Architektury systemów komputerowych

Lista zadań nr 3

Na zajęcia od 17 do 19 marca 2025

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,
  - jeśli nie podano inaczej, 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 (`&&`, `||`, `^^`),
  - operatory porównania (`<`, `>`, `<=` i `>=`),
- Dozwolone:
  - operacje bitowe,
  - przesunięcie bitowe 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. Zastąp instrukcję dzielenia całkowitoliczbowego zmiennej `n` typu `uint32_t` przez stałą 3 przy pomocy operacji mnożenia liczb typu `uint64_t`. Skorzystaj z faktu, że $\frac{x}{k} \equiv x \cdot \frac{1}{k}$. Zapisz $\frac{1}{k}$ przy pomocy **liczby stałopozycyjnej** (ang. *fixed point number*). Przedstaw dowód poprawności swojego rozwiązania. Powiedz też co należałoby zrobić, aby umieć wykonać podobną operację dla dowolnej innej (ale też z góry ustalonej) liczby wymiernej.

Wskazówka: Spróbuj rozwiązać zadanie samodzielnie, a następnie przeczytaj §10.3 książki „*Uczta programistów*".

Zadanie 2. Standard IEEE 754-2008 definiuje liczby zmiennopozycyjne o szerokości 16-bitów. Zapoznaj się z nim, a następnie oblicz ręcznie $(a + b) + c$ oraz $a + (b + c)$, gdzie $a = 3.984375 \cdot 10^{-1}$, $b = 3.4375 \cdot 10^{-1}$ i $c = 1.771 \cdot 10^{3}$, używając liczb w tym formacie. Zapisz wynik binarnie i dziesiętnie. Zaprezentuj działanie algorytmu zaokrąglania liczb zmiennopozycyjnych i podaj definicje bitów **guard**, **round** i **sticky**. Zastanów się jak sumować ciągi liczb zmiennopozycyjnych, żeby zminimalizować błąd.

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

Zadanie 3. Mamy zmienne `x`, `y` i `z` typu `int32_t` ustawione na wybrane przez nas wartości. Konwertujemy je do liczb typu `double` zapisanych w zmiennych `dx`, `dy` i `dz`. Rozważmy poniższe wyrażenia z języka C. Wskaż, które z nich zawsze obliczą się do prawdy i uzasadnij ten fakt. W pozostałych przypadkach podaj kontrprzykład – konkretne wartości zmiennych całkowitoliczbowych.

1. `(float)x == (float)dx`
2. `dx - dy == (double)(x - y)`
3. `(dx + dy) + dz == dx + (dy + dz)`
4. `(dx * dy) * dz == dx * (dy * dz)`
5. `dx / dx == dz / dz`

Wskazówka: Zasady niejawnego rzutowania są wyjaśnione w §6.3.1.4, §6.3.1.5 i §6.3.1.8.

Zadanie 4. Reprezentacje binarne liczb zmiennoprzecinkowych f i g typu `float` zostały załadowane odpowiednio do zmiennych `x` i `y` typu `uint32_t`. Podaj wyrażenie, które:

1. zmieni znak liczby `f`,
2. obliczy wartość $\lfloor log_2|\texttt{f}|\rfloor$ typu `int` dla f w postaci znormalizowanej,
3. zwróci wartość logiczną operacji `f == g`,
4. zwróci wartość logiczną operacji `f < g`.

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

Wskazówka: Spróbuj rozwiązać zadanie samodzielnie, a następnie przeczytaj §15.2 książki „*Uczta programistów*".

Dla poniższych zadań należy podać kompletny algorytm, zatem dozwolona jest cała składnia języka C bez ograniczeń z nagłówka listy zadań. Jednakże należy używać wyłącznie operacji na typie `int32_t` lub `uint32_t`.

Zadanie 5. Binarna reprezentacja liczby zmiennoprzecinkowej f typu `float` została załadowana do zmiennej `x` typu `uint32_t`. Podaj algorytm obliczający $f \cdot 2^i$ wykonujący obliczenia na zmiennej `x` używając wyłącznie operacji na liczbach całkowitych. Osobno rozważ $i \geq 0$ i $i < 0$. Zakładamy, że liczba f jest znormalizowana, ale wynik operacji może dać wartość $\pm\infty$, ± 0 lub liczbę zdenormalizowaną.

Przykładowo, dla `x = 0xC0A00000` i $i = 1$ algorytm powinien zwrócić wartość `0xC1200000`.

Uwaga! Dla uproszczenia należy założyć, że wynik zaokrąglamy w kierunku zera.

Zadanie 6. Napisz ciało funkcji o sygnaturze `int32_t float2int(int32_t f)`, konwertującej binarną reprezentację liczby typu `float` umieszczoną w zmiennej `f` do wartości typu `int32_t`. Wynik należy zaokrąglić w kierunku zera. Jeśli konwersja spowoduje nadmiar lub f ma wartość `NaN`, zwróć wartość `0x80000000` (tj. `MIN_INT`).

Przykładowo, wywołanie `float2int(0xC0A00000)` powinno zwrócić wartość -5.

Zadanie 7 (bonus). Na podstawie artykułów `0x5f3759df`[1] oraz `0x5f3759df` (appendix)[2] zreferuj działanie algorytmu szybkiego przybliżania odwrotności pierwiastka kwadratowego z liczby typu `float`. Należy wyjaśnić podstawy obliczeń na binarnej reprezentacji liczby `x` i pochodzenie stałej `0x5f3759df`.

[1] `http://h14s.p5r.org/2012/09/0x5f3759df.html`
[2] `http://h14s.p5r.org/2012/09/0x5f3759df-appendix.html`