

LISTA 2

① czy te wyrażenia to zawsze prawda?

- x, y to int32_t w U2
- przedstawienie jak w unsigned
- przesunięcie w prawo: asymetryczne
- $(x > 0) \vee (x - 1 < 0)$
 \rightarrow nie dla INT32_MIN
 $\rightarrow x - 1$ skutkuje przedstawieniem do INT32_MAX
- $(x \& \#1111) \neq \#1111 \vee (x \ll 29 < 0)$
 $\rightarrow \#1111 = \dots 0111$
 $\rightarrow (x \& \#1111) \neq \#1111 \Leftrightarrow$ któryś z trzech ostatnich bitów nie jest zapalony
 $\rightarrow (x \ll 29)$ przesunięcie o 29 pozycji
 \rightarrow najstarszy zapalony bit: $3 + 29 = 32$
 \rightarrow to bit znaku, więc $(x \ll 29 < 0)$
- $(x * x) >= 0$ $x = \sqrt{\text{INTMAX}} + 1$
 \rightarrow nie, np. $0x50000000 \dots ? ? ?$ $x = 2^{16} - 1$
 $x^2 = 2^{32} - 2 \cdot 2^{16} + 1$
 $\mod 2^{32}$
 $= -2^{17} + 1$
- $(x < 0) \vee -x \leq 0$ ✓
- $(x > 0) \vee -x \geq 0$
- \rightarrow nie, bo $-\text{INT_MIN} = \text{INT_MIN} < 0$
- $(x | -x) \gg 31 == -1$
- \rightarrow nie bo dla $x = 0$ wynosi 0
- $x + y == (\text{uint32_t})y + (\text{uint32_t})x$
- 1° $x, y \geq 0 \rightarrow$ unsigned takie same jak U2
- 2° $x < 0 \vee y < 0$ dodawanie działa tak samo, tylko interpretacja się zmienia
- $\text{uint } x = x + 2^{32}$
- $x + 2^{32} + y = x + y + 2^{32}$
- overflow! liczymy $\mod 2^{32}$
- wynosi $x + y$

$$\begin{aligned} \circ & x * \sim y + (\text{uint32_t}) y * (\text{uint32_t}) x == -x \\ \sim y &= -y - 1 \\ x \cdot (-y - 1) + ux \cdot uy &== -x \\ -xy - \cancel{x} + ux \cdot uy &== -\cancel{x} \\ ux \cdot uy &== xy \end{aligned}$$

↖ preliczono na unsigned

(2) wyznaczanie liczby zapalonych bitów

- 1) $x = x - ((x \gg 1) \& 010101\dots);$
- 2) $x = (x \& 00110011\dots) + ((x \gg 2) \& 00110011\dots);$
- 3) $c = ((x + (x \gg 4) \& 000011110000\dots) \& 000100000001\dots) \gg 24$

1) zlicza sumy par bitów

$$\left\{ \begin{array}{l} x = 11010000 \\ x \gg 1 = 01101000 \\ \vdots 55\dots = 01000000 \\ x = 10|01|00|00 \end{array} \right\} - = (x \ll 0) + (x \ll 8) + (x \ll 16) + (x \ll 24) !$$

2) sumuje 4-bitowe fragmenty

$$\begin{array}{r} x \quad 10010000 \\ \& 33 \quad 00010000 \\ x \gg 2 \quad 00100100 \\ \& 33 \quad 00100000 \\ + \quad 00110000 \end{array} +$$

3) sumuje 8-bitowe fragmenty

$$\begin{array}{r} x \quad 00110000 \\ x \gg 4 \quad 00000011 \\ + x \quad 00110011 \\ 8^4 1^4 \quad 00000011 \\ \& 6^3 10^4 \quad 000011 \end{array} \rightarrow \text{wynik} = 3$$

- 1) $x = x - ((x >> 1) \& 010101\dots);$
- 2) $x = (x \& 00110011\dots) +$
 $((x >> 2) \& 00110011\dots);$
- 3) $x = ((x + (x >> 4)) \& 000011110000\dots)$
- 4) $x = x + (x >> 8)$
- 5) $x = x + (x >> 16)$
- 6) return $x \& \underbrace{00111111\dots}_{6 \text{ bits}},$ bo max
 $32 = 2^5$

③ wypisanie którego wartości logiczna jest odpowiedź na pytanie
czy $s = x + y$ powoduje over/underflow

$\text{sign_}x = x \gg (N-1);$

$(\text{sign_}x \wedge \text{sign_}s) \neq (\text{sign_}y \wedge \text{sign_}s)$

→ overflow gdy $x, y \geq 0$ a $s < 0$

→ xor daje 1 dla różnych wartości

→ underflow gdy $x, y < 0$ a $s \geq 0$

4) zmienna uint 32 - t x,y przedstawiają
dwie elementowe wektory uint 8 - t
jak szybko obliczyć $z = x + y$?

$$1) S = (x \& 01^7 01^7 \dots) + (y \& 01^7 01^7 \dots)$$

$$2) S = ((x \wedge y) \& 10^7 10^7 \dots) \wedge S$$

1) obliczenie sumy poza najstarszym
bitami każdego uint 8 - t

$$\text{np.: } x = \begin{array}{r} 11111111 \\ \wedge 10000000 \\ \hline 01111111 \end{array} \quad \begin{array}{l} 01^7 \dots \\ + \end{array}$$

$$y = \begin{array}{r} 00000000 \\ \wedge 11111111 \\ \hline 01111111 \end{array}$$

2) dodanie najstarszego bitu tam, gdzie
to nie spowoduje przepiętania

$$x \wedge y = 11111111 \quad 01111111$$

$$\& 10^7 \dots = 10000000 \quad 00000000$$

$$\wedge S = 11111111 \quad 01111111$$

jak szybko obliczyć $z = x - y$?

$$1) d = (x | 10^7 10^7 \dots) - (y \& 01^7 01^7 \dots)$$

$$2) d = ((x \wedge y) | 01^7 01^7 \dots) \wedge d$$

1) ustawia w x najstarsze bity na 1
a w y je zeruje

$$\text{np.: } x = \begin{array}{r} 11111111 \\ | 10^7 \\ \hline 11111111 \end{array} \quad 00000000$$

$$y = \begin{array}{r} 00001111 \\ \wedge 10000000 \\ \hline 00001111 \end{array} \quad \left. \begin{array}{l} 00000000 \\ - \\ 00000000 \end{array} \right\}$$

$$x - y = 11110000 \quad 10000000$$

$$2) \quad \left. \begin{array}{l} x \wedge y = 11110000 \quad 10000000 \\ | 01^7 = 11111110 \quad 11111110 \\ \wedge d = 00001110 \quad 01111111 \\ \wedge () = 11110001 \quad 10000000 \end{array} \right\} \wedge$$

$$x \wedge y = 11110000 \quad 10000000$$

$$| 01^7 = 11111110 \quad 11111110$$

$$\wedge d = 00001110 \quad 01111111$$

$$\wedge () = 11110001 \quad 10000000$$

$$9/4 = 2$$

⑤ oblicz $x \cdot 3/4$ zaokrąglając w dół
→ nie można dopuszczać do under/overflow!

$$x \cdot 3 = x + x \cdot 2$$

int32_t threefourths (int32_t x)
{ return (x + (x << 1)) >> 2
}

$$\begin{aligned} x &= 4k + r \\ k &= x >> 2 \\ r &= x - (k << 2) \\ \text{return } &(k + (k << 1)) + ((r + (r << 1)) >> 2) \end{aligned}$$

$$x \cdot 3/4 = (4k+r) \cdot 3/4 = k \cdot 3 + r \cdot 3/4$$

$$(x >> 1) + (x >> 2) + (x >> 1 \& x \& 1)$$

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

wskazówka: $b ? x : y = b * x + !b * y$

$$\text{abs}(x) = (x \geq 0) * x + !(x \geq 0) * (-x)$$

$$\rightarrow (x \geq 0) \Leftrightarrow \text{bit znaku} = 0$$

$$1) \text{sign} = x \gg 31 \quad \leftarrow 0 \text{ lsb} - 1$$

$$2) (x \wedge \text{sign}) - \text{sign} \quad \text{bo jest mniej przy } \gg$$

1) prawy maskę 000... dla dodatnich

lub 1111... dla ujemnych

2) ~nic nie zmienia dla dodatnich

za to dla ujemnych zachowuje się jak ~

\rightarrow potem $- \text{sign}$ to -0 dla (+)

oraz $+1$ dla (-)

\rightarrow bo $-x$ dla $x \cup u_2$ to $(\sim x) + 1$

$$\text{np.: } x = 1100\dots$$

$$\text{sign} = 1111\dots$$

$$x \wedge \text{sign} = 0011\dots$$

$$- \text{sign} = 00\dots 1$$

+

$$01111\dots$$

$$00000\dots$$

$$01111\dots$$

$$01111\dots$$

\sim brak znian dla
 $x \geq 0$

$$\textcircled{7} \quad \text{sign}(x) = \begin{cases} -1 & \text{da } x < 0 \\ 0 & \text{da } x = 0 \\ 1 & \text{da } x > 0 \end{cases}$$

$$(x \gg 31) \mid -((-x) \gg 31)$$

(+)	0	$\underbrace{1}_{\text{---}}$	$\underbrace{-1}_{\text{---}}$
(-)	-1	0	0

-1 0 1

x	11...11	00...00	00...01
<u>$\gg 31$</u>	11...11	00...00	00...00
$-x$	00...01	00...00	11...11
$-x \gg 31$	00...00	00...00	11...11
<u>$-(\uparrow)$</u>	00...00	00...00	00...01
L R	-1	0	1