# Systemy komputerowe

Lista zadań nr 11

Na zajęcia 21 maja 2025

UWAGA! W trakcie prezentacji rozwiązań należy być przygotowanym do wyjaśnienia pojęć, które zostały oznaczone **wytłuszczoną** czcionką.

**Zadanie 1.** Dany jest następujący kod oraz założenia co do sposobu jego wykonania.

```
1 int x[2][128];
2 int i;
3 int sum = 0;
4
5 for (i = 0; i < 128; i++) {
6   sum += x[0][i] * x[1][i];
7 }
```

- `sizeof(int) = 4`
- Tablica `x` znajduje się w pamięci pod adresem 0x0
- Pamięć podręczna jest początkowo pusta
- Dostępy do pamięci dotyczą jedynie elementów tablicy `x`. Wszystkie pozostałe zmienne kompilator umieścił w rejestrach. Pamięć podręczna nie przechowuje instrukcji.

Oblicz współczynniki chybień przy wykonaniu tego kodu dla każdego z poniższych wariantów

- pamięć podręczna ma 512 bajtów, mapowanie bezpośrednie, rozmiar bloku 16 bajtów
- jak powyżej, z tym że rozmiar pamięci podręcznej to 1024 bajtów
- pamięć podręczna ma 512 bajtów, jest dwudrożna, sekcyjno-skojarzeniowa, używa polityki zastępowania LRU, rozmiar bloku to 16 bajtów

W tym ostatnim przypadku, czy zwiększenie rozmiaru pamięci podręcznej pomoże zredukować współczynnik chybień? A zwiększenie rozmiaru bloku?

**Zadanie 2.** Pracujesz nad programem wyświetlającym animacje na ekranie o rozmiarze 640x480 pikseli. Komputer dysponuje 32KB pamięci podręcznej z mapowaniem bezpośrednim, o rozmiarze bloku 8 bajtów. Używasz następujących struktur danych.

```
1 struct pixel {
2   char r;
3   char g;
4   char b;
5   char a;
6 };
```

```
7 struct pixel buffer[480][640];
8 int i, j;
9 char *cptr;
10 int *iptr;
```

Zakładamy, że `sizeof(char) = 1`, `sizeof(int) = 4`, tablica `buffer` znajduje się w pamięci pod adresem 0x0, pamięć podręczna jest początkowo pusta. Ponadto, dostępy do pamięci dotyczą jedynie elementów tablicy `buffer`. Wszystkie pozostałe zmienne kompilator umieścił w rejestrach. Pamięć podręczna działa w trybie write-back write-allocate.

Jaki jest współczynnik trafień przy wykonaniu poniższego kodu?

```
1 for (j = 639; j >= 0; j--) {
2    for (i = 479; i >= 0; i--) {
3        buffer[i][j].r = 0;
```

```
        buffer[i][j].g = 0;
        buffer[i][j].b = 0;
        buffer[i][j].a = 0;
    }
}
```

**Zadanie 3.** Kod z poprzedniego zadania został zoptymalizowany przez dwóch programistów. Wynik ich pracy znajduje się odpowiednio w lewej i prawej kolumnie poniżej.

```
char *cptr = (char *) buffer;
while (cptr < (((char *) buffer) + 640 * 480 * 4)) {
      *cptr = 0;
       cptr++;
}
```

```
int *iptr = (int *)buffer;
while (iptr < ((int *)buffer + 640*480)) {
      *iptr = 0;
       iptr++;
}
```

Czym różnią się te programy? Czy któryś z nich jest wyraźnie lepszy, jeśli idzie o wykorzystanie pamięci podręcznej? Odpowiedź uzasadnij wyliczając odpowiednie współczynniki trafień.

**Zadanie 4.** Rozważmy alternatywny sposób indeksowania pamięci podręcznej, w którym do wyboru numeru zbioru używa się nie środkowych bitów adresu, lecz najstarszych bitów. Dlaczego to podejście nie jest rozsądne?

**Zadanie 5.** Zwiększanie drożności pamięci podręcznej (przy stałych liczbie i rozmiarze bloku) *statystycznie* prowadzi do zredukowania współczynnika chybień. Niestety, istnieją patologiczne przypadki, w których zwiększanie drożności zwiększa ten współczynnik. Rozważmy dwie organizacje pamięci, z mapowaniem bezpośrednim oraz dwudrożną, obydwie o tym samym rozmiarze bloku i tej samej liczbie bloków. Pamięć dwudrożna używa LRU jako polityki wymiany. Skonstruuj sekwencję dostępów do pamięci, która generuje więcej chybień w przypadku użycia pamięci podręcznej dwudrożniej, niż gdy pamięcią podręczną jest ta z mapowaniem bezpośrednim.

Poniżej znajdują się kolejne zadania dotyczące predyktorów skoków, pochodzące z książki Hennessy, John L., and David A. Patterson. Computer Architecture: A Quantitative Approach **6e**. 2019 (HP6e).

**Zadanie 6.** Wykonaj zadanie 3.17 z HP6e (str. 277).

**Zadanie 7.** (2pkty). Wykonaj zadanie 3.18 z HP6e (str. 278).

**Zadanie 8.** (2pkty). Wykonaj zadanie 3.19 z HP6e (str. 278).

**Uwaga:** Może się zdarzyć, że bufor predykcji skoków (ang. branch-target buffer) nie zawiera wpisu dotyczącego instrukcji skoku, o którą chcemy go zapytać. W tym przypadku płacimy karę (ang. buffer miss penalty). HP6e, s. 228 – 234.