Architektury systemów komputerowych

Lista zadań nr 2

Na zajęcia od 10 do 12 marca 2025

Rozwiązania zadań muszą się trzymać następujących wytycznych:

- Założenia:
  - liczby całkowite są w reprezentacji uzupełnień do dwóch,
  - wartość logiczna prawdy i fałszu odpowiada kolejno wartościom całkowitoliczbowym `1` i `0`,
  - przesunięcie w prawo na liczbach ze znakiem jest przesunięciem arytmetycznym,
  - dane typu `int` mają `N` bitów długości,
  - jeśli nie podano inaczej, rozwiązanie musi działać dla dowolnego `N` o wartości $8 * 2^n$,
  - przyjmujemy dodatkowo, że przepełnienie na liczbach ze znakiem zachowuje się dokładnie tak, jak na liczbach bez znaku.
- Zabronione:
  - wyrażenia warunkowe (`?:`) i wszystkie instrukcje (łącznie z `if`) poza przypisaniem,
  - operacja mnożenia, dzielenia i reszty z dzielenia (`*`, `/`, `%`),
  - operacje logiczne (`&&`, `||`, `!`),
  - operatory porównania (`==`, `!=`, `<`, `>`, `<=` i `>=`),
  - rzutowanie – zarówno jawne jak i niejawne.
- Dozwolone:
  - instrukcja przypisania,
  - operacje bitowe,
  - przesunięcie bitowe w lewo i prawo z argumentem w przedziale `0...N-1`,
  - dodawanie i odejmowanie,
  - stała `N`, stałe własne oraz zdefiniowane w pliku nagłówkowym `limits.h`.

UWAGA! W trakcie prezentacji rozwiązań należy zdefiniować i wyjaśnić pojęcia, które zostały oznaczone **wytłuszczoną** czcionką.

Zadanie 1. Czy poniższe wyrażenia zawsze obliczą się do prawdy dla dwóch dowolnych wartości zmiennych `x` i `y` typu `int32_t`? Jeśli nie to podaj wartości, które prowadzą do obliczenia fałszu.

- `(x > 0) || (x - 1 < 0)`
- `(x & 7) != 7 || (x << 29 < 0)`
- `(x * x) >= 0`
- `x < 0 || -x <= 0`
- `x > 0 || -x >= 0`
- `(x | -x) >> 31 == -1`
- `x + y == (uint32_t)y + (uint32_t)x`
- `x * ~y + (uint32_t)y * (uint32_t)x == -x`

Uwaga! W tym zadaniu przyjmujemy, że przepełnienie na liczbach ze znakiem zachowuje się dokładnie tak, jak na liczbach bez znaku.

Zadanie 2. Napisz ciąg instrukcji w języku `C`, który wyznaczy liczbę zapalonych bitów w zmiennej x.

Uwaga! Oczekiwana złożoność to $O(\log n)$, gdzie n to liczba bitów w słowie. Posłuż się strategią *„dziel i zwyciężaj”*.

Zadanie 3. Napisz wyrażenie zawierające wyłącznie zmienne `x`, `y` i `s`, którego wartością logiczną jest odpowiedź na pytanie czy wykonanie instrukcji `s = x + y` spowodowało **nadmiar** (ang. *overflow*) lub **niedomiar** (ang. *underflow*).

Wskazówka: Spróbuj rozwiązać zadanie samodzielnie, a następnie przeczytaj §2.12 książki *„Uczta programistów”*.

Zadanie 4. Zmienne `x` i `y` o typie `uint32_t` przechowują czteroelementowe wektory typu `uint8_t`. Tj. wektor $\{x_3, x_2, x_1, x_0\}$ reprezentujemy w zmiennej `x` przypisując jej wartość $\sum_{i=0}^{3} x_i \cdot 2^{8i}$. Jak szybko obliczyć zmienną `z` przechowującą wektor $\{z_3, z_2, z_1, z_0\}$, gdzie $z_i = x_i \oplus y_i$, gdy:

- $\oplus$ jest operacją dodawania,
- $\oplus$ jest operacją odejmowania.

Obliczając wynik należy zapobiec wystąpieniu **przeniesienia** (ang. *carry*) lub **pożyczki** (ang. *borrow*) propagujących się do bardziej znaczącego bajtu.

Wskazówka: Spróbuj rozwiązać zadanie samodzielnie, a następnie przeczytaj §2.17 książki „*Uczta programistów*".

Zadanie 5. Uzupełnij ciało funkcji zadeklarowanej następująco:

```
/* Oblicz x * 3 / 4 zaokrąglając w dół. */
int32_t threefourths(int32_t x);
```

Uwaga! Nie można dopuścić do wystąpienia nadmiaru i niedomiaru!

Zadanie 6. Podaj fragment kodu, który oblicza funkcję:

$$abs(x) = \begin{cases} x & \text{dla } x \geq 0 \\ -x & \text{dla } x < 0 \end{cases}$$

Skorzystaj z następującej własności: jeśli `b` jest wartością logiczną, to wyrażenie `b ? x : y` można przetłumaczyć do `b * x + !b * y`. Następnie stwórz analogiczną konstrukcję używając tylko operatorów bitowych.

Wskazówka: Spróbuj rozwiązać zadanie samodzielnie, a następnie przeczytaj §2.4 książki „*Uczta programistów*".

Zadanie 7. Podaj fragment kodu, który oblicza funkcję:

$$sign(x) = \begin{cases} -1 & \text{dla } x < 0 \\ 0 & \text{dla } x = 0 \\ 1 & \text{dla } x > 0 \end{cases}$$

Wskazówka: Spróbuj rozwiązać zadanie samodzielnie, a następnie przeczytaj §2.7 książki „*Uczta programistów*".