Architektury systemów komputerowych

Lista zadań nr 11

Na zajęcia 25 maja 2022

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

Zadanie 1. Rozważmy **pamięć podręczną z mapowaniem bezpośrednim adresowaną bajtowo**. Używamy adresów 32-bitowych w następującym formacie: $(\mathrm{tag}, \mathrm{index}, \mathrm{offset}) = (\mathrm{addr}_{31\ldots10}, \mathrm{addr}_{9\ldots5}, \mathrm{addr}_{4\ldots0})$.

- Jaki jest rozmiar bloku w 32-bitowych słowach?
- Ile wierszy ma nasza pamięć podręczna?
- Jaki jest stosunek liczby bitów składujących metadane do liczby wszystkich bitów?

Zadanie 2. Mamy system z pamięcią operacyjną adresowaną bajtowo. Szerokość szyny adresowej wynosi 12. Pamięć podręczna ma organizację **sekcyjno-skojarzeniową** o dwuelementowych **zbiorach**, a **blok** ma 4 bajty. Dla podanego niżej stanu pamięci podręcznej wyznacz, które bity adresu wyznaczają: offset, indeks, $\mathrm{znacznik}$. Wszystkie wartości numeryczne podano w systemie szesnastkowym.

Indeks	Znacznik	Valid	B0	B1	B2	B3
0	00	1	40	41	42	43
	83	1	FE	97	CC	D0
1	00	1	44	45	46	47
	83	0	–	–	–	–
2	00	1	48	49	4A	4B
	40	0	–	–	–	–
3	FF	1	9A	C0	03	FF
	00	0	–	–	–	–

Określ, które z poniższych operacji odczytu wygenerują **trafienie** albo **chybienie** i ew. jakie wartości wczytają:

Adres	Trafienie?	Wartość
832	...	...
835	...	...
FFD	...	...

Pod jakim adresem w pamięci RAM leży bajt o wartości `0x4A` z naszej pamięci podręcznej?

Zadanie 3. Rozważmy pamięć podręczną z poprzedniego zadania. Mamy następującą sekwencję odwołań do czterobajtowych słów pamięci o adresach zadanych liczbami w systemie szesnastkowym:

```
0 4 10 84 3c e8 c8c a0 4 400 84 10 e8 884 c8c 0
```

Załóż, że na początku pamięć podręczna jest pusta. **Polityka wymiany** to NRU (ang. *Not Recently Used*). Podaj liczbę **wierszy** zastąpionych w wyniku **chybienia wywołanego konfliktem** (ang. *conflict miss*). Ile chybień było **przymusowych** (ang. *compulsory miss*)? Jaka jest efektywność pamięci podręcznej (ang. *hit ratio*)? Podaj w postaci tabelki (ale bez danych bloku) zawartość pamięci podręcznej po wykonaniu powyższych odwołań. Na każdy zbiór podaj kolejnego kandydata na **ofiarę** (ang. *victim*). Ile bitów na zbiór potrzebujesz na przechowanie informacji o tym jak wyznaczyć następną ofiarę?

Wskazówka: Definicje rodzajów chybień można znaleźć w §6.3.1 podręcznika.

Zadanie 4. Powtórz polecenia z poprzedniego zadania dla **w pełni asocjacyjnej** (ang. *fully associative*) pamięci podręcznej posiadającej 8 bloków. Polityka wymiany to LRU (ang. *Least Recently Used*). Reszta danych nie ulega zmianie.

Zadanie 5. Odpowiedz na następujące pytania dotyczące organizacji pamięci podręcznej:

1. Do wyboru zbioru pamięci podręcznej używamy bitów znajdujących się w środku adresu, zaraz przed offsetem bloku. Czemu jest to lepszy pomysł niż używanie najbardziej znaczących bitów adresu?
2. Zdecydowana większość procesorów posiada odrębną pamięć podręczną pierwszego poziomu dla danych i dla instrukcji. Jakie korzyści to przynosi?
3. Które fragmenty pamięci system operacyjny musi skonfigurować w trybie dostępu *write-through*?

Zadanie 6. Rozważamy system z dwupoziomową pamięcią podręczną z **polityką zapisu**: **write-back** i **write-allocate**. Jeśli blok o określonym adresie znajduje się na poziomie L_i to znajduje się również na wszystkich poziomach $j > i$ (ang. *inclusive caches*). Przy pomocy schematu blokowego[1] przedstaw algorytm obsługi zapisu słowa maszynowego do pamięci. Pierwszym elementem diagramu ma być predykat „*Czy słowo jest w L1?*". Pamiętaj o **bicie dirty** i o tym, że pamięć podręczna może być całkowicie wypełniona! Zakładamy, że pamięć podręczna pierwszego poziomu nie może komunikować się bezpośrednio z pamięcią operacyjną. Wiemy też, że pamięć `L2` jest dużo większa niż `L1`.

Zastanów się: Jakie problemy sprawiło by założenie, że pamięć podręczna przechowuje blok o określonym adresie dokładnie na co najwyżej jednym poziomie L_i (ang. *exclusive caches*)?

Zadanie 7. Załóżmy, że dostęp do pamięci głównej trwa $70ns$, a dostępy do pamięci stanowią 36% wszystkich instrukcji. Rozważmy system z pamięcią podręczną o następującej strukturze: `L1` – 2 KiB, współczynnik chybień 8.0%, czas dostępu $0.66ns$ (1 cykl procesora); `L2` – 1 MiB, współczynnik chybień 0.5%, czas dostępu $5.62ns$. Procesor charakteryzuje się współczynnikiem **CPI** (ang. *cycles per instruction*) równym 1.0, jeśli pominiemy instrukcje robiące dostępy do pamięci danych. Odpowiedz na poniższe pytania:

- Jaki jest średni czas dostepu do pamięci dla procesora: tylko z pamięcią podręczną `L1`, z `L1` i `L2`?
- Procesor wykonuje wszystkie instrukcje, łącznie z dostępami do pamięci danych. Oblicz jego `CPI` kiedy posiada: tylko pamięć podręczną `L1`, z `L1` i `L2`.

Uwaga: Zakładamy, że wszystkie instrukcje wykonywane przez program są w pamięci podręcznej `L1i`.

Zadanie 8. Dla czterodrożnej sekcyjno-skojarzeniowej pamięci podręcznej implementujemy politykę zastępowania `LRU`. W każdym zbiorze przechowujemy maksymalnie 8 dodatkowych bitów (nazwijmy je state), które kodują listę wierszy od ostatnio używanego do najdawniej używanego. Funkcja $victim : state \to line$ służy do wyznaczania wiersza ofiary, a funkcja $update : state \times line \to state$ do aktualizacji stanu wiersza, jeśli procesor wygenerował trafienie w wiersz $line$. Podaj implementację funkcji $victim$ i $update$ w języku `C` używając wyłącznie operatorów bitowych, ale nie arytmetycznych! Wytłumacz jak kodujesz listę wierszy przy pomocy $state$.

Wskazówka: W stanie wiersza zakoduj kolejność elementów w wierszu.

Zastanów się: Czemu zależy nam na zminimalizowaniu liczby bitów kodujących $state$, a nie zminimalizowaniu logiki realizującej funkcję «`victim`» i «`update`»?

[1] `https://pl.wikipedia.org/wiki/Schemat_blokowy`