Wpływ architektury procesora na system operacyjny

Krzysztof Kąs, Paweł Bedyński, Krzysztof Niemkiewicz

Wydział Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego

07.11.2008

Plan Prezentacji

gdzie szukac informacji...

tu ...

cat /proc/cpuinfo

i tu...

/sys/devices/system/cpu/

i tutaj

linux/Documentation/cpu-freq/user-guide.txt

hot plug CPU

Documentation/hot-plug.txt

gdzie szukac informacji...

tu ...

cat /proc/cpuinfo

i tu...

/sys/devices/system/cpu/

i tutaj

linux/Documentation/cpu-freq/user-guide.txt

hot plug CPU

Documentation/hot-plug.txt

```
Processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 15
model name : Intel(R) Core(TM)2 Duo CPU     T5250  @ 1.50GHz
stepping : 13
cpu MHz : 1000.000
cache size : 2048 KB
physical id : 0
siblings : 2
core id : 0
cpu cores : 2
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 10
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
pge mca cmov  pat pse36 clflush dts acpi mmx
fxsr sse sse2 ss ht tm pbe nx lm constant_tsc
 arch_perfmon pebs bts pni monitor ds_cpl
est tm2 ssse3 cx16 xtpr lahf_lm
bogomips : 2995.62
clflush size : 64
```

gdzie szukac informacji...

tu ...

cat /proc/cpuinfo

i tu...

/sys/devices/system/cpu/

i tutaj

linux/Documentation/cpu-freq/user-guide.txt

hot plug CPU

Documentation/hot-plug.txt

```
processor : 1
vendor_id : GenuineIntel
cpu family : 6
model : 15
model name : Intel(R) Core(TM)2 Duo CPU     T5250  @ 1.50GHz
stepping : 13
cpu MHz : 1000.000
cache size : 2048 KB
physical id : 0
siblings : 2
core id : 1
cpu cores : 2
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 10
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
pge mca cmov pat pse36 clflush dts acpi mmx
fxsr sse sse2 ss ht tm pbe nx lm constant_tsc
 arch_perfmon pebs bts pni monitor ds_cpl
 est tm2 ssse3 cx16 xtpr lahf_lm
bogomips : 2992.63
clflush size : 64
```

gdzie szukac informacji...

tu ...

cat /proc/cpuinfo

i tu...

/sys/devices/system/cpu/

i tutaj

linux/Documentation/cpu-freq/user-guide.txt

hot plug CPU

Documentation/hot-plug.txt

gdzie szukac informacji...

tu ...

cat /proc/cpuinfo

i tu...

/sys/devices/system/cpu/

i tutaj

linux/Documentation/cpu-freq/user-guide.txt

hot plug CPU

Documentation/hot-plug.txt

```
/sys/devices/system/cpu$ ls
cpu0  cpu1  cpuidle  sched_mc_power_savings

/sys/devices/system/cpu/cpu0$ ls -R
.:
cache  cpufreq  cpuidle  crash_notes  topology
./cache:
index0  index1  index2
./cache/index0: (to samo w index1 i index 2)
coherency_line_size  physical_line_partition  type
level                shared_cpu_map           ways_of_associativity
number_of_sets       size
./cpufreq:
affected_cpus     ondemand                      scaling_driver    stats
cpuinfo_cur_freq  scaling_available_frequencies  scaling_governor
cpuinfo_max_freq  scaling_available_governors    scaling_max_freq
cpuinfo_min_freq  scaling_cur_freq               scaling_min_freq
./cpufreq/ondemand:
ignore_nice_load  sampling_rate      sampling_rate_min
powersave_bias    sampling_rate_max  up_threshold
./cpufreq/stats:
time_in_state  total_trans  trans_table
./cpuidle:
state0  state1  state2
./cpuidle/state0: (to samo w state 1 i state 2)
latency  name  power  time  usage
./topology:
core_id  core_siblings  physical_package_id  thread_siblings
```

gdzie szukac informacji...

tu ...

cat /proc/cpuinfo

i tu...

/sys/devices/system/cpu/

i tutaj

linux/Documentation/cpu-freq/user-guide.txt

hot plug CPU

Documentation/hot-plug.txt

gdzie szukac informacji...

tu ...

cat /proc/cpuinfo

i tu...

/sys/devices/system/cpu/

i tutaj

linux/Documentation/cpu-freq/user-guide.txt

hot plug CPU

Documentation/hot-plug.txt

```
1     CPU frequency and voltage scaling code in the Linux(TM) kernel
2
3
4                       L i n u x    C P U F r e q
5
6                           U S E R   G U I D E
7
8
9                   Dominik Brodowski  <linux@brodo.de>
10
11
12
13   Clock scaling allows you to change the clock speed of the CPUs on the
14    fly. This is a nice method to save battery power, because the lower
15            the clock speed, the less power the CPU consumes.

54 The following processors for the x86 architecture are supported by cpufreq:
55
56 AMD Elan - SC400, SC410
57 AMD mobile K6-2+
58 AMD mobile K6-3+
59 AMD mobile Duron
60 AMD mobile Athlon
61 AMD Opteron
62 AMD Athlon 64
63 Cyrix Media GXm
64 Intel mobile PIII and Intel mobile PIII-M on certain chipsets
65 Intel Pentium 4, Intel Xeon
66 Intel Pentium M (Centrino)
67 National Semiconductors Geode GX
68 Transmeta Crusoe
69 Transmeta Efficeon
70 VIA Cyrix 3 / C3
71 various processors on some ACPI 2.0-compatible systems [*]
```

gdzie szukac informacji...

tu ...

cat /proc/cpuinfo

i tu...

/sys/devices/system/cpu/

i tutaj

linux/Documentation/cpu-freq/user-guide.txt

hot plug CPU

Documentation/hot-plug.txt

gdzie szukac informacji...

tu ...

cat /proc/cpuinfo

i tu...

/sys/devices/system/cpu/

i tutaj

linux/Documentation/cpu-freq/user-guide.txt

hot plug CPU

Documentation/hot-plug.txt

```
1 CPU hotplug Support in Linux(tm) Kernel

40 General Stuff about CPU Hotplug
41 --------------------------------
42
43 Command Line Switches
44 ---------------------
45 maxcpus=n    Restrict boot time cpus to n. Say if you have 4 cpus, using
46              maxcpus=2 will only boot 2. You can choose to bring the
47              other cpus later online, read FAQ's for more info.
48
49 additional_cpus=n (*) Use this to limit hotpluggable cpus. This option sets
50    cpu_possible_map = cpu_present_map + additional_cpus
51
52 (*) Option valid only for following architectures
53 - x86_64, ia64

217 Q: What happens when a CPU is being logically offlined?
218 A: The following happen, listed in no particular order :-)
219
220 - A notification is sent to in-kernel registered modules by sending an event
221   CPU_DOWN_PREPARE or CPU_DOWN_PREPARE_FROZEN, depending on whether or not the
222   CPU is being offlined while tasks are frozen due to a suspend operation in
223   progress
224 - All processes are migrated away from this outgoing CPU to new CPUs.
225   The new CPU is chosen from each process' current cpuset, which may be
226   a subset of all online CPUs.
227 - All interrupts targeted to this CPU is migrated to a new CPU
228 - timers/bottom half/task lets are also migrated to a new CPU
229 - Once all services are migrated, kernel calls an arch specific routine
230   __cpu_disable() to perform arch specific cleanup.
231 - Once this is successful, an event for successful cleanup is sent by an event
232   CPU_DEAD (or CPU_DEAD_FROZEN if tasks are frozen due to a suspend while the
233   CPU is being offlined).
```

Instrukcje "specjalne" dla roznych architektur - Intel MMX (1)

MMX 1997r.

(MultiMedia eXtensions lub Matrix Math eXtensions) to zestaw 57 instrukcji SIMD dla procesorow Pentium i zgodnych. Rozkazy MMX moga realizowac dzialania logiczne i arytmetyczne na liczbach calkowitych. Pierwotnie wprowadzone w 1997 przez Intela dla procesorow Pentium MMX, aktualnie dostepne rowniez na procesory innych producentow.

- Rejestry MMX maja rozmiar 64 bitow, jest ich 8.

  Rejestry MMX w asemblerze nosza nazwy mm0, mm1,... mm7.
- Nowe typy danych -"packed"
  1. 8 - 8 bitow (packed byte),
  2. 4 - 16 bitow (packed word),
  3. 2 - 32 bity (packed dword),
  4. 1 - 64 bity (quad word).

Instrukcje "specjalne" dla roznych architektur - Intel MMX (1)

MMX 1997r.

(MultiMedia eXtensions lub Matrix Math eXtensions) to zestaw 57 instrukcji SIMD dla procesorow Pentium i zgodnych. Rozkazy MMX moga realizowac dzialania logiczne i arytmetyczne na liczbach calkowitych. Pierwotnie wprowadzone w 1997 przez Intela dla procesorow Pentium MMX, aktualnie dostepne rowniez na procesory innych producentow.

- Rejestry MMX maja rozmiar 64 bitow, jest ich 8.

  Rejestry MMX w asemblerze nosza nazwy mm0, mm1,... mm7.
- Nowe typy danych -"packed"
  1. 8 - 8 bitow (packed byte),
  2. 4 - 16 bitow (packed word),
  3. 2 - 32 bity (packed dword),
  4. 1 - 64 bity (quad word).

Instrukcje "specjalne" dla roznych architektur - Intel MMX (2)

PCMPEGB = P-CMP-EQ-B

mm1 = |01001000|01001100|10000100|00101110|11000100|01001110|11100000|00000001|
+
mm2 = |00001000|01001100|10000100|10101111|11000100|01000000|11110111|00000001|
=
mm2 = |00000000|11111111|11111111|00000000|11111111|00000000|00000000|11111111|

operacje:

- artytmetyczne (np. PADDB, PADDW, PADDD)
- porownania (np. PCMPEQB, PCMPEQW, PCMPEQD)
- logiczne (np. POR PAD)
- przesuniecia bitowe (np. PSLLW, PSLLD, PSLLQ)

Zastosowania:

- wyswietlanie grafiki trojwymiarowej: przekształcenia geometryczne, cieniowanie, teksturowanie;
- dekodowanie obrazow JPEG i PNG; filmow MPEG (transformacja DTC)

Instrukcje "specjalne" dla roznych architektur - Intel MMX (2)

PCMPEGB = P-CMP-EQ-B

mm1 = |01001000|01001100|10000100|00101110|11000100|01001110|11100000|00000001|
+
mm2 = |00001000|01001100|10000100|10101111|11000100|01000000|11110111|00000001|
=
mm2 = |00000000|11111111|11111111|00000000|11111111|00000000|00000000|11111111|

operacje:

- artytmetyczne (np. PADDB, PADDW, PADDD)
- porownania (np. PCMPEQB, PCMPEQW, PCMPEQD)
- logiczne (np. POR PAD)
- przesuniecia bitowe (np. PSLLW, PSLLD, PSLLQ)

Zastosowania:

- wyswietlanie grafiki trojwymiarowej: przekształcenia geometryczne, cieniowanie, teksturowanie,
- dekodowanie obrazow JPEG i PNG; filmow MPEG (transformacja DTC)

Instrukcje "specjalne" dla roznych architektur - Intel MMX (2)

PCMPEGB = P-CMP-EQ-B

mm1 = |01001000|01001100|10000100|00101110|11000100|01001110|11100000|00000001|
+
mm2 = |00001000|01001100|10000100|10101111|11000100|01000000|11110111|00000001|
=
mm2 = |00000000|11111111|11111111|00000000|11111111|00000000|00000000|11111111|

operacje:

- artytmetyczne (np. PADDB, PADDW, PADDD)
- porownania (np. PCMPEQB, PCMPEQW, PCMPEQD)
- logiczne (np. POR PAD)
- przesuniecia bitowe (np. PSLLW, PSLLD, PSLLQ)

Zastosowania:

- wyswietlanie grafiki trojwymiarowej: przeksztalcenia geometryczne, cieniowanie, teksturowanie;
- dekodowanie obrazow JPEG i PNG; filmow MPEG (transformacja DTC)

Instrukcje "specjalne" dla roznych architektur - SSE/SSE2/SSE3/SSSE3/SSE4

SSE - Streaming SIMD Extensions

SSE - jest nazwa zestawu instrukcji wprowadzonego w 1999 roku po raz pierwszy w procesorach Pentium III firmy Intel. http://sourceware.org/gdb/papers/linux/linux-sse.html

SSE2 - Streaming SIMD Extensions 2

Technologia ta zostala wprowadzona w procesorach rodziny Pentium 4 oraz Athlon 64. (20 XI 2000)

SSE3 - Streaming SIMD Extensions 3

oznaczany rowniez przez firme Intel jako Prescott New Instructions lub PNI (III 2001)

SSSE3 - Supplemental Streaming SIMD Extensions 3

Intel Core 2 , AMD Phenom (2006)

SSE4 - Streaming SIMD Extensions 4

Intel Core 2, Penryn (VI 2007)

Instrukcje "specjalne" dla roznych architektur - SSE/SSE2/SSE3/SSSE3/SSE4

SSE - Streaming SIMD Extensions

SSE - jest nazwa zestawu instrukcji wprowadzonego w 1999 roku po raz pierwszy w procesorach Pentium III firmy Intel. http://sourceware.org/gdb/papers/linux/linux-sse.html

SSE2 - Streaming SIMD Extensions 2

Technologia ta zostala wprowadzona w procesorach rodziny Pentium 4 oraz Athlon 64. (20 XI 2000)

SSE3 - Streaming SIMD Extensions 3

oznaczany rowniez przez firme Intel jako Prescott New Instructions lub PNI (III 2001)

SSSE3 - Supplemental Streaming SIMD Extensions 3

Intel Core 2 , AMD Phenom (2006)

SSE4 - Streaming SIMD Extensions 4

Intel Core 2, Penryn (VI 2007)

Instrukcje "specjalne" dla roznych architektur - SSE/SSE2/SSE3/SSSE3/SSE4

SSE - Streaming SIMD Extensions

SSE - jest nazwa zestawu instrukcji wprowadzonego w 1999 roku po raz pierwszy w procesorach Pentium III firmy Intel. http://sourceware.org/gdb/papers/linux/linux-sse.html

SSE2 - Streaming SIMD Extensions 2

Technologia ta zostala wprowadzona w procesorach rodziny Pentium 4 oraz Athlon 64. (20 XI 2000)

SSE3 - Streaming SIMD Extensions 3

oznaczany rowniez przez firme Intel jako Prescott New Instructions lub PNI (III 2001)

SSSE3 - Supplemental Streaming SIMD Extensions 3

Intel Core 2 , AMD Phenom (2006)

SSE4 - Streaming SIMD Extensions 4

Intel Core 2, Penryn (VI 2007)

Instrukcje "specjalne" dla roznych architektur - SSE

```
CMPNEQPS      xmm0, xmm1

|   1.0     |   -5.3    |   16.5    |    17.2   | xmm0
|   7.0     |   -5.3    |   16.5    |    17.3   | xmm1
     =           =           =           =
|111..1111|000..0000|000..0000|111..1111| xmm0
```

Typy danych - 4 elementowy wektor (128bit)

1. packed (rownolegle) : wykonujac rownoczesnie 4 niezalezne dzialania zmiennoprzecinkowe na odpowiadajacych sobie elementach wektorow;
2. scalar (skalarne) : wykonujac dzialanie tylko na pierwszych elementach wektorow.

Przesylanie liczb zmiennoprzecinkowych miedzy rejestrami i pamiecia

- MOVAPS, MOVUPS - Przeslanie 4 liczb zmiennoprzecinkowych pomiedzy rejestrem XMM, a pamiecia lub innym rejestrem XMM
- MOVLHPS, MOVHLPS - Przeslanie miedzy rejestrami 64-bitow (2 liczb zmiennoprzecinkowych).

Instrukcje "specjalne" dla roznych architektur - SSE

```
CMPNEQPS      xmm0, xmm1

|   1.0     |   -5.3    |   16.5    |    17.2   | xmm0

|   7.0     |   -5.3    |   16.5    |    17.3   | xmm1
     =           =           =           =
|111..1111|000..0000|000..0000|111..1111| xmm0
```

Typy danych - 4 elementowy wektor (128bit)

1. packed (rownolegle) : wykonujac rownoczesnie 4 niezalezne dzialania zmiennoprzecinkowe na odpowiadajacych sobie elementach wektorow;
2. scalar (skalarne) : wykonujac dzialanie tylko na pierwszych elementach wektorow.

Przesylanie liczb zmiennoprzecinkowych miedzy rejestrami i pamiecia

- MOVAPS, MOVUPS - Przeslanie 4 liczb zmiennoprzecinkowych pomiedzy rejestrem XMM, a pamiecia lub innym rejestrem XMM
- MOVLHPS, MOVHLPS - Przeslanie miedzy rejestrami 64-bitow (2 liczb zmiennoprzecinkowych).

Instrukcje "specjalne" dla roznych architektur - SSE

```
CMPNEQPS      xmm0, xmm1

|   1.0     |   -5.3    |   16.5    |    17.2   | xmm0
|   7.0     |   -5.3    |   16.5    |    17.3   | xmm1
     =           =           =           =
|111..1111|000..0000|000..0000|111..1111| xmm0
```

Typy danych - 4 elementowy wektor (128bit)

1. packed (rownolegle) : wykonujac rownoczesnie 4 niezalezne dzialania zmiennoprzecinkowe na odpowiadajacych sobie elementach wektorow;
2. scalar (skalarne) : wykonujac dzialanie tylko na pierwszych elementach wektorow.

Przesylanie liczb zmiennoprzecinkowych miedzy rejestrami i pamiecia

- MOVAPS, MOVUPS - Przeslanie 4 liczb zmiennoprzecinkowych pomiedzy rejestrem XMM, a pamiecia lub innym rejestrem XMM
- MOVLHPS, MOVHLPS - Przeslanie miedzy rejestrami 64-bitow (2 liczb zmiennoprzecinkowych).

Instrukcje "specjalne" dla roznych architektur - SSE (2)

- SSE
  Pamiec Podreczna:
  - Z = PREFETCH : poziomy 0, 1, 2 ; prefetchNTA (non-temporal)
  - BEZ = MOVNTQ (z rejestru MMX), MOVNTPS (z rejestru SSE)

- SSE3
  - MONITOR ustala początkowy adres zakresu pamięci (rozmiar tego obszaru jest stały, zależny od modelu procesora), który następnie jest automatycznie monitorowany przez procesor
  - MWAIT powoduje wejście w zoptymalizowaną sprzętową pętlę, która jest przerywana dopiero, gdy flaga zostanie ustawiona, a więc gdy nastąpi zapis pod obserwowany adres

Instrukcje "specjalne" dla roznych architektur - SSE (2)

- SSE
  Pamiec Podreczna:
  - Z = PREFETCH : poziomy 0, 1, 2 ; prefetchNTA (non-temporal)
  - BEZ = MOVNTQ (z rejestru MMX), MOVNTPS (z rejestru SSE)

- SSE3
  - MONITOR ustala początkowy adres zakresu pamięci (rozmiar tego obszaru jest stały, zależny od modelu procesora), który następnie jest automatycznie monitorowany przez procesor
  - MWAIT powoduje wejście w zoptymalizowaną sprzętową pętlę, która jest przerywana dopiero, gdy flaga zostanie ustawiona, a więc gdy nastąpi zapis pod obserwowany adres

Instrukcje "specjalne" dla roznych architektur - SSE (2)

- SSE
  Pamiec Podreczna:
  - Z = PREFETCH : poziomy 0, 1, 2 ; prefetchNTA (non-temporal)
  - BEZ = MOVNTQ (z rejestru MMX), MOVNTPS (z rejestru SSE)

- SSE3
  - MONITOR ustala poczatkowy adres zakresu pamieci (rozmiar tego obszaru jest staly, zalezny od modelu procesora), ktory nastepnie jest automatycznie monitorowany przez procesor
  - MWAIT powoduje wejscie w zoptymalizowana sprzetowa petle, ktora jest przerywana dopiero, gdy flaga zostanie ustawiona, a wiec gdy nastapi zapis pod obserwowany adres.

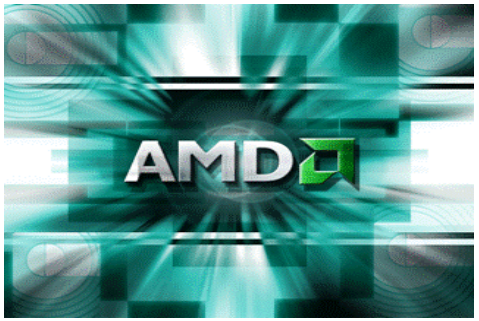
Instrukcje "specjalne" dla roznych architektur - SSE (2)

- SSE
  Pamiec Podreczna:
  - Z = PREFETCH : poziomy 0, 1, 2 ; prefetchNTA (non-temporal)
  - BEZ = MOVNTQ (z rejestru MMX), MOVNTPS (z rejestru SSE)

- SSE3
  - MONITOR ustala poczatkowy adres zakresu pamieci (rozmiar tego obszaru jest staly, zalezny od modelu procesora), ktory nastepnie jest automatycznie monitorowany przez procesor
  - MWAIT powoduje wejscie w zoptymalizowana sprzetowa petle, ktora jest przerywana dopiero, gdy flaga zostanie ustawiona, a wiec gdy nastapi zapis pod obserwowany adres.

Instrukcje "specjalne" dla roznych architektur - przyszlosc SSE5/AVX

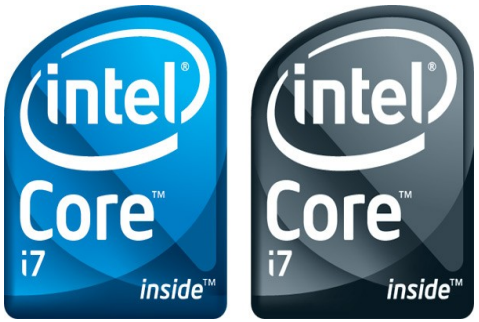
AMD - SSE5



- 2009 - planowane pojawienie sie pierwszego procesora
- 82 nowe rozkazy
- rozkazy 4 argumentowe (RISC)

$$w = y \pm (\pm x \cdot z)$$

INTEL - AVX



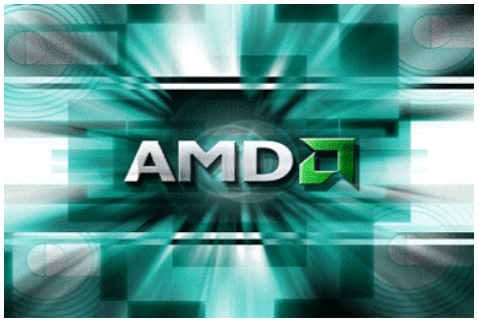
- 2010 - planowanie pojawienie sie pierwszego procesora
- ponad 250 nowych instrukcji
- 256 bitowe rejestry (ymm0 ... ymm15) (w dalszych wersjach 512 i 1024 bitowe)
- Sprzetowe wsparcie dla szyfrowania AES
  http://software.intel.com/en-us/articles/advanced-encryption-standard-aes-instructions-set

Instrukcje "specjalne" dla roznych architektur - przyszlosc SSE5/AVX

AMD - SSE5

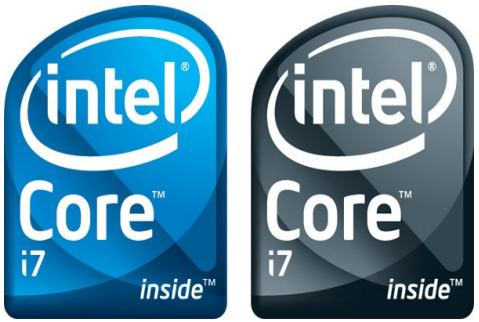
- 2009 - planowane pojawienie sie pierwszego procesora
- 82 nowe rozkazy
- rozkazy 4 argumentowe (RISC)

$$w = y \pm (\pm x \cdot z)$$



INTEL - AVX

- 2010 - planowanie pojawienie sie pierwszego procesora
- ponad 250 nowych instrukcji
- 256 bitowe rejestry (ymm0 . . . ymm15) (w dalszych wersjach 512 i 1024 bitowe)
- Sprzetowe wsparcie dla szyfrowania AES http://software.intel.com/en-us/articles/advanced-encryption-standard-aes-instructions-set



RISC vs CICS

CISC lata 60-te/70-te

Complex Instruction Set Computers

- duza liczba instrukcji 100 - ....
- mala liczba rejestrow - specjalizowane
- duza liczba trybow adresowania
- zlozone instrukcje wymagajace kilku-kilkunastu cykli zegara

RISC lata 80-te

Reduced Instruction Set Computers

- ograniczona liczba instrukcji
- duza liczba rejestrow - uniwersalne
- redukcja trybow adresowania
- wszystkie rozkazy wykonuja sie w jedym cyklu maszynowym (pipelining)

- Complex od zlozonosci poszczegolnych instrukcji a nie od ich ilosci (RISC czesto wykonuje wiecej instrukcji niz CISC)

RISC vs CICS

CISC lata 60-te/70-te

Complex Instruction Set Computers

- duza liczba instrukcji 100 - ....
- mala liczba rejestrow - specjalizowane
- duza liczba trybow adresowania
- zlozone instrukcje wymagajace kilku-kilkunastu cykli zegara

RISC lata 80-te

Reduced Instruction Set Computers

- ograniczona liczba instrukcji
- duza liczba rejestrow - uniwersalne
- redukcja trybow adresowania
- wszystkie rozkazy wykonuja sie w jedym cyklu maszynowym (pipelining)

- Complex od zlozonosci poszczegolnych instrukcji a nie od ich ilosci (RISC czesto wykonuje wiecej instrukcji niz CISC)

RISC vs CICS

CISC lata 60-te/70-te

Complex Instruction Set Computers

- duza liczba instrukcji 100 - ....
- mala liczba rejestrow - specjalizowane
- duza liczba trybow adresowania
- zlozone instrukcje wymagajace kilku-kilkunastu cykli zegara

RISC lata 80-te

Reduced Instruction Set Computers

- ograniczona liczba instrukcji
- duza liczba rejestrow - uniwersalne
- redukcja trybow adresowania
- wszystkie rozkazy wykonuja sie w jedym cyklu maszynowym (pipelining)

- Complex od zlozonosci poszczegolnych instrukcji a nie od ich ilosci (RISC czesto wykonuje wiecej instrukcji niz CISC)

RISC vs CICS

CISC lata 60-te/70-te

Complex Instruction Set Computers

- duza liczba instrukcji 100 - ....
- mala liczba rejestrow - specjalizowane
- duza liczba trybow adresowania
- zlozone instrukcje wymagajace kilku-kilkunastu cykli zegara

RISC lata 80-te

Reduced Instruction Set Computers

- ograniczona liczba instrukcji
- duza liczba rejestrow - uniwersalne
- redukcja trybow adresowania
- wszystkie rozkazy wykonuja sie w jedym cyklu maszynowym (pipelining)

- Complex od zlozonosci poszczegolnych instrukcji a nie od ich ilosci (RISC czesto wykonuje wiecej instrukcji niz CISC)

RISC vs CICS

CISC lata 60-te/70-te

Complex Instruction Set Computers

- duza liczba instrukcji 100 - ....
- mala liczba rejestrow - specjalizowane
- duza liczba trybow adresowania
- zlozone instrukcje wymagajace kilku-kilkunastu cykli zegara
  - Complex od zlozonosci poszczegolnych instrukcji a nie od ich ilosci (RISC czesto wykonuje wiecej instrukcji niz CISC)

RISC lata 80-te

Reduced Instruction Set Computers

- ograniczona liczba instrukcji
- duza liczba rejestrow - uniwersalne
- redukcja trybow adresowania
- wszystkie rozkazy wykonuja sie w jedym cyklu maszynowym (pipelining)

RISC vs CICS

CISC lata 60-te/70-te

Complex Instruction Set Computers

- duza liczba instrukcji 100 - ....
- mala liczba rejestrow - specjalizowane
- duza liczba trybow adresowania
- zlozone instrukcje wymagajace kilku-kilkunastu cykli zegara

RISC lata 80-te

Reduced Instruction Set Computers

- ograniczona liczba instrukcji
- duza liczba rejestrow - uniwersalne
- redukcja trybow adresowania
- wszystkie rozkazy wykonuja sie w jedym cyklu maszynowym (pipelining)

- Complex od zlozonosci poszczegolnych instrukcji a nie od ich ilosci (RISC czesto wykonuje wiecej instrukcji niz CISC)

RISC vs CICS problemy

CISC

Complex Instruction Set Computers

- duzo trybow odwolan do pamieci
- skomplikowane instrukcje
- duza liczba odwolan do pamieci (koszt czasu)

RISC

Reduced Instruction Set Computers

- ?
- ?
- ?

RISC vs CICS problemy

CISC

Complex Instruction Set Computers

- duzo trybow odwolan do pamieci
- skomplikowane instrukcje
- duza liczba odwolan do pamieci (koszt czasu)

RISC

Reduced Instruction Set Computers

- ?
- ?
- ?

RISC vs CICS problemy

CISC

Complex Instruction Set Computers

- duzo trybow odwolan do pamieci
- skomplikowane instrukcje
- duza liczba odwolan do pamieci (koszt czasu)

RISC

Reduced Instruction Set Computers

- ?
- ?
- ?

RISC vs CICS problemy

CISC

Complex Instruction Set Computers

- duzo trybow odwolan do pamieci
- skomplikowane instrukcje
- duza liczba odwolan do pamieci (koszt czasu)

RISC

Reduced Instruction Set Computers

- ?
- ?
- ?

RISC vs CICS problemy

CISC

Complex Instruction Set Computers

- duzo trybow odwolan do pamieci
- skomplikowane instrukcje
- duza liczba odwolan do pamieci (koszt czasu)

RISC

Reduced Instruction Set Computers

- ?
- ?
- ?

RISC vs CICS czym jest x86?

- do 80386 procesory rodziny x86 byly typowymi CISC-ami
- od 80486 wewnetrzne architektury coraz bardziej zaczynaja przypominac RISC (od zewnatrz paskudne CISC-y)

RISC vs CICS czym jest x86?

- do 80386 procesory rodziny x86 byly typowymi CISC-ami
- od 80486 wewnetrzne architektury coraz bardziej zaczynaja przypominac RISC (od zewnatrz paskudne CISC-y)

RISC vs CICS czym jest x86?

- do 80386 procesory rodziny x86 byly typowymi CISC-ami
- od 80486 wewnetrzne architektury coraz bardziej zaczynaja przypominac RISC (od zewnatrz paskudne CISC-y)

RISC

- krotki czas wykonywania rozkazu
- duza czestotliwosc zegara sugeruje ze operacje nie moga byc "obszerne"

RISC vs CICS czym jest x86?

- do 80386 procesory rodziny x86 byly typowymi CISC-ami
- od 80486 wewnetrzne architektury coraz bardziej zaczynaja przypominac RISC (od zewnatrz paskudne CISC-y)

RISC

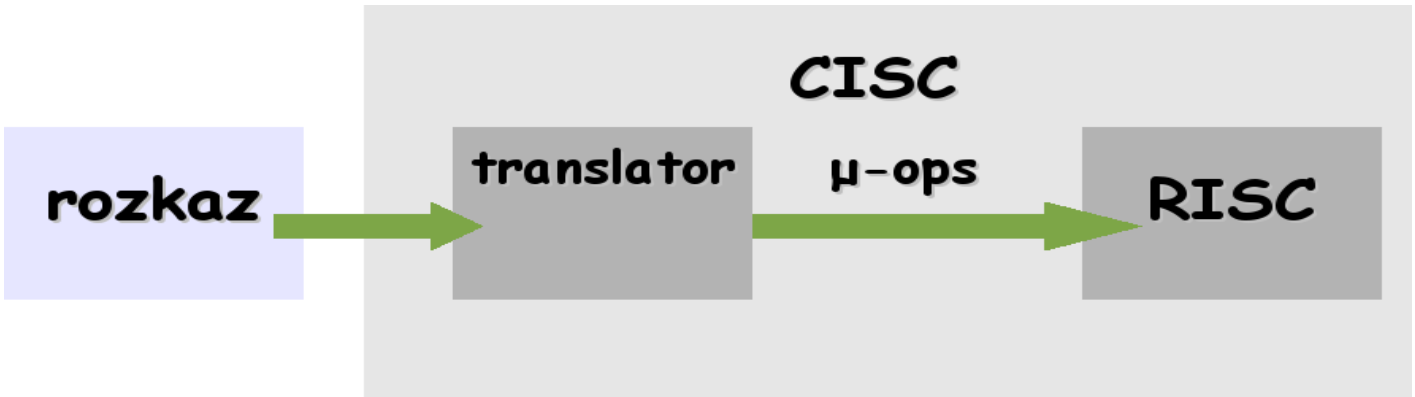
- krotki czas wykonywania rozkazu
- duza czestotliwosc zegara sugeruje ze operacje nie moga byc "obszerne"

CISC

- duza liczba instrukcji (bardzo duza)
- skomplikowane instrukcje ... i tak sa rozbijane na mniejsze

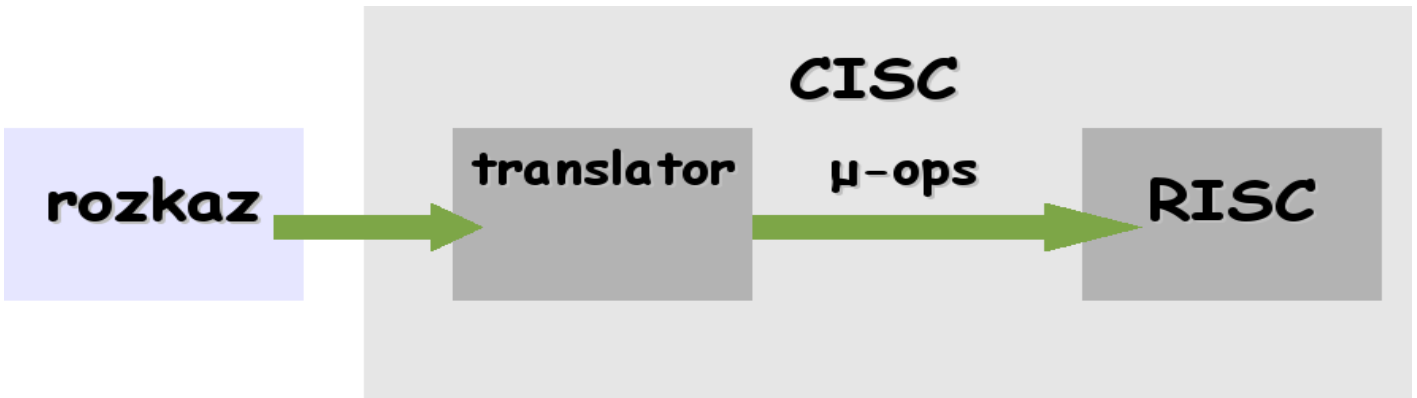
RISC vs CICS czym jest x86?

- do 80386 procesory rodziny x86 byly typowymi CISC-ami
- od 80486 wewnetrzne architektury coraz bardziej zaczynaja przypominac RISC (od zewnatrz paskudne CISC-y)



RISC vs CICS czym jest x86?

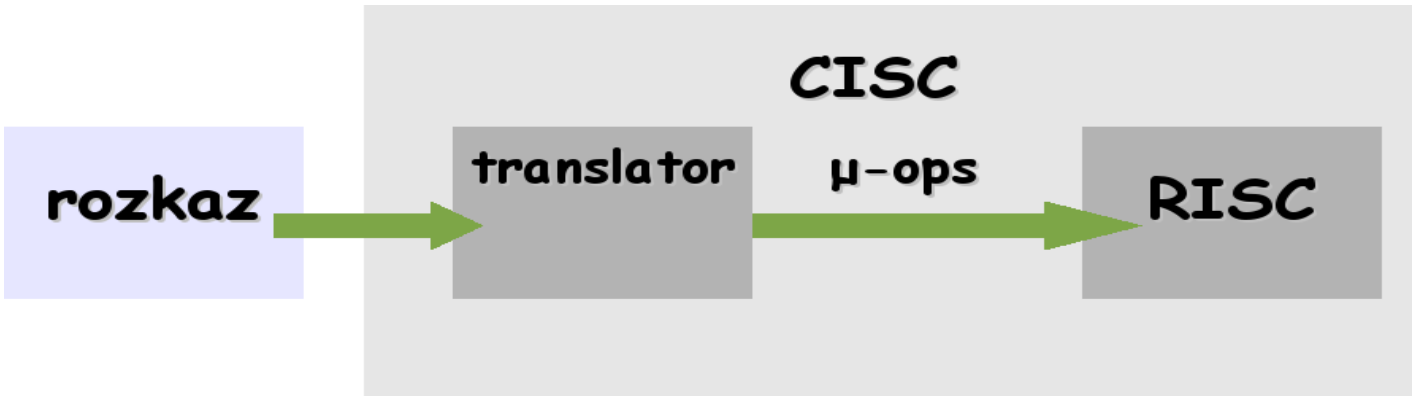
- do 80386 procesory rodziny x86 byly typowymi CISC-ami
- od 80486 wewnetrzne architektury coraz bardziej zaczynaja przypominac RISC (od zewnatrz paskudne CISC-y)



PostRISC?

RISC vs CICS czym jest x86?

- do 80386 procesory rodziny x86 byly typowymi CISC-ami
- od 80486 wewnetrzne architektury coraz bardziej zaczynaja przypominac RISC (od zewnatrz paskudne CISC-y)



PostRISC?

http://community.tomshardware.pl/comments.php?DiscussionID=135563

Pipelining - co to ?

etapy przetwarzania rozkazow:

1. pobranie rozkazu z pamieci operacyjnej
2. dekodowanie rozkazu, pobieranie danych z rejestrow
3. wykonanie instrukcji, lub obliczenie adresu
4. dostep do rejestru
5. zapis wyniku

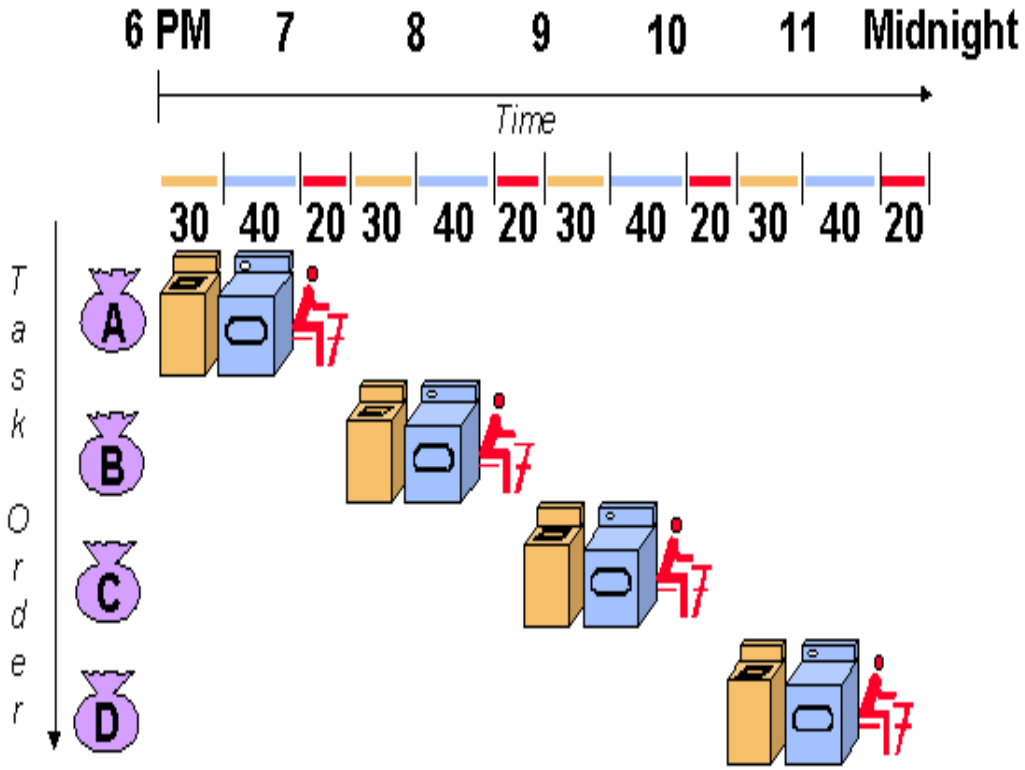
architektury RISC maja przewage poniewaz ich instrukcje sa male

Pipelining - co to ?

etapy przetwarzania rozkazow:

1. pobranie rozkazu z pamieci operacyjnej
2. dekodowanie rozkazu, pobieranie danych z rejestrow
3. wykonanie instrukcji, lub obliczenie adresu
4. dostep do rejestru
5. zapis wyniku

architektury RISC maja przewage poniewaz ich instrukcje sa male

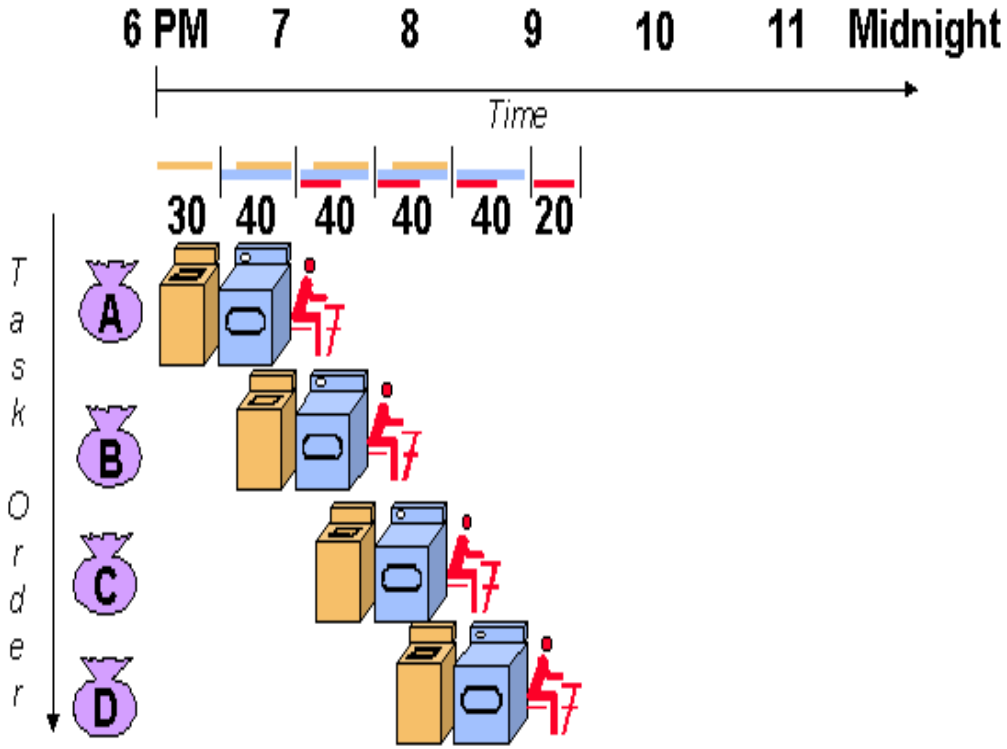


Pipelining - co to ?

etapy przetwarzania rozkazow:

1. pobranie rozkazu z pamieci operacyjnej
2. dekodowanie rozkazu, pobieranie danych z rejestrow
3. wykonanie instrukcji, lub obliczenie adresu
4. dostep do rejestru
5. zapis wyniku

architektury RISC maja przewage poniewaz ich instrukcje sa male



Pipelining - problemy

problem1

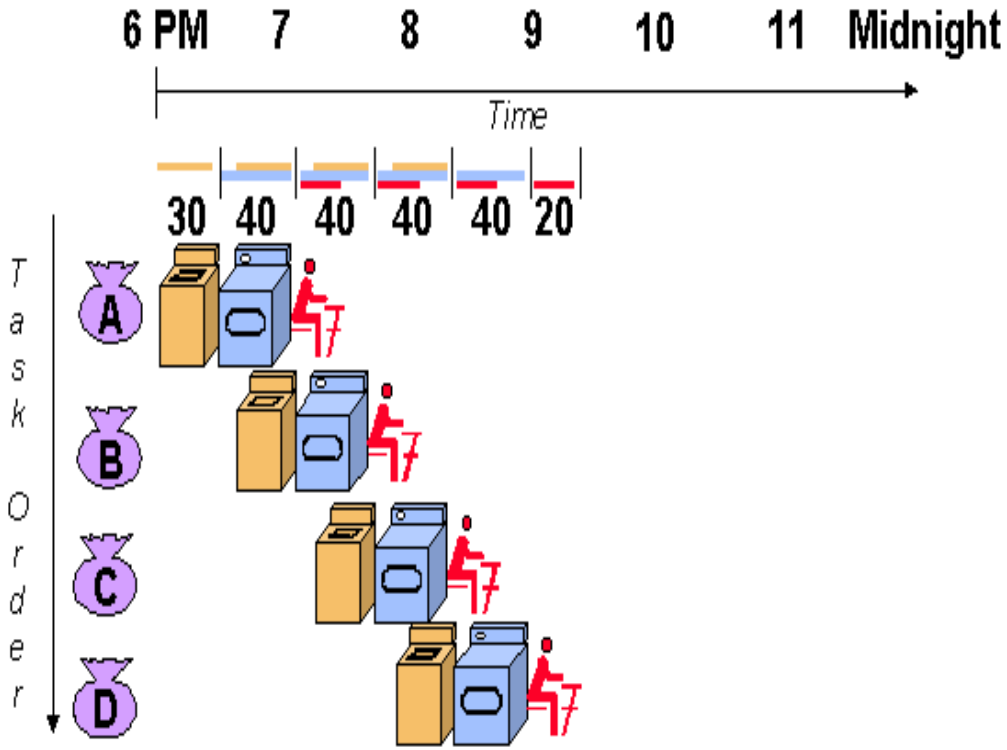
```
add $r3, $r2, $r1
add $r5, $r4, $r3

/* dodaj $r2 do $r1 i
wynik umiesc w $r3 */
```

problem2

```
Loop :
      add $r3, $r2, $r1
      sub $r6, $r5, $r4
      beq $r3, $r6, Loop
//dalsze instrukcje

/* beq to skok do Loop:
jesli $r3==$r6 */
```



Pipelining - problemy

problem1

```
add $r3, $r2, $r1
add $r5, $r4, $r3

/* dodaj $r2 do $r1 i
wynik umiesc w $r3 */
```

problem2

```
Loop :
      add $r3, $r2, $r1
      sub $r6, $r5, $r4
      beq $r3, $r6, Loop
//dalsze instrukcje

/* beq to skok do Loop:
jesli $r3==$r6 */
```

Rozwiazania:

- Zamiana kolejnosci
  (w przykladzie 1 wepchanie innych NIEZALEZNYCH instrukcji moze uratowac sprawe)
  reorganizacja kolejnosci jest zadaniem kompilatora, ma on za zadanie zminimalizowac liczbe przestojow
- Przewidywanie wyniku
  na ogol petle piszemy po to by wykonywaly sie wiecej niz raz
  procesor 'zgaduje' prawdopodobna sciezke i wykonuje ja w przod
- Przewidywanie dynamiczne
  pamietanie historii wyborow przy kazdym rozgalezieniu
  nawet 90% skutecznosc trafienia - algorytm Yeh'a
- Superskalarnosc
  osobne potoki - powielone czesci sprzetu (ew specjalizacja w typach obliczen)

Pipelining - problemy

problem1

```
add $r3, $r2, $r1
add $r5, $r4, $r3

/* dodaj $r2 do $r1 i
wynik umiesc w $r3 */
```

problem2

```
Loop :
      add $r3, $r2, $r1
      sub $r6, $r5, $r4
      beq $r3, $r6, Loop
//dalsze instrukcje

/* beq to skok do Loop:
jesli $r3==$r6 */
```

Rozwiazania:

- Zamiana kolejnosci
  (w przykladzie 1 wepchanie innych NIEZALEZNYCH instrukcji moze uratowac sprawe)
  reorganizacja kolejnosci jest zadaniem kompilatora, ma on za zadanie zminimalizowac liczbe przestojow
- Przewidywanie wyniku
  na ogol petle piszemy po to by wykonywaly sie wiecej niz raz
  procesor 'zgaduje' prawdopodobna sciezke i wykonuje ja w przod
- Przewidywanie dynamiczne
  pamietanie historii wyborow przy kazdym rozgalezieniu
  nawet 90% skutecznosc trafienia - algorytm Yeh'a
- Superskalarnosc
  osobne potoki - powielone czesci sprzetu (ew specjalizacja w typach obliczen)

Pipelining - problemy

problem1

```
add $r3, $r2, $r1
add $r5, $r4, $r3

/* dodaj $r2 do $r1 i
wynik umiesc w $r3 */
```

problem2

```
Loop :
      add $r3, $r2, $r1
      sub $r6, $r5, $r4
      beq $r3, $r6, Loop
//dalsze instrukcje

/* beq to skok do Loop:
jesli $r3==$r6 */
```

Rozwiazania:

- Zamiana kolejnosci
  (w przykladzie 1 wepchanie innych NIEZALEZNYCH instrukcji moze uratowac sprawe)
  reorganizacja kolejnosci jest zadaniem kompilatora, ma on za zadanie zminimalizowac liczbe przestojow
- Przewidywanie wyniku
  na ogol petle piszemy po to by wykonywaly sie wiecej niz raz
  procesor 'zgaduje' prawdopodobna sciezke i wykonuje ja w przod
- Przewidywanie dynamiczne
  pamietanie historii wyborow przy kazdym rozgalezieniu
  nawet 90% skutecznosc trafienia - algorytm Yeh'a
- Superskalarnosc
  osobne potoki - powielone czesci sprzetu (ew specjalizacja w typach obliczen)

Pipelining - problemy

problem1

```
add $r3, $r2, $r1
add $r5, $r4, $r3

/* dodaj $r2 do $r1 i
wynik umiesc w $r3 */
```

problem2

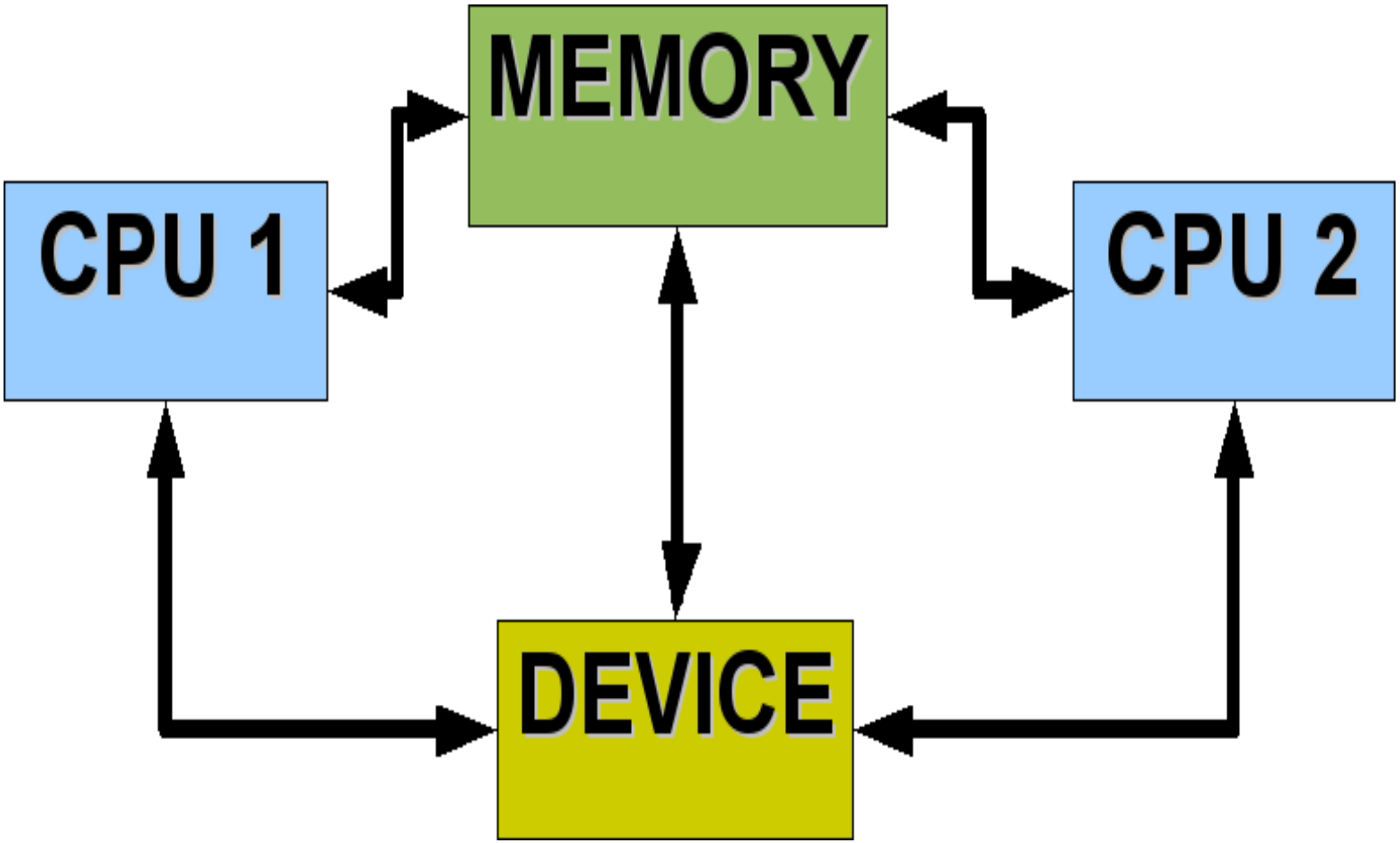
```
Loop :
      add $r3, $r2, $r1
      sub $r6, $r5, $r4
      beq $r3, $r6, Loop
//dalsze instrukcje

/* beq to skok do Loop:
jesli $r3==$r6 */
```

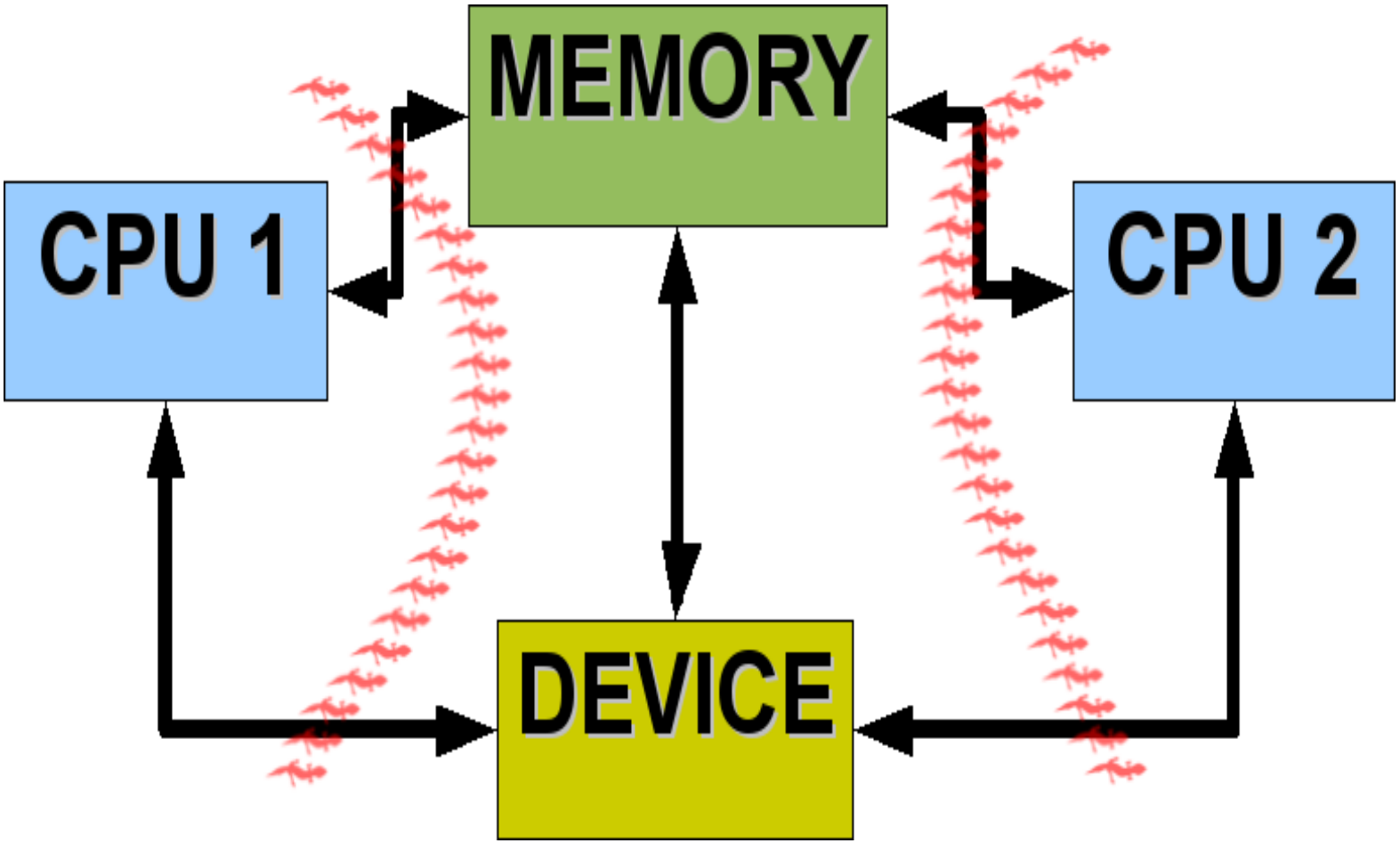
Rozwiazania:

- Zamiana kolejnosci
  (w przykladzie 1 wepchanie innych NIEZALEZNYCH instrukcji moze uratowac sprawe)
  reorganizacja kolejnosci jest zadaniem kompilatora, ma on za zadanie zminimalizowac liczbe przestojow
- Przewidywanie wyniku
  na ogol petle piszemy po to by wykonywaly sie wiecej niz raz
  procesor 'zgaduje' prawdopodobna sciezke i wykonuje ja w przod
- Przewidywanie dynamiczne
  pamietanie historii wyborow przy kazdym rozgalezieniu
  nawet 90% skutecznosc trafienia - algorytm Yeh'a
- Superskalarnosc
  osobne potoki - powielone czesci sprzetu (ew specjalizacja w typach obliczen)

model



model



po co? (1)

`A==1; B==2`

CPU 1

- A=3
- B=4

CPU 2

- x=A
- y=B

```
STORE A=3,      STORE B=4,      x=LOAD A->3,    y=LOAD B->4
STORE A=3,      STORE B=4,      y=LOAD B->4,    x=LOAD A->3
STORE A=3,      x=LOAD A->3,    STORE B=4,      y=LOAD B->4
STORE A=3,      x=LOAD A->3,    y=LOAD B->2,    STORE B=4
STORE A=3,      y=LOAD B->2,    STORE B=4,      x=LOAD A->3
STORE A=3,      y=LOAD B->2,    x=LOAD A->3,    STORE B=4
STORE B=4,      STORE A=3,      x=LOAD A->3,    y=LOAD B->4
STORE B=4, ...
...
```

razem 24 przeploty, a mozliwe wyniki ...

po co? (1)

`A==1; B==2`

CPU 1

- A=3
- B=4

```
STORE A=3,      STORE B=4,
STORE A=3,      STORE B=4,
STORE A=3,      x=LOAD A->3,
STORE A=3,      x=LOAD A->3,
STORE A=3,      y=LOAD B->2,
STORE A=3,      y=LOAD B->2,
STORE B=4,      STORE A=3,
STORE B=4, ...
...
```

CPU 2

- x=A
- y=B

```
x=LOAD A->3,    y=LOAD B->4
y=LOAD B->4,    x=LOAD A->3
STORE B=4,      y=LOAD B->4
y=LOAD B->2,    STORE B=4
STORE B=4,      x=LOAD A->3
x=LOAD A->3,    STORE B=4
x=LOAD A->3,    y=LOAD B->4
```

razem 24 przeploty, a mozliwe wyniki ...

po co? (2)

```
A == 1, B == 2, C = 3, P == &A, Q == &C
```

CPU 1

- B=4
- P=&B

CPU 2

- Q=P
- D=*Q

```
(Q == &A) and (D == 1)
(Q == &B) and (D == 2)
(Q == &B) and (D == 4)
```

UWAGA: CPU2 nigdy nie zaladuje C do D poniewaz CPU zaladuje P do Q zanim rozpocznie ladowanie *Q

po co? (2)

```
A == 1, B == 2, C = 3, P == &A, Q == &C
```

CPU 1

- B=4
- P=&B

CPU 2

- Q=P
- D=*Q

```
(Q == &A) and (D == 1)
(Q == &B) and (D == 2)
(Q == &B) and (D == 4)
```

UWAGA: CPU2 nigdy nie zaladuje C do D poniewaz CPU zaladuje P do Q zanim rozpocznie ladowanie *Q

co mamy gwarantowane...

Z perspektywy pojedynczego procesora jego instrukcje 'zalezne' wykonuja sie szeregowo

CPU 1

- Q=P ; D=*Q;
- Q = LOAD P, D = LOAD *Q

'zachodzace' instrukcje zapisu/odczytu ('wewnatrz' jednego procesora) beda szeregowane

CPU 1

- a = *X; *X = b;
- a = LOAD *X, STORE *X = b

CPU 1

- *X = c; d = *X;
- STORE *X = c, d = LOAD *X

co mamy gwarantowane...

Z perspektywy pojedynczego procesora jego instrukcje 'zalezne' wykonuja sie szeregowo

CPU 1

- Q=P ; D=*Q;
- Q = LOAD P, D = LOAD *Q

'zachodzace' instrukcje zapisu/odczytu ('wewnatrz' jednego procesora) beda szeregowane

CPU 1

- a = *X; *X = b;
- a = LOAD *X, STORE *X = b

CPU 1

- *X = c; d = *X;
- STORE *X = c, d = LOAD *X

co mamy gwarantowane...

Z perspektywy pojedynczego procesora jego instrukcje 'zalezne' wykonuja sie szeregowo

CPU 1

- Q=P ; D=*Q;
- Q = LOAD P, D = LOAD *Q

'zachodzace' instrukcje zapisu/odczytu ('wewnatrz' jednego procesora) beda szeregowane

CPU 1

- a = *X; *X = b;
- a = LOAD *X, STORE *X = b

CPU 1

- *X = c; d = *X;
- STORE *X = c, d = LOAD *X

Co musimy, a czego nie mozemy zakladac..

Nie mozemy zakladac, ze niezalezne odczyty/zapisy wykonuja sie szeregowo
tzn. dla X = *A; Y = *B; *D = Z; mozemy miec:

```
X = LOAD *A,  Y = LOAD *B,  STORE *D = Z
X = LOAD *A,  STORE *D = Z, Y = LOAD *B
Y = LOAD *B,  X = LOAD *A,  STORE *D = Z
Y = LOAD *B,  STORE *D = Z, X = LOAD *A
STORE *D = Z, X = LOAD *A,  Y = LOAD *B
STORE *D = Z, Y = LOAD *B,  X = LOAD *A
```

Musimy zakladac, ze zalezne (overlapping) zapisy/odczyty moga byc mergowane lub pomijane
tzn. X = *A; Y = *(A + 4); mozemy miec:

```
X = LOAD *A; Y = LOAD *(A + 4);
Y = LOAD *(A + 4); X = LOAD *A;
{X, Y} = LOAD {*A, *(A + 4) };
```

a dla *A = X; *A = Y; mozemy miec:

```
STORE *A = X; Y = LOAD *A;
STORE *A = Y = X;
```

Co musimy, a czego nie mozemy zakladac..

Nie mozemy zakladac, ze niezalezne odczyty/zapisy wykonuja sie szeregowo
tzn. dla X = *A; Y = *B; *D = Z; mozemy miec:

```
X = LOAD *A,  Y = LOAD *B,  STORE *D = Z
X = LOAD *A,  STORE *D = Z, Y = LOAD *B
Y = LOAD *B,  X = LOAD *A,  STORE *D = Z
Y = LOAD *B,  STORE *D = Z, X = LOAD *A
STORE *D = Z, X = LOAD *A,  Y = LOAD *B
STORE *D = Z, Y = LOAD *B,  X = LOAD *A
```

Musimy zakladac, ze zalezne (overlapping) zapisy/odczyty moga byc mergowane lub pomijane
tzn. X = *A; Y = *(A + 4); mozemy miec:

```
X = LOAD *A; Y = LOAD *(A + 4);
Y = LOAD *(A + 4); X = LOAD *A;
{X, Y} = LOAD {*A, *(A + 4) };
```

a dla *A = X; *A = Y; mozemy miec:

```
STORE *A = X; Y = LOAD *A;
STORE *A = Y = X;
```

Co musimy, a czego nie mozemy zakladac..

Nie mozemy zakladac, ze niezalezne odczyty/zapisy wykonuja sie szeregowo
tzn. dla X = *A; Y = *B; *D = Z; mozemy miec:

```
X = LOAD *A,  Y = LOAD *B,  STORE *D = Z
X = LOAD *A,  STORE *D = Z, Y = LOAD *B
Y = LOAD *B,  X = LOAD *A,  STORE *D = Z
Y = LOAD *B,  STORE *D = Z, X = LOAD *A
STORE *D = Z, X = LOAD *A,  Y = LOAD *B
STORE *D = Z, Y = LOAD *B,  X = LOAD *A
```

Musimy zakladac, ze zalezne (overlapping) zapisy/odczyty moga byc mergowane lub pomijane
tzn. X = *A; Y = *(A + 4); mozemy miec:

```
X = LOAD *A; Y = LOAD *(A + 4);
Y = LOAD *(A + 4); X = LOAD *A;
{X, Y} = LOAD {*A, *(A + 4) };
```

a dla *A = X; *A = Y; mozemy miec:

```
STORE *A = X; Y = LOAD *A;
STORE *A = Y = X;
```

Co to jest MEMORY BARRIER

Niezalezne operacje na pamieci moga byc wykonywane efektywnie w losowej kolejnosci ale moze to stwarzac problemy (np CPU-CPU lub I/O). Czasem jednak wymagane jest by kompilator i CPU byly poinformowane o pewnych restrykcjach.

MEMORY BARRIER

Memory barriers nakladaja czesciowy porzadek na operacjach po dwoch stronach bariery. Powoduja ze sekwencja zdarzen na pamieci staje sie dla innych czesci systemu imitacja sytuacji, w ktorej to CPU wykonuje te operacje w takiej, a nie innej kolejnosci

Co to jest MEMORY BARRIER

Niezalezne operacje na pamieci moga byc wykonywane efektywnie w losowej kolejnosci ale moze to stwarzac problemy (np CPU-CPU lub I/O). Czasem jednak wymagane jest by kompilator i CPU byly poinformowane o pewnych restrykcjach.

MEMORY BARRIER

Memory barriers nakladaja czesciowy porzadek na operacjach po dwoch stronach bariery. Powoduja ze sekwencja zdarzen na pamieci staje sie dla innych czesci systemu imitacja sytuacji, w ktorej to CPU wykonuje te operacje w takiej, a nie innej kolejnosci

Co to jest MEMORY BARRIER

Niezalezne operacje na pamieci moga byc wykonywane efektywnie w losowej kolejnosci ale moze to stwarzac problemy (np CPU-CPU lub I/O). Czasem jednak wymagane jest by kompilator i CPU byly poinformowane o pewnych restrykcjach.

MEMORY BARRIER

Memory barriers nakladaja czesciowy porzadek na operacjach po dwoch stronach bariery. Powoduja ze sekwencja zdarzen na pamieci staje sie dla innych czesci systemu imitacja sytuacji, w ktorej to CPU wykonuje te operacje w takiej, a nie innej kolejnosci

Bogactwo barier ...

1. WMB - write (STORE) memory barrier
2. data dependency barriers
3. RMB - read (LOAD) memory barrier
4. general memory barrier
5. LOCK / UNLOCK

Czy wszedzie bariery sa potrzebne ?

Jak sie maja te gwarancje do roznych architektur?
linux/arch/alpha/include/asm/barrier.h

Bogactwo barier ...

1. WMB - write (STORE) memory barrier
2. data dependency barriers
3. RMB - read (LOAD) memory barrier
4. general memory barrier
5. LOCK / UNLOCK

Czy wszedzie bariery sa potrzebne ?

Jak sie maja te gwarancje do roznych architektur?
linux/arch/alpha/include/asm/barrier.h

Bogactwo barier ...

1. WMB - write (STORE) memory barrier
2. data dependency barriers
3. RMB - read (LOAD) memory barrier
4. general memory barrier
5. LOCK / UNLOCK

Czy wszedzie bariery sa potrzebne ?

Jak sie maja te gwarancje do roznych architektur?
linux/arch/alpha/include/asm/barrier.h

Bogactwo barier ...

1. WMB - write (STORE) memory barrier
2. data dependency barriers
3. RMB - read (LOAD) memory barrier
4. general memory barrier
5. LOCK / UNLOCK

Czy wszedzie bariery sa potrzebne ?

Jak sie maja te gwarancje do roznych architektur?
linux/arch/alpha/include/asm/barrier.h

Bogactwo barier ...

1. WMB - write (STORE) memory barrier
2. data dependency barriers
3. RMB - read (LOAD) memory barrier
4. general memory barrier
5. LOCK / UNLOCK

Czy wszedzie bariery sa potrzebne ?

Jak sie maja te gwarancje do roznych architektur?
linux/arch/alpha/include/asm/barrier.h

Bogactwo barier ...

1. WMB - write (STORE) memory barrier
2. data dependency barriers
3. RMB - read (LOAD) memory barrier
4. general memory barrier
5. LOCK / UNLOCK

Czy wszedzie bariery sa potrzebne ?

Jak sie maja te gwarancje do roznych architektur?
linux/arch/alpha/include/asm/barrier.h

Bogactwo barier ...

1. WMB - write (STORE) memory barrier
2. data dependency barriers
3. RMB - read (LOAD) memory barrier
4. general memory barrier
5. LOCK / UNLOCK

Czy wszedzie bariery sa potrzebne ?

Jak sie maja te gwarancje do roznych architektur?
linux/arch/alpha/include/asm/barrier.h

Czego nie mozemy zalozyc o MEMORY BARRIER w LINUX KERNEL

NIE mozemy zalozyc ze

1. operacja dostepu do pamieci wykonana przed zalozeniem bariery bedzie ukonczona w chwili ukonczenia operacji bariery (bariera moze byc postrzegana jako linia ktorej niektore instrukcje nie moga przekraczac)
2. bariera na jednym CPU ma bezposredni wplyw na drugi CPU lub inne urzadzenia (co z posrednim?)
3. jeden CPU bedzie widzial w prawidlowej kolejnosci efekty dostepu do pamieci innego CPU nawet jesli ten drugi uzywa MEMORY BARRIER (odsylam do SMP BARRIER PAIRING)
4. inny hardware nie zmieni kolejnosci wykonywania instrukcji dostepu do pamieci

Czego nie mozemy zalozyc o MEMORY BARRIER w LINUX KERNEL

NIE mozemy zalozyc ze

1. operacja dostepu do pamieci wykonana przed zalozeniem bariery bedzie ukonczona w chwili ukonczenia operacji bariery (bariera moze byc postrzegana jako linia ktorej niektore instrukcje nie moga przekraczac)
2. bariera na jednym CPU ma bezposredni wplyw na drugi CPU lub inne urzadzenia (co z posrednim?)
3. jeden CPU bedzie widzial w prawidlowej kolejnosci efekty dostepu do pamieci innego CPU nawet jesli ten drugi uzywa MEMORY BARRIER (odsylam do SMP BARRIER PAIRING)
4. inny hardware nie zmieni kolejnosci wykonywania instrukcji dostepu do pamieci

Czego nie mozemy zalozyc o MEMORY BARRIER w LINUX KERNEL

NIE mozemy zalozyc ze

1. operacja dostepu do pamieci wykonana przed zalozeniem bariery bedzie ukonczona w chwili ukonczenia operacji bariery (bariera moze byc postrzegana jako linia ktorej niektore instrukcje nie moga przekraczac)
2. bariera na jednym CPU ma bezposredni wplyw na drugi CPU lub inne urzadzenia (co z posrednim?)
3. jeden CPU bedzie widzial w prawidlowej kolejnosci efekty dostepu do pamieci innego CPU nawet jesli ten drugi uzywa MEMORY BARRIER (odsylam do SMP BARRIER PAIRING)
4. inny hardware nie zmieni kolejnosci wykonywania instrukcji dostepu do pamieci

Czego nie mozemy zalozyc o MEMORY BARRIER w LINUX KERNEL

NIE mozemy zalozyc ze

1. operacja dostepu do pamieci wykonana przed zalozeniem bariery bedzie ukonczona w chwili ukonczenia operacji bariery (bariera moze byc postrzegana jako linia ktorej niektore instrukcje nie moga przekraczac)
2. bariera na jednym CPU ma bezposredni wplyw na drugi CPU lub inne urzadzenia (co z posrednim?)
3. jeden CPU bedzie widzial w prawidlowej kolejnosci efekty dostepu do pamieci innego CPU nawet jesli ten drugi uzywa MEMORY BARRIER (odsylam do SMP BARRIER PAIRING)
4. inny hardware nie zmieni kolejnosci wykonywania instrukcji dostepu do pamieci

Czego nie mozemy zalozyc o MEMORY BARRIER w LINUX KERNEL

NIE mozemy zalozyc ze

1. operacja dostepu do pamieci wykonana przed zalozeniem bariery bedzie ukonczona w chwili ukonczenia operacji bariery (bariera moze byc postrzegana jako linia ktorej niektore instrukcje nie moga przekraczac)
2. bariera na jednym CPU ma bezposredni wplyw na drugi CPU lub inne urzadzenia (co z posrednim?)
3. jeden CPU bedzie widzial w prawidlowej kolejnosci efekty dostepu do pamieci innego CPU nawet jesli ten drugi uzywa MEMORY BARRIER (odsylam do SMP BARRIER PAIRING)
4. inny hardware nie zmieni kolejnosci wykonywania instrukcji dostepu do pamieci

anomalie

Rozpatrzmy uklad:

```
CPU 1              CPU 2
================== ==================
{ A == 1, B == 2, C = 3, P == &A, Q == &C }
B = 4;
<write barrier>
P = &B
                   Q = P;
                   D = *Q;
```

W oczywisty sposob wystepuje tu data dependency i wydaje sie ze na koncy Q musi byc albo &A lub &B oraz ze

```
(Q == &A) implies (D == 1)
(Q == &B) implies (D == 4)
```

Jednak wiedza CPU2 o P moze byc updateowana PRZED wiedza o B, co prowadzi do nastepujacego przeplotu

```
(Q == &B) and (D == 2) ????
```

architektura DEC Alpha

anomalie

Rozpatrzmy uklad:

```
CPU 1              CPU 2
================== ==================
{ A == 1, B == 2, C = 3, P == &A, Q == &C }
B = 4;
<write barrier>
P = &B
                   Q = P;
                   D = *Q;
```

W oczywisty sposob wystepuje tu data dependency i wydaje sie ze na koncy Q musi byc albo &A lub &B oraz ze

```
(Q == &A) implies (D == 1)
(Q == &B) implies (D == 4)
```

Jednak wiedza CPU2 o P moze byc updateowana PRZED wiedza o B, co prowadzi do nastepujacego przeplotu

```
(Q == &B) and (D == 2) ????
```

architektura DEC Alpha

anomalie

Rozpatrzmy uklad:

```
CPU 1              CPU 2
================== ==================
{ A == 1, B == 2, C = 3, P == &A, Q == &C }
B = 4;
<write barrier>
P = &B
                   Q = P;
                   D = *Q;
```

W oczywisty sposob wystepuje tu data dependency i wydaje sie ze na koncy Q musi byc albo &A lub &B oraz ze

```
(Q == &A) implies (D == 1)
(Q == &B) implies (D == 4)
```

Jednak wiedza CPU2 o P moze byc updateowana PRZED wiedza o B, co prowadzi do nastepujacego przeplotu

```
(Q == &B) and (D == 2) ????
```

architektura DEC Alpha

anomalie - rozwiazanie

Rozwiazanie:

```
CPU 1                CPU 2
================     ================
{ A == 1, B == 2, C = 3, P == &A, Q == &C }
B = 4;
<write barrier>
P = &B
                     Q = P;
                     <data dependency barrier>
                     D = *Q;
```

wprowadzenie data dependency barrier powoduje ze zajdzie jedna z tych 2 'oczekiwanych' sytuacji

MEMORY BARRIERS w LINUX KERNEL 1

COMPILER BARRIER - barrier();

- prosta blokada powodujaca ze zadna operacja dostepu nie moze przedostac sie z jednej strony na druga
- ta blokada jest osobna wzgledem procesora wiec ...
- procesor moze pozniej przeorganizowac kolejnosc wykonywania instrukcji ?!

MEMORY BARRIERS w LINUX KERNEL 2

CPU MEMORY BARRIERS

TYPE	MANDATORY	SMP CONDITIONAL
GENERAL	mb()	smp_mb()
WRITE	wmb()	smp_wmb()
READ	rmb()	smp_rmb()
DATA DEPENDENCY	read_barrier_depends()	smp_read_barrier_depends()

oraz, troche bardziej zaawansowane:

- *set_mb*(*var*, *value*)
- *set_wbm*(*var*, *value*)

przypisuja wartosc do zmiennej a potem (w drugim przypadku) ustawiaja write-barrier

MEMORY BARRIERS w LINUX KERNEL 3

CPU MEMORY BARRIERS

```
(*) smp_mb__before_atomic_dec();
(*) smp_mb__after_atomic_dec();
(*) smp_mb__before_atomic_inc();
(*) smp_mb__after_atomic_inc();
```

Zobaczmy na przykladzie:

```
obj->dead = 1;
smp_mb__before_atomic_dec();
atomic_dec(&obj->ref_count);
```

To powoduje ze flaga 'dead' zostanie ustawiona zanim zmieni sie licznik. wiecej informacji w *Documentation\atomic_ops.txt*

```
(*) smp_mb__before_clear_bit(void);
(*) smp_mb__after_clear_bit(void);
```

Zabezpiecza operacje przed wywolaniem cleara przed 'przeskoczeniem' po clearu

```
smp_mb__before_clear_bit();
clear_bit( ... );
```

MEMORY BARRIERS w LINUX KERNEL 4

Posrednie wywolanie MEMORY BARRIER

Niektore z funkcji jadra linuxa posrednio wywoluje Memory Barriers (locking, scheduling, memory allocation)
scheduling, memory allocation

Co moze byc gwarantowane ?

Co z LOCKING Functions?
Jadro linuxa ma kilka konstrukcji typu LOCK m.in.

- spin locks
- mutexy
- semafory

Posrednie wywolanie MEMORY BARRIER

Niektore z funkcji jadra linuxa posrednio wywoluje Memory Barriers (locking, scheduling, memory allocation)
scheduling, memory allocation - full memory barrier

Co moze byc gwarantowane ?

Co z LOCKING Functions?
Jadro linuxa ma kilka konstrukcji typu LOCK m.in.

- spin locks
- mutexy
- semafory

MEMORY BARRIERS w LINUX KERNEL 4

Posrednie wywolanie MEMORY BARRIER

Niektore z funkcji jadra linuxa posrednio wywoluje Memory Barriers (locking, scheduling, memory allocation)
scheduling, memory allocation - full memory barrier

Co moze byc gwarantowane ?

Co z LOCKING Functions?
Jadro linuxa ma kilka konstrukcji typu LOCK m.in.

- spin locks
- mutexy
- semafory

Posrednie wywolanie MEMORY BARRIER

Niektore z funkcji jadra linuxa posrednio wywoluje Memory Barriers (locking, scheduling, memory allocation)
scheduling, memory allocation - full memory barrier

Co moze byc gwarantowane ?

Co z LOCKING Functions?
Jadro linuxa ma kilka konstrukcji typu LOCK m.in.

- spin locks
- mutexy
- semafory

MEMORY BARRIERS w LINUX KERNEL 5

LOCKING FUNCTIONS

1. implikacje operacji LOCK
2. implikacje operacji UNLOCK
3. implikacje LOCK(1) ; LOCK(2)
4. implikajce LOCK ; UNLOCK
5. implikacje LOCK'a ktory sie nie powiodl

wiec (1),(2),(4) UNLOCK z nastepujacym po nim LOCK sa rownowazne full barrier natomiast LOCK z nastepujacym po nim UNLOCK juz nie

UWAGA!

Nastepstwem tego ze LOCK i UNLOCK moga byc barierami 'w jedna strone' jest to ze efekty instrukcji spoza sekcji krytycznej moga 'przesiaknac' do wnetrza sekcji krytycznej

MEMORY BARRIERS w LINUX KERNEL 5

LOCKING FUNCTIONS

1. implikacje operacji LOCK
2. implikacje operacji UNLOCK
3. implikacje LOCK(1) ; LOCK(2)
4. implikajce LOCK ; UNLOCK
5. implikacje LOCK'a ktory sie nie powiodl

wiec (1),(2),(4) UNLOCK z nastepujacym po nim LOCK sa rownowazne full barrier natomiast LOCK z nastepujacym po nim UNLOCK juz nie

UWAGA!

Nastepstwem tego ze LOCK i UNLOCK moga byc barierami 'w jedna strone' jest to ze efekty instrukcji spoza sekcji krytycznej moga 'przesiaknac' do wnetrza sekcji krytycznej

MEMORY BARRIERS w LINUX KERNEL 5

LOCKING FUNCTIONS

1. implikacje operacji LOCK
2. implikacje operacji UNLOCK
3. implikacje LOCK(1) ; LOCK(2)
4. implikajce LOCK ; UNLOCK
5. implikacje LOCK'a ktory sie nie powiodl

wiec (1),(2),(4) UNLOCK z nastepujacym po nim LOCK sa rownowazne full barrier natomiast LOCK z nastepujacym po nim UNLOCK juz nie

UWAGA!

Nastepstwem tego ze LOCK i UNLOCK moga byc barierami 'w jedna strone' jest to ze efekty instrukcji spoza sekcji krytycznej moga 'przesiaknac' do wnetrza sekcji krytycznej

MEMORY BARRIERS w LINUX KERNEL 5

LOCKING FUNCTIONS

1. implikacje operacji LOCK
2. implikacje operacji UNLOCK
3. implikacje LOCK(1) ; LOCK(2)
4. implikajce LOCK ; UNLOCK
5. implikacje LOCK'a ktory sie nie powiodl

wiec (1),(2),(4) UNLOCK z nastepujacym po nim LOCK sa rownowazne full barrier natomiast LOCK z nastepujacym po nim UNLOCK juz nie

UWAGA!

Nastepstwem tego ze LOCK i UNLOCK moga byc barierami 'w jedna strone' jest to ze efekty instrukcji spoza sekcji krytycznej moga 'przesiaknac' do wnetrza sekcji krytycznej

MEMORY BARRIERS w LINUX KERNEL 5

LOCKING FUNCTIONS

1. implikacje operacji LOCK
2. implikacje operacji UNLOCK
3. implikacje LOCK(1) ; LOCK(2)
4. implikajce LOCK ; UNLOCK
5. implikacje LOCK'a ktory sie nie powiodl

wiec (1),(2),(4) UNLOCK z nastepujacym po nim LOCK sa rownowazne full barrier natomiast LOCK z nastepujacym po nim UNLOCK juz nie

UWAGA!

Nastepstwem tego ze LOCK i UNLOCK moga byc barierami 'w jedna strone' jest to ze efekty instrukcji spoza sekcji krytycznej moga 'przesiaknac' do wnetrza sekcji krytycznej

MEMORY BARRIERS w LINUX KERNEL 5

LOCKING FUNCTIONS

1. implikacje operacji LOCK
2. implikacje operacji UNLOCK
3. implikacje LOCK(1) ; LOCK(2)
4. implikajce LOCK ; UNLOCK
5. implikacje LOCK'a ktory sie nie powiodl

wiec (1),(2),(4) UNLOCK z nastepujacym po nim LOCK sa rownowazne full barrier natomiast LOCK z nastepujacym po nim UNLOCK juz nie

UWAGA!

Nastepstwem tego ze LOCK i UNLOCK moga byc barierami 'w jedna strone' jest to ze efekty instrukcji spoza sekcji krytycznej moga 'przesiaknac' do wnetrza sekcji krytycznej

MEMORY BARRIERS w LINUX KERNEL 5

LOCKING FUNCTIONS

1. implikacje operacji LOCK
2. implikacje operacji UNLOCK
3. implikacje LOCK(1) ; LOCK(2)
4. implikajce LOCK ; UNLOCK
5. implikacje LOCK'a ktory sie nie powiodl

wiec (1),(2),(4) UNLOCK z nastepujacym po nim LOCK sa rownowazne full barrier natomiast LOCK z nastepujacym po nim UNLOCK juz nie

UWAGA!

Nastepstwem tego ze LOCK i UNLOCK moga byc barierami 'w jedna strone' jest to ze efekty instrukcji spoza sekcji krytycznej moga 'przesiaknac' do wnetrza sekcji krytycznej

LOCKING FUNCTIONS - przyklady

```
*A = a; *B = b;
LOCK
*C = c; *D = d;
UNLOCK
*E = e; *F = f;
```

Nastepujaca kolejnosc zdarzen jest mozliwa

```
LOCK, {*F,*A}, *E, {*C,*D}, *B, UNLOCK

{*F,*A} oznacza jednoczesny dostep.
```

Ale zadna z tych juz nie

```
{*F,*A}, *B,    LOCK, *C, *D,   UNLOCK, *E
*A, *B, *C,     LOCK, *D,       UNLOCK, *E, *F
*A, *B,         LOCK, *C,       UNLOCK, *D, *E, *F
*B,             LOCK, *C, *D,   UNLOCK, {*F,*A}, *E
```

LOCKING FUNCTIONS - przyklady

```
*A = a; *B = b;
LOCK
*C = c; *D = d;
UNLOCK
*E = e; *F = f;
```

Nastepujaca kolejnosc zdarzen jest mozliwa

```
LOCK, {*F,*A}, *E, {*C,*D}, *B, UNLOCK
```

```
{*F,*A} oznacza jednoczesny dostep.
```

Ale zadna z tych juz nie

```
{*F,*A}, *B,    LOCK, *C, *D,   UNLOCK, *E
*A, *B, *C,     LOCK, *D,       UNLOCK, *E, *F
*A, *B,         LOCK, *C,       UNLOCK, *D, *E, *F
*B,             LOCK, *C, *D,   UNLOCK, {*F,*A}, *E
```

LOCKING FUNCTIONS - przyklady

```
*A = a; *B = b;
LOCK
*C = c; *D = d;
UNLOCK
*E = e; *F = f;
```

Nastepujaca kolejnosc zdarzen jest mozliwa

```
LOCK, {*F,*A}, *E, {*C,*D}, *B, UNLOCK

{*F,*A} oznacza jednoczesny dostep.
```

Ale zadna z tych juz nie

```
{*F,*A}, *B,     LOCK, *C, *D,    UNLOCK, *E
*A, *B, *C,      LOCK, *D,        UNLOCK, *E, *F
*A, *B,          LOCK, *C,        UNLOCK, *D, *E, *F
*B,              LOCK, *C, *D,    UNLOCK, {*F,*A}, *E
```

Plan Prezentacji

Czy 64-bitowe adresowanie to dzisiaj konieczność?

Czy 64-bitowe adresowanie to dzisiaj konieczność?

system operacyjny	*pamięć dostępna dla jednej aplikacji*	*maksymalna obsługiwana wielkość pamięci RAM*
Windows XP Professional (32-bit)	*4 GB*	*4 GB*
Windows XP Professional (64-bit)	*16 TB*	*128 GB*
Windows Server 2003 Standard (32-bit)	*4 GB*	*4 GB*
Windows Server 2003 Datacenter (32-bit)	*4 GB*	*128 GB (przy wsparciu sprzętu)*
Windows Server 2003 Datacenter (64-bit)	*16 TB*	*512 GB*

Przydzielanie pamięci dla aplikacji w systemie Windows

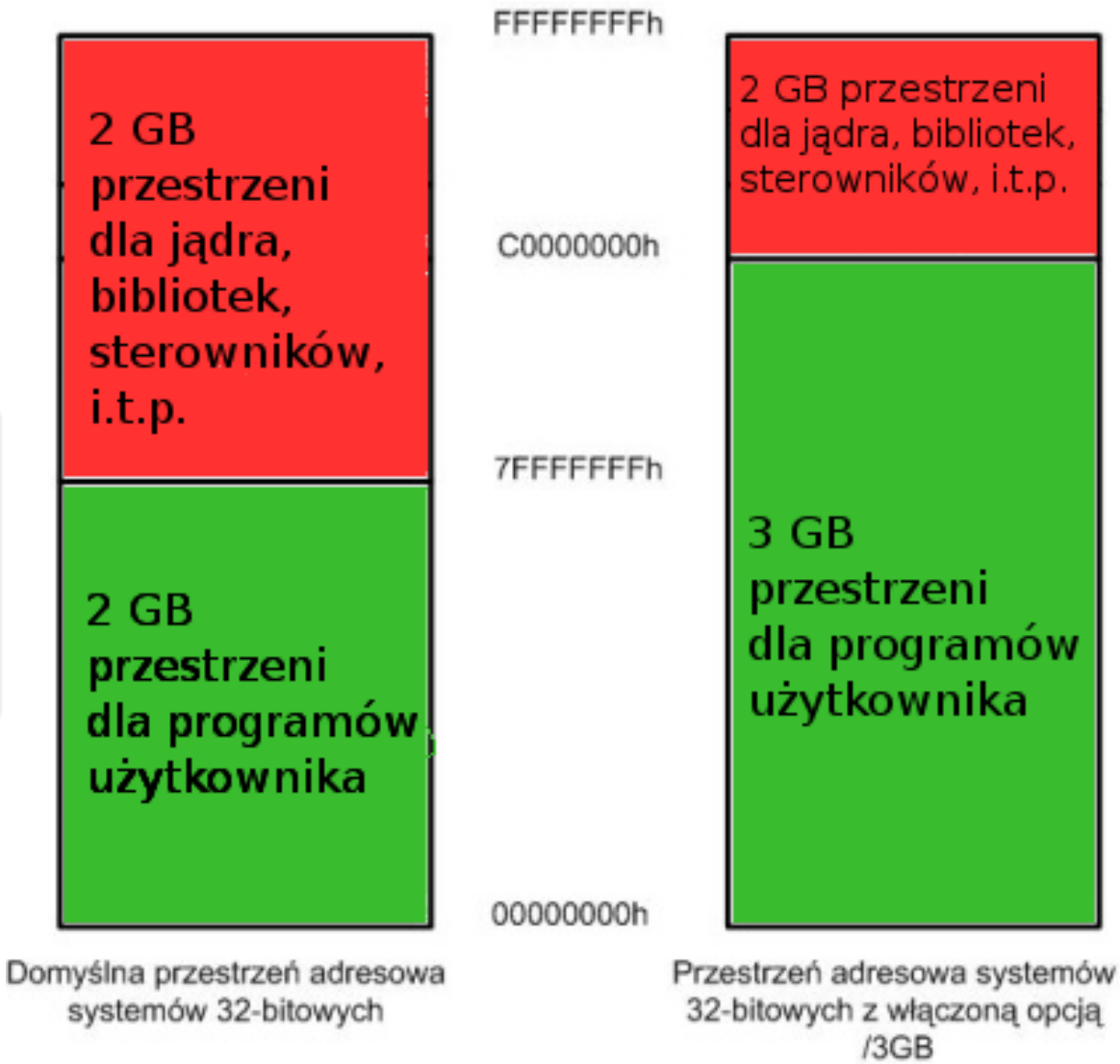
W 32-bitowym Windowsie mamy standardowo do dyspozycji jeszcze mniej niż 4 GB pamięci...

... bo system operacyjny też jej potrzebuje.

Przydzielanie pamięci dla aplikacji w systemie Windows

W 32-bitowym Windowsie mamy standardowo do dyspozycji jeszcze mniej niż 4 GB pamięci...

... bo system operacyjny też jej potrzebuje.



Przydzielanie pamięci dla aplikacji w systemie Windows

W 32-bitowym Windowsie mamy standardowo do dyspozycji jeszcze mniej niż 4 GB pamięci...

... bo system operacyjny też jej potrzebuje.

Istnieją jednak mechanizmy pozwalające zaadresować dużo więcej niż 4 GB, bez konieczności przechodzenia na tryb 64-bitowy.

PAE - Physical Address Extension

- wprowadzone w Pentium Pro
- rozszerza fizyczną przestrzeń adresową w 32-bitowej architekturze poprzez zastosowanie 36-bitowych adresów
- wymagane wsparcie systemu operacyjnego (np. Windows XP, Windows Vista, Windows Server, Linux 2.6)
- można zaadresować do 64 GB pamięci RAM

PAE - Physical Address Extension

- wprowadzone w Pentium Pro
- rozszerza fizyczną przestrzeń adresową w 32-bitowej architekturze poprzez zastosowanie 36-bitowych adresów
- wymagane wsparcie systemu operacyjnego (np. Windows XP, Windows Vista, Windows Server, Linux 2.6)
- można zaadresować do 64 GB pamięci RAM

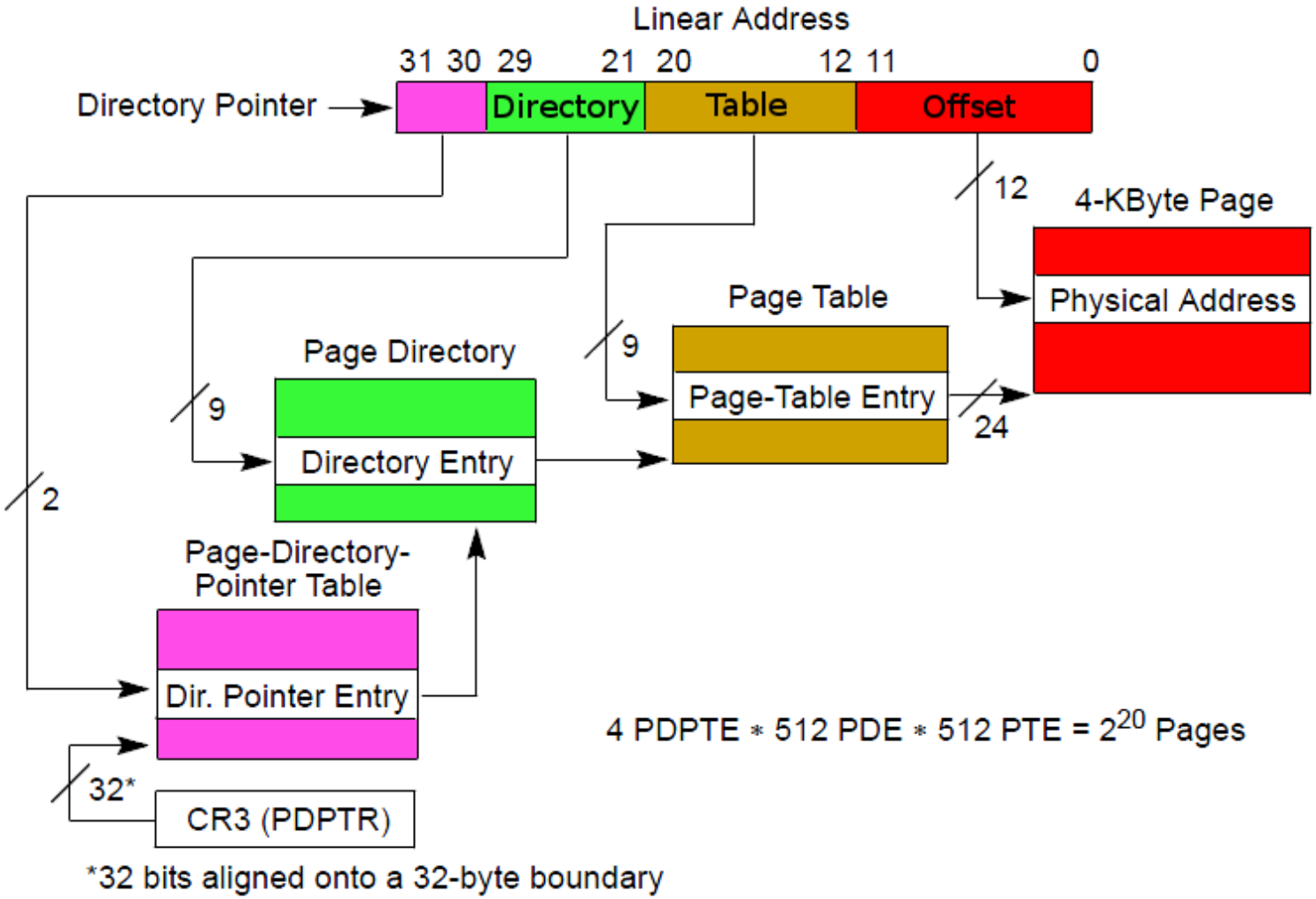
PAE - Physical Address Extension

- wprowadzone w Pentium Pro
- rozszerza fizyczną przestrzeń adresową w 32-bitowej architekturze poprzez zastosowanie 36-bitowych adresów
- wymagane wsparcie systemu operacyjnego (np. Windows XP, Windows Vista, Windows Server, Linux 2.6)
- można zaadresować do 64 GB pamięci RAM

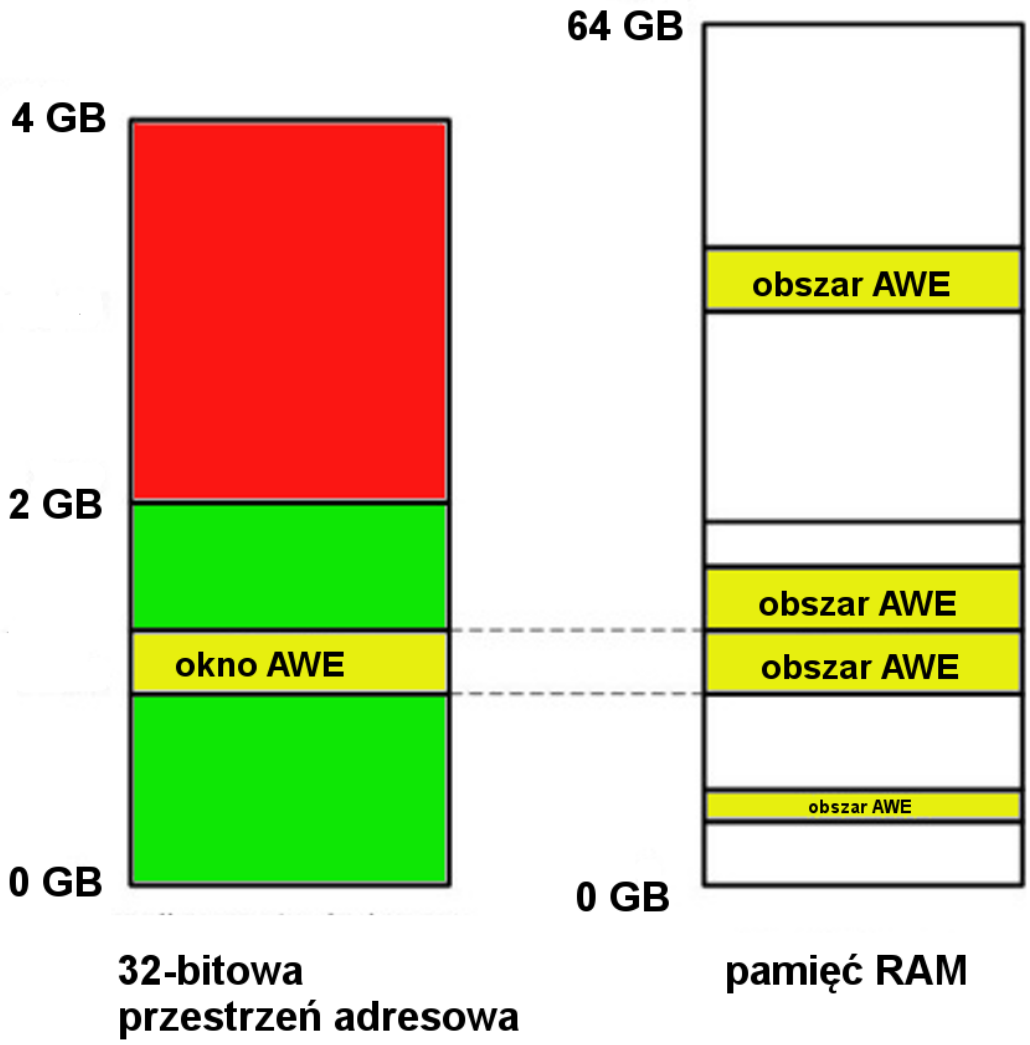
PAE - Physical Address Extension

- wprowadzone w Pentium Pro
- rozszerza fizyczną przestrzeń adresową w 32-bitowej architekturze poprzez zastosowanie 36-bitowych adresów
- wymagane wsparcie systemu operacyjnego (np. Windows XP, Windows Vista, Windows Server, Linux 2.6)
- można zaadresować do 64 GB pamięci RAM

PAE - trójpoziomowe stronicowanie



AWE - Address Windowing Extensions



AWE - Address Windowing Extensions

Ograniczenia:

- procesy nie mogą współdzielić stron pamięci
- nie można tej samej ramce w pamięci RAM przypisać kilku różnych adresów logicznych w ramach jednego procesu
- jeśli wiele aplikacji używa AWE, to mogą one wyczerpać całą dostępną pamięć fizyczną, uniemożliwiając kolejnym korzystanie z tego mechanizmu
- aplikacje używające AWE nie mogą być emulowane na innej platformie.

AWE - Address Windowing Extensions

Ograniczenia:

- procesy nie mogą współdzielić stron pamięci
- nie można tej samej ramce w pamięci RAM przypisać kilku różnych adresów logicznych w ramach jednego procesu
- jeśli wiele aplikacji używa AWE, to mogą one wyczerpać całą dostępną pamięć fizyczną, uniemożliwiając kolejnym korzystanie z tego mechanizmu
- aplikacje używające AWE nie mogą być emulowane na innej platformie.

AWE - Address Windowing Extensions

Ograniczenia:

- procesy nie mogą współdzielić stron pamięci
- nie można tej samej ramce w pamięci RAM przypisać kilku różnych adresów logicznych w ramach jednego procesu
- jeśli wiele aplikacji używa AWE, to mogą one wyczerpać całą dostępną pamięć fizyczną, uniemożliwiając kolejnym korzystanie z tego mechanizmu
- aplikacje używające AWE nie mogą być emulowane na innej platformie.

AWE - Address Windowing Extensions

Ograniczenia:

- procesy nie mogą współdzielić stron pamięci
- nie można tej samej ramce w pamięci RAM przypisać kilku różnych adresów logicznych w ramach jednego procesu
- jeśli wiele aplikacji używa AWE, to mogą one wyczerpać całą dostępną pamięć fizyczną, uniemożliwiając kolejnym korzystanie z tego mechanizmu
- aplikacje używające AWE nie mogą być emulowane na innej platformie.

AWE - Address Windowing Extensions

Ograniczenia:

- procesy nie mogą współdzielić stron pamięci
- nie można tej samej ramce w pamięci RAM przypisać kilku różnych adresów logicznych w ramach jednego procesu
- jeśli wiele aplikacji używa AWE, to mogą one wyczerpać całą dostępną pamięć fizyczną, uniemożliwiając kolejnym korzystanie z tego mechanizmu
- aplikacje używające AWE nie mogą być emulowane na innej platformie.

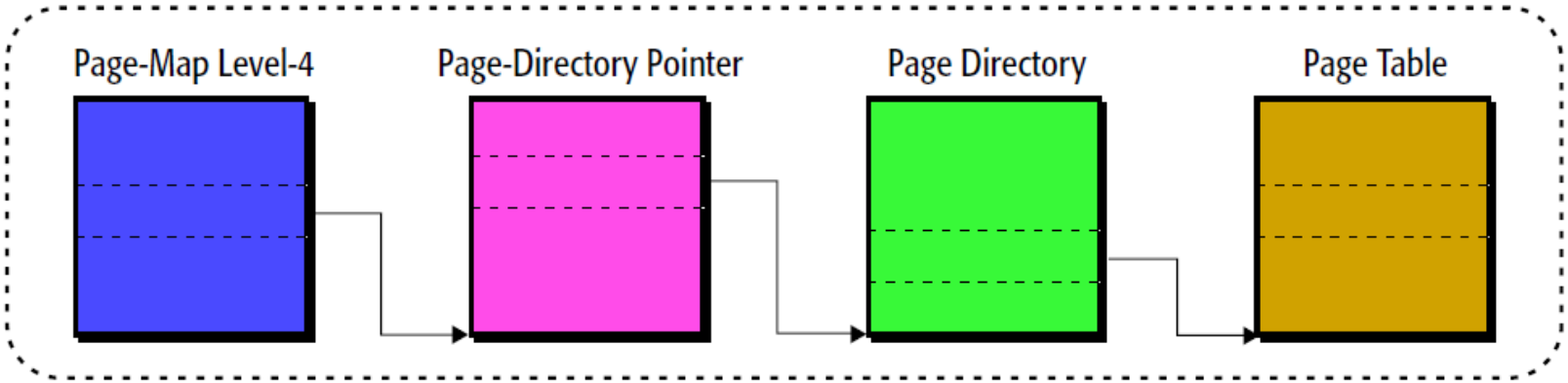
AWE - Address Windowing Extensions

Alternatywą w Linuksie dla AWE jest funkcja systemowa mmap() (która mapuje pliki lub urządzenia do pamięci).

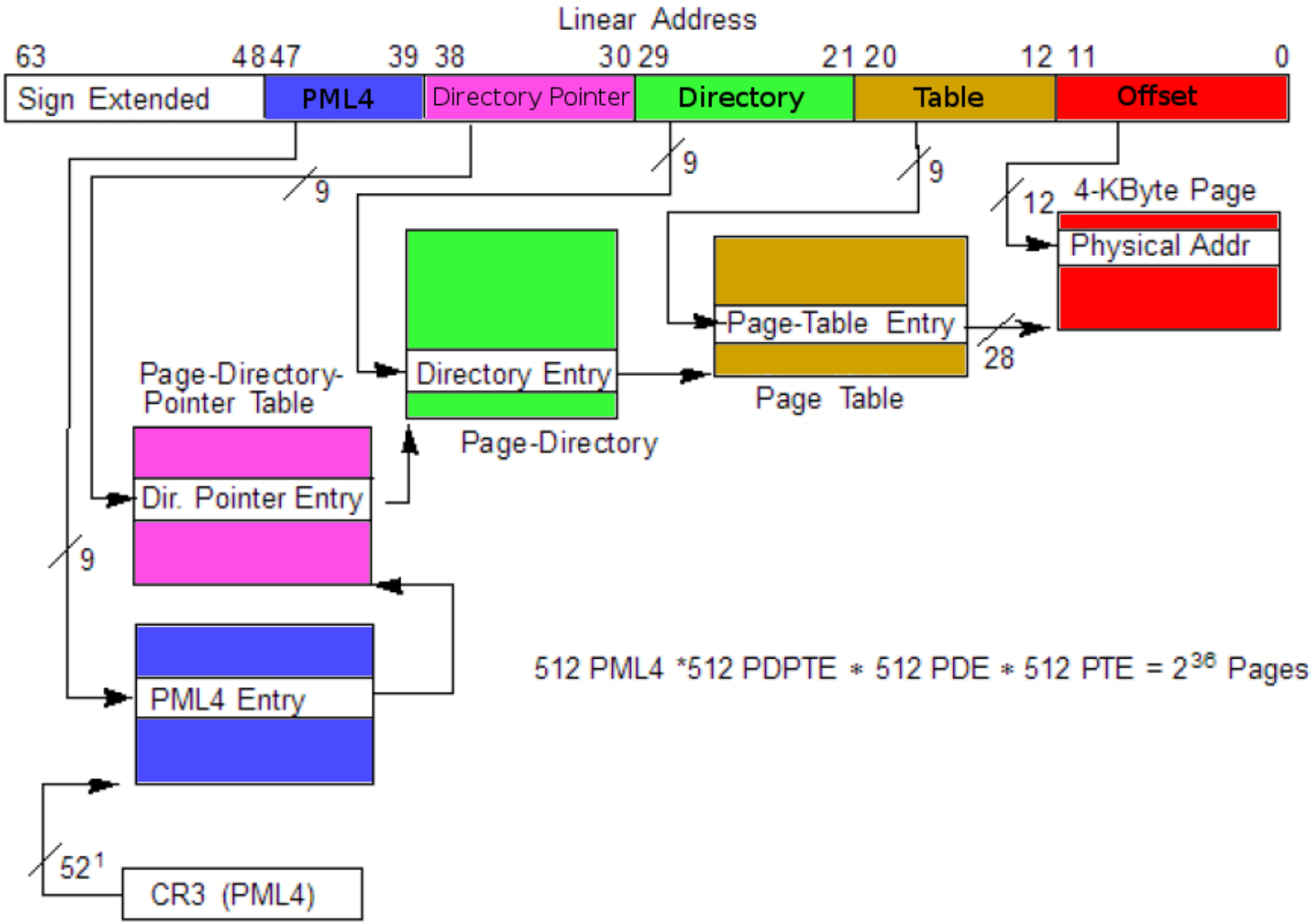
Adresowanie w trybie 64-bitowym

Adresowanie w trybie 64-bitowym

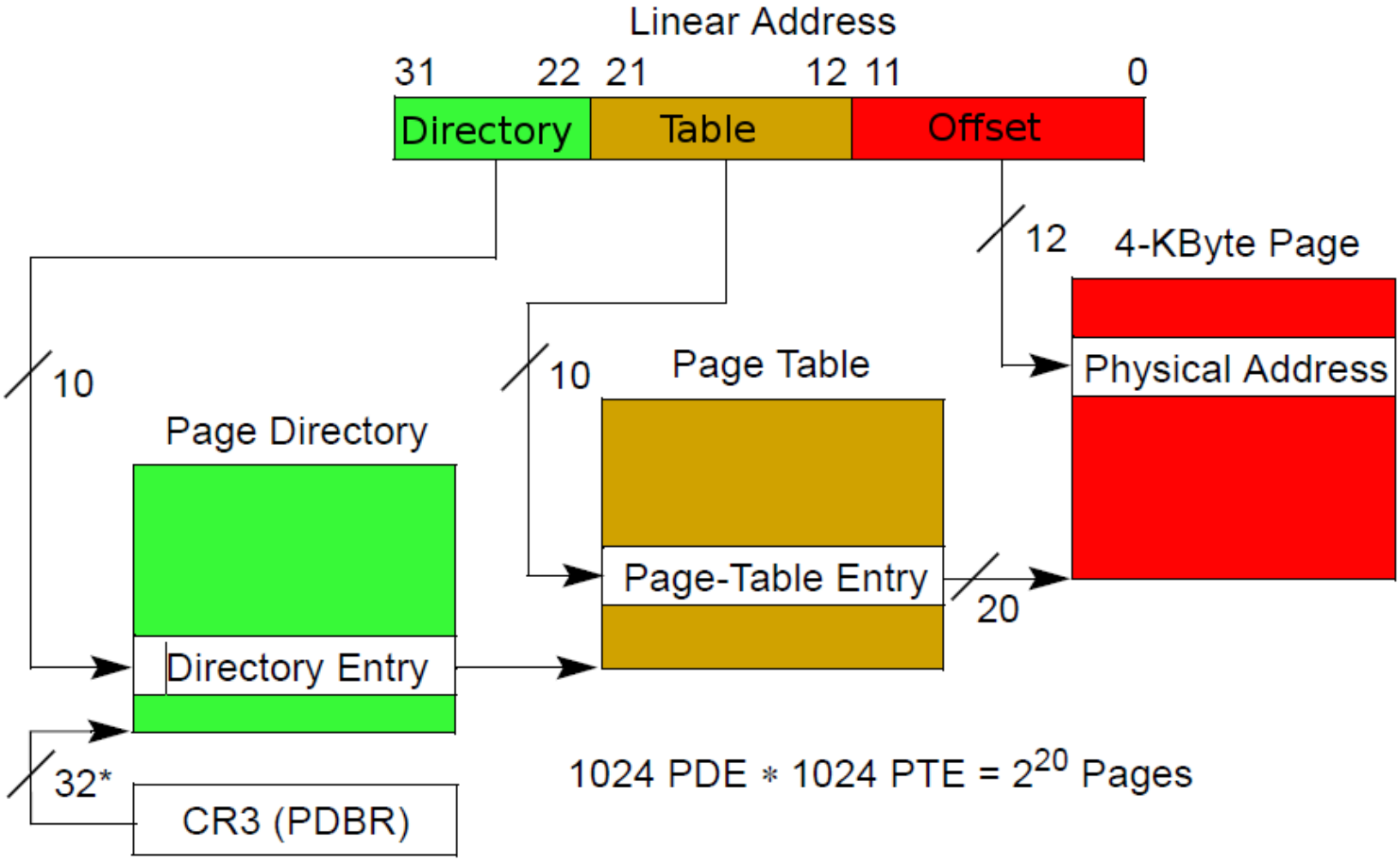
W trybie IA-32e, przy stronie 4 KB, stosuje się czteropoziomowe stronicowanie:



Adresowanie w trybie 64-bitowym



Dla porównania w trybie 32-bitowym:



64-bity a segmentacja

- istnieje "płaska" 64-bitowa liniowa przestrzeń adresowa, bez segmentacji
- rejestry CS, DS, ES, SS są traktowane jako zero
- rejestry FS, GS pełnią rolę tylko pomocniczą

64-bity a segmentacja

- istnieje "płaska" 64-bitowa liniowa przestrzeń adresowa, bez segmentacji
- rejestry CS, DS, ES, SS są traktowane jako zero
- rejestry FS, GS pełnią rolę tylko pomocniczą

64-bity a segmentacja

- istnieje "płaska" 64-bitowa liniowa przestrzeń adresowa, bez segmentacji
- rejestry CS, DS, ES, SS są traktowane jako zero
- rejestry FS, GS pełnią rolę tylko pomocniczą

Wielkość stron

Domyślnie 4 KB (Small Page Size).
Jest możliwa większa wartość w trybie PSE (Page Size Extensions):

- 4 MB dla x86
- 2 MB dla x64 i x86 w trybie PAE

Wielkość stron

Domyślnie 4 KB (Small Page Size).

Jest możliwa większa wartość w trybie PSE (Page Size Extensions):

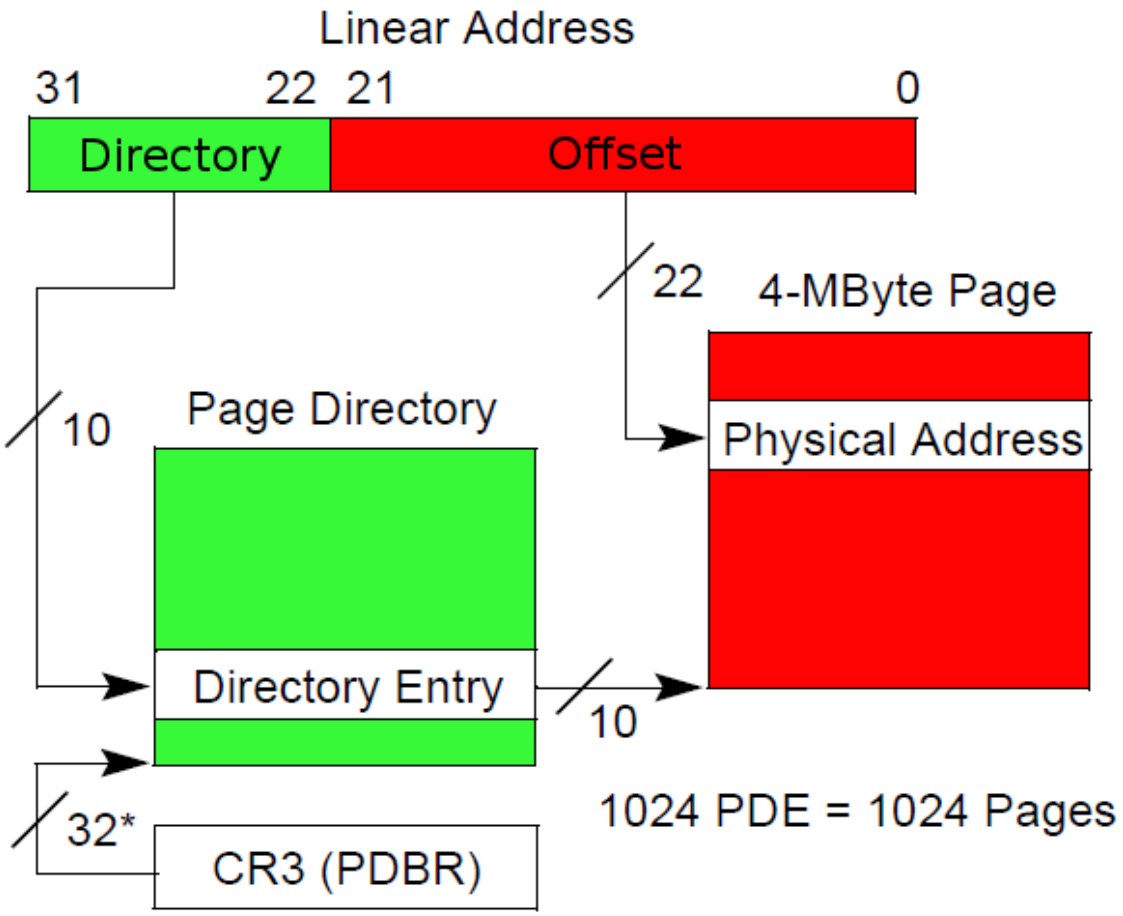
- 4 MB dla x86
- 2 MB dla x64 i x86 w trybie PAE

Wielkość stron

Domyślnie 4 KB (Small Page Size).
Jest możliwa większa wartość w trybie PSE (Page Size Extensions):

- 4 MB dla x86
- 2 MB dla x64 i x86 w trybie PAE

Wielkość stron - PSE w trybie 32-bitowym:

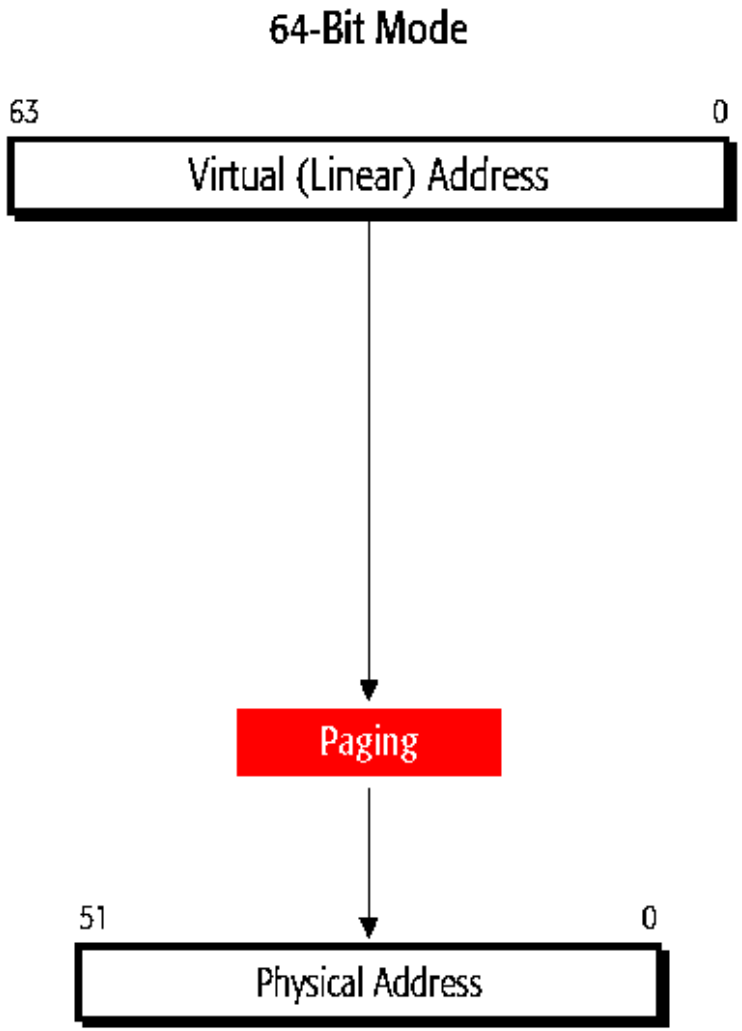


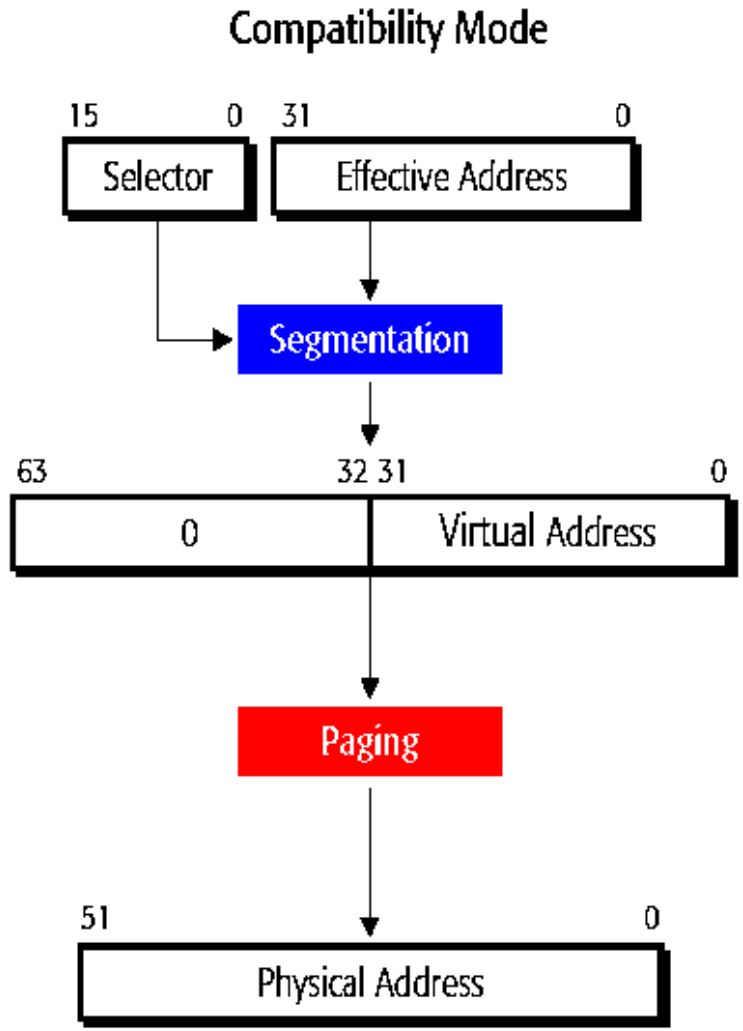
*32 bits aligned onto a 4-KByte boundary.

Tryby pracy procesora 64-bitowego

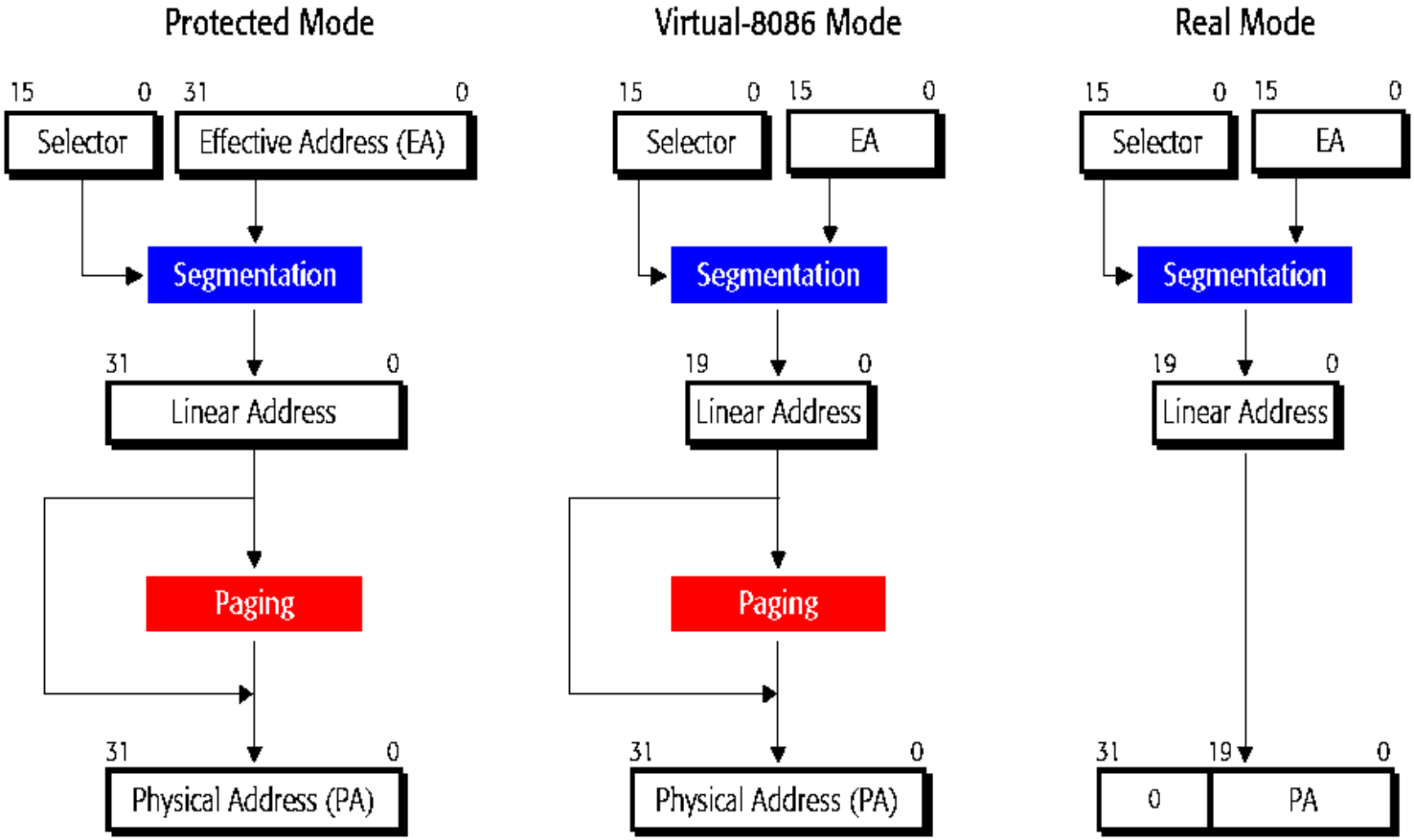
- long mode
- legacy mode

long mode


64-Bit Mode
63
0
Virtual (Linear) Address
Paging
51
0
Physical Address


Compatibility Mode
15
0
31
0
Selector
Effective Address
Segmentation
63
32 31
0
0
Virtual Address
Paging
51
0
Physical Address

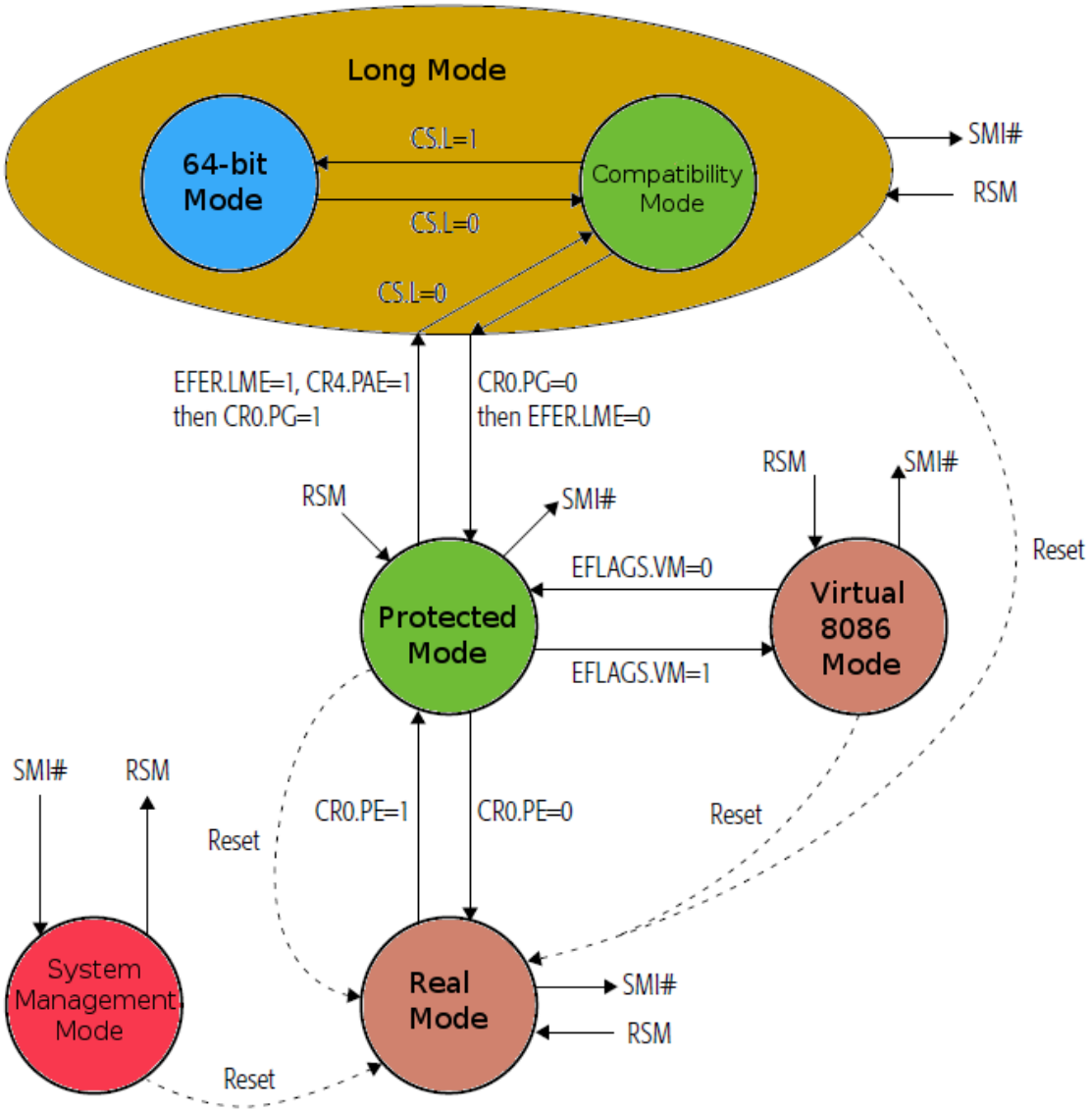
legacy mode


Protected Mode
15
0
31
0
Selector
Effective Address (EA)
Segmentation
31
0
Linear Address
Paging
31
0
Physical Address (PA)
Virtual-8086 Mode
15
0
15
0
Selector
EA
Segmentation
19
0
Linear Address
Paging
31
0
Physical Address (PA)
Real Mode
15
0
15
0
Selector
EA
Segmentation
19
0
Linear Address
31
19
0
0
PA

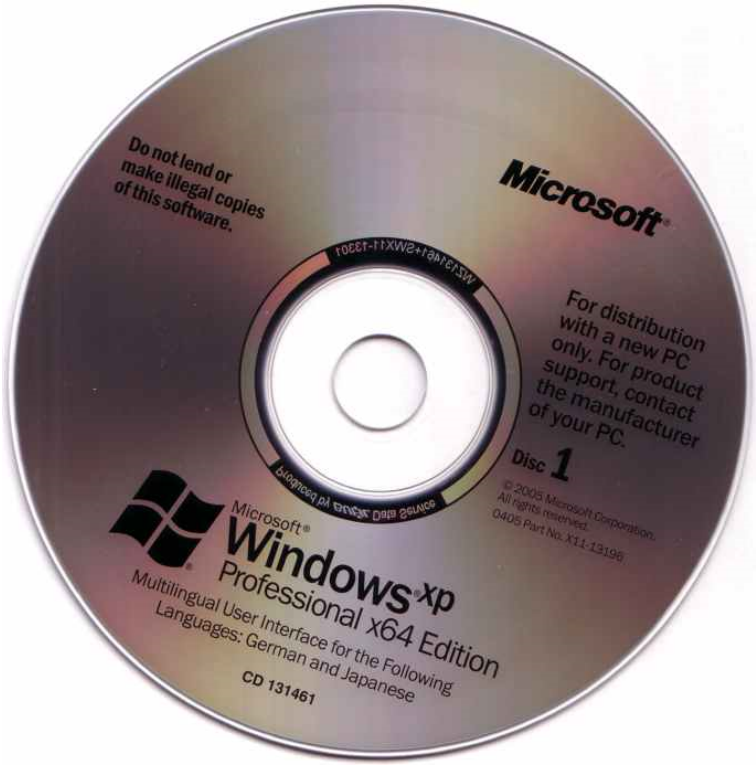
Tryby pracy procesora 64-bitowego - podsumowanie

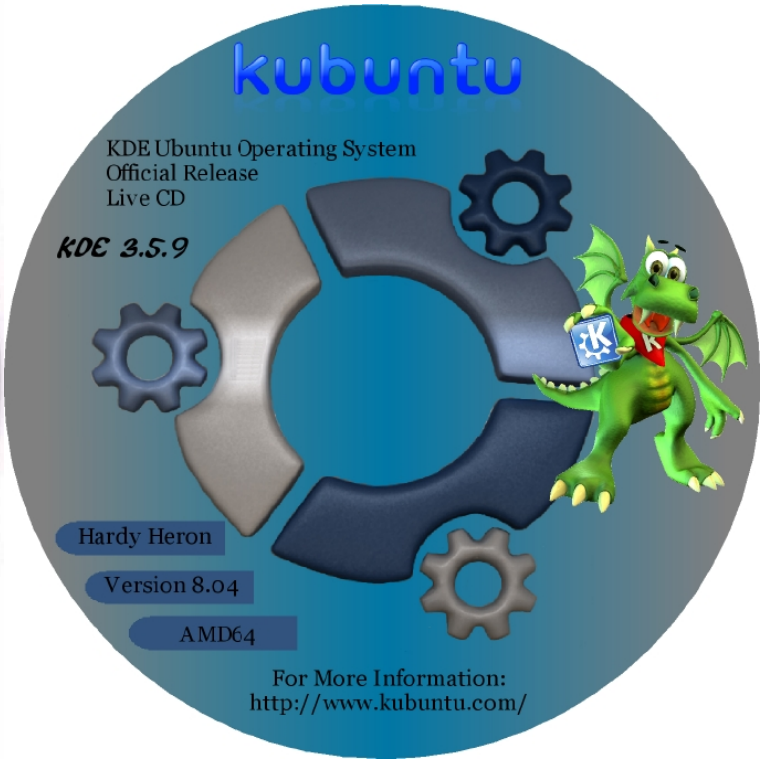
Operating Mode		Operating System Required	Application Recompile Required	Defaults		Register Extensions	Typical
				Address Size (bits)	Operand Size (bits)		GPR Width (bits)
Long Mode	64-Bit Mode	64-bit OS	yes	64	32	yes	64
	Compatibility Mode		no	32		no	32
				16	16		16
Legacy Mode	Protected Mode	Legacy 32-bit OS	no	32	32	no	32
				16	16		
	Virtual-8086 Mode			16	16		16
	Real Mode	Legacy 16-bit OS					

Tryby pracy procesora 64-bitowego - podsumowanie



64-bitowe systemy operacyjne - przykłady





64-bitowe edycje systemu Windows ...

... mają problemy z 32-bitowymi sterownikami

- dlatego logo "Certified for Vista" może uzyskać tylko sprzęt, do którego zostaną dostarczone sterowniki 32 i 64-bitowe

64-bitowe edycje systemu Windows ...

... posiadają następujące ograniczenia pamięciowe:

- maksymalnie 16 TB logicznej przestrzeni adresowej (8 TB dla procesów jądra i 8 TB dla procesów użytkownika)
- nie więcej niż 128 GB (Windows XP) lub 1 TB (Windows Server 2003) pamięci RAM

64-bitowe edycje systemu Windows ...

... umożliwiają skorzystanie z technologii WOW64 (Windows-on-Windows 64-bit)

- ale nie można łączyć kodu 32-bitowego z 64-bitowym

64-bitowy Linux

- był jednym z pierwszych 64-bitowych systemów operacyjnych
- jest w nim możliwa 64-bitowa rekompilacja programów, które wykorzystują inne, 32-bitowe programy

64-bitowy Linux

- był jednym z pierwszych 64-bitowych systemów operacyjnych
- jest w nim możliwa 64-bitowa rekompilacja programów, które wykorzystują inne, 32-bitowe programy

64-bitowy Mac OS X

- jądro uruchamiane jest jako 32-bitowy proces, dzięki czemu działają 32-bitowe sterowniki urządzeń
- jednocześnie wspierane są 64-bitowe procesy użytkownika
- technologia "universal binary" - wszystkie wersje programu w jednym pliku

64-bitowy Mac OS X

- jądro uruchamiane jest jako 32-bitowy proces, dzięki czemu działają 32-bitowe sterowniki urządzeń
- jednocześnie wspierane są 64-bitowe procesy użytkownika
- technologia "universal binary" - wszystkie wersje programu w jednym pliku

64-bitowy Mac OS X

- jądro uruchamiane jest jako 32-bitowy proces, dzięki czemu działają 32-bitowe sterowniki urządzeń
- jednocześnie wspierane są 64-bitowe procesy użytkownika
- technologia "universal binary" - wszystkie wersje programu w jednym pliku

Spójność pamięci cache w procesorach Intela i AMD

- Procesor nie zapewnia, że dane znajdujące się w jego pamięci podręcznej są zawsze spójne z pamięcią RAM.
- Musi więc to zapewnić system operacyjny.
- Pomocna jest instrukcja WBINVD.

Spójność pamięci cache w procesorach Intela i AMD

- Procesor nie zapewnia, że dane znajdujące się w jego pamięci podręcznej są zawsze spójne z pamięcią RAM.
- Musi więc to zapewnić system operacyjny.
- Pomocna jest instrukcja WBINVD.

Spójność pamięci cache w procesorach Intela i AMD

- Procesor nie zapewnia, że dane znajdujące się w jego pamięci podręcznej są zawsze spójne z pamięcią RAM.
- Musi więc to zapewnić system operacyjny.
- Pomocna jest instrukcja WBINVD.

MTRR - memory type range registers

- Rejestry MTRR pozwalają powiązać typ pamięci z zakresem przestrzeni adresowej.
- Dzięki temu procesor może optymalizować operacje na różnych typach pamięci, np. RAM, ROM, mapowane do pamięci urządzenia I/O.
- Strong Uncacheable (UC) - lokacje w pamięci nie są nigdy sprowadzane do pamięci cache.
- W systemach wieloprocesorowych system operacyjny musi utrzymywać spójność MTRR (bo oczywiście różne procesory muszą używać tych samych wartości MTRR).
- Procesory AMD i Intela nie zapewniają bowiem sprzętowego wsparcia dla utrzymania tej spójności.

MTRR - memory type range registers

- Rejestry MTRR pozwalają powiązać typ pamięci z zakresem przestrzeni adresowej.
- Dzięki temu procesor może optymalizować operacje na różnych typach pamięci, np. RAM, ROM, mapowane do pamięci urządzenia I/O.
- Strong Uncacheable (UC) - lokacje w pamięci nie są nigdy sprowadzane do pamięci cache.
- W systemach wieloprocesorowych system operacyjny musi utrzymywać spójność MTRR (bo oczywiście różne procesory muszą używać tych samych wartości MTRR).
- Procesory AMD i Intela nie zapewniają bowiem sprzętowego wsparcia dla utrzymania tej spójności.

MTRR - memory type range registers

- Rejestry MTRR pozwalają powiązać typ pamięci z zakresem przestrzeni adresowej.
- Dzięki temu procesor może optymalizować operacje na różnych typach pamięci, np. RAM, ROM, mapowane do pamięci urządzenia I/O.
- Strong Uncacheable (UC) - lokacje w pamięci nie są nigdy sprowadzane do pamięci cache.
- W systemach wieloprocesorowych system operacyjny musi utrzymywać spójność MTRR (bo oczywiście różne procesory muszą używać tych samych wartości MTRR).
- Procesory AMD i Intela nie zapewniają bowiem sprzętowego wsparcia dla utrzymania tej spójności.

MTRR - memory type range registers

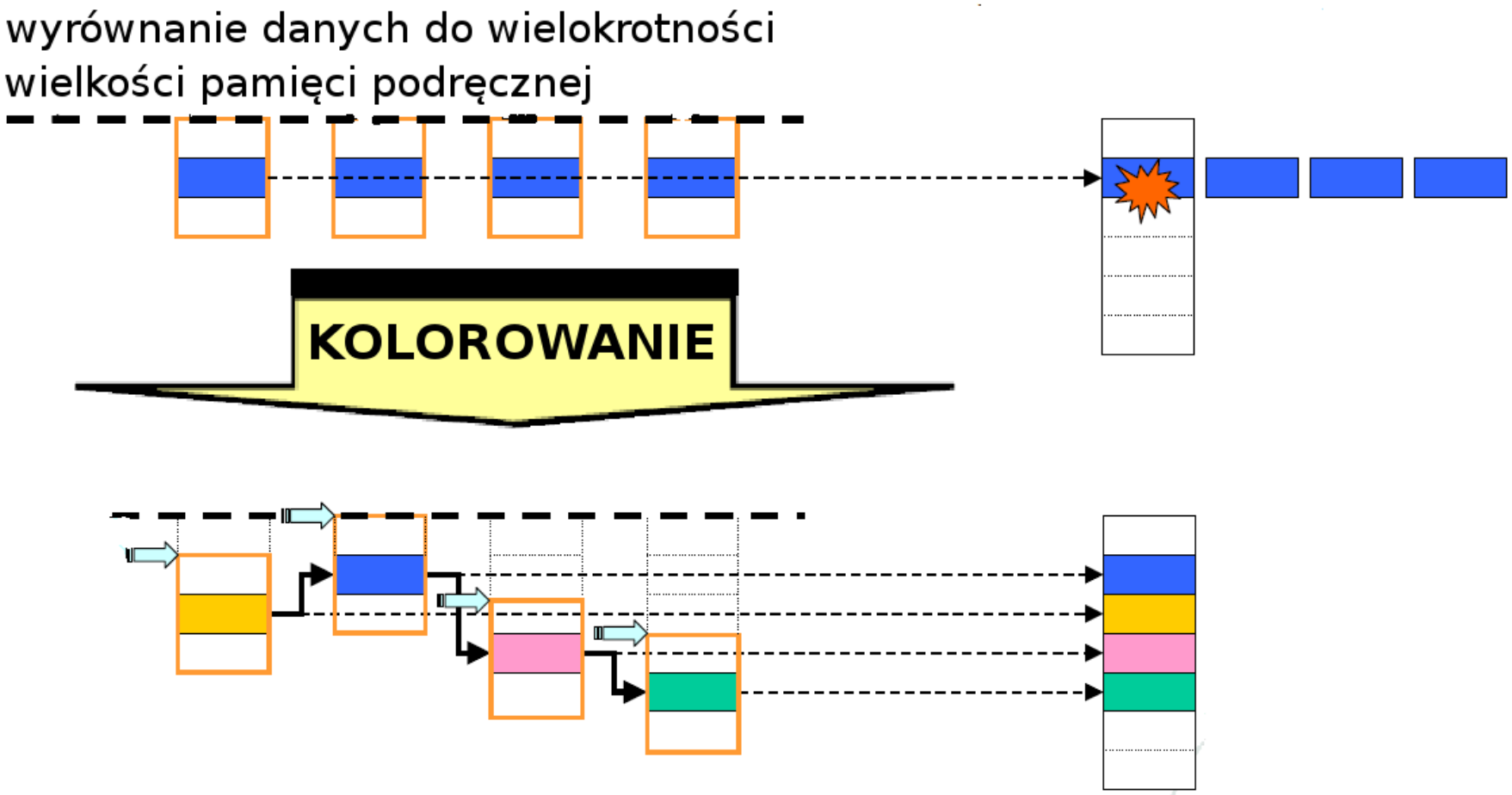
- Rejestry MTRR pozwalają powiązać typ pamięci z zakresem przestrzeni adresowej.
- Dzięki temu procesor może optymalizować operacje na różnych typach pamięci, np. RAM, ROM, mapowane do pamięci urządzenia I/O.
- Strong Uncacheable (UC) - lokacje w pamięci nie są nigdy sprowadzane do pamięci cache.
- W systemach wieloprocesorowych system operacyjny musi utrzymywać spójność MTRR (bo oczywiście różne procesory muszą używać tych samych wartości MTRR).
- Procesory AMD i Intela nie zapewniają bowiem sprzętowego wsparcia dla utrzymania tej spójności.

MTRR - memory type range registers

- Rejestry MTRR pozwalają powiązać typ pamięci z zakresem przestrzeni adresowej.
- Dzięki temu procesor może optymalizować operacje na różnych typach pamięci, np. RAM, ROM, mapowane do pamięci urządzenia I/O.
- Strong Uncacheable (UC) - lokacje w pamięci nie są nigdy sprowadzane do pamięci cache.
- W systemach wieloprocesorowych system operacyjny musi utrzymywać spójność MTRR (bo oczywiście różne procesory muszą używać tych samych wartości MTRR).
- Procesory AMD i Intela nie zapewniają bowiem sprzętowego wsparcia dla utrzymania tej spójności.

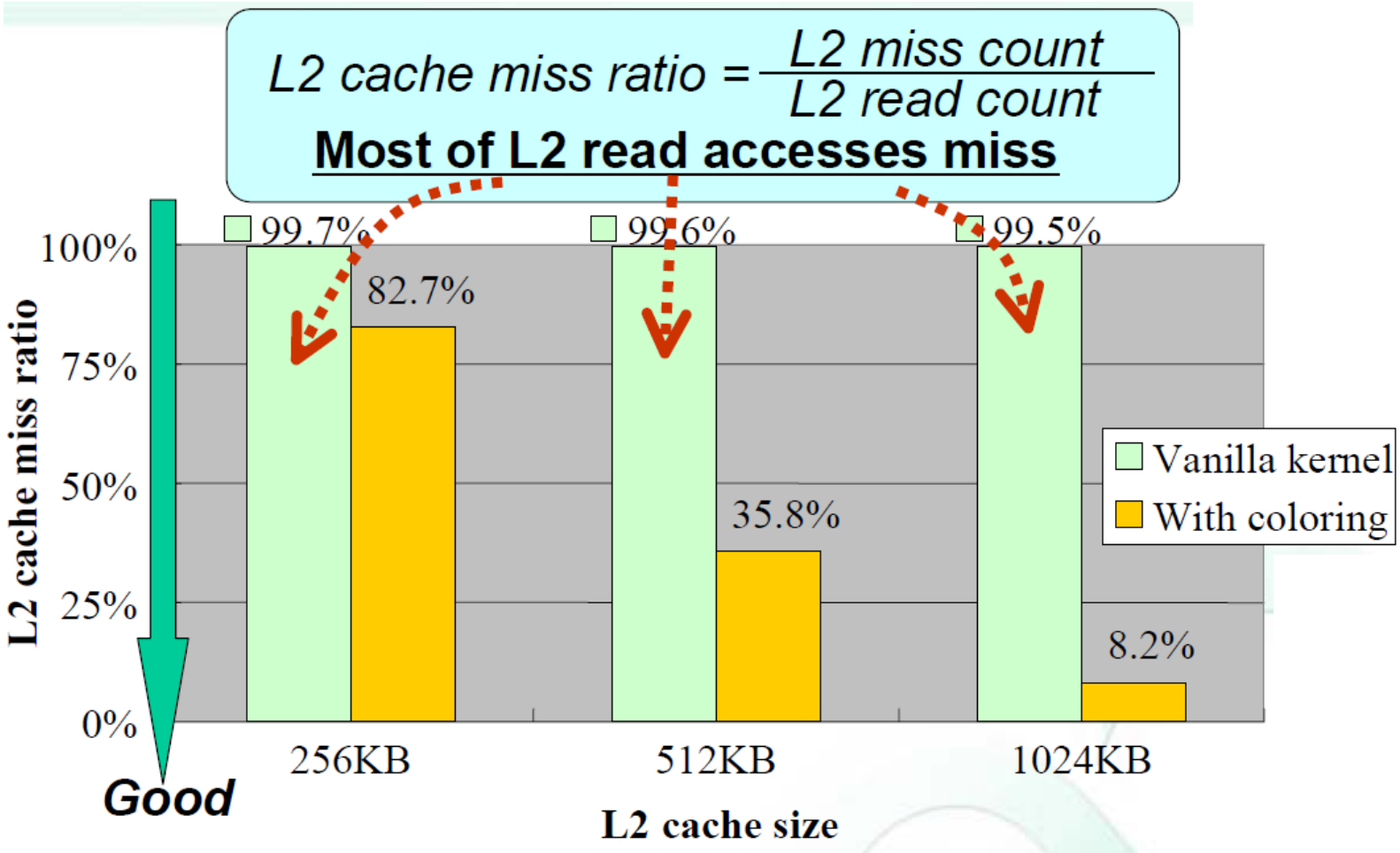
Kolorowanie - przykład: przyspieszanie schedulera w Linuksie

Kolorowanie - przykład: przyspieszanie schedulera w Linuksie

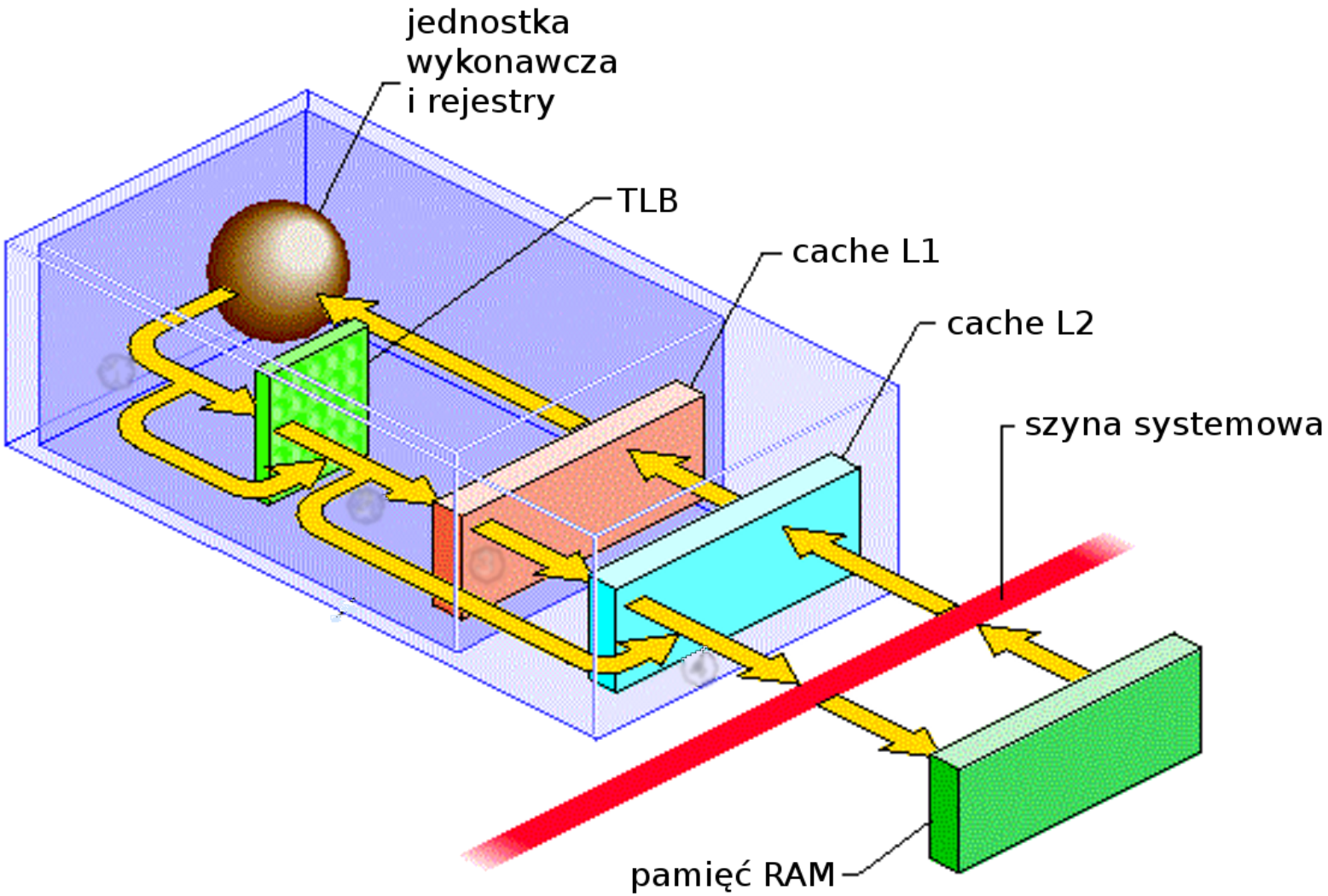


Kolorowanie - przykład: przyspieszanie schedulera w Linuksie

Różnica w wydajności jest szczególnie widoczna w przypadku Linuksa 2.4 (test przeprowadzony w laboratorium firmy Fujitsu)



TLB - przypomnienie



TLB - spójność

- za każdym razem, kiedy katalog lub tablica stron ulega zmianie, system operacyjny musi natychmiast unieważnić odpowiedni wpis w TLB (poprzez instrukcję INVLPG)
- konieczność modyfikacji TLB zachodzi między innymi podczas przełączania kontekstu

System operacyjny może radzić sobie z tym problemem na dwa sposoby:

- przez opróżnianie całej zawartości TLB przy zmianie kontekstu
- przez zapisywanie identyfikatorów przestrzeni adresowej (ASID - address space identifiers) razem z każdym wpisem w TLB.

TLB - spójność

- za każdym razem, kiedy katalog lub tablica stron ulega zmianie, system operacyjny musi natychmiast unieważnić odpowiedni wpis w TLB (poprzez instrukcję INVLPG)
- konieczność modyfikacji TLB zachodzi między innymi podczas przełączania kontekstu

System operacyjny może radzić sobie z tym problemem na dwa sposoby:

- przez opróżnianie całej zawartości TLB przy zmianie kontekstu
- przez zapisywanie identyfikatorów przestrzeni adresowej (ASID - address space identifiers) razem z każdym wpisem w TLB.

TLB - spójność

- za każdym razem, kiedy katalog lub tablica stron ulega zmianie, system operacyjny musi natychmiast unieważnić odpowiedni wpis w TLB (poprzez instrukcję INVLPG)
- konieczność modyfikacji TLB zachodzi między innymi podczas przełączania kontekstu

System operacyjny może radzić sobie z tym problemem na dwa sposoby:

- przez opróżnianie całej zawartości TLB przy zmianie kontekstu
- przez zapisywanie identyfikatorów przestrzeni adresowej (ASID - address space identifiers) razem z każdym wpisem w TLB.

TLB - spójność

- za każdym razem, kiedy katalog lub tablica stron ulega zmianie, system operacyjny musi natychmiast unieważnić odpowiedni wpis w TLB (poprzez instrukcję INVLPG)
- konieczność modyfikacji TLB zachodzi między innymi podczas przełączania kontekstu

System operacyjny może radzić sobie z tym problemem na dwa sposoby:

- przez opróżnianie całej zawartości TLB przy zmianie kontekstu
- przez zapisywanie identyfikatorów przestrzeni adresowej (ASID - address space identifiers) razem z każdym wpisem w TLB.

TLB - spójność

- za każdym razem, kiedy katalog lub tablica stron ulega zmianie, system operacyjny musi natychmiast unieważnić odpowiedni wpis w TLB (poprzez instrukcję INVLPG)
- konieczność modyfikacji TLB zachodzi między innymi podczas przełączania kontekstu

System operacyjny może radzić sobie z tym problemem na dwa sposoby:

- przez opróżnianie całej zawartości TLB przy zmianie kontekstu
- przez zapisywanie identyfikatorów przestrzeni adresowej (ASID - address space identifiers) razem z każdym wpisem w TLB.

TLB - spójność

- za każdym razem, kiedy katalog lub tablica stron ulega zmianie, system operacyjny musi natychmiast unieważnić odpowiedni wpis w TLB (poprzez instrukcję INVLPG)
- konieczność modyfikacji TLB zachodzi między innymi podczas przełączania kontekstu

System operacyjny może radzić sobie z tym problemem na dwa sposoby:

- przez opróżnianie całej zawartości TLB przy zmianie kontekstu
- przez zapisywanie identyfikatorów przestrzeni adresowej (ASID - address space identifiers) razem z każdym wpisem w TLB.

Obsługa chybienia w TLB

Są dwie możliwości:

- sprzętowe zarządzanie TLB (np. w procesorach IA32) - chybienie obsługiwane przez procesor
- programowe zarządzanie TLB (np. w procesorach IA64) - chybienie przekazywane jako wyjątek systemowi operacyjnemu

Obsługa chybienia w TLB

Są dwie możliwości:

- sprzętowe zarządzanie TLB (np. w procesorach IA32) - chybienie obsługiwane przez procesor
- programowe zarządzanie TLB (np. w procesorach IA64) - chybienie przekazywane jako wyjątek systemowi operacyjnemu

Obsługa chybienia w TLB

Są dwie możliwości:

- sprzętowe zarządzanie TLB (np. w procesorach IA32) - chybienie obsługiwane przez procesor
- programowe zarządzanie TLB (np. w procesorach IA64) - chybienie przekazywane jako wyjątek systemowi operacyjnemu

Zasięg bufora TLB

- Zwiększając n-krotnie rozmiar strony, zwiększamy n-krotnie zasięg TLB ...
- ... w rezultacie osiągając wyższy współczynnik trafień w TLB.
- Windows XP dla procesorów zgodnych z IA32 używa zawsze stron wielkości 4 KB.

Zasięg bufora TLB

- Zwiększając *n*-krotnie rozmiar strony, zwiększamy *n*-krotnie zasięg TLB ...
- ... w rezultacie osiągając wyższy współczynnik trafień w TLB.
- Windows XP dla procesorów zgodnych z IA32 używa zawsze stron wielkości 4 KB.

Zasięg bufora TLB

- Zwiększając *n*-krotnie rozmiar strony, zwiększamy *n*-krotnie zasięg TLB ...
- ... w rezultacie osiągając wyższy współczynnik trafień w TLB.
- Windows XP dla procesorów zgodnych z IA32 używa zawsze stron wielkości 4 KB.

Zasięg bufora TLB

- Zwiększając *n*-krotnie rozmiar strony, zwiększamy *n*-krotnie zasięg TLB ...
- ... w rezultacie osiągając wyższy współczynnik trafień w TLB.
- Windows XP dla procesorów zgodnych z IA32 używa zawsze stron wielkości 4 KB.

Plan Prezentacji

Very Long Instruction Word

Bardzo długie instrukcje

- W jednej instrukcji kodujemy wiele atomowych operacji które procesor wykona równolegle
- Przykład (arch. SHARC - Super Harvard Architecture Single-Chip Computer)

```
f12=f0*f4, f8=f8+f12, f0=dm(i0,m3), f4=pm(i8,m9);
```

- Zbiór możliwych instrukcji zależy od liczby jednostek wykonawczych
- Rozwiązywanie problemów związanych z zrównoleglaniem kodu, wyborem które instrukcje będą wykonywane jako następne (if-y) itp jest przerzucone na kompilator

Zalety

- Uproszczony hardware
- Zapewne lepsze przewidywania co do wyboru gałęzi kodu
- Jeśli mamy dobry kompilator to powinniśmy mieć lepsze rezulaty niż dla potokowania
- Niezgodność z innymi architekturami, złożone kompilatory
- Czasem instrukcje są w znacznej mierze puste

Very Long Instruction Word

Bardzo długie instrukcje

- W jednej instrukcji kodujemy wiele atomowych operacji które procesor wykona równolegle
- Przykład (arch. SHARC - Super Harvard Architecture Single-Chip Computer)

```
f12=f0*f4, f8=f8+f12, f0=dm(i0,m3), f4=pm(i8,m9);
```

- Zbiór możliwych instrukcji zależy od liczby jednostek wykonawczych
- Rozwiązywanie problemów związanych z zrównoleglaniem kodu, wyborem które instrukcje będą wykonywane jako następne (if-y) itp jest przerzucone na kompilator

Zalety

- Uproszczony hardware
- Zapewne lepsze przewidywania co do wyboru gałęzi kodu
- Jeśli mamy dobry kompilator to powinniśmy mieć lepsze rezulaty niż dla potokowania
- Niezgodność z innymi architekturami, złożone kompilatory
- Czasem instrukcje są w znacznej mierze puste

Very Long Instruction Word

Bardzo długie instrukcje

- W jednej instrukcji kodujemy wiele atomowych operacji które procesor wykona równolegle
- Przykład (arch. SHARC - Super Harvard Architecture Single-Chip Computer)

```
f12=f0*f4, f8=f8+f12, f0=dm(i0,m3), f4=pm(i8,m9);
```

- Zbiór możliwych instrukcji zależy od liczby jednostek wykonawczych
- Rozwiązywanie problemów związanych z zrównoleglaniem kodu, wyborem które instrukcje będą wykonywane jako następne (if-y) itp jest przerzucone na kompilator

Zalety

- Uproszczony hardware
- Zapewne lepsze przewidywania co do wyboru gałęzi kodu
- Jeśli mamy dobry kompilator to powinniśmy mieć lepsze rezulaty niż dla potokowania
- Niezgodność z innymi architekturami, złożone kompilatory
- Czasem instrukcje są w znacznej mierze puste

Very Long Instruction Word

Bardzo długie instrukcje

- W jednej instrukcji kodujemy wiele atomowych operacji które procesor wykona równolegle
- Przykład (arch. SHARC - Super Harvard Architecture Single-Chip Computer)

```
f12=f0*f4, f8=f8+f12, f0=dm(i0,m3), f4=pm(i8,m9);
```

- Zbiór możliwych instrukcji zależy od liczby jednostek wykonawczych
- Rozwiązywanie problemów związanych z zrównoleglaniem kodu, wyborem które instrukcje będą wykonywane jako następne (if-y) itp jest przerzucone na kompilator

Zalety i wady

- Uproszczony hardware
- Zapewne lepsze przewidywania co do wyboru gałęzi kodu
- Jeśli mamy dobry kompilator to powinniśmy mieć lepsze rezulaty niż dla potokowania
- Niezgodność z innymi architekturami, złożone kompilatory
- Czasem instrukcje są w znacznej mierze puste

Very Long Instruction Word

Bardzo długie instrukcje

- W jednej instrukcji kodujemy wiele atomowych operacji które procesor wykona równolegle
- Przykład (arch. SHARC - Super Harvard Architecture Single-Chip Computer)

```
f12=f0*f4, f8=f8+f12, f0=dm(i0,m3), f4=pm(i8,m9);
```

- Zbiór możliwych instrukcji zależy od liczby jednostek wykonawczych
- Rozwiązywanie problemów związanych z zrównoleglaniem kodu, wyborem które instrukcje będą wykonywane jako następne (if-y) itp jest przerzucone na kompilator

Zalety i wady

- Uproszczony hardware
- Zapewne lepsze przewidywania co do wyboru gałęzi kodu
- Jeśli mamy dobry kompilator to powinniśmy mieć lepsze rezulaty niż dla potokowania
- Niezgodność z innymi architekturami, złożone kompilatory
- Czasem instrukcje są w znacznej mierze puste

Wieloprocesorowosc, wielordzeniowosc

Wieloprocesorowość

- Komputery z wieloma procesorami
- Głównie superkomputery o naprawdę dużej liczbie procesorów i specjalnej budowie
- Specjalne SO
- Wieloprocesorowość jest raczej rzadko spotykana w komputerach klasy PC

Wielordzeniowość

- Dość nowe rozwiązanie
- Projektowane dla komputerów klasy PC
- Obecnie powszechnie stosowane
- Popularne SO muszą je obsługiwać aby pozostać popularnymi

Wieloprocesorowosc, wielordzeniowosc

Wieloprocesorowość

- Komputery z wieloma procesorami
- Głównie superkomputery o naprawdę dużej liczbie procesorów i specjalnej budowie
- Specjalne SO
- Wieloprocesorowość jest raczej rzadko spotykana w komputerach klasy PC

Wielordzeniowość

- Dość nowe rozwiązanie
- Projektowane dla komputerów klasy PC
- Obecnie powszechnie stosowane
- Popularne SO muszą je obsługiwać aby pozostać popularnymi

Wieloprocesorowosc, wielordzeniowosc

Wieloprocesorowość

- Komputery z wieloma procesorami
- Głównie superkomputery o naprawdę dużej liczbie procesorów i specjalnej budowie
- Specjalne SO
- Wieloprocesorowość jest raczej rzadko spotykana w komputerach klasy PC

Wielordzeniowość

- Dość nowe rozwiązanie
- Projektowane dla komputerów klasy PC
- Obecnie powszechnie stosowane
- Popularne SO muszą je obsługiwać aby pozostać popularnymi

Dlaczego wiele rdzeni?

Problemy z jednordzeniowymi procesorami

- Prawo Moore'a
- Granice możliwości technicznych

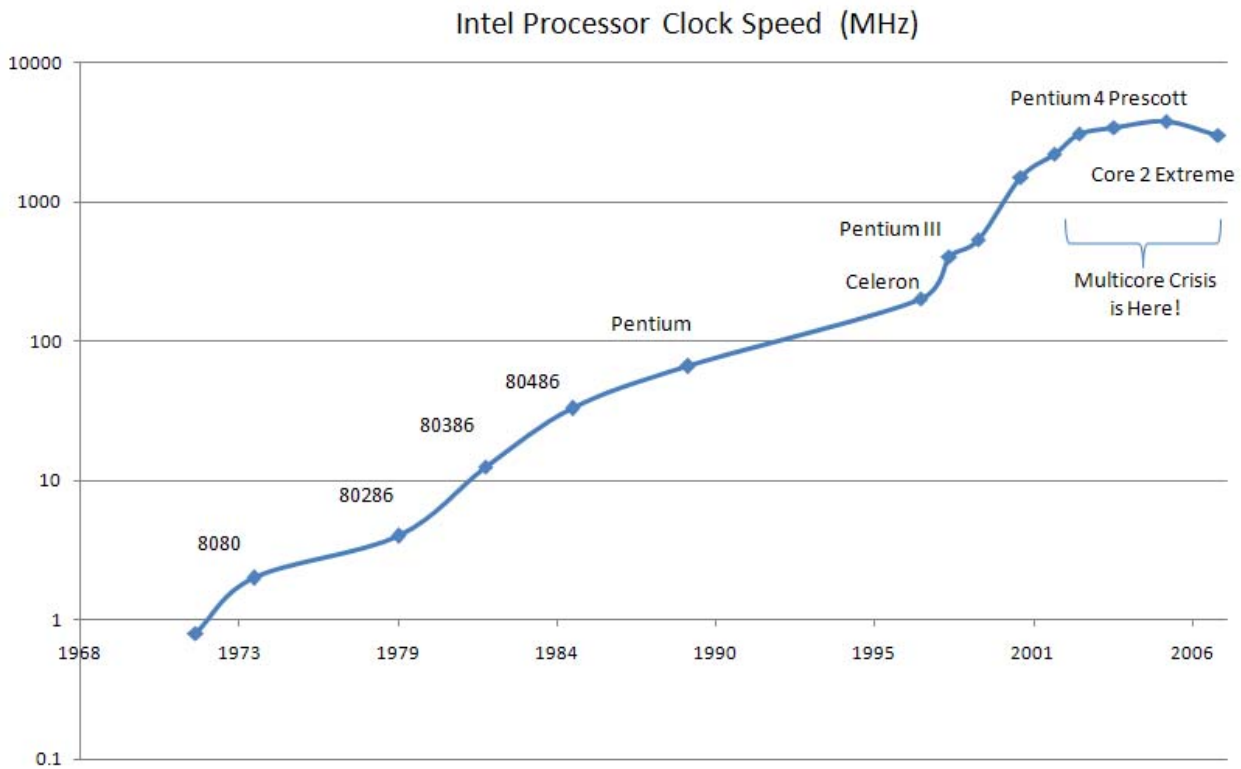
Wiele rdzeni

- Mniejsze wydzielanie ciepła i zużycie energii
- Teoretycznie n-razy szybszy procesor przy tej samej prędkości rdzenia
- Lepsze od kilku procesorów bo wspólny cache

Problemy

- Mało programów równoległych
- „Often it's cheaper to buy proper computer than to rewrite software"
- Zapychanie pasma dostępu do pamięci i urządzeń (np karty sieciowej)

Dlaczego wiele rdzeni?



Prędkość pojedyńczego procesora/rdzenia w funkcji czasu

Dlaczego wiele rdzeni?

Problemy z jednordzeniowymi procesorami

- Prawo Moore'a
- Granice możliwości technicznych

Wiele rdzeni

- Mniejsze wydzielanie ciepła i zużycie energii
- Teoretycznie n-razy szybszy procesor przy tej samej prędkości rdzenia
- Lepsze od kilku procesorów bo wspólny cache

Problemy

- Mało programów równoległych
- „Often it's cheaper to buy proper computer than to rewrite software"
- Zapychanie pasma dostępu do pamięci i urządzeń (np karty sieciowej)

Dlaczego wiele rdzeni?

Problemy z jednordzeniowymi procesorami

- Prawo Moore'a
- Granice możliwości technicznych

Wiele rdzeni

- Mniejsze wydzielanie ciepła i zużycie energii
- Teoretycznie n-razy szybszy procesor przy tej samej prędkości rdzenia
- Lepsze od kilku procesorów bo wspólny cache

Problemy

- Mało programów równoległych
- „Often it's cheaper to buy proper computer than to rewrite software"
- Zapychanie pasma dostępu do pamięci i urządzeń (np karty sieciowej)

Dlaczego wiele rdzeni?

Problemy z jednordzeniowymi procesorami

- Prawo Moore'a
- Granice możliwości technicznych

Wiele rdzeni

- Mniejsze wydzielanie ciepła i zużycie energii
- Teoretycznie n-razy szybszy procesor przy tej samej prędkości rdzenia
- Lepsze od kilku procesorów bo wspólny cache

Problemy

- Mało programów równoległych
- „Often it's cheaper to buy proper computer than to rewrite software"
- Zapychanie pasma dostępu do pamięci i urządzeń (np karty sieciowej)

Dlaczego wiele rdzeni?

Problemy z jednordzeniowymi procesorami

- Prawo Moore'a
- Granice możliwości technicznych

Wiele rdzeni

- Mniejsze wydzielanie ciepła i zużycie energii
- Teoretycznie n-razy szybszy procesor przy tej samej prędkości rdzenia
- Lepsze od kilku procesorów bo wspólny cache

Problemy

- Mało programów równoległych
- „Often it's cheaper to buy proper computer than to rewrite software"
- Zapychanie pasma dostępu do pamięci i urządzeń (np karty sieciowej)

Architektura asynchroniczna (1970-80)

AMP,ASMP,Master - Slave

- Każdy procesor jest traktowany niezależnie
- Procesory mogą być różnego przeznaczenia, różnie taktowane
- Często mają dostęp do różnych urządzeń preferyjnych
- Procesy muszą specyfikować na którym procesorze mają pracować
- Łatwo o wąskie gardło, nierównomierne rozłożenie obciążenia
- Master - Slave czyli jeden główny procesor z SO, pozostałe procesory do szybkich obliczeń

Współczesne przykłady

- Procesor Cell(PS3 ale nie tylko...)
- GPU

Architektura asynchroniczna (1970-80)

AMP,ASMP,Master - Slave

- Każdy procesor jest traktowany niezależnie
- Procesory mogą być różnego przeznaczenia, różnie taktowane
- Często mają dostęp do różnych urządzeń preferyjnych
- Procesy muszą specyfikować na którym procesorze mają pracować
- Łatwo o wąskie gardło, nierównomierne rozłożenie obciążenia
- Master - Slave czyli jeden główny procesor z SO, pozostałe procesory do szybkich obliczeń

Współczesne przykłady

- Procesor Cell(PS3 ale nie tylko...)
- GPU

Architektura asynchroniczna (1970-80)

AMP,ASMP,Master - Slave

- Każdy procesor jest traktowany niezależnie
- Procesory mogą być różnego przeznaczenia, różnie taktowane
- Często mają dostęp do różnych urządzeń preferyjnych
- Procesy muszą specyfikować na którym procesorze mają pracować
- Łatwo o wąskie gardło, nierównomierne rozłożenie obciążenia
- Master - Slave czyli jeden główny procesor z SO, pozostałe procesory do szybkich obliczeń

Współczesne przykłady

- Procesor Cell(PS3 ale nie tylko...)
- GPU

Architektura asynchroniczna (1970-80)

AMP,ASMP,Master - Slave

- Każdy procesor jest traktowany niezależnie
- Procesory mogą być różnego przeznaczenia, różnie taktowane
- Często mają dostęp do różnych urządzeń preferyjnych
- Procesy muszą specyfikować na którym procesorze mają pracować
- Łatwo o wąskie gardło, nierównomierne rozłożenie obciążenia
- Master - Slave czyli jeden główny procesor z SO, pozostałe procesory do szybkich obliczeń

Współczesne przykłady

- Procesor Cell(PS3 ale nie tylko...)
- GPU

Architektura asynchroniczna (1970-80)

AMP,ASMP,Master - Slave

- Każdy procesor jest traktowany niezależnie
- Procesory mogą być różnego przeznaczenia, różnie taktowane
- Często mają dostęp do różnych urządzeń preferyjnych
- Procesy muszą specyfikować na którym procesorze mają pracować
- Łatwo o wąskie gardło, nierównomierne rozłożenie obciążenia
- Master - Slave czyli jeden główny procesor z SO, pozostałe procesory do szybkich obliczeń

Współczesne przykłady

- Procesor Cell(PS3 ale nie tylko...)
- GPU

Architektura asynchroniczna (1970-80)

AMP,ASMP,Master - Slave

- Każdy procesor jest traktowany niezależnie
- Procesory mogą być różnego przeznaczenia, różnie taktowane
- Często mają dostęp do różnych urządzeń preferyjnych
- Procesy muszą specyfikować na którym procesorze mają pracować
- Łatwo o wąskie gardło, nierównomierne rozłożenie obciążenia
- Master - Slave czyli jeden główny procesor z SO, pozostałe procesory do szybkich obliczeń

Współczesne przykłady

- Procesor Cell(PS3 ale nie tylko...)
- GPU

SMP,BMP

SMP

- Wszystkie procesory traktowane równorzędnie
- Procesy mogą swobodnie migrować między procesorami
- SO powinien zapewniać równomierne obciążenie procesorów
- Obecnie najczęściej stosowane
- W linuxie od 1.2.33

Cache

- Poważnym problemem technicznym jest synchronizacja cache
- Zazwyczaj zapewniana sprzętowo, ale ...
- BMP - może jednak mnie wolności?

SMP,BMP

SMP

- Wszystkie procesory traktowane równorzędnie
- Procesy mogą swobodnie migrować między procesorami
- SO powinien zapewniać równomierne obciążenie procesorów
- Obecnie najczęściej stosowane
- W linuxie od 1.2.33

Cache

- Poważnym problemem technicznym jest synchronizacja cache
- Zazwyczaj zapewniana sprzętowo, ale ...
- BMP - może jednak mnie wolności?

SMP,BMP

SMP

- Wszystkie procesory traktowane równorzędnie
- Procesy mogą swobodnie migrować między procesorami
- SO powinien zapewniać równomierne obciążenie procesorów
- Obecnie najczęściej stosowane
- W linuxie od 1.2.33

Cache

- Poważnym problemem technicznym jest synchronizacja cache
- Zazwyczaj zapewniana sprzętowo, ale ...
- BMP - może jednak mnie wolności?

SMP,BMP

SMP

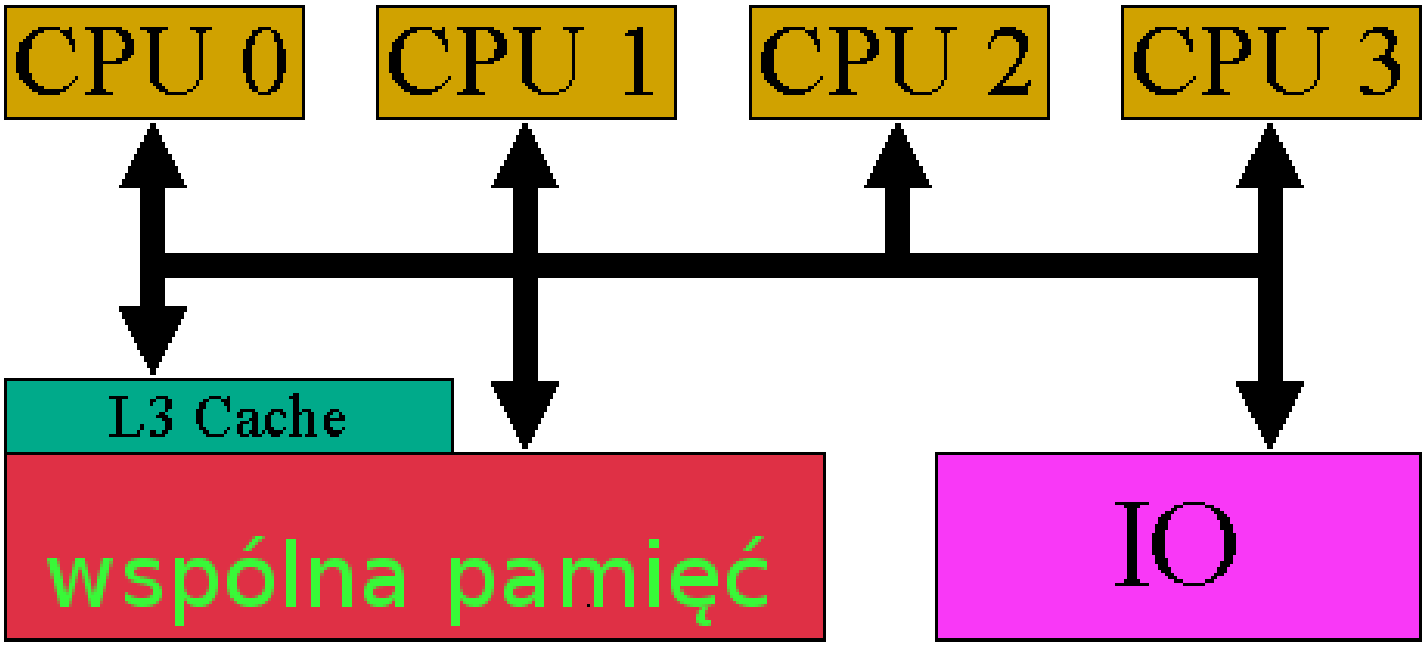
- Wszystkie procesory traktowane równorzędnie
- Procesy mogą swobodnie migrować między procesorami
- SO powinien zapewniać równomierne obciążenie procesorów
- Obecnie najczęściej stosowane
- W linuxie od 1.2.33

Cache

- Poważnym problemem technicznym jest synchronizacja cache
- Zazwyczaj zapewniana sprzętowo, ale ...
- BMP - może jednak mnie wolności?

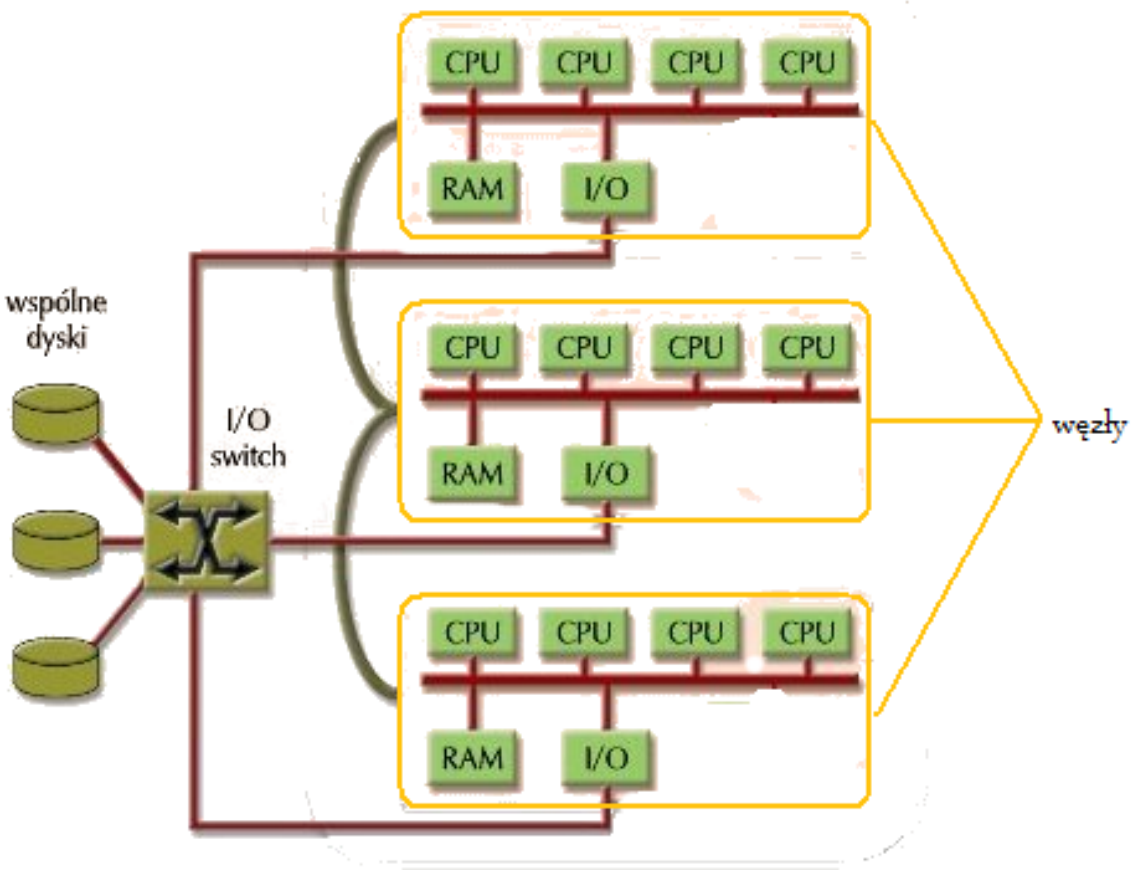
SMP

UMA, czyli Uniform Memory Access



NUMA, ccNUMA

Non-Uniform Memory Access lub Non-Uniform Memory Architecture, czasami też DSM - Distributed Shared Memory



Obecnie w praktyce ccNUMA.

Utrzymywanie spójności pamięci podręcznych

Źródła kłopotów

- lokalne pamięci podręczne
- buforowanie operacji pisania

Modele spójności - podobnie jak na BD

- Spójność ścisła
- Spójność sekwencyjna
- Spójność słaba, na wyjściu, na wejściu

Implementacja

- Algorytm scentralizowanego zarządcy
- Dynamiczny algorytm zarządcy rozproszonego
- Jawne blokowanie zasobów
- Linda

Utrzymywanie spójności pamięci podręcznych

Źródła kłopotów

- lokalne pamięci podręczne
- buforowanie operacji pisania

Modele spójności - podobnie jak na BD

- Spójność ścisła
- Spójność sekwencyjna
- Spójność słaba, na wyjściu, na wejściu

Implementacja

- Algorytm scentralizowanego zarządcy
- Dynamiczny algorytm zarządcy rozproszonego
- Jawne blokowanie zasobów
- Linda

Utrzymywanie spójności pamięci podręcznych

Źródła kłopotów

- lokalne pamięci podręczne
- buforowanie operacji pisania

Modele spójności - podobnie jak na BD

- Spójność ścisła
- Spójność sekwencyjna
- Spójność słaba, na wyjściu, na wejściu

Implementacja

- Algorytm scentralizowanego zarządcy
- Dynamiczny algorytm zarządcy rozproszonego
- Jawne blokowanie zasobów
- Linda

Plan Prezentacji

Przykre wspomnienia z PW

Filozofia

- Problem pięciu filozofów (raczej rzadko spotykany w SO)
- Wyłączność dostępu do zasobów
- Problem czytelników i pisarzy
- Zapewnienie „normalnej“ semantyki programów na wielu procesorach

„Normalna“ semantyka

```
x = 5;
x = x + 1;
if (x!=6)printf("Spokojnie, to tylko PW");
```

Przykre wspomnienia z PW

Filozofia

- Problem pięciu filozofów (raczej rzadko spotykany w SO)
- Wyłączność dostępu do zasobów
- Problem czytelników i pisarzy
- Zapewnienie „normalnej“ semantyki programów na wielu procesorach

„Normalna“ semantyka

```
x = 5;
x = x + 1;
if (x!=6)printf("Spokojnie, to tylko PW");
```

Przykre wspomnienia z PW

Filozofia

- Problem pięciu filozofów (raczej rzadko spotykany w SO)
- Wyłączność dostępu do zasobów
- Problem czytelników i pisarzy
- Zapewnienie „normalnej“ semantyki programów na wielu procesorach

„Normalna“ semantyka

```
x = 5;
x = x + 1;
if (x!=6)printf("Spokojnie, to tylko PW");
```

Rozwiązania w linuxie

Znane nam rozwiązania w linuxie

- `atomic_t, atomic_read(v),atomic_set(v, i),atomic_dec_and_test(v) ...`
- blokowanie przerwań
- `spinlock_t`
- `rw_lock_t`
- semafory($< 2.6.24$), mutexy($> 2.6.16$) – „my" jesteśmy akurat pomiędzy

Rozwiązania w linuxie

Znane nam rozwiązania w linuxie

- `atomic_t, atomic_read(v),atomic_set(v, i),atomic_dec_and_test(v) ...`
- blokowanie przerwań
- `spinlock_t`
- `rw_lock_t`
- semafory($< 2.6.24$), mutexy($> 2.6.16$) - „my" jesteśmy akurat pomiędzy

Rozwiązania w linuxie

Znane nam rozwiązania w linuxie

- `atomic_t, atomic_read(v),atomic_set(v, i),atomic_dec_and_test(v) ...`
- blokowanie przerwań
- `spinlock_t`
- `rw_lock_t`
- semafory($< 2.6.24$), mutexy($> 2.6.16$) - „my" jesteśmy akurat pomiędzy

Rozwiązania w linuxie

Znane nam rozwiązania w linuxie

- `atomic_t, atomic_read(v),atomic_set(v, i),atomic_dec_and_test(v) ...`
- blokowanie przerwań
- `spinlock_t`
- `rw_lock_t`
- semafory($< 2.6.24$), mutexy($> 2.6.16$) - „my“ jesteśmy akurat pomiędzy

Rozwiązania w linuxie

Znane nam rozwiązania w linuxie

- `atomic_t, atomic_read(v),atomic_set(v, i),atomic_dec_and_test(v) ...`
- blokowanie przerwań
- `spinlock_t`
- `rw_lock_t`
- semafory($< 2.6.24$), mutexy($> 2.6.16$) - „my“ jesteśmy akurat pomiędzy

Rozwiązania w Windows

Porady dla twórców sterowników dla Windows

- Różne poziomy `IRQL`
- słowo kluczowe `volatile` - mało skuteczne, semantyka nie jest dokładnie wyspecyfikowana
- operacje atomowe, analogicznie jak w linuxie
- `InterlockedXxx`(szybkie) i `ExInterlockedXxx`
- Spinlocki
- `Mutex, fast mutex, and executive resource`

Rozwiązania w Windows

Porady dla twórców sterowników dla Windows

- Różne poziomy `IRQL`
- słowo kluczowe `volatile` - mało skuteczne, semantyka nie jest dokładnie wyspecyfikowana
- operacje atomowe, analogicznie jak w linuxie
- `InterlockedXxx`(szybkie) i `ExInterlockedXxx`
- Spinlocki
- `Mutex, fast mutex, and executive resource`

Rozwiązania w Windows

Porady dla twórców sterowników dla Windows

- Różne poziomy `IRQL`
- słowo kluczowe `volatile` - mało skuteczne, semantyka nie jest dokładnie wyspecyfikowana
- operacje atomowe, analogicznie jak w linuxie
- `InterlockedXxx`(szybkie) i `ExInterlockedXxx`
- Spinlocki
- `Mutex, fast mutex, and executive resource`

Rozwiązania w Windows

Porady dla twórców sterowników dla Windows

- Różne poziomy `IRQL`
- słowo kluczowe `volatile` - mało skuteczne, semantyka nie jest dokładnie wyspecyfikowana
- operacje atomowe, analogicznie jak w linuxie
- `InterlockedXxx`(szybkie) i `ExInterlockedXxx`
- Spinlocki
- `Mutex, fast mutex, and executive resource`

Rozwiązania w Windows

Porady dla twórców sterowników dla Windows

- Różne poziomy `IRQL`
- słowo kluczowe `volatile` - mało skuteczne, semantyka nie jest dokładnie wyspecyfikowana
- operacje atomowe, analogicznie jak w linuxie
- `InterlockedXxx`(szybkie) i `ExInterlockedXxx`
- Spinlocki
- `Mutex, fast mutex, and executive resource`

Rozwiązania w Windows

Porady dla twórców sterowników dla Windows

- Różne poziomy `IRQL`
- słowo kluczowe `volatile` - mało skuteczne, semantyka nie jest dokładnie wyspecyfikowana
- operacje atomowe, analogicznie jak w linuxie
- `InterlockedXxx`(szybkie) i `ExInterlockedXxx`
- Spinlocki
- `Mutex, fast mutex, and executive resource`

Scheduler w linuxie - stary i nowy

Stary(jądra 2.4.x)

- analogiczny jak dla jednego procesora
- jest jedna globalna kolejka procesów gotowych(wtedy był jeszcze Big Kernel Lock)
- wolny procesor bierze najlepszy proces z kolejki
- brak jakichkolwiek rozwiązań ustalających optymalne rozłożenie procesów

Nowy(jądra 2.6.x)

- osobna kolejka dla kazdego procesora
- Load balancing co 200ms(albo 1ms)
- jak dobierac procesy miedzy procesorami?
- pamiętajmy o potrzebie synchronizacji cache

Scheduler w linuxie - stary i nowy

Stary(jądra 2.4.x)

- analogiczny jak dla jednego procesora
- jest jedna globalna kolejka procesów gotowych(wtedy był jeszcze Big Kernel Lock)
- wolny procesor bierze najlepszy proces z kolejki
- brak jakichkolwiek rozwiązań ustalających optymalne rozłożenie procesów

Nowy(jądra 2.6.x)

- osobna kolejka dla kazdego procesora
- Load balancing co 200ms(albo 1ms)
- jak dobierac procesy miedzy procesorami?
- pamiętajmy o potrzebie synchronizacji cache

Scheduler w linuxie - stary i nowy

Stary(jądra 2.4.x)

- analogiczny jak dla jednego procesora
- jest jedna globalna kolejka procesów gotowych(wtedy był jeszcze Big Kernel Lock)
- wolny procesor bierze najlepszy proces z kolejki
- brak jakichkolwiek rozwiązań ustalających optymalne rozłożenie procesów

Nowy(jądra 2.6.x)

- osobna kolejka dla kazdego procesora
- Load balancing co 200ms(albo 1ms)
- jak dobierac procesy miedzy procesorami?
- pamiętajmy o potrzebie synchronizacji cache

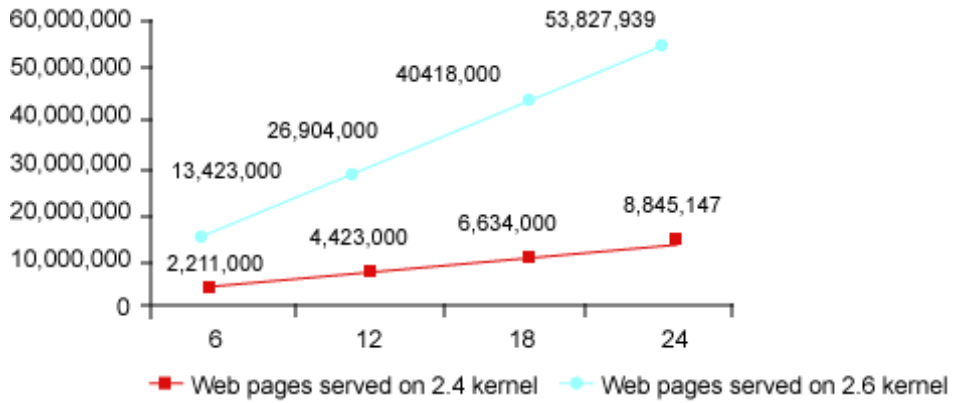
Scheduler w linuxie - stary i nowy

Stary(jądra 2.4.x)

- analogiczny jak dla jednego procesora
- jest jedna globalna kolejka procesów gotowych(wtedy był jeszcze Big Kernel Lock)
- wolny procesor bierze najlepszy proces z kolejki
- brak jakichkolwiek rozwiązań ustalających optymalne rozłożenie procesów

Nowy(jądra 2.6.x)

- osobna kolejka dla kazdego procesora
- Load balancing co 200ms(albo 1ms)
- jak dobierac procesy miedzy procesorami?
- pamiętajmy o potrzebie synchronizacji cache



Load balancing

Procedura wyrównywania obciążenia procesorów

- wyszukanie najbardziej zatłoczonej kolejki
- wybór kolejki expired lub active
- wybór procesu o najmniejszej wartości priorytetu który nie jest wykonywany i nie ma danych w cache'u
- przeniesienie procesu
- i tak dalej aż te dwie kolejki nie bedą zbalansowane

Load balancing

Procedura wyrównywania obciążenia procesorów

- wyszukanie najbardziej zatłoczonej kolejki
- wybór kolejki expired lub active
- wybór procesu o najmniejszej wartości priorytetu który nie jest wykonywany i nie ma danych w cache'u
- przeniesienie procesu
- i tak dalej aż te dwie kolejki nie bedą zbalansowane

Load balancing

Procedura wyrównywania obciążenia procesorów

- wyszukanie najbardziej zatłoczonej kolejki
- wybór kolejki expired lub active
- wybór procesu o najmniejszej wartości priorytetu który nie jest wykonywany i nie ma danych w cache'u
- przeniesienie procesu
- i tak dalej aż te dwie kolejki nie bedą zbalansowane

Load balancing

Procedura wyrównywania obciążenia procesorów

- wyszukanie najbardziej zatłoczonej kolejki
- wybór kolejki expired lub active
- wybór procesu o najmniejszej wartości priorytetu który nie jest wykonywany i nie ma danych w cache'u
- przeniesienie procesu
- i tak dalej aż te dwie kolejki nie bedą zbalansowane

Load balancing

Procedura wyrównywania obciążenia procesorów

- wyszukanie najbardziej zatłoczonej kolejki
- wybór kolejki expired lub active
- wybór procesu o najmniejszej wartości priorytetu który nie jest wykonywany i nie ma danych w cache'u
- przeniesienie procesu
- i tak dalej aż te dwie kolejki nie bedą zbalansowane

Load balancing

Procedura wyrównywania obciążenia procesorów

- wyszukanie najbardziej zatłoczonej kolejki
- wybór kolejki expired lub active
- wybór procesu o najmniejszej wartości priorytetu który nie jest wykonywany i nie ma danych w cache'u
- przeniesienie procesu
- i tak dalej aż te dwie kolejki nie bedą zbalansowane

Ogólne wymagania

Prawo Amdahl'a

$$W = \frac{1}{(1-P)+\frac{P}{S}}$$

- W - czas wykonania programu z wieloma procesorami w stosunku do czasu na pojedynczym procesorze
- P - czesc programu którą można wykonać równolegle
- S - liczba procesorów logicznych do dyspozycji
- Zakładamy, że nasze procesory logiczne są takiej wydajności jak ten jeden pierwotny

Inne ograniczenia

- Dostęp do szyny danych
- Dostęp do pamięci
- Dostęp do struktur jądra
- Dostęp do dysku, sieci itd.
- W praktyce dobrze jeśli dla dwóch rdzeni mamy 60% szybsze wykonanie

Ogólne wymagania

Prawo Amdahl'a

$$W = \frac{1}{(1-P)+\frac{P}{S}}$$

- W - czas wykonania programu z wieloma procesorami w stosunku do czasu na pojedynczym procesorze
- P - czesc programu którą można wykonać równolegle
- S - liczba procesorów logicznych do dyspozycji
- Zakładamy, że nasze procesory logiczne są takiej wydajności jak ten jeden pierwotny

Inne ograniczenia

- Dostęp do szyny danych
- Dostęp do pamięci
- Dostęp do struktur jądra
- Dostęp do dysku, sieci itd.
- W praktyce dobrze jeśli dla dwóch rdzeni mamy 60% szybsze wykonanie

Ogólne wymagania

Prawo Amdahl'a

$$W = \frac{1}{(1-P)+\frac{P}{S}}$$

- W - czas wykonania programu z wieloma procesorami w stosunku do czasu na pojedynczym procesorze
- P - czesc programu którą można wykonać równolegle
- S - liczba procesorów logicznych do dyspozycji
- Zakładamy, że nasze procesory logiczne są takiej wydajności jak ten jeden pierwotny

Inne ograniczenia

- Dostęp do szyny danych
- Dostęp do pamięci
- Dostęp do struktur jądra
- Dostęp do dysku, sieci itd.
- W praktyce dobrze jeśli dla dwóch rdzeni mamy 60% szybsze wykonanie

Ogólne wymagania

Prawo Amdahl'a

$$W = \frac{1}{(1-P)+\frac{P}{S}}$$

- W - czas wykonania programu z wieloma procesorami w stosunku do czasu na pojedynczym procesorze
- P - czesc programu którą można wykonać równolegle
- S - liczba procesorów logicznych do dyspozycji
- Zakładamy, że nasze procesory logiczne są takiej wydajności jak ten jeden pierwotny

Inne ograniczenia

- Dostęp do szyny danych
- Dostęp do pamięci
- Dostęp do struktur jądra
- Dostęp do dysku, sieci itd.
- W praktyce dobrze jeśli dla dwóch rdzeni mamy 60% szybsze wykonanie

Ogólne wymagania

Prawo Amdahl'a

$$W = \frac{1}{(1-P)+\frac{P}{S}}$$

- W - czas wykonania programu z wieloma procesorami w stosunku do czasu na pojedynczym procesorze
- P - czesc programu którą można wykonać równolegle
- S - liczba procesorów logicznych do dyspozycji
- Zakładamy, że nasze procesory logiczne są takiej wydajności jak ten jeden pierwotny

Inne ograniczenia

- Dostęp do szyny danych
- Dostęp do pamięci
- Dostęp do struktur jądra
- Dostęp do dysku, sieci itd.
- W praktyce dobrze jeśli dla dwóch rdzeni mamy 60% szybsze wykonanie

Grupy aplikacji przystosowanych do wielowątkowości...

Aplikacje biznesowe

- serwery WWW i serwery aplikacji, np Apache,Tomcat, JBoss...
- SAPy
- Bazy danych

Naukowe

- SETI, fizyka cząstek elementarnych
- alg. genetyczne (AI), Blast

Inne

- Łamanie haseł, zwłaszcza brute-force
- ...

Grupy aplikacji przystosowanych do wielowątkowości...

Aplikacje biznesowe

- serwery WWW i serwery aplikacji, np Apache,Tomcat, JBoss...
- SAPy
- Bazy danych

Naukowe

- SETI, fizyka cząstek elementarnych
- alg. genetyczne (AI), Blast

Inne

- Łamanie haseł, zwłaszcza brute-force
- ...

Grupy aplikacji przystosowanych do wielowątkowości...

Aplikacje biznesowe

- serwery WWW i serwery aplikacji, np Apache,Tomcat, JBoss...
- SAPy
- Bazy danych

Naukowe

- SETI, fizyka cząstek elementarnych
- alg. genetyczne (AI), Blast

Inne

- Łamanie haseł, zwłaszcza brute-force
- ...

Grupy aplikacji przystosowanych do wielowątkowości...

Aplikacje biznesowe

- serwery WWW i serwery aplikacji, np Apache,Tomcat, JBoss...
- SAPy
- Bazy danych

Naukowe

- SETI, fizyka cząstek elementarnych
- alg. genetyczne (AI), Blast

Inne

- Łamanie haseł, zwłaszcza brute-force
- ...

... i inne typy aplikacji które też chcą skorzystać z nowych możliwości

Jeszcze inne :)...

- Bioshock, UT3 , Call Of Duty 4, Company of Heroes ,Crysis ....

... i inne typy aplikacji które też chcą skorzystać z nowych możliwości

Jeszcze inne :)...

- Bioshock, UT3 , Call Of Duty 4, Company of Heroes ,Crysis ....

SISD,SIMD,MISD,MIMD

Podział architektur ze względu na liczbę potoków

- SISD(Single Instruction, Single Data) - komputery z jednym procesorem logicznym
- MISD(Multiple Instruction, Single Data) - rzadko spotykane
- SIMD(Single Instruction, Multiple Data) - jednostki wektorowe, GPU
- MIMD(Multiple Instruction, Multiple Data) - komputery z wieloma procesorami, systemy rozproszone

Obecnie ten podział traci na znaczeniu

- Prawie każdy komputer jest MIMD
- Prawie każdy komputer zawiera jakieś elementy typu SIMD

SISD,SIMD,MISD,MIMD

Podział architektur ze względu na liczbę potoków

- SISD(Single Instruction, Single Data) - komputery z jednym procesorem logicznym
- MISD(Multiple Instruction, Single Data) - rzadko spotykane
- SIMD(Single Instruction, Multiple Data) - jednostki wektorowe, GPU
- MIMD(Multiple Instruction, Multiple Data) - komputery z wieloma procesorami, systemy rozproszone

Obecnie ten podział traci na znaczeniu

- Prawie każdy komputer jest MIMD
- Prawie każdy komputer zawiera jakieś elementy typu SIMD

SISD,SIMD,MISD,MIMD

Podział architektur ze względu na liczbę potoków

- SISD(Single Instruction, Single Data) - komputery z jednym procesorem logicznym
- MISD(Multiple Instruction, Single Data) - rzadko spotykane
- SIMD(Single Instruction, Multiple Data) - jednostki wektorowe, GPU
- MIMD(Multiple Instruction, Multiple Data) - komputery z wieloma procesorami, systemy rozproszone

Obecnie ten podział traci na znaczeniu

- Prawie każdy komputer jest MIMD
- Prawie każdy komputer zawiera jakieś elementy typu SIMD

SISD,SIMD,MISD,MIMD

Podział architektur ze względu na liczbę potoków

- SISD(Single Instruction, Single Data) - komputery z jednym procesorem logicznym
- MISD(Multiple Instruction, Single Data) - rzadko spotykane
- SIMD(Single Instruction, Multiple Data) - jednostki wektorowe, GPU
- MIMD(Multiple Instruction, Multiple Data) - komputery z wieloma procesorami, systemy rozproszone

Obecnie ten podział traci na znaczeniu

- Prawie każdy komputer jest MIMD
- Prawie każdy komputer zawiera jakieś elementy typu SIMD

SISD,SIMD,MISD,MIMD

Podział architektur ze względu na liczbę potoków

- SISD(Single Instruction, Single Data) - komputery z jednym procesorem logicznym
- MISD(Multiple Instruction, Single Data) - rzadko spotykane
- SIMD(Single Instruction, Multiple Data) - jednostki wektorowe, GPU
- MIMD(Multiple Instruction, Multiple Data) - komputery z wieloma procesorami, systemy rozproszone

Obecnie ten podział traci na znaczeniu

- Prawie każdy komputer jest MIMD
- Prawie każdy komputer zawiera jakieś elementy typu SIMD

SISD,SIMD,MISD,MIMD

Podział architektur ze względu na liczbę potoków

- SISD(Single Instruction, Single Data) - komputery z jednym procesorem logicznym
- MISD(Multiple Instruction, Single Data) - rzadko spotykane
- SIMD(Single Instruction, Multiple Data) - jednostki wektorowe, GPU
- MIMD(Multiple Instruction, Multiple Data) - komputery z wieloma procesorami, systemy rozproszone

Obecnie ten podział traci na znaczeniu

- Prawie każdy komputer jest MIMD
- Prawie każdy komputer zawiera jakieś elementy typu SIMD

SISD,SIMD,MISD,MIMD

Podział architektur ze względu na liczbę potoków

- SISD(Single Instruction, Single Data) - komputery z jednym procesorem logicznym
- MISD(Multiple Instruction, Single Data) - rzadko spotykane
- SIMD(Single Instruction, Multiple Data) - jednostki wektorowe, GPU
- MIMD(Multiple Instruction, Multiple Data) - komputery z wieloma procesorami, systemy rozproszone

Obecnie ten podział traci na znaczeniu

- Prawie każdy komputer jest MIMD
- Prawie każdy komputer zawiera jakieś elementy typu SIMD

SISD,SIMD,MISD,MIMD

Podział architektur ze względu na liczbę potoków

- SISD(Single Instruction, Single Data) - komputery z jednym procesorem logicznym
- MISD(Multiple Instruction, Single Data) - rzadko spotykane
- SIMD(Single Instruction, Multiple Data) - jednostki wektorowe, GPU
- MIMD(Multiple Instruction, Multiple Data) - komputery z wieloma procesorami, systemy rozproszone

Obecnie ten podział traci na znaczeniu

- Prawie każdy komputer jest MIMD
- Prawie każdy komputer zawiera jakieś elementy typu SIMD

Systemy rozproszone

Systemy rozproszone

- Sieciowy system operacyjny - współdzielone pliki
- Rozproszony system operacyjny (wiele komputerów) - komunikaty
- Rozproszony system operacyjny (wieloprocesorowy) - współdzielona pamięć
- Ciąg dalszy nastąpi...

Systemy rozproszone

Systemy rozproszone

- Sieciowy system operacyjny - współdzielone pliki
- Rozproszony system operacyjny (wiele komputerów) - komunikaty
- Rozproszony system operacyjny (wieloprocesorowy) - współdzielona pamięć
- Ciąg dalszy nastąpi...

Systemy rozproszone

Systemy rozproszone

- Sieciowy system operacyjny - współdzielone pliki
- Rozproszony system operacyjny (wiele komputerów) - komunikaty
- Rozproszony system operacyjny (wieloprocesorowy) - współdzielona pamięć
- Ciąg dalszy nastąpi...

Systemy rozproszone

Systemy rozproszone

- Sieciowy system operacyjny - współdzielone pliki
- Rozproszony system operacyjny (wiele komputerów) - komunikaty
- Rozproszony system operacyjny (wieloprocesorowy) - współdzielona pamięć
- Ciąg dalszy nastąpi...

Systemy rozproszone

Systemy rozproszone

- Sieciowy system operacyjny - współdzielone pliki
- Rozproszony system operacyjny (wiele komputerów) - komunikaty
- Rozproszony system operacyjny (wieloprocesorowy) - współdzielona pamięć
- Ciąg dalszy nastąpi...