Wpływ architektury komputera na system operacyjny

Tomasz Badowski, Kamil Herba, Łukasz Kidziński

11 listopada 2008

Gdzie szukać wpływu?

Uproszczony schemat działania świata:
Potrzeba -> Realizacja -> Korzyści -> Nowe potrzeby
Spróbujemy spojrzeć skąd brały się kolejne zmiany.

Trochę liczb

Moc procesora

- Cena FLOPS:
  - ok 1,100,000,000,000$ (1.1 trylion $) za GFLOPS (potrzeba 17 milionów jednostek IBM 1620)
  - 2008: ok 0.20$ za GFLOPS (wystarcza Sony PS3)
- moc co 1,5 roku większa o 100%

Pamięć:

- Ceny:
  - 1978: 512K pamięci VAX-11/780 za 30,000$
  - 2008: 1MB dysku twardego SATA za 0,001$
- podwaja się co dwa lata

Po co komu OS?

Zanim pojawiły się systemy operacyjne

- Każdy program musiał znać architekturę
- Nie dało się uruchomić kilku programów jednocześnie
- ani nawet jeden po drugim

Lepsze komputery i zapotrzebowanie na aplikacje zapoczątkowało rozwój systemów.
Pojawiały się problemy:

- Zamazywanie nie tego co trzeba
- Potrzebna informacja kiedy zmienić taśmę
- Chęć wyższego poziomu (języków)

lata 60-te - wielozadaniowość/wieloprocesorowość

1961 - CTSS - Compatible Time-Sharing System
Okazuje się, że pierwsze schedulery mają już ponad 45 lat.
System używany na IBM 7094. Innowacje

- 5K z 32K przeznaczone dla systemu
- Przerwanie zegarowe
- Scheduler jako kolejka wielopoziomowa
- Sprzętowe zarządzanie pamięcią
- Instrukcje wyłapywane przez system

Wejście wyjście to standardowe urządzenia IBM. Sześć kanałów danych:

- drukarki, czytniki kart perforowanych, dziurkacze
- zapis na taśmy
- pamięć bębnowa
- wyświetlacze wektorowe
- IBM 7750 transmission control unit - 112 terminali

lata 60-te - wielozadaniowość/wieloprocesorowość

1961 - Multics - przodek UNIXa

- wszystko jest plikiem
- problemem było to, że maszyny wspierały tylko 1MB segmenty
- trzeba było wymyślić podział segmentów
- dynamiczne linkowanie
- stos dla każdego procesu i każdego trybu

lata 80-te - ethernet, sieć i grupy robocze

1979 - Lisp machine
1983 - Novell NetWare "file sharing instead of disk sharing"
pierwszy system zorientowany na pracę sieciową
1984 - Berkeley Software Distribution - Berkeley sockets

lata 90-te - komputery osobiste

W komputerach osobistych

- interpreter BASIC jako surowy OS
- uruchamianie programów
- nie potrzebny prawdziwy system
- single-task i single-user
- 4KB - 256KB RAMu
- system operacyjny jeszcze bardziej pogorszyłby osiągi

Im więcej może komputer, tym mniej chce robić człowiek.

Im więcej może komputer, tym mniej chce robić człowiek.

- Wykonywanie procesów jeden po drugim

Im więcej może komputer, tym mniej chce robić człowiek.

- Wykonywanie procesów jeden po drugim
- Dzielenie czasu

Im więcej może komputer, tym mniej chce robić człowiek.

- Wykonywanie procesów jeden po drugim
- Dzielenie czasu
- Gry, muzyka, filmy

Im więcej może komputer, tym mniej chce robić człowiek.

- Wykonywanie procesów jeden po drugim
- Dzielenie czasu
- Gry, muzyka, filmy
- Urządzenia przenośne

Im więcej może procesor, tym mniej chce robić programista.

Wprowadzanie danych

Im więcej może procesor, tym mniej chce robić programista.

Wprowadzanie danych

- karty z dziurkami

Im więcej może procesor, tym mniej chce robić programista.

Wprowadzanie danych

- karty z dziurkami
- wiersz poleceń

Im więcej może procesor, tym mniej chce robić programista.

Wprowadzanie danych

- karty z dziurkami
- wiersz poleceń
- GUI

Języki

Im więcej może procesor, tym mniej chce robić programista.

Wprowadzanie danych

- karty z dziurkami
- wiersz poleceń
- GUI

Języki

- Assembler

Im więcej może procesor, tym mniej chce robić programista.

Wprowadzanie danych

- karty z dziurkami
- wiersz poleceń
- GUI

Języki

- Assembler
- Fortran

Im więcej może procesor, tym mniej chce robić programista.

Wprowadzanie danych

- karty z dziurkami
- wiersz poleceń
- GUI

Języki

- Assembler
- Fortran
- C

Im więcej może procesor, tym mniej chce robić programista.

Wprowadzanie danych

- karty z dziurkami
- wiersz poleceń
- GUI

Języki

- Assembler
- Fortran
- C
- Java, .NET

Sieci komputerowe

Izolacja -> dial-up -> LAN i WAN
Problemy

Sieci komputerowe

Izolacja -> dial-up -> LAN i WAN

Problemy

- obsługa połączenia

Sieci komputerowe

Izolacja -> dial-up -> LAN i WAN

Problemy

- obsługa połączenia
- dostęp do odpowiednich danych

Sieci komputerowe

Izolacja -> dial-up -> LAN i WAN

Problemy

- obsługa połączenia
- dostęp do odpowiednich danych
- bezpieczeństwo

Nowe idee

Sieci komputerowe

Izolacja -> dial-up -> LAN i WAN

Problemy

- obsługa połączenia
- dostęp do odpowiednich danych
- bezpieczeństwo

Nowe idee

- architektura client-server

Sieci komputerowe

Izolacja -> dial-up -> LAN i WAN

Problemy

- obsługa połączenia
- dostęp do odpowiednich danych
- bezpieczeństwo

Nowe idee

- architektura client-server
- sockety

Inne trendy

Duże dyski:

Inne trendy

Duże dyski:

- usuwanie plików traci znaczenie

Inne trendy

Duże dyski:

- usuwanie plików traci znaczenie
- dodatkowe miejsce na indeksy i statystyki

Inne trendy

Duże dyski:

- usuwanie plików traci znaczenie
- dodatkowe miejsce na indeksy i statystyki
- struktura danych uzależniona od wyszukiwania

Inne trendy

Duże dyski:

- usuwanie plików traci znaczenie
- dodatkowe miejsce na indeksy i statystyki
- struktura danych uzależniona od wyszukiwania
- nie stronicujemy ważnych procesów

Trendy procesorów

- Przyspieszanie procesorów nie idzie już tak gwałtownie.
- Wieloprocesorowość w maszynach domowych.
- Przyszłość leży w kompilatorach, wieloprcesorowości i w systemach operacyjnych

Jakie mamy wsparcie?

Aby wspierać system operacyjny wprowadzono takie rozwiązania jak

- zegar systemowy
- synchronizacja przez atomowe 'test-and-set'
- ochrona pamięci
- obsługa I/O
- przerwania i wyjątki
- tryb chroniony
- chronione instrukcje
- wywołania systemowe

Jakie mamy wsparcie?

Niektóre instrukcje są dostępne TYLKO dla OS.
W szczególności tylko OS może

- komunikować się z I/O
- zarządzać pamięcią
  - dostęp do tablic stron
  - TLB
- zmieniać 'bit trybu'

Jak to jest z tym trybem chronionym?

- wymagada architektura wspierająca co najmniej dwa tryby
- bit ustawiany w rejestrze procesora
- chronione instrukcje tylko w trybie jądra

Dlaczego nie przyjęła się 4 stopniowa ochrona (rings)?

Jak to jest z tym trybem chronionym?

- wymagada architektura wspierająca co najmniej dwa tryby
- bit ustawiany w rejestrze procesora
- chronione instrukcje tylko w trybie jądra

Dlaczego nie przyjęła się 4 stopniowa ochrona (rings)?

- wszystko fajnie jak OS jest pisany specjalnie pod architekturę
- tak jest już rzadko - przeważnie chcemy jak największą abstrakcję

CISC vs RISC

CISC też miał zalety!

- wspieranie języków wysokiego poziomu
- krótsze programy

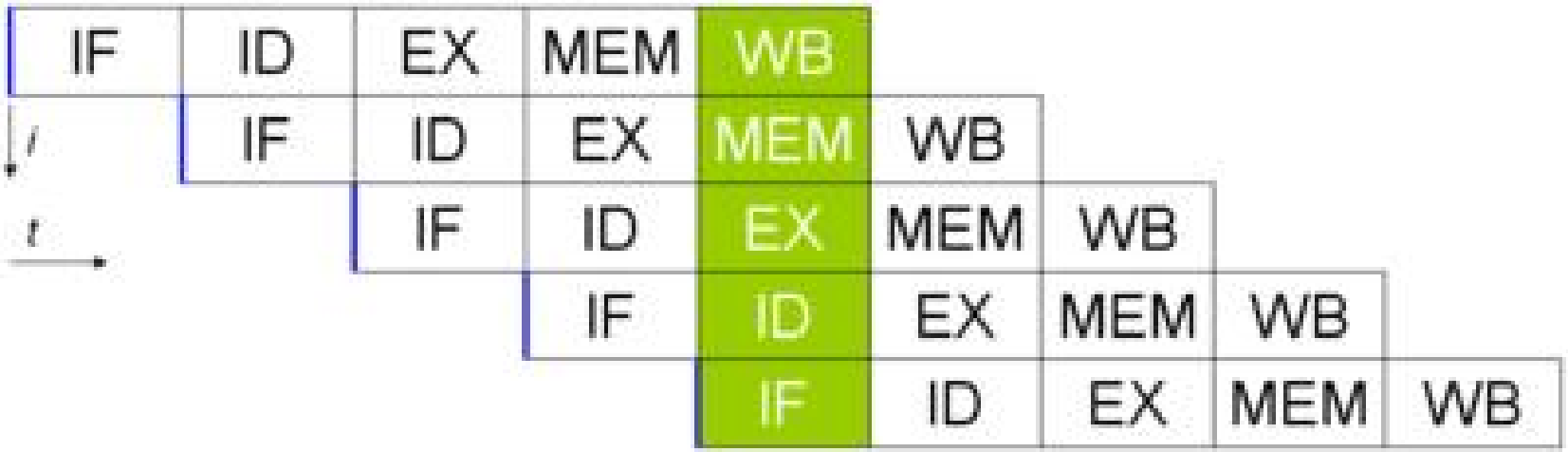
To już przeszłość.
Ale trzeba było zachować zgodność.
x86 -> interface CISC, translator RISC

Problemy: Wait state

Rozwiązania:

- CPU cache (instruction prefetch)
- branch prediction
- pipeline
- simultaneous multithreading
- Hardware scout(!)

Pipeline


IF
ID
EX
MEM
WB
IF
ID
EX
MEM
WB
IF
ID
EX
MEM
WB
IF
ID
EX
MEM
WB
IF
ID
EX
MEM
WB
i
t

Dobrodziejstwa x86 -> ulepszenia w systemach

Task State Segment.
Trzyma takie informacje jak:

- Processor register state
- I/O Port permissions
- Inner level stack pointers
- Previous TSS link

Możliwość szybszego przełączania kontekstu.
Linux tworzy tylko po jednym TSS na każdy CPU.

Dobrodziejstwa x86 -> ulepszenia w systemach

MMX.
8 rejestrów 64bitowych do arytmetyki
Wszystko fajnie, ale zbiory instrukcji nie dają semantyki w stylu stosu.
System operacyjny(!) musi sobie z tym poradzić jakoś inaczej.
3DNow!
Operacje zmiennoprzecinkowe + śtosowe"instrukcje => system nie musi się martwić

Dobrodziejstwa x86 -> ulepszenia w systemach

SSE.
Możliwości podobne jak 3DNow! tylko więcej i lepiej.
Intel dorzucił Ęnhanced mode".
Gdy system dowie się o próbie wejścia może włączyć Enhanced mode.
Jeśli nie wie, że coś takiego istnieje to zostaje w Protected mode.

Dobrodziejstwa x86 -> ulepszenia w systemach

Physical Address Extension (PAE)
Odwoływanie się do pamięci większej niż 4GB na komputerach o 32-bitowych.

Co dalej?

To co dziś mamy w desktopach będziemy mieć w kieszeni.
To co dziś mamy w mainframe'ach dostaniemy desktopach.

Rodzaje systemów wieloprocesorowych.

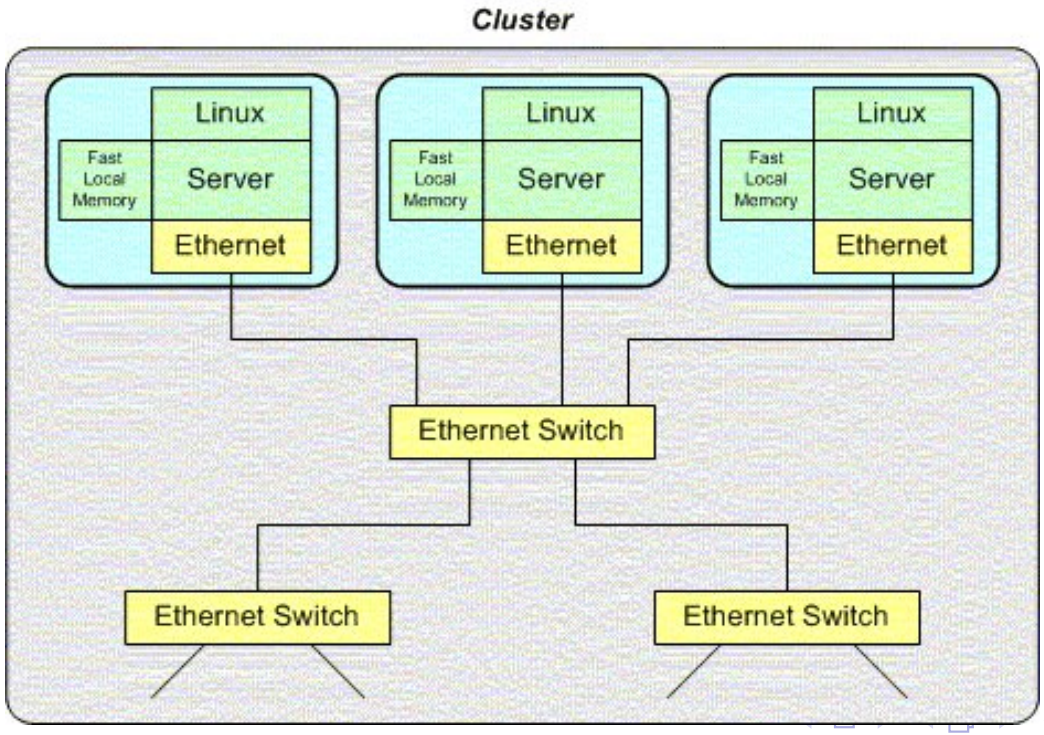
Wieloprocesorowość — jest to skoordynowane wykonywanie programów przez więcej niż jeden procesor w komputerze lub na kilku komputerach.

System wieloprocesorowy — według definicji ANSI (Amerykańskiego Państwowego Instytutu Normalizacji) z 1970r. jest to maszyna cyfrowa złożona z dwu lub więcej jednostek przetwarzania sterowanych centralnie. Zatem jest to pojedynczy komputer posiadający więcej niż jeden procesor, pracujący pod kontrolą jednego systemu operacyjnego.

- Systemy takie nazywamy też ściśle powiąznymi (tightly coupled).
- Wieloprocesorowość jest też wykorzystywana

Systemy luźno powiązane.

Systemy luźno powiązane — zwane też klastrami (ang. clusters) są złożone z wielu komputerów połaczonych szybkim systemem komunikacji (najczęściej Gigabit Ethernet). Przykładem jest Linux Beowulf cluster.



Systemy ściśle powiązane i procesory wielordzeniowe.

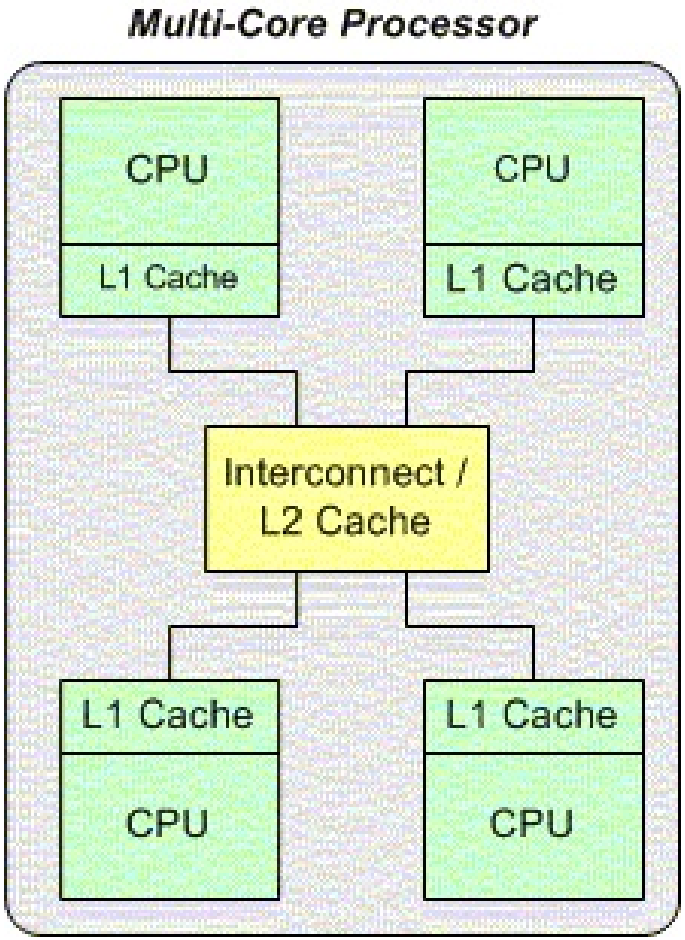
Systemy ściśle powiązne — wiele procesorów dzieli szynę, zegar czasem pamięć i urządzenia zewnętrzne.

- Może występować centralna zewnętrzna pamieć dzielona dla wszystkich procesorów (UMA – Uniform Memory Access), bądź zarówno własna jak i dzielona (NUMA Non-Uniform Memory Access, np. w architekturze Cray).

Najbardziej skrajną odmianą systemów ściśle powiązanych są komputery wykorzystujące procesory wielordzeniowe.

Prosecor wielordzeniowy — składa się z kilku jednostek obliczeniowych - rdzeni, umieszczonych w pojedynczym układzie scalonym, na jednej kości.

Procesor wielordzeniowy



Zalety systemów wieloprocesorowych.

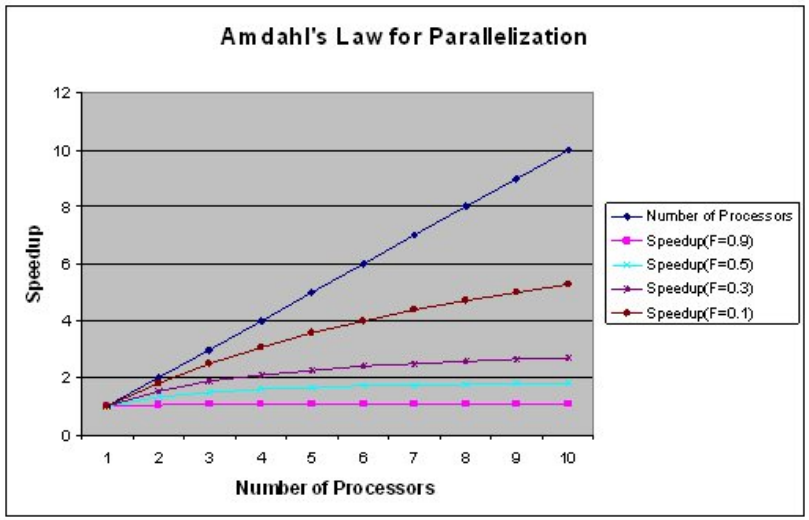
- Większa przepustowość - korzystając z większej liczby procesorów możemy wykonać większą pracę w tym samym czasie. Nie przyspieszamy jednak N razy używając N procesorów. Spowolnienie jest spowodowane m. in.
  - utratą czasu na koordynację działań wszystkich części,
  - rywalizacją o zasoby dzielone

  Przede wszystkim jednak nie wszystkie problemy nadają się do zrównoleglenia.

Prawo Amdahla

$$Przyspieszenie = \frac{1}{F + (1 - F)/N}$$

F to procent problemu, który nie da się zrównoleglić.

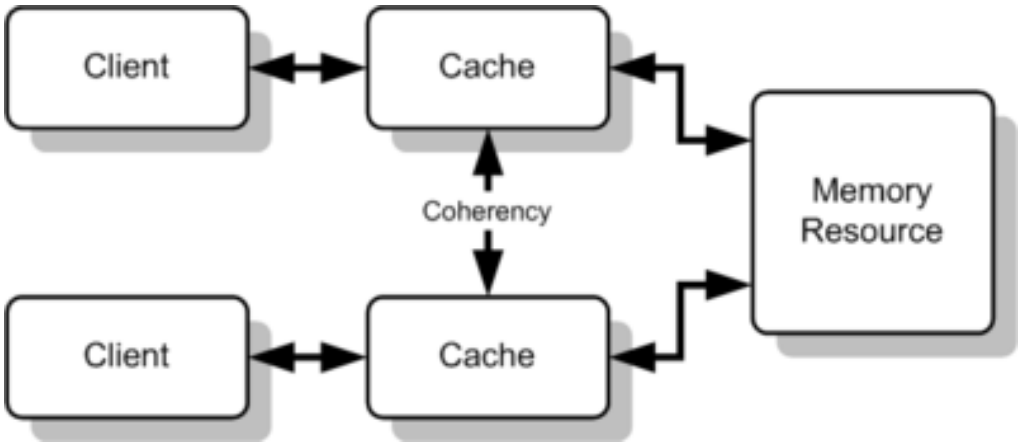


Zalety i wady systemów wieloprocesorowych c. d.

- Systemy wieloprocesorowe pozwalają na zmniejszenie kosztów, ponieważ dzielą urządzenia i pamięć zewnętrzna oraz źródło zasilania.
- Niezawodność - w systemach wieloprocesorowych jest mozliwe kontynuowanie pracy przy awarii jednego z procesorów

Wady systemów wieloprocesorowych

- Pełne wykorzystanie ich możliwości wymaga trudniejszego programowania równoległego.
- W przypadku pamięci współdzielonej powstaje spowolnienie spowodowane koniecznością utrzymywania spójności pamięci podręcznej. Jest to prawdopodobine najpoważniejsza przyczyna ograniczonej skalowalności tego typu architektur.



Dodatkowe wady i zalety systemów wielordzeniowych.

Zalety

- Umieszczenie procesorów na jednym rdzeniu umożliwia przyspieszenie układu spójności pamięci podręcznej
- Bliskość poprawia jakość sygnałów między procesorami i zmniejsza czas komunikacji mniejsze zużycie energii niż oddzielnych procesorów.

Wady

- Wykorzystanie tej samej szyny przez dwa procesory zmniejszyć wydajność jeśli szybkość przesyłania informacji jest czynnikiem ograniczającym.
- Większe problemy z chłodzeniem niż dla niezaleznych procesorów.

Duża popularność procesorów wielordzeniowych.

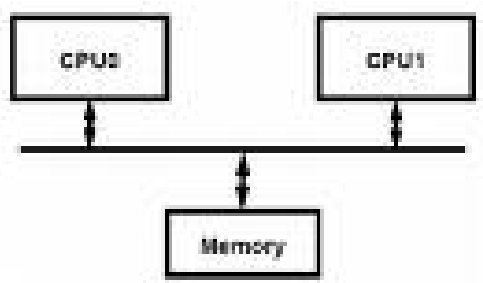
Producent	Nazwa	Opis
IBM	POWER4	SMP, dual CPU
IBM	POWER5	SMP, dual CPU
AMD	AMD X2	SMP, dual CPU
Intel	Xeon	SMP, dual or quad CPU
Intel	Core2 Duo	SMP, dual CPU
ARM	MPCore	SMP, up to four CPUs
IBM	Xenon	SMP, three Power PC CPUs
IBM	Cell Processor	ASMP, nine CPUs

We are dedicating all of our future product development to multicore designs. ... This is a sea change in computing
Paul Otellini, President, Intel (2005)

Mniej oczywiste wykorzystanie wielordzeniowości - GPU, procesory sygnałowe, wbudowane.

Modele pracy systemów wieloprocesorowych.

Najczęstszy model pracy wielu procesorów to **wieloprzetwarzanie symetryczne** (ang. symmetric multiprocessing - SMP), w którym każdym procesor pełni równorzędną rolę i ma dostęp do wspólnej pamięci.



Obecnie w niektórych systemach (np. konsole gier Sony PS3) występuje też **wieloprzetwarzanie asymetryczne** (ang. asymmetric multiprocessing - ASMP), w którym główny procesor przydziela zadania procesorom podporządkowanym (master – slave). Procesor nadrzędny ma dostęp do systemowych struktur danych, podejmuje decyzje planistyczne i wykonuje operacje IO. Przy wielu procesorach staje się to mało wydajne, ponieważ główny procesor może stanowić wąskie gardło.

Zależność modelu pracy od architektury i systemu operacyjnego.

Rozróżnienie między powyższymi modelami może być uwarunkowane sprzętowo albo przez system operacyjny.

- Mogą istnieć mechanizmy sprzętowe do rozróżniania procesorów o różnych rolach w systemie.
- Rozróżnienie może pochodzić jedynie od systemu, np. SunOS dla komputerów Sun umożliwia przetwarzanie asymetryczne, natomiast wersja 5 tego systemu (Solaris 3), wykorzystując ten sam sprzęt umożliwia przetwarzanie symetryczne.

Współczesne systemy operacyjne umożliwiają przetwarzanie symetryczne. Np. Windows NT, Solaris, Digital Unix, OS/2 i Linux.

Zmiany jądra Linuxa a SMP.

Lepsze wykorzystanie SMP przez jądro 2.6 w stusunku do jądra 2.4

- Wersja 2.4
  - pojedyncza kolejka dla wszystkich procesorów
  - Wybór przez procesor kolejnego procesu zajmował czas O(n) i wiązał się z założeniem blokady na kolejkę. Jeden procesor wykonywał przeliczanie priorytetów, podczas gdy inne procesory czekały na jej zakończenie.
- Wersja 2.6
  - Szeregowanie w czasie O(1)
  - Każdy procesor ma własną kolejkę active i expired i zakłada własne blokady. Proces wykonuje się możliwie długo na tym samym procesorze, co zwiększa efektywność wykorzystania cache
  - Co 200ms równoważone jest obciążenie procesorów.
    - Wadą tej czynności jest brak danych o procesie w cache nowego procesora (cold cache) i konieczność ich przesłania.

Jakie funkcje Linux 2.6 Schedulera wykorzystują SMP?

Poniżej wymieniam najważniejsze funkcje schedulera

schedule	Główna funkcja schedulera. Wybiera do wykonania procesy o najwyższym priorytecie.
load_balance	Sprawdza czy występuje brak równowagi obciążenia procesorów, jeśli tak to przesuwa procesy między procesorami.
effective_prio	Zwraca dynamiczny priorytet procesu.
recalc_task_prio	Określa bonus lub karę za czas przestoju.
source_load	"Konserwatywnie" oblicza obciążenie danego CPU, z którego proces może zostać zabrany.
target_load	"Liberalnie" oblicza obciążenie danego CPU, do którego proces może zostać przekazany.

source_load, target_load - wyjaśnienie działania.

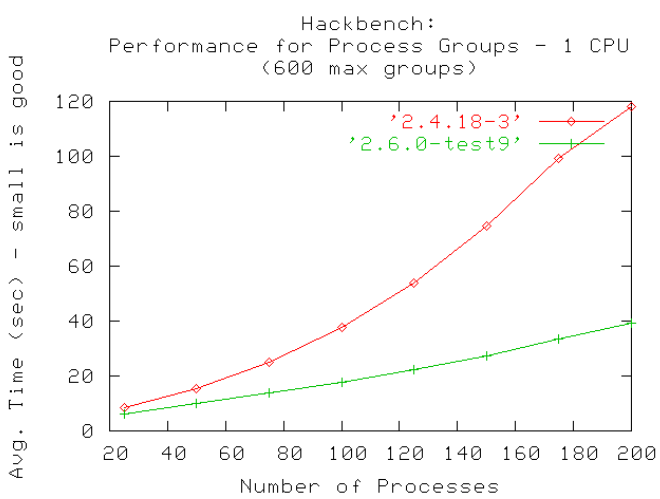
Linux-2.6.8 kernel, plik sched.c

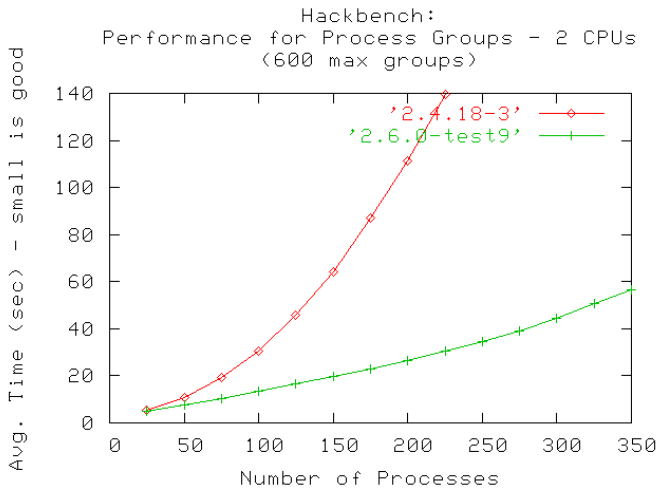
```
/*
* Return a low guess at the load of a migration-source cpu.
*
* We want to under-estimate the load of migration sources, to
* balance conservatively.
*/
static inline unsigned long source_load(int cpu)
{
runqueue_t *rq = cpu_rq(cpu);
unsigned long load_now = rq->nr_running * SCHED_LOAD_SCALE;
return min(rq->cpu_load, load_now);
}
/*
* Return a high guess at the load of a migration-target cpu
*/
static inline unsigned long target_load(int cpu)
{
runqueue_t *rq = cpu_rq(cpu);
unsigned long load_now = rq->nr_running * SCHED_LOAD_SCALE;
return max(rq->cpu_load, load_now);
}
```

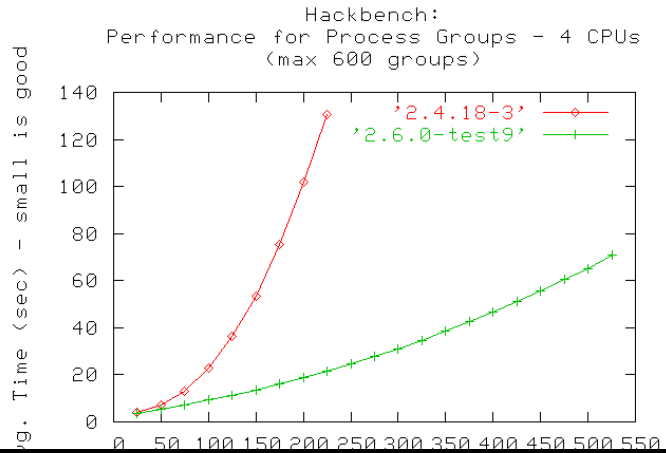
cpu_load jest aktualizowane przy wywołaniu rebalance_tick() jako (zaokrąglona w górę) średnia poprzedniej i aktualnej wartości (nr_running * SCHED_LOAD_SCALE).

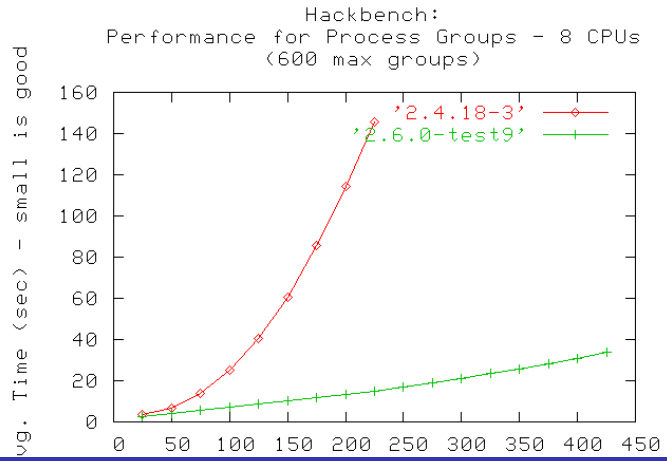
Testy poprawy wydajności.

Jądro Linuxa 2.4.18 było porównywane z jądrem 2.6.0-test9 kernel, przy użyciu programu hackbench. Oprócz poprawy wydajności i większej skalowalności, scheduler umożliwia lepsze wykorzystanie zasobów, dzięki czemu przy domyślnych ustawieniach jądra w systemie może działać więcej procesów niz wcześniej.









Jak wyzwolić moc SMP w programach użytkowych?

Wątki

- SMP jest najlepiej wykorzystywany przez programy wielowątkowe. Jedną z najopularniejszych implementacji wątków jest POSIX threads.
- Programy wielowątkowe ze współdzieloną pamięcią doskonale nadają się do wykorzystania w architekturach wielordzeniowych, ponieważ procesory dzielą wspólną pamięć.

Bezpośrenie wykorzystywanie wątków wymaga dużej ostrożności, ponadto żeby zrównoleglić w ten sposób istniejący progam trzeba wprowadzić wiele zmian i nowego kodu. Dlatego często stosuje się bardziej wysokopoziomowe metody zrównoleglania programów na architektury wielordzeniowe:

- OpenMP
- MPI

OpenMP.

- OpenMP to zestaw tzw. dyrektyw, czyli jakby komentarzy, podpowiadających kompilatorowi w jaki sposób można zrównoleglić kod. OpenMP pozwala zrównoleglać programy napisane w językach Fortran i C.
- Jest to bardzo wygodne z punktu widzenia programisty rozwiązanie, ponieważ pozwala na bardzo szybkie zrównoleglenie istniejącego kodu. OpenMP stosuje się w przypadku architektur o pamięci współdzielonej
- Podobnie jak wątki OpenMP jest idealny dla architektur wielordzeniowych.

Przykład programu w OpenMP.

Program liczący $\pi = 4 \int_0^1 \frac{1}{1+x^2}$

```
1. program openmpi
2. implicit none
3. integer i, n
4. real sumpi, mypart, x
5. parameter (n = 1000000)
6.
7. sumpi = 0.0
8. do i = 1, n
9. x = (i + 0.5) / n
10. mypart = 1.0/(1.0 + x**2)
11. sumpi = sumpi + mypart
12. end do
13. sumpi = 4.0 * sumpi / n
14. print *, sumpi
15. end program
```

```
1. program openmpi
2. implicit none
3. integer i, n
4. real sumpi, mypart, x
5. parameter (n = 1000000)
6.
7. sumpi = 0.0
8. !$omp parallel do default(none)
9. !$omp& private(i, x, mypart)
10. !$omp& shared(n, sumpi)
11. do i = 1, n
12. x = (i + 0.5) / n
13. mypart = 1.0/(1.0 + x**2)
14. !$omp critical
15. sumpi = sumpi + mypart
16. !$omp end critical
17. end do
18. sumpi = 4.0 * sumpi / n
19. print *, sumpi
20. end program
```

MPI.

- W przeciwieństwie do rozwiązań opartych na wątkach MPI wykorzystuje komunikaty do przekazywania informacji z pamięci jednego procesora do drugiego.
- Metoda ta jest wykorzystywana przede wszystkim w systemach z pamięcia rozproszoną, ale może być też stosowana dla programowania na architektury wielordzeniowe.
- W programach wykorzystujących MPI w systemach SMP procesory mają przydzielony pewien własny obszar pamięci a komunikaty przeznaczone dla innych procesorów zapisują do pamięci współdzielonej.

Bibliografia

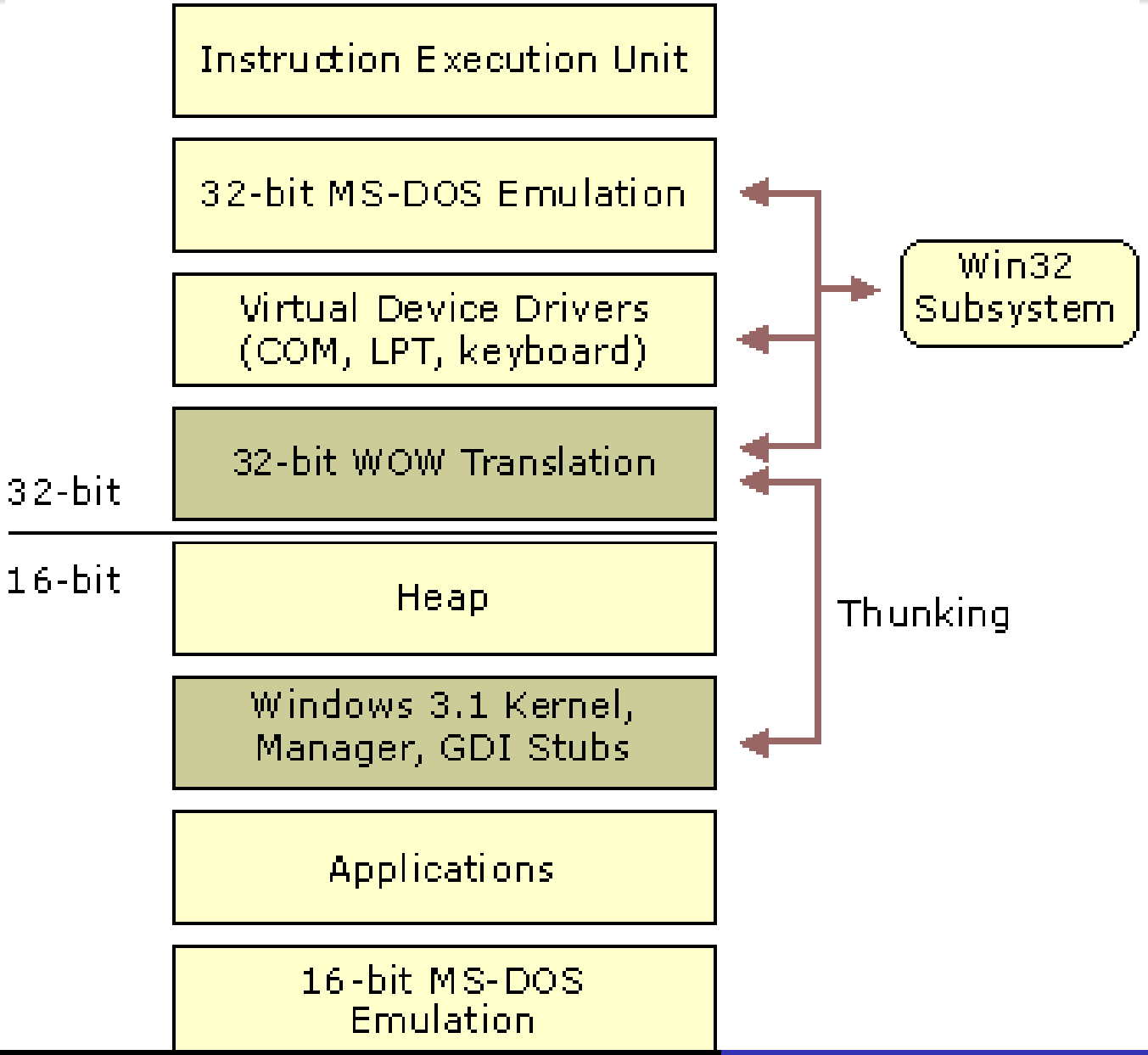
1. A. Silberschatz, P. B. Galvin. Podstawy systemów operacyjnych, WNT 2006.
2. M. Tim Jones. Inside the Linux scheduler. www.ibm.com/developerworks/linux/library/l-scheduler/
3. M. Tim Jones. Linux and symmetric multiprocessing. www.ibm.com/developerworks/library/l-linux-smp/
4. Douglas Eadline, Ph.D. The Multicore Programming Challenge. www.linux-mag.com/id/4312
5. Josh Aas. Understanding the Linux 2.6.8.1 CPU Scheduler c 2005 Silicon Graphics, Inc. (SGI) 17th February 2005
6. Linux Process Scheduler Improvements in Version 2.6.0. devresources.linux-foundation.org/craiger/hackbench/
7. Plik sched.c. www.gelato.unsw.edu.au/lxr/source/kernel/sched.c
8. OpenMP. www.icm.edu.pl/kdm/OpenMP
9. Wikipedia
10. M. Friedman, O. Pentakalos. Windows 2000 Performance Guide - Help for Administrators and Application Developers, Pentakalosoreilly.com/catalog/w2kperf/chapter/ch05.html

aplikacje 16-bitowe

Procesory 286 i wcześniejsze miały architekturę 16-bitową. Mogły adresować do 2 MB pamięci (21 bitów). Oczywiście nie było mowy o żadnym stronicowaniu.

Rozwiązanie

- Dosemu i VDM(Virtual Dos Machine - Win NT) - użycie sprzętowego trybu virtual 8086 mode
- Dosbox - programowa emulacja 286 - możliwość uruchamiania na procesorach z rodzin innych niż x86, amd64


Instruction Execution Unit
32-bit MS-DOS Emulation
Virtual Device Drivers
(COM, LPT, keyboard)
32-bit WOW Translation
Win32
Subsystem
32-bit
16-bit
Heap
Thunking
Windows 3.1 Kernel,
Manager, GDI Stubs
Applications
16-bit MS-DOS
Emulation

procesory 64-bitowe

- rozmiar pamięci, którą można zaadresować - ponad 16 miliardów Gigabajtów
- lepsza obsługa dużych plików
- obróbka danych o rozmiarze 2 razy większym niż dotychczas (grafika, obliczenia naukowe)
- brak problemu z datami powyżej roku 2038 (time_t przechowywane jako liczba 64-bitowa)

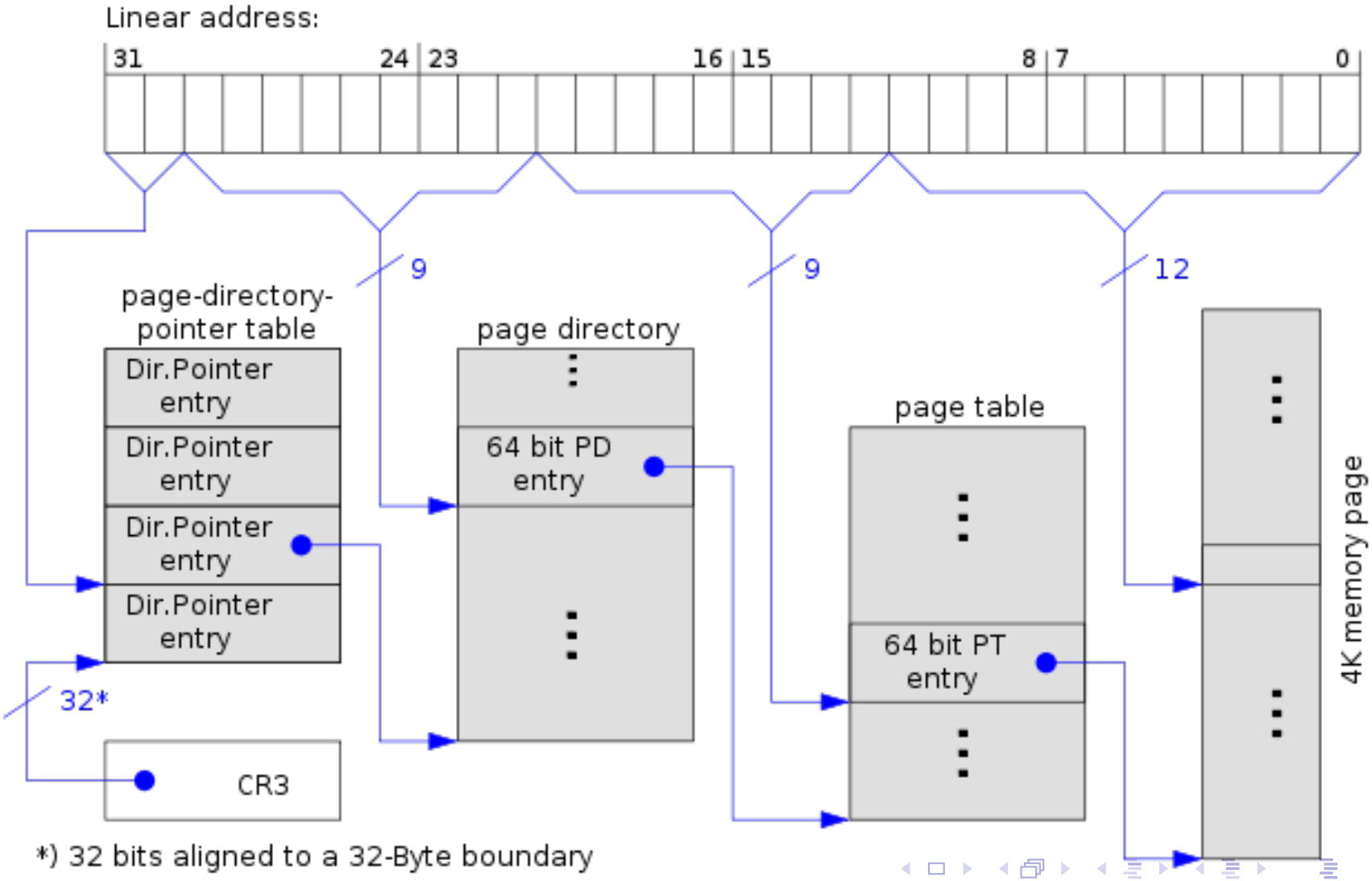
zastosowania dużych przestrzeni adresowych

- bazy danych
- data mining
- Web cache i wyszukiwarki
- CAD i CAE
- niektóre rodzaje obliczeń naukowych

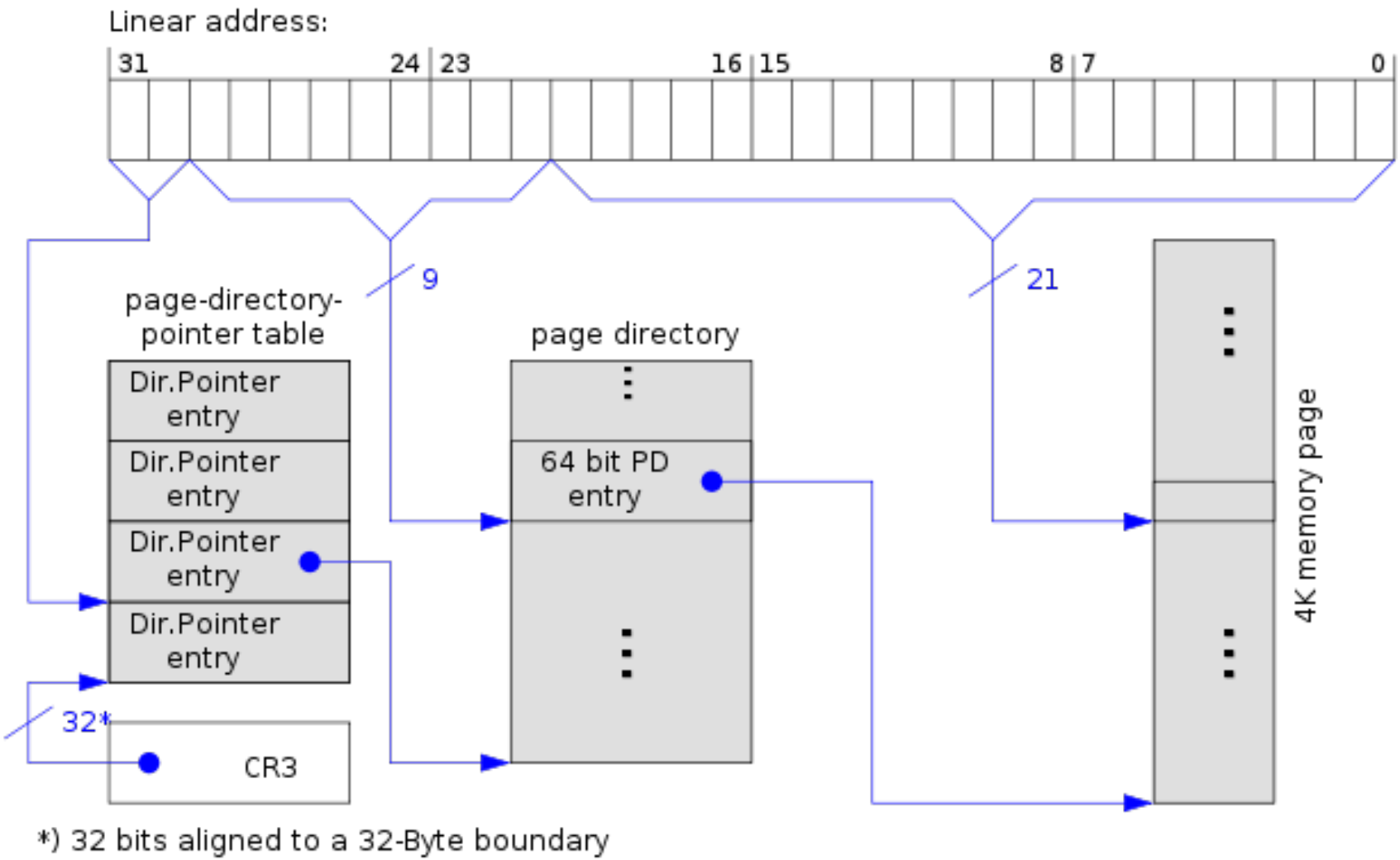
jak obsłużyć więcej pamięci bez 64 bitów ?

- PAE - poprawione stronicowanie, machanizm sprzętowy
- AWE (Address Windows Extensions) - mechanizm programowy
- mmap() - mapowanie plików do pamięci, mechanizm programowy

PAE - adresowanie stron o romiarze 4K



PAE - adresowanie stron o rozmiarze 2 MB

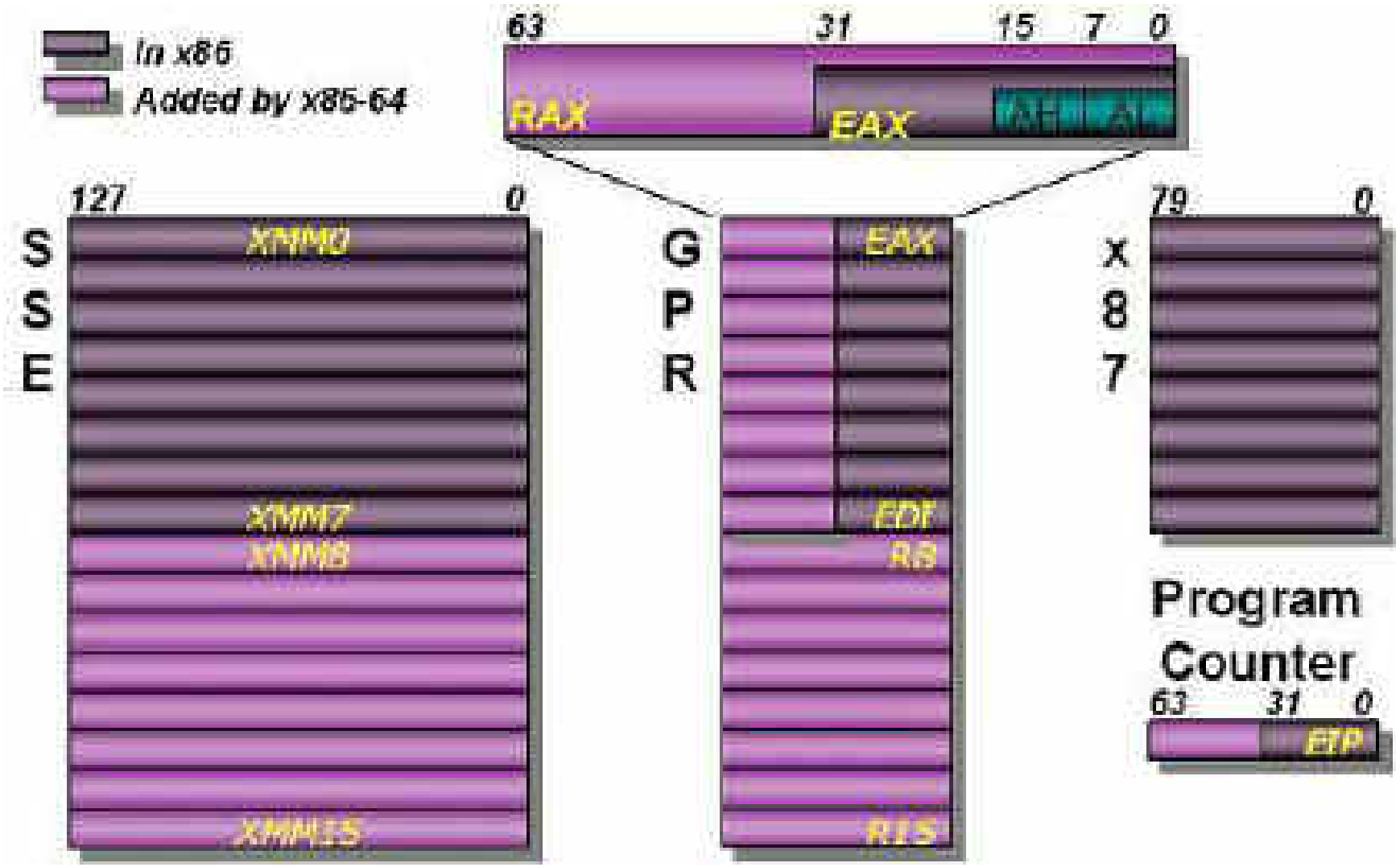


architektury 64-bitowe

- Itanium
- Sparc
- Power PC
- x86-64 (AMD64, Intel64) - CISC

archiektura x86-64

- rozszerzenie instrukcji do 64 bitów (zachowanie 32-bitowych odpowiedników)
  movl eax, ebx - wersja 32-bitowa
  movq rax, rbx - wersja 64-bitowa
- małe wyjątki np. stos - brak pushl
- tryb pracy long mode (tryb 64-bitowy, z możliwością przełączania dla aplikacji 32- i 16-bitowych)
- rozszerzenie eax,ebx, etc. do rax, rbx, etc. (kompatybilność wsteczna)
- nowe rejestry rxb, rxw, rxd, rx gdzie x jest z przedziału 8-15


In x86
Added by x86-64
63
31
15
7
0
RAX
EAX
127
0
SSE
XMM0
XMM7
XMM8
XMM15
GPR
EAX
EDI
R8
R15
x87
79
0
Program Counter
63
31
0
EIP

problemy z 64 bitami

- zmiana rozmiaru wskaźników (nie można konwertować wskaźników na inty)
- różne rozmiary pewnych stałych (np. maksymalna wartość zmiennej long)
- różne modele danych (Microsoft i VC++ inny model niż wszystkie Unixopodobne)

modele danych

Data model	short	int	long	long long	pointers
LP64	16	32	64	64	64
ILP64	16	64	64	64	64
SILP64	64	64	64	64	64
LLP64	16	32	32	64	64

WOW64

- Itanium - własny zestaw instrukcji - konieczność emulacji
- x86-64 - kompatybilność wsteczna - przełączanie trybu procesora (procesory z rejestrami 32- i 64-bitowymi)
- odrobinę inne zachowanie rejestru systemu windows w systemie 32 i 64-bitowym
- zastępowanie sterowników z \ WINDOWS \ SYSTEM32 tymi z \ WINDOWS \ SYSWOW64

Max OS X - 32-bitowe jądro z obsługą aplikacji 64-bitowych

- jądro - proces 32-bitowy (procesor 64-bitowy - zmiana trybu)
- programy 32-bitowe mogą tworzyć procesy 64-bitowe
- brak konieczności pisania 64-bitowych sterowników
- jądro tłumaczy zajmuje się kopiowaniem między 32- i 64-bitowymi przestrzeniami adresowymi

sterowniki

- w przeszłości głównie komputery biznesowe (brak zapotrzebowania na kamery, bluetooth, etc.)
- komunikacja bezpośrednia ze sprzętem (brak możliwości emulacji jak dla zwykłych aplikacji)
- problemy z adresowaniem adresów wyższych niż 4 GB w sterownikach 32-bitowych (np DMA)

sterowniki

- w przeszłości głównie komputery biznesowe (brak zapotrzebowania na kamery, bluetooth, etc.)
- komunikacja bezpośrednia ze sprzętem (brak możliwości emulacji jak dla zwykłych aplikacji)
- problemy z adresowaniem adresów wyższych niż 4 GB w sterownikach 32-bitowych (np DMA)

  uniemożliwienie adresowania powyżej 4 GB

  napisanie nowego sterownika

aplikacje 32-bitowe w linuksie 64-bitowym

- brak wielu pluginów do przeglądarek 64-bitowych (np. Flash), co zniechęca przeciętnego użytkownika
- konieczność kompilowania oprogramowania z źródeł, bo w sieciowych repozytoriach brak wersji 64-bitowych
- problem z oprogramowaniem o kodzie zamkniętym (wymagają przepisania przez producentów, a to kosztuje, zajmuje sporo czasu, a zapotrzebowanie jest niewielkie)

rozwiązania

- kompatybilność wsteczna (np AMD64) - wymuszanie na instalatorze używania pakietów z innej architektury

```
sudo dpkg -i --force-architecture packagename_i386.deb
```

- program linux32 oszukujący uruchamiany program co do architektury

```
hendrik@hendrik:~$ uname -a
Linux hendrik 2.6.15-25-amd64-k8 #1 SMP PREEMPT Wed Jun
11:39:18 UTC 2006 x86_64 GNU/Linux
hendrik@hendrik:~$ linux32 uname -a
Linux hendrik 2.6.15-25-amd64-k8 #1 SMP PREEMPT Wed Jun
11:39:18 UTC 2006 i686 GNU/Linux
```

Fragmenty kodu programu linux32

```
#ifdef STUPID_DEFAULT
#define DFL_PER PER_LINUX32_3GB
```

systemy 64-bitowe - przeciw

- przeciętny użytkownik, który nie ma więcej niz 4 GB RAM-u nie potrzebuje adresowania 64-bitowego

systemy 64-bitowe - przeciw

- przeciętny użytkownik, który nie ma więcej niz 4 GB RAM-u nie potrzebuje adresowania 64-bitowego
- istnieją mechanizmy umożliwiające obsługę większej ilości RAM-u bez 64 bitów

systemy 64-bitowe - przeciw

- przeciętny użytkownik, który nie ma więcej niz 4 GB RAM-u nie potrzebuje adresowania 64-bitowego
- istnieją mechanizmy umożliwiające obsługę większej ilości RAM-u bez 64 bitów
- w wielu zastosowaniach - oprogramowanie biurowe, surfowanie po sieci, słuchanie muzyki nie będzie poprawy

systemy 64-bitowe - przeciw

- przeciętny użytkownik, który nie ma więcej niz 4 GB RAM-u nie potrzebuje adresowania 64-bitowego
- istnieją mechanizmy umożliwiające obsługę większej ilości RAM-u bez 64 bitów
- w wielu zastosowaniach - oprogramowanie biurowe, surfowanie po sieci, słuchanie muzyki nie będzie poprawy
- część aplikacji działa wolniej (wirtualna maszyna Javy - brak szybszej wersji kompilatora)

systemy 64-bitowe - przeciw

- przeciętny użytkownik, który nie ma więcej niz 4 GB RAM-u nie potrzebuje adresowania 64-bitowego
- istnieją mechanizmy umożliwiające obsługę większej ilości RAM-u bez 64 bitów
- w wielu zastosowaniach - oprogramowanie biurowe, surfowanie po sieci, słuchanie muzyki nie będzie poprawy
- część aplikacji działa wolniej (wirtualna maszyna Javy - brak szybszej wersji kompilatora)
- spora część oprogramowania nie działa (długotrwałe zmiany, koszty kupna nowego)

systemy 64-bitowe - przeciw

- przeciętny użytkownik, który nie ma więcej niz 4 GB RAM-u nie potrzebuje adresowania 64-bitowego
- istnieją mechanizmy umożliwiające obsługę większej ilości RAM-u bez 64 bitów
- w wielu zastosowaniach - oprogramowanie biurowe, surfowanie po sieci, słuchanie muzyki nie będzie poprawy
- część aplikacji działa wolniej (wirtualna maszyna Javy - brak szybszej wersji kompilatora)
- spora część oprogramowania nie działa (długotrwałe zmiany, koszty kupna nowego)

Dodatkowe materiały

Notatki Łukasza Kidzińskiego o wpływie architektury komputera na system operacyjny