Architektury systemów komputerowych 2017

Lista zadań nr 3

Na zajęcia 15 i 16 marca 2017

Jeśli nie stwierdzono inaczej, 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; rozwiązanie musi działać dla dowolnego `N` będącego wielokrotnością 8.
- Zabronione:
  - wyrażenia warunkowe (`?:`) i wszystkie instrukcje poza przypisaniem,
  - operacja mnożenia, dzielenia i reszty z dzielenia,
  - operacje logiczne (`&&`, `||`, `^^`),
  - porównania (<, >, <= i >=).
- Dozwolone:
  - operacje bitowe,
  - przesunięcie w lewo i prawo z argumentem w przedziale `0...N - 1`,
  - dodawanie i odejmowanie,
  - test równości (==) i nierówności (`!=`),
  - stała `N`, stałe własne oraz zdefiniowane w pliku nagłówkowym `<limits.h>`.

Zadanie 1. Mamy dwa wektory $\vec{x}$ i $\vec{y}$ długości $n = 2^k$, których elementami są liczby w zakresu $[0, 1)$ zapisane w formacie `Q16.16`. Pokaż jak w języku `C` obliczyć wynik, o typie `Q16.16`, poniższego wyrażenia, a następnie oszacuj błąd obliczeń. Wszystkie wyniki pośrednie muszą być 32-bitowymi liczbami całkowitymi. Nie można osobno liczyć części całkowitej i ułamkowej. Można używać operatorów mnożenia i dzielenia.

$$d(\vec{x}, \vec{y}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2$$

Zadanie 2. Zastąp instrukcję dzielenia całkowitoliczbowego zmiennej `n` typu `int32_t` przez stałą 3 przy pomocy operacji mnożenia liczb typu `int64_t`. Przedstaw dowód poprawności swojego rozwiązania. Instrukcja dzielenia działa zgodnie z wzorem podanym na wykładzie, tj.:

$$\texttt{div3}(n) = \begin{cases} \lfloor \frac{n}{3} \rfloor & \text{dla } n \geq 0 \\ \lceil \frac{n}{3} \rceil & \text{dla } n < 0 \end{cases}$$

Wskazówka: Zauważ, że $\frac{x}{k} \equiv x * \frac{1}{k}$. Rozważ osobno liczby ujemne i nieujemne.

Zadanie 3. Standard IEEE 754-2008 definiuje liczby zmiennopozycyjne o szerokości 16-bitów. Zapisz ciąg bitów reprezentujący liczbę $1.5625 \cdot 10^{-1}$. Porównaj zakres liczbowy i dokładność w stosunku do liczb zmiennopozycyjnych pojedynczej precyzji (`float`).

Zadanie 4. Oblicz ręcznie $3.984375 \cdot 10^{-1} + 3.4375 \cdot 10^{-1} + 1.771 \cdot 10^{3}$ używając liczb w formacie z poprzedniego zadania. Zapisz wynik binarnie i dziesiętnie. Czy wynik się zmieni jeśli najpierw wykonamy drugie dodawanie?

UWAGA! Domyślną metodą zaokrąglania w obliczeniach zmiennoprzecinkowych jest *round-to-even*.

Zadanie 5. Załóżmy, że zmienne `x`, `f` i `d` są odpowiednio typów `int`, `float` i `double`. Ich wartości są dowolne, ale `f` i `d` nie mogą równać się $+\infty$, $-\infty$ lub *NaN*. Czy każde z poniższych wyrażeń zostanie obliczone do prawdy? Jeśli nie to podaj wartości zmiennych, dla których wyrażenie zostanie obliczone do fałszu.

1. `x == (int32_t)(double) x`
2. `x == (int32_t)(float) x`
3. `d == (double)(float) d`
4. `f == (float)(double) f`
5. `f == -(-f)`
6. `1.0 / 2 == 1 / 2.0`
7. `d * d >= 0.0`
8. `(f + d) - f == d`

Zadanie 6. Reprezentacja liczby zmiennoprzecinkowej typu `float` została załadowana do zmiennej `x` typu `int32_t`. Podaj algorytm mnożenia `x` przez 2^i. Uwzględnij przypadki brzegowe — tj. kiedy zmienna `x` ma wartość *NaN*, $\pm\infty$, ± 0 lub jest liczbą zdenormalizowaną.

UWAGA! Należy podać algorytm, zatem dozwolona jest cała składnia języka `C` bez ograniczeń z nagłówka listy zadań.

Zadanie 7. Reprezentacje binarne liczb zmiennoprzecinkowych `xf` i `yf` typu `float` zostały załadowane odpowiednio do zmiennych `x` i `y` typu `uint32_t`. Podaj wyrażenie, które:

1. zmieni znak liczby `x`,
2. obliczy wartość $\lfloor log_2|\mathtt{x}| \rfloor$ typu `int` dla $\mathtt{x} \neq 0$,
3. zwróci wartość logiczną operacji `x == y`,
4. zwróci wartość logiczną operacji `x <= y`.

Pamiętaj, że dla liczb zmiennopozycyjnych w standardzie IEEE 754 zachodzi $-0 \equiv +0$. Można pominąć rozważanie wartości *NaN*.

Zadanie 8. Uzupełnij ciało funkcji zadeklarowanej następująco:

```
/* Skonwertuj reprezentację liczby float do wartości int32_t. */
int32_t float2int(int32_t f);
```

Zaokrąglij liczbę w kierunku zera. Jeśli konwersja spowoduje nadmiar lub f ma wartość *NaN*, zwróć 0x80000000. Dla czytelności napisz najpierw rozwiązanie z instrukcjami warunkowymi. Potem przepisz je, by zachować zgodność z wytycznymi z nagłówka listy.