Lista 14

Sunday, 11 June 2023 11:20

as23_lista_14

Architektury systemów komputerowych

Lista zadań nr 14

Na zajęcia 14 i 20 czerwca 2023

UWAGA! W trakcie prezentacji należy być gotowym do zaliczeniowego pytania **wylosowanego** zaciernią.

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. Dane rozważanego procesu, który łącznie używa 1GiB swoją **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 adresowania TLB od adresowania pamięci podręcznej procesora? Kiedy fałdowanie 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.: liczba liczba wpisów, liczba wpisów w zbiorze, itd.

Wskazówka: Przed uruchomieniem programu należy załadować moduł jako o nazwie `cpuid`.

Zadanie 3. Zakładamy taki sam sprzętowy model podsystemu pamięci jak na slajdach do wykładu „Wirtual Memory: Systems" (strony 9-16). Powstrzyj proces translacji adresów i adresowania pamięci podręcznej dla adresów: 0x027c, 0x0340 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	00	1	-	-	0	07	02	1
1	03	20	1	-	-	0	02	17	1	-	-	0
2	-	-	0	-	-	0	-	-	0	-	-	0
3	-	-	0	03	00	1	04	34	1	-	-	0

Zawartość TLB

Idx	Tag	V	B0	B1	B2	B3
0	19	1	99	11	23	11
1	-	0	-	-	-	-
2	18	1	00	02	04	08
3	-	0	-	-	-	-
4	32	1	43	60	8F	09
5	00	1	36	72	F0	1B
6	-	0	-	-	-	-
7	16	1	11	C2	67	5D
8	24	1	3A	00	51	89
9	-	0	-	-	-	-
A	20	1	93	15	DA	3B
B	-	0	-	-	-	-
C	-	0	-	-	-	-
D	16	1	04	96	34	15
E	13	1	B3	77	1B	10
F	08	1	AF	BB	01	D7

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	20	1
0E	11	1
0F	04	1

Zawartość tablicy stron

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

Zadanie 4. W tym zadaniu będziemy analizować w jaki sposób system operacyjny musi aktualizować tablicę stron wraz z kolejnymi dostępami do pamięci głównej. Załóż, że strony są rozmiaru 4KiB. TLB jest **w pełni asocjacyjne** z zastępowaniem LRU. Najwyższa wartość pola LRU koduje najdłuższego kandydata na **ofiarę** (ang. victim). Jeśli potrzebujesz **wtoczyć** (ang. swap-in) stronę z dysku użyj następnego numeru **ramki** (ang. page frame) wolnego od najwyższego 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 była 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	08
04	1	09
05	1	0B
06	0	dysk
07	0	dysk
08	0	dysk
09	0	dysk
0A	0	dysk
0B	1	03
0C	1	brak

Początkowa zawartość tablicy stron

Valid	Tag	LRU	PPN
1	08	0	0C
1	07	1	04
1	03	2	06
1	06	3	09

Początkowa zawartość TLB

Adres
1234
0B63
365c
871b
9ee8
3140
c0d8

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 [5, §4-5] 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 A/X, ID i D/D [5, §4-6] wpływa na sprawdzenie 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*) [1, §3.4.2]. W jaki sposób algorytm ten korzysta z bitów A i D [5, §4-6] 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 miss). Rozważ wariant pesymistyczny i optymistyczny dla czterodniowego 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?

Wskazówka: 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 wykorzystać równoległy dostęp do TLB i pamięci podręcznej, stosując schemat pamięci indeksowanej wirtualnie i znakowanej fizycznie [2, §2.4.1]. Jakie korzyści to przynosi?

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

Zadanie 9. Wiemy, że pamięć podręczna TLB jest niezbędna do przyspieszenia szybkiej translacji adresów. Kiedy w systemie zachodzi potrzeba **przełączenia przestrzeni adresowej**? Czemu należałoby wtedy wyczyścić zawartość TLB? Czemu chcielibyśmy tego uniknąć? 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 architekci procesora decydują 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?

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/dam/develop/public/us/en/documents/325462-sdm-vol-1-2a-2b-3a-3b-3c-3d-4.pdf

[5] https://en.wikipedia.org/wiki/CPU_cache#Misses_and_synonyms_problems

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. Dane rozważanego procesu, który łącznie używa 1GiB swoją **wirtualnej przestrzeni adresowej**. Podaj maksymalny i minimalny rozmiar tablicy stron (a) jednopoziomowej oraz (b) **dwupoziomowej**.

ZADANIE

- 32 bitowy adres wirtualny
- strona 4 KiB
- tablicy stron 1024 wpisy
- PTE - 4 bajty
- używamy 1 GiB wirtualnej pamięci ciągłej

$1\,KiB = 2^{10}$

$1\,MiB = 2^{20}$

$1\,GiB = 2^{30}$

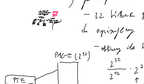
a) jednopoziomowa

- 12 bitów przesunięcia, do wyciągnięcia 2^{32} pamięć
- dzielimy do tablicy jednostki stron

$\frac{2^{32}}{2^{12}}$ $2^2 = 2^{20} = 4\,MiB$

b) dwupoziomowa, jeśli nie używamy, tablicy stron to nie tworzymy dla wpisy tablicy stron

* przypadek najgorszy $PAGE = 2^{10}(4KB)$



$PAGE = (2^{10})$

- 1 tablica stron

$\frac{2^{32}}{2^{12}} = 2^{20}$ PTE

2^{20} PTE $\cdot 4\,B = 4\,MiB$

- ile wpisów mamy 1 GiB, przydał jest przydzielony, jak całe strony są zajęte

$2^{10} \cdot 2^{10} = 2^{20}$ ile tyle całych stron w PTE

$1\,GiB = 2^{30}$

$\frac{2^{30}}{2^{12}} = 2^{18} = 256$ PTE, tyle na 1 tablicy

$\text{Rozmiar} = \underbrace{2^{10} \cdot 2^{10}}_{\text{1 tablica stron}} \cdot 4 + \underbrace{2^{8} \cdot 2^{10}}_{\text{tablice pośrednie}} \cdot 4 = 2^{18} \cdot 4 = 1\,MiB$

* przypadek najgorszy $PAGE = 2^{10}(4KB)$

- ile wpisów mamy 1 GiB, przydał jest przydzielony, jak całe strony są zajęte

$2^{10} \cdot 2^{10} = 2^{20}$ ile tyle całych stron w PTE

$1\,GiB = 2^{30}$

$\frac{2^{30}}{2^{12}} = 2^{18} = 256$ PTE, tyle na 1 tablicę stron

$\text{Rozmiar} = \underbrace{2^{10} \cdot 2^{10}}_{\text{1 tablica stron}} \cdot 4 + \underbrace{2^{8} \cdot 2^{10}}_{\text{tablice pośrednie}} \cdot 4 = 2^{18} \cdot 4 = 1\,MiB$

$\frac{2^{32}}{2^{10}} = 2^{22}$ PTE

$2^{22} \cdot 4 = 16\,MiB$

$\frac{2^{22}}{2^{10}} = 2^{12}$ tablic pośrednich

$2^{12} \cdot 4 = 16\,KiB$

$\text{Rozmiar} = 16\,MiB + 16\,KiB$

$\frac{2^{32}}{2^{12}} = 2^{20}$ PTE

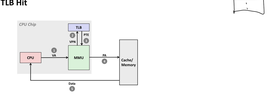
$2^{20} \cdot 4 = 4\,MiB$

$\frac{2^{20}}{2^{10}} = 2^{10}$ tablic pośrednich

$2^{10} \cdot 4 = 4\,KiB$

$\text{Rozmiar} = 4\,MiB + 4\,KiB$

TLB Hit



- aby uniknąć wszelkie tablicy stron używamy sprzętowej tablicy stron

- adresujemy za pomocą VPN, to jest jego tag

A Two-Level Page Table Hierarchy
Unused
Unused
Unused