Architektury systemów komputerowych

Lista zadań nr 14

Na zajęcia 10 i 11 czerwca 2024

UWAGA! W trakcie prezentacji należy być gotowym do zdefiniowania pojęć oznaczonych **wytłuszczoną** czcionką.

Zadanie 1. Procesor posługuje się 32-bitowymi **adresami wirtualnymi**, rozmiar **strony** ma 4KiB, katalog tablicy stron ma 1024 wpisy, a rozmiar **wpisu tablicy stron** zajmuje 4 bajty. Dalej rozważamy proces, który łącznie używa 1GiB swojej **wirtualnej przestrzeni adresowej**. Podaj maksymalny i minimalny rozmiar tablicy stron (a) jednopoziomowej oraz (b) **dwupoziomowej**.

Zadanie 2. Czemu dodanie **TLB** (ang. *translation lookaside buffer*) do **jednostki zarządzania pamięcią** (ang. *memory management unit*) przyspiesza *translację adresów*? W jaki sposób różni się adresowanie TLB od adresowania pamięci podręcznej procesora? Kiedy ładowanie wpisu tablicy stron do TLB zakończy się wygenerowaniem wyjątku procesora? Czemu procesory używają osobnych TLB do tłumaczenia adresów instrukcji i danych? Posługując się narzędziem `cpuid`[1] wyświetl następujące informacje dot. struktury TLB swojego procesora tj.: łączna liczba wpisów, liczba wpisów w zbiorze, itd.

Wskazówka: Przed uruchomieniem programu należy załadować moduł jądra o nazwie «`cpuid`».

Zadanie 3. Zakładamy taki sam uproszczony model podsystemu pamięci jak na slajdach do wykładu „*Virtual Memory: Systems*” (strony 9–16). Powtórz proces translacji adresów i adresowania pamięci podręcznej dla adresów: `0x027c`, `0x03a9` i `0x0040` zakładając poniższy stan TLB, pamięci podręcznej i tablicy stron.

Zbiór	Tag	PPN	V	Tag	PPN	V	Tag	PPN	V	Tag	PPN	V
0	–	–	0	09	0D	1	–	–	0	07	02	1
1	03	2D	1	–	–	0	02	17	1	–	–	0
2	–	–	0	–	–	0	–	–	0	–	–	0
3	–	–	0	03	0D	1	0A	34	1	–	–	0

Zawartość TLB

Idx	Tag	V	B0	B1	B2	B3
0	19	1	99	11	23	11
1	–	0	–	–	–	–
2	1B	1	00	02	04	08
3	–	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	–	0	–	–	–	–
7	16	1	11	C2	DF	03
8	24	1	3A	00	51	89
9	–	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	–	0	–	–	–	–
C	–	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	09	1	DE	01	C4	32

Zawartość pamięci podręcznej

VPN	PPN	V
00	28	1
01	–	0
02	33	1
03	02	1
04	–	0
05	16	1
06	–	0
07	–	0
08	13	1
09	17	1
0A	09	1
0B	–	0
0C	–	0
0D	2D	1
0E	11	1
0F	0D	1

Zawartość tablicy stron

[1] `http://www.etallen.com/cpuid.html`

Zadanie 4. W tym zadaniu będziemy analizowali w jaki sposób system operacyjny musi aktualizować tablicę stron wraz z kolejnymi dostępami do pamięci głównej. Załóż, że strony są rozmiarze 4KiB, TLB jest **w pełni asocjacyjne** z zastępowaniem LRU. Najwyższa wartość pola LRU koduje najlepszego kandydata na **ofiarę** (ang. *victim*). Jeśli potrzebujesz **wtoczyć** (ang. *swap-in*) stronę z dysku użyj następnego numeru **ramki** (ang. *page frame*) większego od największego istniejącego w tablicy stron. Dla poniższych danych podaj ostateczny stan TLB i tablicy stron po wykonaniu wszystkich dostępów do pamięci. Dla każdej operacji dostępu do pamięci wskaż czy było to trafienie w TLB, trafienie w tablicę stron, czy też **błąd strony**.

VPN	Valid	PPN
00	1	05
01	0	dysk
02	0	dysk
03	1	06
04	1	09
05	1	0B
06	0	dysk
07	1	04
08	0	dysk
09	0	dysk
0A	1	03
0B	1	0C
0C	0	brak

Początkowa zawartość tablicy stron

Valid	Tag	LRU	PPN
1	0B	0	0C
1	07	1	04
1	03	2	06
0	04	3	09

Początkowa zawartość TLB

Adres
123d
08b3
365c
871b
bee6
3140
c049

Ciąg dostępów do pamięci

Zadanie 5. Na podstawie [1, §9.7.1] opisz dokładnie pola **deskryptorów stron** (ang. *page table entry*) i **deskryptorów katalogów stron** (ang. *page directory entry*) dla architektury x86-64. Jak wartość bitów CD i WT [4, §4.9] wpływa na domyślną politykę zarządzania pamięcią podręczną dla zawartości danej strony? Uzasadnij, że dla stron, które odwzorowują pamięć urządzeń wejścia-wyjścia, jądro systemu operacyjnego musi odpowiednio konfigurować te bity. Jak wartość bitów R/W, XD i U/S [4, §4.6] wpływa na uprawnienia procesora do dostępu do pamięci? Jak jądro korzysta z tych bitów do skonfigurowania wirtualnej przestrzeni adresowej procesu?

Zadanie 6. Przedstaw algorytm **zastępowania stron NRU** (ang. *Not Recently Used*) [3, §3.4.2]. W jaki sposób algorytm ten korzysta z bitów A i D [4, §4.8] należących do *wpisu tablicy stron* x86-64? Niektóre architektury nie implementują tych bitów. Jak w takim razie je zasymulować posługując się procedurami obsługi *błędu strony* i *przerwania zegarowego*.

Zadanie 7. Wyznacz maksymalny rozmiar **zbioru roboczego** procesu, dla którego nie będzie on generował nowych chybień w TLB (ang. *TLB reach*). Rozważ wariant pesymistyczny i optymistyczny dla czterodrożnego TLB o 64 wpisach. Jak zmieni się oszacowanie, jeśli zezwolimy na używanie **dużych stron** (ang. *huge pages*) o rozmiarze 4MiB? Jakie programy mogą skorzystać na używaniu dużych stron? W jaki sposób zakodować dużą stronę w hierarchicznej tablicy stron?

Wskazkówka: Duże strony w procesorach x86 wprowadzono w Pentium Pro wraz z rozszerzeniem Page Size Extension[2].

Zadanie 8. Na wykładzie przyjęliśmy, że translacja adresów jest wykonywana przed dostępem do pamięci podręcznej, co wprowadza opóźnienie w przetwarzaniu instrukcji. Taki schemat określa się mianem pamięci podręcznej **indeksowanej** i **znakowanej adresami fizycznymi** (ang. *physically-indexed, physically-tagged*). Wyjaśnij jak wykonywać równolegle dostęp do TLB i pamięci podręcznej, stosując schemat pamięci *indeksowanej wirtualnie i znakowanej fizycznie* [2, §2.4.1]. Jakie korzyści to przynosi?

Wskazówka: Posłuż się ostatnim slajdem do wykładu „*Virtual Memory: Systems*”, ale wytłumacz to szczegółowo!

[2] https://en.wikipedia.org/wiki/Page_Size_Extension

Zadanie 9. Wiemy, że pamięć podręczna TLB jest niezbędna do przeprowadzania szybkiej translacji adresów. Kiedy w systemie zachodzi potrzeba **przełączenia przestrzeni adresowej**? Czemu należałoby wtedy wyczyścić zawartość TLB? Czemu chcielibyśmy tego unikać? Jak ten problem rozwiązuje wprowadzenie **identyfikatorów przestrzeni adresowych** [2, §2.4.2] [4, §4.10.1]?

Zadanie 10 (bonus). Celem zwiększenia wydajności dostępów do pamięci architekt procesora decyduje się na użycie schematu *pamięci indeksowanej i znakowanej adresami wirtualnymi* (ang. *virtually-indexed, virtually-tagged*). Na podstawie [2, §4.2.1] wyjaśnij jak w takim przypadku może zamanifestować się problem homonimów i synonimów[3]?

Literatura

[1] „*Computer Systems: A Programmer's Perspective*"
Randal E. Bryant, David R. O'Hallaron; Pearson; 3rd edition, 2016

[2] „*Memory Systems: Cache, DRAM, Disk*"
Bruce Jacob, Spencer W. Ng, David T. Wang; *Morgan Kaufmann*, 2008

[3] „*Modern Operating Systems*"
Andrew Tanenbaum, Herbert Bos; *Pearson*, 4th edition, 2014

[4] „*Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1*"
https://software.intel.com/content/www/us/en/develop/download/intel-64-and-ia-32-architectures-sdm-volume-3a-system-programming-guide-part-1.html

[3] https://en.wikipedia.org/wiki/CPU_cache#Homonym_and_synonym_problems