

Architektury systemów komputerowych

Lista zadań nr 3

Na zajęcia 12 i 18 marca 2024

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 `int32_t` przez stałą 3 przy pomocy operacji mnożenia liczb typu `int64_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. Instrukcja dzielenia działa zgodnie z wzorem podanym na wykładzie, tj.:

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

Najpierw rozwiąż zadanie dla przypadku $n \geq 0$, a potem uogólnij to na pełen przedział liczb.

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

Zadanie 2. Mamy dwa wektory \vec{x} i \vec{y} długości $n = 2^k$, których elementami są liczby w zakresie $[0, 1)$ zapisane w **reprezentacji stałopozycyjnej** Q0.32. Chcemy obliczyć wynik typu Q0.32 poniższego wyrażenia:

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

Zauważ, że wyniki pośrednie muszą być zapisane w reprezentacji Q32.32. Nie można osobno liczyć części całkowitej i ułamkowej. Można używać wszystkich konstrukcji języka C poza liczbami zmiennopozycyjnymi.

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 $(a + b) + c$ 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 formacie z poprzedniego zadania. 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 5. 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 mogą obliczyć się do fałszu. Podaj kontrprzykład – konkretne wartości naszych 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 6. 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 $\lfloor \log_2 |f| \rfloor$ 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 7. 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.

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

Zadanie 8. Napisz ciało procedury o sygnaturze «int32_t int2float(int32_t i)» konwertującej wartość całkowitoliczbową do binarnej reprezentacji liczby typu «float». Wynik należy zaokrąglić do najbliższej liczby parzystej – w tym celu wyznacz wartość bitów guard, round i sticky. Do wyznaczenia pozycji wiodącej jedynek można użyć funkcji «_builtin_clz», tj. instrukcji wbudowanej¹ w kompilator gcc.

Wskazówka: Rozwiązanie jest łatwiejsze do zrozumienia, jeśli mantysę zapisze się na najbardziej znaczących bitach.

Zadanie 9. Uzupełnij ciało poniższej procedury 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). Kod procedury zaczyna się wyodrębnieniem poszczególnych elementów liczby zmiennopozycyjnej:

```
1 int32_t float2int(int32_t f) {
2     int32_t s = f >> 31;          /* -1 jeśli znak był ujemny */
3     int32_t e = (f >> 23 & 255) - 127; /* wykładnik po odjęciu bias */
4     uint32_t m = f << 8 | 0x80000000; /* mantysa 1.xxx... dosunięta do lewej */
5     /* TODO */
6 }
```

Wskazówka: Wzorcówka ma dodatkowe cztery linie kodu i używa jednej instrukcji warunkowej!

Zadanie 10 (bonus). Na podstawie artykułów 0x5f3759df² oraz 0x5f3759df (appendix)³ 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<https://gcc.gnu.org/onlinedocs/gcc/Other-Builtins.html>

²<http://h14s.p5r.org/2012/09/0x5f3759df.html>

³<http://h14s.p5r.org/2012/09/0x5f3759df-appendix.html>