeregowania zaproponowanym przez Ingo Molnár'a

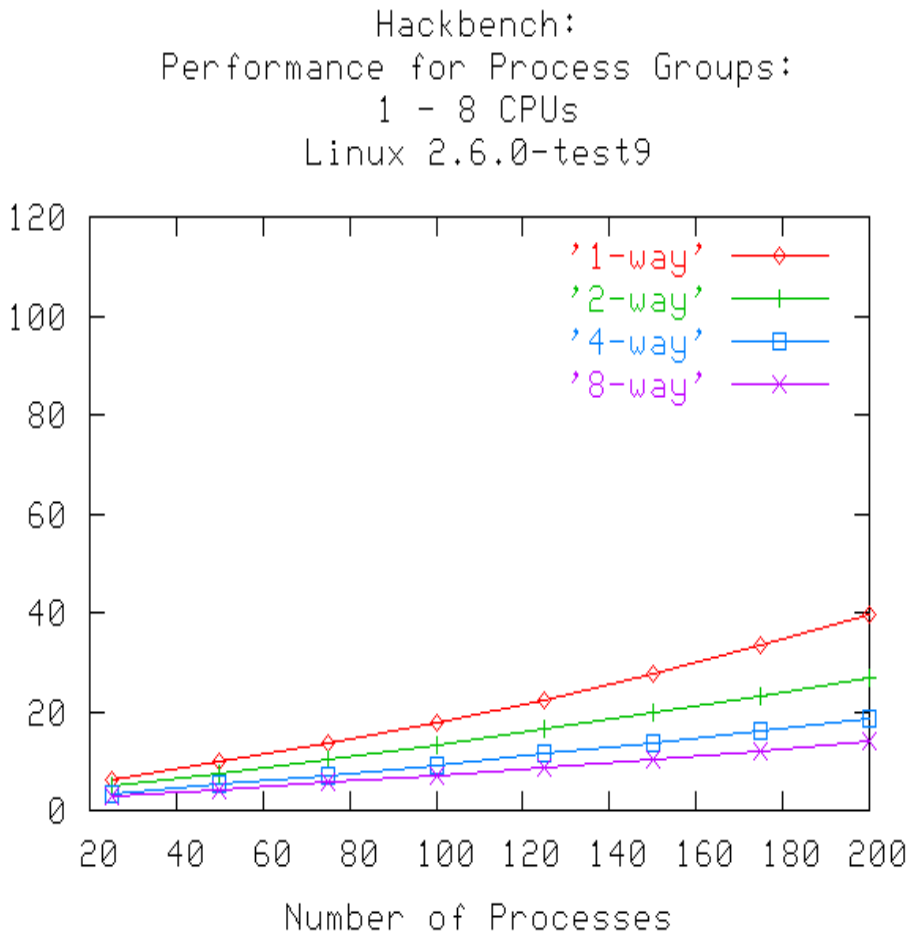
Scheduler w jądrach 2.4 i 2.6

Jądro v2.4	Jądro v2.6
Szeregowanie procesów w czasie O(n)	Szeregowanie procesów w czasie O(1)
Jest jedna kolejka procesów współdzielona przez wszystkie procesory	Każdy procesor ma na własność dwie kolejki procesów: active i expired
Przeszeregowanie zadań wiąże się z założeniem spinlocka na kolejke procesów	Podczas przeszeregowywania nie są blokowane kolejki pozostałych procesorów
Brak równoważenia pracy procesorów	Równoważenie pracy procesorów co 200ms – w razie potrzeby przenosi procesy między kolejkami (funkcja load_balance)
Nie dba o dobre wykorzystanie cache procesora	Dba o maksymalne wykorzystanie cache procesora – procesy trafiaja do tego samego procesora
Nie skalowalny	Poprawa skalowalności

Testy potwierdzające teorię

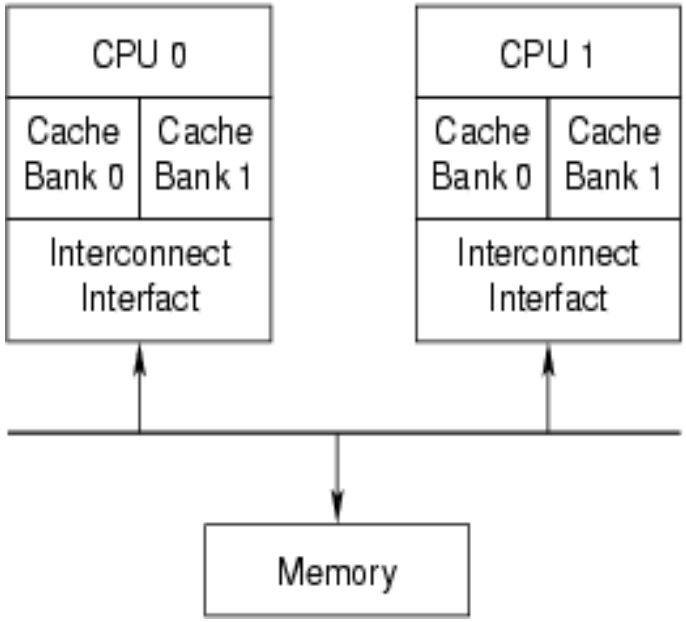






Problemy z SMP

- Cache składa się z niezależnych działających równolegle pul
- W systemach wieloprocesorowych może to się objawiać nieintuicyjnymi przeplotami np:



- CPU 0 CPU 1
- {A == 1; B == 2}
- A = 3; x = A;
- B = 4; y = B;
- Możliwe 24 możliwości przeplotów i 4 różne wartości (x,y)
- Nieintuicyjne przeploty zarówno w zapisie jak i odczycie

Rozwiązanie linuxa

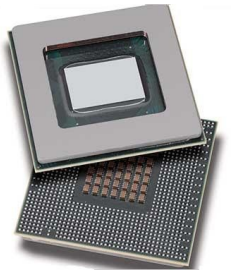
- Zmiana kolejności wykoniania instrukcji wymaga wsparcia sprzętowego
- W kodzie linuxa(v2.0) bardzo dobrze wyabstrachowano różnice na konkretnych CPU i zaimplementowano bariery pamięci:
  - Ogólne (smp_mb())
  - Odczytu (smp_rmb())
  - Zapisu (smp_wmb())
  - Zależności danych (smp_read_barrier_depends())

	Loads Reordered After Loads?	Loads Reordered After Stores?	Stores Reordered After Stores?	Stores Reordered After Loads?	Atomic Instructions Reordered With Loads?	Atomic Instructions Reordered With Stores?	Dependent Loads Reordered?	Incoherent Instruction Cache/Pipeline?
Alpha	Y	Y	Y	Y	Y	Y	Y	Y
AMD64	Y			Y				
IA64	Y	Y	Y	Y	Y	Y		Y
(PA–RISC)	Y	Y	Y	Y				
PA–RISC CPUs								
POWER	Y	Y	Y	Y	Y	Y		Y
SPARC RMO	Y	Y	Y	Y	Y	Y		Y
(SPARC PSO)			Y	Y		Y		Y
SPARC TSO				Y				Y
x86	Y	Y		Y				Y
(x86 OOStore)	Y	Y	Y	Y				Y
zSeries				Y				Y

Bariery pamięci w praktyce

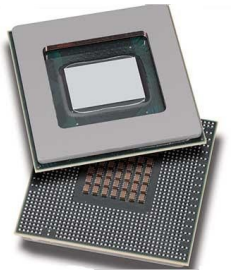
- Bariery pamięci gwarantują nam częściowy porządek
- CPU 1 CPU 2
  - ========== ==========
  - a = 1; }---- ---> { v = c
  - b = 2; } \ / { w = d
  - <write barrier> \ <read barrier>
  - c = 3; } / \ { x = a;
  - d = 4; }---- --->{ y = b;
- Bariery pamięci są wykorzystywane w semaforach, spinlockach, RCU, a także w niektórych operacjach atomowych – które stanowią interfejs dla programisty

Windows – SMP i ccNUMA



- Wsparcie od wersji Windows 2000
- Zmiany w mechanizmach synchronizacji
- Nowy scheduler dla procesorów wielordzeniowych
- Nowe informacje przechowywane w strukturach jądra – Dispatcher Database

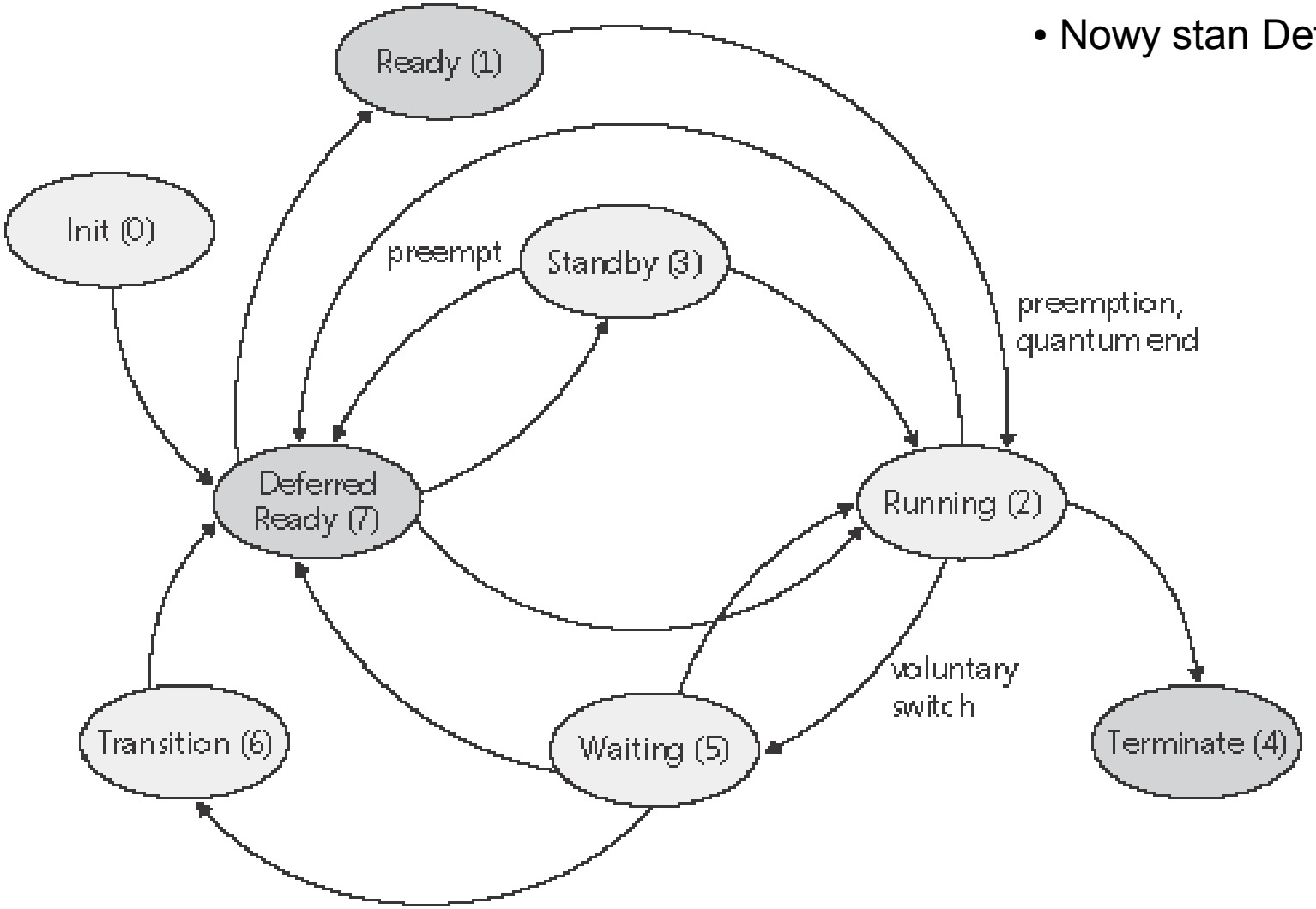
Dispatcher Database



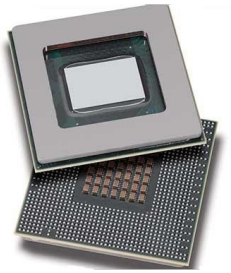
- Wiedza potrzebna do szeregowania procesów
- Dodano informacje o stanie procesorów
  - Ciekawostka: Licencja na liczbę procesorów :)
- Od (Win 2003) po kolejce p. gotowych na procesor
- Dodatkowe spinlock'i:
  - dispatcher, zmiana kontekstu – XP
  - processor control block (PRCB) – Windows 2003
- Blokady wszystkich struktur jądra jeszcze istnieją(Win 2003)
  - Na przykład podczas modyfikacji struktur synchronizacji
- Nowy stan i kolejki procesów – deffered ready

Stany wątków w Windowsie

- Nowy stan Deferred Ready

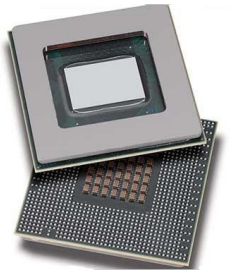


Synchronizacja



- Poważne zmiany dotyczące dostępu do struktur jądra
  - Kiedyś zapewniano wyłączny dostęp podnocząc IRQL
  - W SMP dwa wątki mogą to zrobić jednocześnie
- Obecnie wykorzystywane mechanizmy:
  - Spinlocki (zwykłe i kolejkowane)
  - Atomowe porównania, zmiana +/- 1,
  - Atomowe operacje na listach

Windows – process affinity

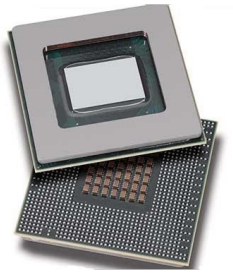


- Process affinity
  - Ograniczenie wykonania wątku do wybranych procesorów
- Jak to zrobić?
  - *SetThreadAffinity, SetProcessAffinity*
  - *Imagecfg tool in the Windows 2000 Server Resource Kit Supplement 1 (flagi -a , -u)*
- Może zmniejszyć liczbę czasu CPU jaki proces otrzymuje
- Może też poprawić wydajność, mniejsze migotanie cache'u
- Trik na tymczasowe obejście synchronizacyjnych bug'ów

Idealny i ostatni procesor wątku

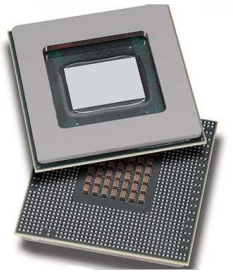
- Idealny – preferowany przez wątek
- Cykliczny wybór idealnego procesora dla wątków w procesie
  - Ukryte założenie o równym rozłożeniu pracy
  - *SetThreadIdealProcessor*
  - Uwzględnia procesory hiperwątkowe
  - Dla architektur NUMA cyklicznie przydziela procesy do idealnych węzłow
    - Dygresja: Funkcje operujące na pamięci w węźle

Wieloprocesorowe algorytmy szeregowania wątków



- Gdy procesor wolny (idle):
  - idealny → poprzedni → aktualny(scheduler) → inny
- Gdy wszystkie zajęte, to wykonanie na procesorze idealnym
- Brak wyrównywania obciążenia rdzeni
  - Procesor bezczynny → zaczyna sam szukać wątku
- Gwarancja, że wątek o najwyższym priorytecie jest wykonywany

Niespójność cache'u i rozwiązanie w AIX'ie

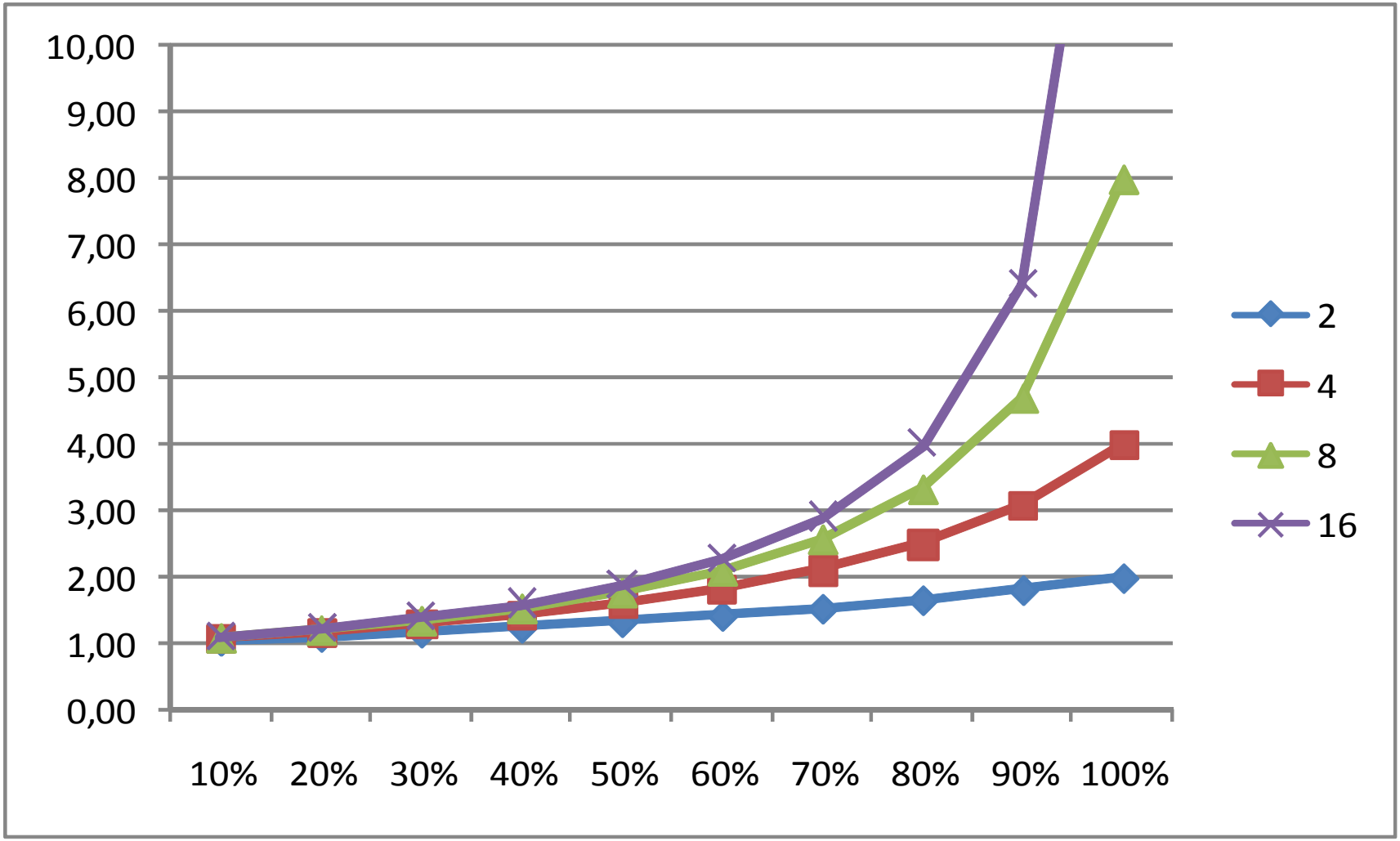


- Dwa procesory mogą korzystać z tej samej komórki pamięci, które zostaną ściągnięte do cache'a → łatwo o niespójność
- Snooping logic – rozgłaszanie i nasłuchiwanie na szynie
- Cross invalidation – po zmianie zawartości komórki pamięci w cache'u unieważnienie tej komórki w innych cache'ach
- Przydzielanie wątków do procesorów uwzględnia pamięć cache
  - Skłonność do przydzielenia procesu do procesora X zależy proporcjonalnie od rozmiaru roboczego cache'u wątku i odwrotnie proporcjonalnie od długości czasu jaki minął, gdy ostatnio wątek pracował na procesorze
  - Opłacalne dla wątków z dużą ilością obliczeń

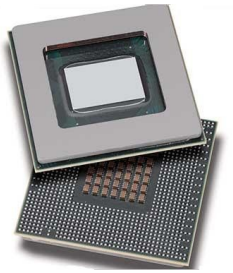
Korzyści z wielordzeniowości – Prawo Amdahla

- Korzyść z n-krotnego przyspieszenia części procesu, zajmującej t * 100% czasu to: $\frac{1}{(1-t)+\frac{t}{n}}$
- Dominująca część czasu zużywana jest przez czynność nieprzyspieszoną
- Dobry wynik to przyspieszenie działania o 60% przy dwóch rdzeniach
  - To oznacza 2-krotne przyspieszenie ok.80% czynności wykonywanych przez proces

Korzyści z wielordzeniowości – Prawo Amdahla



Wydajność w systemach wielordzeniowych



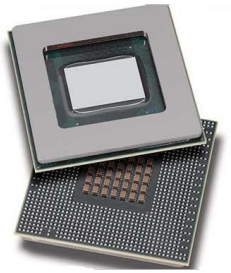
- Szybkość przetwarzania zależy od:
  - Zbalansowania obciążąnie pracą procesorów
  - Rywalizacji o blokady
  - Przywiązania wątków do jednego procesora
  - Rywalizacji o pamięć i szynę
  - Kosztu chybień w pamięci cache
- Wydajność jednego programu, jeśli nie jest wielowątkowy może się zmniejszyć

Programy potrafiące wykorzystać wielordzeniowość

- Wszystkie rodzaje serwerów: Apache, Tomcat, JBoss
- Obliczeniowe z możliwością zrównoleglania: algorytmy genetyczne, Blast, operacje macierzowe
- Kompilatory
- Łamacze haseł, szczególnie metodą brute-force
- Najnowsze gry 3D

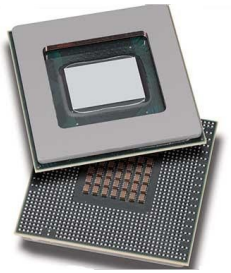
- Ogólna prawidłowość: Aplikacje wykonujące dużą ilość niezależnych obliczeń

Test dla procesorów wielordzeniowych



- Pokaz testów wykonanych przy pomocy programu Nuclearus na Windows XP na procesorze dwurdzeniowym
  - Bez ograniczeń
  - Z ograniczeniem programu testującego tylko do jednego procesora (przypisanie „affinity”)

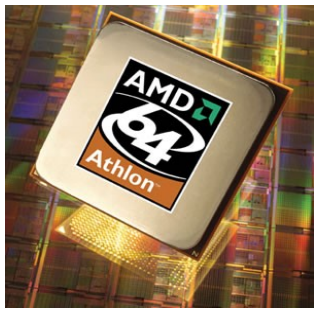
64 bity - wprowadzenie



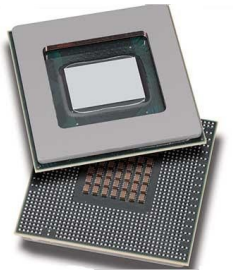
- Słowo procesora ma 64 bity
  - czyli dwa razy tyle co w systemie 32 bitowym
- Daleko idące implikacje dla systemu
  - szybkość obliczeń
  - obsługa pamięci
  - I/O
  - wsteczna kompatybilność!

Po co nam 64 bity?

- Adresowanie większej ilości pamięci
  - szczególnie per-proces
- Mapowanie w całości dużych plików i dysków
  - na przykład dysku DVD/blu-ray
- (Czasami) (nieco) szybsze działanie
- Marketing wielkich firm



Zastosowanie



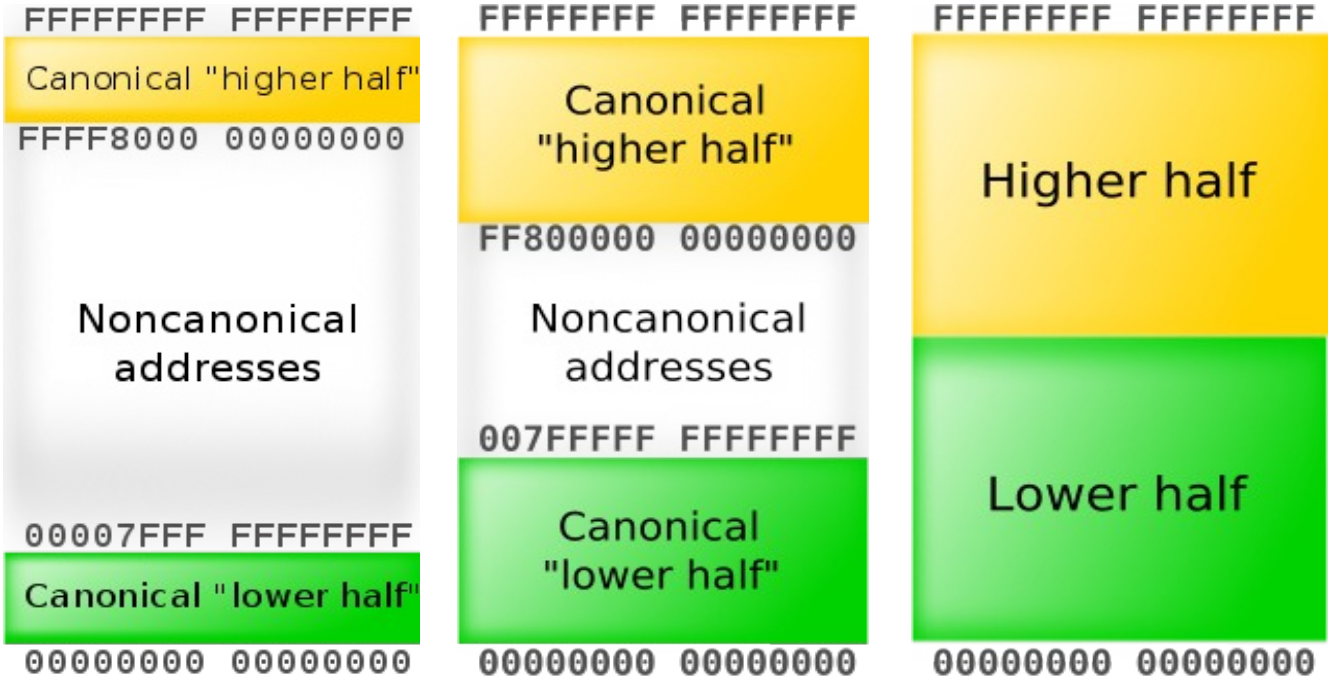
- Systemy serwerowe
- Bazy danych
- Ośrodki badawcze przetwarzające dużo danych
  - Genewa - CERN Linux, 64 bity

Fakty i mity

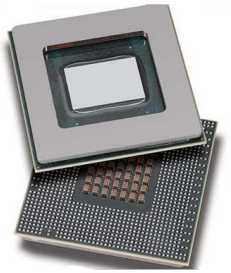
- tylko 64 bitowa aplikacja używa 64 bitowych danych?
  - sizeof(long long) == 64 !
    - ale przesyłane w postaci pary słów
- kernel 32 bit oznacza system 32 bitowy?
  - rzadko kiedy potrzebuje więcej niż 4 GB pamięci
    - może obsługiwać 64bity jedynie na zewnątrz

Architektura a pamięć

- teoretycznie adresujemy aż 16 EB
  - = 18.446.744.073.709.551.616 bajtów
  - obecnie używanych jest tylko 48 bitów adresu (Linux: 43)

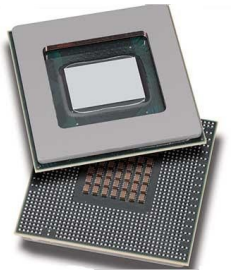


Pamięć – jak nią zarządzać?



- W systemach 32 bitowych
  - 3 poziomy systemu tablic/katalogów/stron
- W systemach 64 bitowych
  - „Page-Map Level 4 Table” + zwiększona ilość wpisów
- Przy 48 bitach adresu i 4 KB na stronę:
  - pełne tablice to 512 mb danych
    - porównanie z rozmiarami cache'a

W świecie Intela



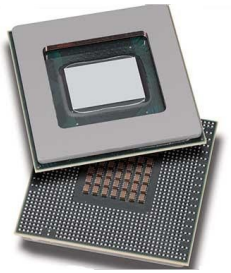
- 2x więcej rejestrów w x86-64 niż w poprzednikach
- 2x większe rejestry
  - szybsze działanie samego procesora
- Większe słowo zajmuje więcej w TLB!
  - Częściej chybiamy – mamy mniej słów
    - Zysk z obliczeń tracony na odczyt z pamięci!

Od strony programisty

- Zmieniony model danych - LP64

Data type	ILP32 size	ILP32 alignment	LP64 size	LP64 alignment
char	1 byte	1 byte	1 byte	1 byte
short	2 bytes	2 bytes	2 bytes	2 bytes
int	4 bytes	4 bytes	4 bytes	4 bytes
long	4 bytes	4 bytes	**8 bytes**	**8 bytes**
pointer	4 bytes	4 bytes	**8 bytes**	**8 bytes**
size_t	4 bytes	4 bytes	**8 bytes**	**8 bytes**
long long	8 bytes	4 bytes	8 bytes	**8 bytes**

Od strony programisty # 2



- Uwaga na wskaźniki!

```
int *c = malloc(...);
int *d = (int *)((int)c + 4); // ŹLE!
```

- Poprawka – rzutowanie na pewniejszy (char*)

```
int *d = (int *)((char *)c + 4); // OK
```

Od strony programisty

- Kolejna pułapka - maski

```
long fun(long param)
{
    long maska = ~0x3; // 0xfffffffc lub 0xfffffffffffffffc
    return (param & maska);
}
```

- Uniwersalne rozwiązanie – ifdef – czy dobre?

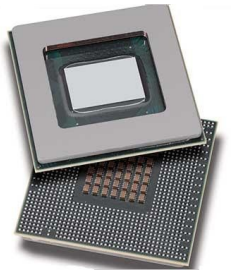
```
#ifdef _LP64
// 64-bit
#else
// 32-bit
#endif
```

Kompatybilność z 32 bit

- Wiele używanych aplikacji jest 32 bitowych
  - Nie każdy ma linuksa i może skompilować z -m64
    - SO musi zapewnić kompatybilność wstecz!

16bit DOS + Win32 + 64 bit =

Przykład - Windows



- Dawniej – Windows\System i Windows\System32
  - Logiczny sposób na utrzymanie kompatybilności
- „Oczywisty krok naprzód” - Windows\System64
  - Realia - Windows\wow64...

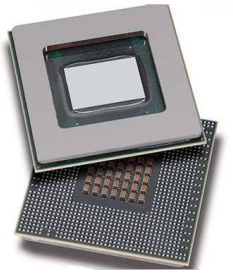
...DLL HELL

Wydajność + testy

- 32 bit = 100%
  - Logiczny sposób na utrzymanie kompatybilności
- Ubuntu Eddy Eft (UT 2004 DM Rankin) - 93.0%
- Windows XP 32/64 bit (Quake III) – 94.4 %
- Mac OS X (geekbench) - 107.9%

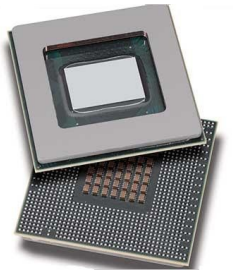
...a teraz policzymy silnię

Bibliografia



- http://www.mimuw.edu.pl/~marpe/arch/index.html
- http://cse.stanford.edu/class/sophomore-college/projects-00/risc
- http://arstechnica.com/articles/paedia/cpu/pipelining-1.ars/1
- http://www.embedded.com/98/9810fe3.htm
- http://www.frazpc.pl/artykuly/633/Architektura/procesorow/Intel/Core/i7/ (Nehalem)
- http://www.faqs.org/docs/Linux-HOWTO/SMP-HOWTO.html
- http://www.ibm.com/developerworks/library/l-linux-smp/
- http://www.ibm.com/developerworks/linux/library/l-scheduler/

Bibliografia



- http://rainbow.mimuw.edu.pl/SO/Projekt04-05/temat4-g5/#c3-2
- http://students.mimuw.edu.pl/SO/Wyklady-html/03_schedule/3_schedule.html
- http://rainbow.mimuw.edu.pl/SO/Linux/Temat08/SMP.html#migracja-zarzadzanie
- http://www.linuxjournal.com/article/8211
- Dokumentacja linuxa: memory-barriers.txt
- Dokumentacja systemu AIX - Multiprocessing http://publib.boulder.ibm.com/infocenter/systems/scope/aix/topic/com.ibm.aix.prftungd/doc/prftungd/intro_mulitproc.htm?tocNode=int_121443
- Mark E. Russinovich, David A. Solomon Microsoft® Windows® Internals, Fourth Edition: Microsoft Windows Server™ 2003, Windows XP, and Windows 2000, Microsoft Press, 2004

Bibliografia

- Wikipedia- hasła: 64-bit, X86-64, SMP, Multicore
- https://twiki.cern.ch/twiki/bin/view/Atlas/PlatformsAndCompilers
- http://searchenterprisedesktop.techtarget.com/tip/0,289483,sid192_gci1204121,00.html
- http://searchwindowsserver.techtarget.com/news/article/0,289142,sid68_gci1041454,00.html
- http://developer.apple.com/DOCUMENTATION/Darwin/Conceptual/64bitPorting

Programy testujące

- Multicore CPU Benchmark Nuclearus - http://nuc-rus.narod.ru/eng.htm
- Test 64-bit - http://students.mimuw.edu.pl/~ts248384/inne/64bit.zip

Bibliografia

- Wikipedia- hasła: 64-bit, X86-64, SMP, Multicore
- https://twiki.cern.ch/twiki/bin/view/Atlas/PlatformsAndCompilers
- http://searchenterprisedesktop.techtarget.com/tip/0,289483,sid192_gci1204121,00.html
- http://searchwindowsserver.techtarget.com/news/article/0,289142,sid68_gci1041454,00.html
- http://developer.apple.com/DOCUMENTATION/Darwin/Conceptual/64bitPorting