2. ZARYS ORGANIZACJI MASZYNY TYPOWEJ

[2.1. Arytmetyka uzupełnieniowa, 2.2. Krótki opis maszyny typowej, 2.3. Kod rozkazowy]

2.1. ARYTMETYKA UZUPEŁNIENIOWA

2.1.0. Zajmiemy się obecnie omówieniem podstawowych zasad binarnej arytmetyki uzupełnieniowej, w której $(n+1)$-bitowe liczby dwójkowowymierne należą do przedziału $<-1, 1-2^{-n}>$ (punkt 1.2). Liczba dodatnia x, jak wiemy, jest przedstawiona w tej arytmetyce następująco:

$$0.\alpha_1\alpha_2\alpha_3 \ldots \alpha_{n-1}\alpha_n,$$

gdzie α_i są to kolejne cyfry dwójkowe (binarne) rozwinięcia liczby x.

Natomiast gdy x jest liczbą ujemną, wówczas w arytmetyce uzupełnieniowej zastępujemy ją liczbą $2+x$, która z kolei ma postać

$$1.\alpha'_1\alpha'_2\alpha'_3 \ldots \alpha'_{n-1}\alpha'_n,$$

gdzie α' $(i = 1, \ldots, n)$ są to cyfry rozwinięcia liczby $1+x$.

Rozpatrzmy kilka przykładów przedstawiania liczb ujemnych:

liczba $\frac{11}{32}$ ma postać

$$0.010110\ldots 0;$$

natomiast liczba $-\frac{11}{32}$ ma postać

$$2-\frac{11}{32}=\frac{53}{32}, \quad 1.101010\ldots 0,$$

liczba $\frac{115}{128}$ ma postać:

$$0.11100110\ldots 0,$$

natomiast liczba $-\frac{115}{128}$ ma postać

$$2-\frac{115}{128}=\frac{141}{128}, \quad 1.00011010\ldots 0.$$

2.1.1. Algorytm uzupełnienia. Jeżeli mamy liczbę dodatnią x, to dla otrzymania w arytmetyce uzupełnieniowej liczby $-x$ wykonujemy operacje $2-x$. Podobnie jeśli mamy liczbę $-x$, reprezentowaną jako $2-x$, a chcemy otrzymać liczbę x, to musimy wykonać operację $2-(2-x)=x$. Dla praktycznego obliczania uzupełnienia liczb podamy obecnie algorytm.

Niech liczba x będzie postaci

$$\alpha_0 \cdot \alpha_1\alpha_2 \ldots \alpha_{n-1}\alpha_n;$$

wprowadzimy pomocniczą liczbę binarną

$$\omega_0 \cdot \omega_1 \omega_2 \ldots \omega_{n-1} \omega_n,$$

której cyfry są określone przez następujące związki:

$$\omega_i = 0$$

wtedy i tylko wtedy, gdy $\alpha_i = \alpha_{i+1} = \ldots = \alpha_n = 0$ (gdzie $i = n, \ldots 1,0$),

$$\omega_i = 1$$

wtedy i tylko wtedy, gdy istnieje wskaźnik całkowity nieujemny j, spełniający warunek $i \leqslant j \leqslant n$, dla którego $\alpha_j = 1$.

Kolejne cyfry uzupełnienia liczby x wyznaczamy korzystając ze związku

$$\alpha'_i = \omega_i - \alpha_i \omega_{i+1} \text{ (gdzie } i = 0,1, \ldots, n).$$

Dla udowodnienia powyższego algorytmu wystarczy zauważyć, że

$$\alpha_i + \alpha'_i = \begin{cases} 0, & \text{gdy} \quad \omega_i = \omega_{i+1} = 0, \\ 1, & \text{gdy} \quad \omega_i = \omega_{i+1} = 1, \\ 2, & \text{gdy} \quad \omega_i = 1, \omega_{i+1} = 0, \end{cases}$$

oraz skorzystać ze związku

$$2 = \sum_{i=0}^{k} 2^{-i} + 2^{-k}.$$

Przekroczenie zakresu przy uzupełnianiu następuje wtedy i tylko wtedy, gdy $x = -1$.

Przykład 2-1. Obliczamy uzupełnienie liczby

$$\frac{25}{32} = 0.1100100;$$

korzystając z przedstawionego wyżej algorytmu otrzymamy

$$1.0011100;$$

łatwo sprawdzić, że otrzymaliśmy liczbę $\frac{39}{32}$, która jest uzupełnieniem liczby $\frac{25}{32}$.

2.1.2. Algorytm sumowania szeregowego. Niech będą dane dwie dodatnie liczby binarne (dwójkowe), o skończonych rozwinięciach. Załóżmy, że obie te liczby są tej samej długości

$$x_1 = \sum_{i=0}^{n} \alpha_i(x_1) 2^{-i},$$

$$x_2 = \sum_{i=0}^{n} \alpha_i(x_2) 2^{-i}.$$

Dodawanie szeregowe polega na dodawaniu do siebie kolejnych współczynników, począwszy od współczynnika przy najniższych potęgach dwójki.

Jeśli oznaczymy cyfry wyniku przez $\alpha_i(y)$, a przeniesienia z dodawania przez p_i, to algorytm dodawania możemy opisać za pomocą tabl. 2-1, realizującej wzór

$$\alpha_i(x_1) + \alpha_i(x_2) + p_{i+1} = \alpha_i(y) + p_i,$$

gdzie $\alpha_i, p_i \in \{0,1\}$ dla $i = n, n-1, \ldots, 1,0$.

Tablica 2-1

$\alpha_i(x_1)$	$\alpha_i(x_2)$	p_{i+1}	$\alpha_i(y)$	p_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Przy dodaniu do siebie współczynników przy 2^{-n} kładziemy $p_{n+1} = 0$. Niech będą dane dwie liczby x_1, x_2 z przedziału $<-1, 1-2^{-n}>$. Oznaczmy dalej przez y_1 przedstawienie liczby x_1 w arytmetyce uzupełnieniowej, przez y_2 zaś przedstawienie liczby x_2 w arytmetyce uzupełnieniowej. Musimy rozpatrzyć cztery przypadki:

1) $x_1 \geqslant 0$, $x_2 \geqslant 0$, wówczas $y_1 = x_1$, $y_2 = x_2$,
2) $x_1 \geqslant 0$, $x_2 < 0$, wówczas $y_1 = x_1$, $y_2 = 2 + x_2$,
3) $x_1 < 0$, $x_2 \geqslant 0$, wówczas $y_1 = 2 + x_1$, $y_2 = x_2$,
4) $x_1 < 0$, $x_2 < 0$, wówczas $y_1 = 2 + x_1$, $y_2 = 2 + x_2$.

Rozpatrzmy pierwszy przypadek

$$y_1 + y_2 = x_1 + x_2,$$

czyli dodawanie w tym przypadku dwóch liczb w arytmetyce uzupełnieniowej sprowadza się do zwykłego dodawania liczb w rozwinięciu binarnym.

Przypadek drugi i trzeci rozpatrzmy łącznie

$$y_1 + y_2 = 2 + x_1 + x_2 = 2 + (x_1 + x_2).$$

Z kolei musimy rozważyć dwa podprzypadki:

1) gdy $x_1 + x_2 < 0$,
2) gdy $x_1 + x_2 \geqslant 0$.

W pierwszym z wymienionych podprzypadków wynik dodawania daje poprawne

przedstawienie liczby w arytmetyce uzupełnieniowej, w drugim należy zmniejszyć o 2; jest to równoważne nieuwzględnianiu pozycji α_{-1} wyniku.

Rozpatrzmy czwarty przypadek:

$$y_1 + y_2 = 4 + x_1 + x_2 = 2 + 2 + x_1 + x_2,$$

ponieważ $x_1 + x_2 < 0$, aby otrzymać prawidłową postać, wynik należy zmniejszyć o 2, jest to równoważne nieuwzględnianiu pozycji α_{-1} wyniku.

Jak wynika z powyższych rozważań, dodawanie liczb z przedziału $< -1, 1-2^{-n} >$ w arytmetyce uzupełnieniowej sprowadza się do szeregowego dodawania liczb binarnych i zaniedbywania pozycji α_{-1}, odpowiadającej pierwszej potędze dwójki 2^1.

Przekroczenie zakresu. Przy dodawaniu (odejmowaniu) liczb z przedziału $< -1, 1-2^{-n} >$ może nastąpić przekroczenie zakresu. Przy dodawaniu przekroczenie zakresu może nastąpić w dwóch przypadkach:

1) gdy $x_1 > 0$, $x_2 > 0$ oraz $y = x_1 + x_2 \geqslant 1$,
2) gdy $x_1 < 0$, $x_2 < 0$ oraz $y = x_1 + x_2 < -1$.

Oznaczymy zerowy bit liczby x przez $\alpha_0(x)$; bit przekroczenie zakresu oznaczymy przez ϱ (jeśli nastąpiło przekroczenie zakresu, to $\varrho = 1$, jeśli zaś nie nastąpiło, to $\varrho = 0$). Korzystając z powyższych oznaczeń w tabl. 2-2 przedstawiliśmy przekroczenie zakresu w zależności od zerowych bitów argumentów i wyniku dodawania.

Tablica 2-2

$\alpha_0(x_1)$	$\alpha_0(x_2)$	$\alpha_0(y)$	ϱ
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

W przyjętym przez nas systemie przedstawiania liczb ujemnych operację odejmowania sprowadzamy do operacji dodawania odjemnej z uzupełnieniem odjemnika.

2.1.3. Algorytm mnożenia przez $\frac{1}{2}$. Jeśli mamy liczbę dodatnią

$$0.\alpha_1\alpha_2 \ldots \alpha_n,$$

to mnożąc ją przez $\frac{1}{2}$ otrzymujemy w wyniku liczbę

$$0.0\alpha_1\alpha_2 \ldots \alpha_n.$$

Podobnie jeśli mamy liczbę ujemną

$$1.\alpha_1\alpha_2 \ldots \alpha_n,$$

to w wyniku otrzymamy liczbę

$$1.1\alpha_1\alpha_2 \ldots \alpha_n.$$

Ogólnie algorytm mnożenia przez $\frac{1}{2}$ dla liczb binarnych w arytmetyce uzupełnieniowej ma postać:

$$\tfrac{1}{2}(\alpha_0 \cdot \alpha_1\alpha_2 \ldots \alpha_n) = \alpha_0 \cdot \alpha_0\alpha_1\alpha_2 \ldots \alpha_n.$$

2.1.4. Algorytm mnożenia przez 2. Mnożenie dowolnej liczby binarnej w arytmetyce uzupełnieniowej

$$\alpha_0 \cdot \alpha_1\alpha_2 \ldots \alpha_n,$$

przez 2 jest wykonalne (bez przekroczenia zakresu) tylko w przypadku, gdy

$$\alpha_0 = \alpha_1,$$

wówczas

$$2(\alpha_0 \cdot \alpha_1\alpha_2 \ldots \alpha_n) = \alpha_1 \cdot \alpha_2\alpha_3 \ldots \alpha_n,$$

w przypadku zaś, gdy

$$\alpha_0 \neq \alpha_1$$

mnożenie wyprowadza poza przedział $< -1, 1-2^{-n+1} >$, czyli następuje przekroczenie zakresu.

2.1.5. Algorytm mnożenia. Niech będzie dana para liczb zwana dalej mnożną i mnożnikiem. Dalej będziemy oznaczali mnożną literą M, mnożnik zaś literą F

$$M = m_0 \cdot m_1 m_2 \ldots m_{n-1},^{(1)}$$
$$F = f_0 \cdot f_1 f_2 \ldots f_{n-1}.$$

Dopisujemy do F na n-tej pozycji 0, otrzymamy

$$F = f_0 \cdot f_1 f_2 \ldots f_{n-1}\, 0.$$

Rozważmy dalej ciąg par

$$0, f_{n-1}; \quad f_{n-2}, f_{n-1}; \quad \ldots; \quad f_{k-1}, f_k; \quad \ldots; \quad f_0, f_1.$$

Będziemy tworzyli ciąg iloczynów cząstkowych

$$A_0, A_1, \ldots, A_n,$$

gdzie pierwszy wyraz ciągu przyjmujemy równy 0:

$$A_0 = 0;$$

dalsze zaś wyrazy tego ciągu tworzymy korzystając z reguł podanych w tabl. 2-3, roz-

(1) Przy czym zakładamy, że $M \neq -1$, w przypadku gdy $M = -1$, zamiast omawianego dalej algorytmu należy zastosować algorytm uzupełnienia, omawiany w punkcie 2.1.1, do liczby F, aby otrzymać poprawny wynik.

ważając kolejne pary f_k, f_{k-1}. Iloczyn A jest równy podwojonemu n-temu iloczynowi cząstkowemu A_n:

$$A = 2A_n.$$

Dowód powyższego algorytmu, podanego przez Booth'a, nie nastręcza trudności.

Tablica 2-3

f_{k-1}	f_k	A_{n-k+1}	(gdzie $k = n, n-1, \ldots, 1$)
0	0	$\frac{1}{2} A_{n-k}$	
0	1	$\frac{1}{2}(A_{n-k} + M)$	
1	0	$\frac{1}{2}(A_{n-k} - M)$	
1	1	$\frac{1}{2} A_{n-k}$	

Wystarczy zauważyć, że w przypadku, gdy F ma na k-tej pozycji jedynkę, pozostałe zaś bity są zerami, to dwa kroki algorytmu mające istotny wpływ na wynik, odpowiadające dwóm parom f_{k+1}, f_k i f_k, f_{k-1} są równoważne pomnożeniu przez 2^{-k},

$$-M \cdot 2^{-k} + M \cdot 2^{-k+1} = M \cdot (2^{-k+1} - 2^{-k}) = M \cdot 2^{-k}$$

Dalej dowód przebiega indukcyjnie.

Uwaga: W przypadku gdy M i F są liczbami n-bitowymi, wynik mnożenia jest liczbą $2n$-1 bitową.

Przykład 2-2. Niech $M = \frac{3}{4}$, zaś $F = \frac{5}{8}$. W arytmetyce uzupełnieniowej mają postać:

$$M = 0.110, \quad F = 0.101.$$

Rozważmy kolejne pary i skonstruujemy kolejne iloczyny cząstkowe:

$$\begin{aligned}
f_3 = 1, \quad f_4 = 0, \quad A_1 &= \tfrac{1}{2}(-M) = 1.1010, \\
f_2 = 0, \quad f_3 = 1, \quad A_2 &= \tfrac{1}{2}(A_1 + M) = 0.00110, \\
f_1 = 1, \quad f_2 = 0, \quad A_3 &= \tfrac{1}{2}(A_2 - M) = 1.101110, \\
f_0 = 0, \quad f_1 = 1, \quad A_4 &= \tfrac{1}{2}(A_3 + M) = 0.0011110,
\end{aligned}$$

$$A = 2A_4 = 0.011110 = \frac{15}{32}.$$

Przykład 2-3. Niech $M = -\frac{11}{32}$, zaś $F = \frac{11}{32}$. W arytmetyce uzupełnieniowej mają one postać:

$$M = 1.10101, \quad F = 0.01011.$$

Rozważmy kolejne pary i skonstruujemy kolejne iloczyny cząstkowe:

$$\begin{array}{llll} f_5 = 1, & f_6 = 0, & A_1 = \tfrac{1}{2}(-M) & = 0.001011, \\ f_4 = 1, & f_5 = 1, & A_2 = \tfrac{1}{2}A_1 & = 0.0001011, \\ f_3 = 0, & f_4 = 1, & A_3 = \tfrac{1}{2}(A_2 + M) & = 0.1011111, \\ f_2 = 1, & f_3 = 0, & A_4 = \tfrac{1}{2}(A_3 - M) & = 0.000110111, \\ f_1 = 0, & f_2 = 1, & A_5 = \tfrac{1}{2}(A_4 + M) & = 1.1110000111, \\ f_0 = 0, & f_1 = 0, & A_6 = \tfrac{1}{2}A_5 & = 1.11110000111, \end{array}$$

$$A = 2A_6 = 1.1110000111 = -\frac{121}{1024}.$$

2.1.6. Algorytm zaokrąglenia. Niech będzie dana liczba binarna w arytmetyce uzupełnieniowej o $n + s$ bitach. Przypuśćmy, że potrzebne nam jest zaokrąglenie tej liczby do n bitów, w tym celu musimy sprawdzić znak liczby; jeśli liczba jest dodatnia, to musimy dodać do niej jedynkę na $n + 1$ pozycji, czyli dodać 2^{-n+1};

$$0.\alpha_1\alpha_2 \ldots \alpha_{n-1}\alpha_n \ldots \alpha_{n+s} + \underbrace{0.00 \ldots 010}_{n+1 \text{ bitów}} \ldots 0.$$

W przypadku zaś, gdy liczba jest ujemna, musimy dodać do niej ciąg jedynek począwszy od pierwszej pozycji, a skończywszy na $n + 1$ pozycji, czyli odjąć 2^{-n+1}; uprzednio kładąc $\alpha_{n+1} = 1$;

$$1.\alpha_1\alpha_2 \ldots \alpha_{n-1}\alpha_n \ldots \alpha_{n+s} + \underbrace{1.11 \ldots 110}_{n+1 \text{ bitów}} \ldots 0$$

wzór algebraiczny opisujący algorytm zaokrąglenia liczby x ma postać:

$$z(x) = x + 2^{-n+1}[2 - 2\alpha_0 + \alpha_{n+1}(\alpha_0 - 1)].$$

Przekroczenie zakresu może nastąpić tylko w dwu przypadkach: w pierwszym, jeśli liczba jest dodatnia (czyli $\alpha_0 = 0$) i bity $\alpha_1, \alpha_2, \ldots, \alpha_n$ są jedynkami, w drugim przypadku, jeśli liczba jest ujemna (czyli $\alpha_0 = 1$) i bity $\alpha_1, \alpha_2, \ldots, \alpha_n$ są zerami.

2.1.7. Algorytm dzielenia. Niech będzie dana para liczb: dzielna i dzielnik. Oznaczmy dzielnik literą M, dzielną zaś literą A,

$$A = a_0 \cdot a_1 a_2 \ldots a_{n-1},$$

$$M = m_0 \cdot m_1 m_2 \ldots m_{n-1}.$$

Nasz algorytm musi zapewniać wykonanie dzielenia, tylko w przypadku gdy

$$|A| < |M|, \tag{2-1}$$

ponieważ w przeciwnym razie nastąpiłoby przekroczenie zakresu. Algorytm nasz będzie konstruował nam dwa ciągi: ciąg pomocniczy i ciąg ilorazów częściowych.

Sprawdzenie warunku (2-1). Będziemy rozróżniali dwa przypadki:

$$\begin{array}{ll} \text{jeśli } a_0 = m_0, & \text{to } A_1 = A - M, \\ \text{jeśli } a_0 \neq m_0, & \text{to } A_1 = A + M. \end{array}$$

Tablica 2-4

a_0	m_0	$\|A\| < \|M\|$ wtedy i tylko wtedy, gdy
0	0	$a_0^1 = 1$
0	1	$a_0^1 = 1$
1	0	$a_0^1 = 0$
1	1	$a_0^1 = 0$

Oznaczając bit uzupełnienia A_1 (czyli bit o wskaźniku zero) przez a_0^1 możemy scharakteryzować za pomocą tabl. 2-4 wszystkie przypadki spełniania nierówności (2-1).

W przypadku gdy nie jest spełniona nierówność (2-1); należy:

$$\text{jeśli } a_0 = m_0, \quad \text{to } A_1 + M = A - M + M = A,$$
$$\text{jeśli } a_0 \neq m_0, \quad \text{to } A_1 - M = A + M - M = A,$$

po czym musi być generowany sygnał nadmiaru.

W przypadku spełnienia nierówności (2-1) wykonujemy kolejne kroki algorytmu.

Pierwszy krok algorytmu:

$$\text{jeśli } a_0^1 = m_0, \quad \text{to } A_2 = 2A_1 - M \quad \text{oraz } I_1 = 1,$$
$$\text{jeśli } a_0^1 \neq m_0, \quad \text{to } A_2 = 2A_1 + M \quad \text{oraz } I_1 = 0.$$

k-ty krok algorytmu (gdzie $k = 2,3,\ldots,n$):

$$\text{jeśli } a_0^k = m_0, \quad \text{to } A_{k+1} = 2A_k - M \quad \text{oraz } I_k = I_{k-1} + 2^{-k+1},$$
$$\text{jeśli } a_0^k \neq m_0, \quad \text{to } A_{k+1} = 2A_k + M \quad \text{oraz } I_k = I_{k-1}.$$

Przez a_0^k oznaczyliśmy bit uzupełnienia (czyli bit o wskaźniku zero) liczby A_k. Iloraz $I = I_n$, zaś reszta z dzielenia R wyraża się wzorem:

$$\text{jeśli } a_0^{n+1} = m_0, \quad \text{to } R = 2^{-n}\,(A_{n+1} - M),$$
$$\text{jeśli } a_0^{n+1} \neq m_0, \quad \text{to } R = 2^{-n}\,(A_{n+1} + M).$$

Przez a_0^{n+1} oznaczyliśmy bit uzupełnienia (czyli bit o wskaźniku zero) liczby A_{n+1}.

Przykład 2-4.

$$A = -\frac{3}{8} = 1.101, \quad M = \frac{1}{2} = 0.100$$

Sprawdzenie nierówności (2-1):

$$1 \neq 0, \quad A_1 = 0.001,$$

jak widać z tabl. 2-4, nierówność jest spełniona, wobec czego możemy przystąpić do dzielenia.

pierwszy krok: $0 = 0$, $A_2 = 1.110$, $I_1 = 1.000$,
drugi krok: $1 \neq 0$, $A_3 = 0.000$, $I_2 = 1.000$,
trzeci krok: $0 = 0$, $A_4 = 1.100$, $I_3 = 1.000$,
czwarty krok: $1 \neq 0$, $A_5 = 1.100$, $I_4 = 1.010$,

$$I = I_4 = 1.010 = -\frac{3}{4},$$

$$R = 2^{-4}\ (1.100 + 0.100) = 0.0000.$$

Przykład 2-5.

$$A = -\frac{9}{64} = 1.110111, \quad M = \frac{3}{8} = 0.011.$$

Sprawdzenie nierówności (2-1):

$$1 \neq 0, \quad A_1 = 0{,}001111,$$

jak widać z tabl. 2-4, nierówność jest spełniona, wobec czego możemy przystąpić do dzielenia.

pierwszy krok:	$0 = 0$,	$A_2 = 0.00011$,	$I_1 = 1.00000$,
drugi krok:	$0 = 0$,	$A_3 = 1.1101$,	$I_2 = 1.10000$,
trzeci krok:	$1 \neq 0$,	$A_4 = 0{,}000$,	$I_3 = 1.10000$,
czwarty krok:	$0 = 0$,	$A_5 = 1.101$,	$I_4 = 1.10100$,
piąty krok:	$1 \neq 0$,	$A_6 = 1.101$,	$I_5 = 1.10100$,
szósty krok:	$1 \neq 0$,	$A_7 = 1.101$,	$I_6 = 1.10100$,

$$I = I_6 = 1.10100 = -\frac{3}{8},$$

$$R = 2^{-6}\ (1.101 + 0.011) = 0.00000.$$

2.2. KRÓTKI OPIS MASZYNY TYPOWEJ

2.2.0. Obecnie omówimy zasady organizacji małej UPMC. Maszyna taka mimo niskich parametrów i prostoty organizacji, wystarczy do wyłożenia zasadniczych elementów programowania w całej rozciągłości. Przedstawiona niżej typowa mała UPMC będzie maszyną szeregową stałoprzecinkową, pracującą w binarnej arytmetyce uzupełnieniowej. Maszyna wykonuje operacje na liczbach dwójkowo wymiernych z przedziału $<-1,\ 1-2^{-33}>$.

W maszynie wyróżniamy dwa rodzaje słów:

1) słowa krótkie o długości 17 pozycji binarnych — w skrócie 17 B[1],

2) słowa długie o długości 34 pozycji binarnych — w skrócie 34 B.

Rozkazy (instrukcje) dla maszyny są kodowane binarnie i przedstawione za pomocą słów krótkich.

Maszyna może wykonywać wszystkie operacje arytmetyczne, logiczne zarówno na słowach krótkich, jak i na słowach długich. Zasadniczo słowa 17 bitowe służą do przedstawiania rozkazów, a słowa 34 bitowe — liczb. Gdy używamy słowa 17 bitowe jako

[1] Literę B będziemy czytali bit, 17B czytamy 17 bitów.