mgr inż. JACEK DUNAJ
mgr inż. KAZIMIERZ MALISZEWSKI
Przemysłowy Instytut Automatyki
i Pomiarów MERA–PIAP
Warszawa

BIBLIOTEKA PROCEDUR MATEMATYCZNYCH — BIBLIOTEKA „P"

Wersja 1

Niniejsza praca zawiera opis opracowanej w Przemysłowym Instytucie Automatyki i Pomiarów biblioteki matematycznej realizującej podstawowe operacje i funkcje zmiennoprzecinkowe na mikroprocesorze Intel 8086. W pracy zdefiniowano minimalny zbiór formatów danych stało- i zmiennoprzecinkowych oraz opisano zrealizowane operacje. Za wzór przyjęto dane i operacje realizowane przez procesor (koprocesor) arytmetyczny Intel 8087.

1. Wprowadzenie

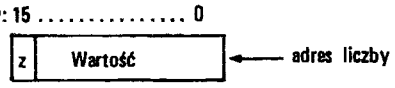
Koprocesor arytmetyczny Intel 8087 realizuje normę IEEE na arytmetykę zmiennoprzecinkową [5,7]. W dalszym tekście podaje się często w nawiasach angielskie nazwy terminów używane w opisie koprocesora 8087; wprowadzono też kilka nowych nazw. Niniejsza realizacja nie zakłada emulacji koprocesora: nie wprowadza się stosu zmiennoprzecinkowego, środowiska (enyironment) i nie realizuje się instrukcji koprocesora. Określa się natomiast zbiór procedur wzorowany w pewnym zakresie na instrukcjach koprocesora. Procedury te wywołuje się w języku assemblera instrukcjami CALL, wraz z przekazaniem adresów argumentów przez stos procesora Intel 8086. Zakłada się, że dana kopia biblioteki może być w danej chwili wykorzystywana tylko przez jedno zadanie jednego procesora Intel 8086. Poza przypadkiem opisanym w rozdziale 7 nie wolno przerwać programu wykorzystującego bibliotekę i użyć jej przez inny program.

2. Formaty danych

Niniejszy rozdział zawiera opis formatów danych używanych przez bibliotekę „P". Nowe pojęcia użyte w opisie będą szczegółowo omówione w rozdziale 3.

We wszystkich formatach bity ponumerowane są od najmniej do najbardziej znaczącego, poczynając od zera. Adres danej jest adresem najmniej znaczącego bajtu, tj. skrajnego prawego, który zaczyna się bitem nr 0. Adresy kolejnych bajtów rosną monotonicznie w lewo. Najbardziej znaczący bit „z" oznacza znak: z=0 dodatni, z=1 ujemny. Formaty całkowite (p. 2.1, 2.2) przedstawiają liczby ujemne w uzupełnieniu do 2. Formaty zmiennoprzecinkowe mają postać znak—moduł. Formaty opisane w p. 2.3 i 2.4 nazywamy formatami upakowanymi, a opisany w p. 2.5 rozpakowanym. Formaty opisane w p. 2.1—2.3 obsługuje także koprocesor, natomiast formaty z p. 2.4 i 2.5 określono tylko dla biblioteki „P".

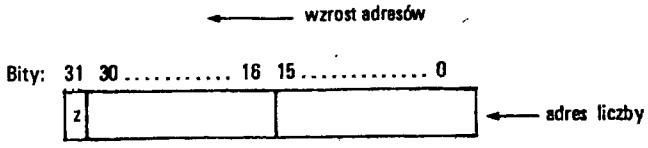
2.1. Liczba całkowita 16–bitowa, 1–słowowa (Word Integer)



Zakres: $-2^{15} \leqslant x \leqslant 2^{15} - 1$ tj. $-32768 \leqslant x \leqslant 32767$

Wartość -2^{15} (postać 10...0) jako argument wejściowy jest traktowana jako numeryczna. Gdy jest ona wynikiem, któremu nie towarzyszy sygnalizacja błędu I (patrz dalej) – ma wartość numeryczną. Natomiast w przypadku wyniku z błędem I jest to wartość całkowita nienumeryczna „niezdefiniowana" (integer indefinite). Używa się jej, gdy nie można dostarczyć prawidłowego wyniku.

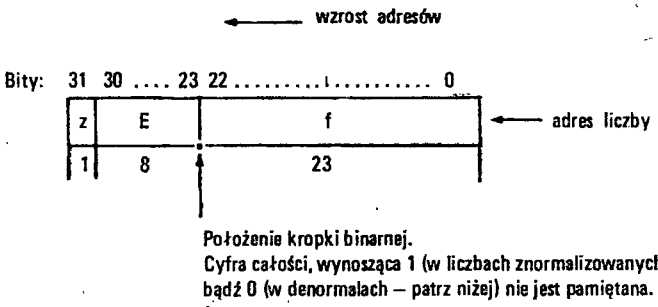
2.2. Liczba całkowita 32–bitowa, 2–słowowa (Short Integer)



Zakres: $-2^{31} \leqslant x \leqslant 2^{31} - 1$ tj. w przybliżeniu: $-2 * 10^9 \leqslant x \leqslant 2 * 10^9$

Wartość -2^{31} (postać 10...0) jest traktowana jako numeryczna, gdy jest argumentem wejściowym lub wyjściowym bez sygnalizacji błędu I. Jeżeli zaś jest to wynik, któremu towarzyszy sygnalizacja błędu I, to jest to wartość „nienumeryczna niezdefiniowana".

2.3. Liczba zmiennoprzecinkowa krótka (Short Real)



z – oznacza znak całej liczby. Liczba przeciwna do danej różni się od niej tylko znakiem.
E – pamiętany wykładnik
f – ułamek

Zakresy wartości pól:

$0 \leqslant E \leqslant MaxE = 255$

$0 \leqslant f \leqslant 1 - 2^{-23}$

Minimalna i maksymalna wartość pola E (0 i 255) określa specjalną interpretację liczby, jak podano dalej. Dla pozostałych wartości E jest przebazowanym wykładnikiem z bazą równą 127, tj. E =e + 127, gdzie e jest rzeczywistym wykładnikiem dwójki.

Wartość liczby określa tabela:

z	E	e	f	Wartość liczby	Rodzaj liczby
	$1 \leqslant E \leqslant 254$	/1/ $-126 \leqslant e \leqslant 127$	$0 \leqslant f \leqslant 1-2^{-23}$	$(-1)^z * 2^{-127} * 1.f$	Znormalizowanu (normal)
D o w o l n y	0	/1/ -126	$0 \leqslant f \leqslant 1-2^{-23}$	$(-1)^z * 2^{-126} * 0.f$	Denormal
	0	–	0	$(-1)^z * 0$	Zero ze znakiem
	255 (111 .... 1)	/2/ 128	0	$(-1)^z * \infty$	Nieskończoność ze znakiem
	255	128	≠ 0	–	Wartość nienumeryczna
1	255	128	1/2 (1000 ... 0)	–	Wartość nienumeryczna „niezdefiniowana"

Uwagi:

/1/ Wartość tę będziemy oznaczać jako eden formatu krótkiego,
/2/ Wartość tę będziemy oznaczać jako ensk formatu krótkiego. gdzie ensk = MaxE – baza.

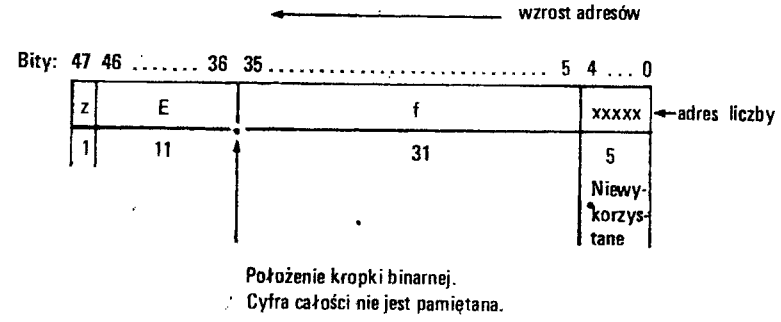
Zakres wartości modułu liczby zmiennoprzecikowej krótkiej x, różnej od 0:

normale: $2^{-126} \leqslant |x| \leqslant 2^{127} * (2-2^{-23}) < 2^{128}$; dokładność ok. 7 cyfr dziesiętnych

denormale: $2^{-149} \leqslant |x| \leqslant 2^{-126} * (1-2^{-23}) < 2^{-126}$

2.4. Liczba zmiennoprzecinkowa średnia (Middle Real)

Liczba zmiennoprzecinkowa średnia ma postać upakowaną w trzech słowach:



Znaczenie pól jest analogiczne, jak dla liczby zmiennoprzecinkowej krótkiej.
Zakresy wartości pól:

$0 \leqslant E \leqslant MaxE = 2047$

$0 \leqslant f \leqslant 1 - 2^{-31}$

Minimalna i maksymalna wartość pola E (0 i 2047) określa specjalną interpretację liczby, jak podano dalej. Dla pozostałych wartości liczbę traktuje się jako znormalizowaną i E jest przebazowanym wykładnikiem z bazą równą 1023, tj. E = e + 1023, gdzie e jest rzeczywistym wykładnikiem dwójki.

Wartość liczby określa tabela:

z	E	e	f	Wartość liczby	Rodzaj liczby
Dowolny	$1 \leqslant E \leqslant 2046$	/1/ $-1022 \leqslant e \leqslant 1023$	$0 \leqslant f \leqslant 1-2^{-31}$	$(-1)^z * 2^{E-1023} * 1.f$	Znormalizowana (normal)
	0	/1/ -1022	$0 \leqslant f \leqslant 1-2^{-31}$	$(-1)^z * 2^{-1022} * 0.f$	Denormal
	0	–	0	$(-1)^z * 0$	Zero ze znakiem
	2047 (111 .... 1)	/2/ 1024	0	$(-1)^z * \infty$	Nieskończoność ze znakiem
	2047	1024	≠ 0	–	Wartość nienumeryczna
1	2047	1024	1/2 (1000 ... 0)	–	Wartość nienumeryczna „niezdefiniowana"

Uwagi:

/1/ Jest to eden formatu średniego
/2/ Jest to ensk formatu średniego, ensk = MaxE – 1023

Zakres wartości modułu liczby zmiennoprzecinkowej średniej x, różnej od 0:

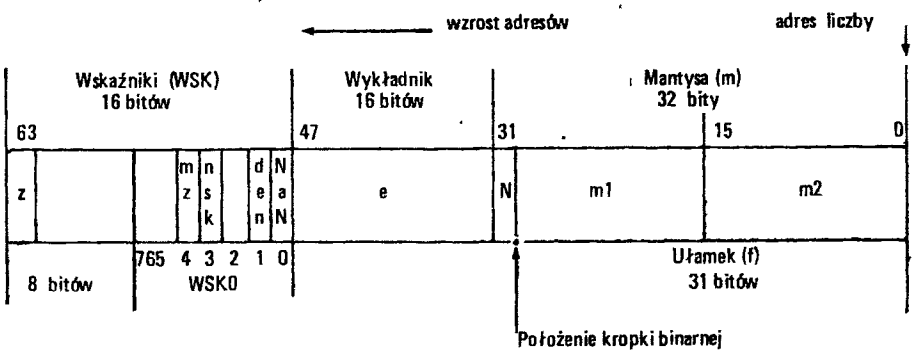
normale: $2^{-1022} \leqslant |x| \leqslant 2^{1023} * (2-2^{-31}) < 2^{1024}$; dokładność ok. 9 cyfr dzies.

denormale: $2^{-1053} \leqslant |x| \leqslant 2^{-1022} * (1-2^{-31}) < 2^{-1022}$

Bity 0 – 4 najmniej znaczącego bajtu liczby są ignorowane przy odczycie i nie zmieniane przez bibliotekę przy zapisie do pamięci.

2.5. Liczba zmiennoprzecinkowa rozpakowana (Unpacked Real)

Biblioteka wykonuje obliczenia na liczbach o przedstawieniu wewnętrznym rozpakowanym w 4 słowach:



Opis pól:

z – znak liczby w zapisie znak–moduł: 0 oznacza „+", 1 oznacza „–"

Wskaźniki:

W młodszym bajcie najbardziej znaczącego słowa liczby (WSK0) ustawia się 1–bitowe wskaźniki określające rodzaj liczby. Nieustawienie żadnego wskaźnika oznacza, że liczba ma wartość numeryczną, określoną przez z–znak, e–wykładnik, m–mantysę. W danej liczbie może być ustawiony tylko jeden wskaźnik. Znaczenie wskaźników określa tabela podana niżej.

e – wykładnik rzeczywisty liczby (bez bazy)

m = N.f mantysa liczby z jawnym bitem całości N.

Wartość liczby określa tabela:

z	WSKO	e	m	Wartość liczby	Rodzaj liczby
Dowolny	0	$-32768 \leqslant e \leqslant 32767$	$1 \leqslant m \leqslant 2-2^{-31}$	$(-1)^z * 2^e * m$	Normal różny od zera
	mz = 1	obojętne	obojętne	$(-1)^z * 0$	Zero normalne ze znakiem
	nsk = 1	obojętne	obojętne	$(-1)^z * \infty$	Nieskończoność ze znakiem
	/2/ den = 1	/1/ $-32768 \leqslant e \leqslant 32767$	$1 \leqslant m \leqslant 2-2^{-31}$	$(-1)^z * 2^e * m$	Normal powstały z rozpakowania i normalizacji denormala
	NaN = 1	obojętne	N = 1, f ≠ 0	–	Wartość nienumeryczna
1	NaN = 1	obojętne	N = 1, f = 1/2 (1100 ... 0)	–	Wartość nienumeryczna „niezdefiniowana"

Uwagi:

/1/ Zakres e dopuszczalny jak dla wszystkich normali, jakkolwiek po normalizacji denormala formatu upakowanego krótkiego lub średniego otrzymuje się: $-1053 \leqslant e \leqslant -127$

/2/ Bit „den" może być ustawiony tylko w wyniku podprogramu rozpakowującego, a nigdy w wyniku operacji arytmetycznej.

Zakres wartości modułu liczby zmiennoprzecinkowej rozpakowanej x różnej od zera wynosi:

$$2^{-32768} \leqslant |x| \leqslant 2^{32767} * (2-2^{-31}) < 2^{32768}$$

dokładność około 9 cyfr dziesiętnych.

W prawidłowych obliczeniach liczba o wartości numerycznej jest zawsze znormalizowana (N=1). Próba działania na nieznormalizowanym argumencie rozpakowanym powoduje sygnalizację błędu „niedozwolona operacja". Liczba może zostać zdenormalizowana tylko dla zapakowania niedomiarowego wyniku do formatu upakowanego, ale wtedy wynik rozpakowany ginie.

Zaletą formatu rozpakowanego jest większa szybkość przetwarzania w porównaniu z formatami upakowanymi i mniejsze prawdopodobieństwo powstania nadmiaru, dzięki szerokiemu zakresowi wykładnika. Postać ta nadaje się szczególnie do obliczeń pośrednich w złożonych operacjach o argumentach i wynikach w formacie upakowanym krótkim.

3. Określenie pojęć i zasad działania na liczbach zmiennoprzecinkowych

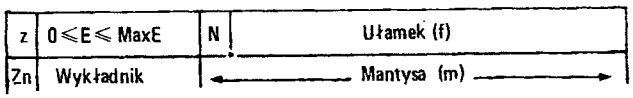
Niniejsza arytmetyka zmiennoprzecinkowa, wynikająca z normy [5,7], różni się bardzo od tradycyjnego rozwiązania, np. maszyn rodziny PDP–11. Do nowych pojęć należą liczby zdenormalizowane (denormale), realizujące tzw. stopniowy niedomiar (gradual underflow), następnie liczby anormalne (unnormals) oraz reprezentacje nieskończoności i wartości nienumerycznych. Nowe jest też ustalenie, że wiodący bit mantysy (oznaczony w niniejszym opisie przez N) ma wagę 1, a nie 1/2 jak dotychczas. Ponadto baza dodana do wykładnika, równa dla formatu krótkiego 127 (zamiast 128) powoduje, że iloczyn modułów największej i najmniejszej niezerowej liczby znormalizowanej wynosi prawie 4 (a nie – prawie 2). Zapisując to w uproszczonej formie mamy:

$$min = 4/Max \quad lub \quad MAX = 4/min.$$

Zatem, jeżeli x jest liczbą znormalizowaną u dołu zakresu normali, to wyrażenia 1/x, 2/x, 3/x i π/x (które są mniejsze od 4/min=MAX) nie spowodują nadmiaru. Natomiast, jeśli x jest liczbą znormalizowaną u góry zakresu, to podobne wyrazy są nieco mniejsze od 4/MAX=min i będą reprezentowane ze zmniejszoną dokładnością w dziedzinie denormali.

Podamy teraz kilka definicji dotyczących liczb zmiennoprzecinkowych upakowanych, określając też bit całości N, który w liczbie jawnie nie występuje (podobnie jak i zaznaczona na rysunku kropka binarna).

Postać liczby jest następująca:



Liczba składa się z jednobitowego znaku z, wykładnika E, domyślnego bitu całości N i ułamka f.

Mantysą m będziemy nazywać liczbę m=N.f (w całym niniejszym opisie kropka oddziela część całkowitą od ułamkowej).

MaxE=111...1 jest największą możliwą wartością wykładnika i wynosi 255 dla formatu krótkiego, zaś 2047 dla średniego.

Minimalna (0) i maksymalna (MaxE) wartość wykładnika określa specjalną interpretację liczby. Dla pozostałych wartości, tj. $0 < E < MaxE$, E jest przebazowanym wykładnikiem: E=e+B, gdzie e jest rzeczywistą potęgą dwójki, a B jest bazą, równą dla formatu krótkiego 127, a dla średniego 1023.

LICZBA ZNORMALIZOWANA (normal) ma przebazowany wykładnik w zakresie $0<E<MaxE$ oraz mantysę bez wiodących zer, tj. z bitem N = 1. Wartość liczby wynosi:

$$x = (-1)^z * 2^{E-B} * m = (-1)^z * 2^{E-B} * 1.f$$

ZERO ZNORMALIZOWANE (true zero) ma pola E, m równe 0 i wartość $(-1)^z * 0$. Znak zera jest w miarę możności uwzględniany w obliczeniach, jednak oba zera, dodatnie i ujemne, należy uważać za tę samą liczbę i w porównaniu przyjmować za równe.

LICZBA ANORMALNA (unnormal) ma zakres wykładnika taki, jak liczba znormalizowana, a w mantysie co najmniej jedno zero wiodące, a więc N=0. Wartość liczby x wynosi:

$$x = (-1)^z * 2^{E-B} * m = (-1)^z * 2^{E-B} * 0.f, \text{ gdzie } 0 < E < MaxE$$

Taka liczba może istnieć tylko w formacie wewnętrznym, z jawnym bitem N, który rozróżnia normale od anormali. Koprocesor dopuszcza tę postać, biblioteka „P" — nie.

ZERO ANORMALNE (pseudo zero) jest szczególnym przypadkiem anormala, gdy mantysa jest zerem przy $0 < E < MaxE$.

Liczby anormalne mogą powstać w wyniku operacji na normalach i denormalach, np. przy mnożeniu normala przez denormal. Koprocesor dopuszcza takie dane i nie traktuje wyniku anormalnego jako błędny, dopóki nie trzeba go przesłać w postaci upakowanej (gdyż anormal nie ma reprezentacji w formatach upakowanych). Niniejsza biblioteka, normalizując argumenty denormalne, wyklucza możliwość powstania wyniku anormalnego. Gdyby taki wynik jednak powstał, to będzie sygnalizowany błąd I, połączony w razie zamaskowania, z wydaniem wartości „niezdefiniowanej" (zob.dalej).

NORMALIZACJA polega na przesuwaniu w lewo mantysy liczby w formacie wewnętrznym wraz ze zmniejszaniem wykładnika o 1 przy przesunięciu o 1 bit. Normalizacja kończy się, gdy bit całości stanie się równy 1. Normalizacja zera anormalnego polega na zastąpieniu go przez zero znormalizowane (E=m=0).

DENORMALIZACJA jest operacją przeciwną do normalizacji. Ma zastosowanie przy pakowaniu normali o rzeczywistym wykładniku mniejszym od minimalnego wykładnika liczb znormalizowanych eden (eden = — 126 dla formatu krótkiego, —1022 dla średniego). Denormalizacja w postaci rozpakowanej liczby polega na przesuwaniu mantysy w prawo ze zwiększaniem wykładnika o 1 co 1 bit.

Proces kończy się, gdy wykładnik osiągnie wartość minimalną dla liczb znormalizowanych, eden, bądź mantysa się całkowicie wyzeruje. W pierwszym przypadku otrzymuje się denormal, dla któreqo e = eden, m = 0.f, w drugim przypadku zaokrąglona liczba jest zerem znormalizowanym.
Moduł denormala wynosi:

$$2^{eden} * 0.f$$

Natomiast moduł normala o minimalnym wykładniku wynosi:

$$2^{eden.} * 1.f$$

Z powyższych wzorów wynika, że przy hipotetycznym zmnieszaniu modułu liczby, po osiągnięciu wykładnika eden, liczba dąży do zera nie przez zmniejszanie wykładnika, ale mantysy. Ponieważ bitu N się nie pamięta, więc dla odróżnienia denormali od normali o wykładniku eden przyjęto

z definicji dla denormali przebazowany wykładnik E = eden + B – 1 = 0, podczas gdy dla normali minimalny przebazowany wykładnik jest E = eden B = 1 (gdyż eden = – B + 1). Wykładnik E = 0 przy f≠0 świadczy więc o denormalności liczby. W tradycyjnej realizacji arytmetyki zmiennoprzecinkowej (bez denormali) wykładnik E = 0 określał tylko liczbę równą 0, niezależnie od wartości f. Było tak dlatego, że zero zmiennoprzecinkowe (E = N = f = 0) nie mogło być odróżnione od liczby (E = 0, N = 1 f = 0). „Marnowały się" przez to reprezentacje o wartościach E = 0, f≠0 i przy zmniejszaniu modułu liczby po dojściu do E = 1, f = 0 tj.

$$2^{1-B} * 1.0,$$

następował skok do 0. Niniejsze rozwiązanie po dojściu do

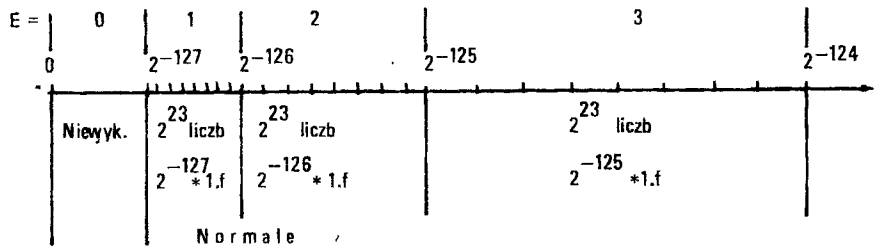
$$2^{1-B} = 2^{eden}$$

w dziedzinie normali, realizuje płynne przejście do 0 w dziedzinie denormali, tj. liczb

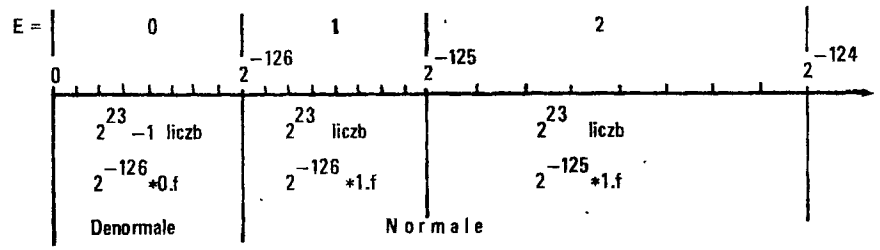
$$2^{eden} * 0.f.$$

Ceną tego rozwiązania jest mniejsza dokładność liczb denormalnych.

Poniższe rysunki porównują dół zakresu modułów liczb zmiennoprzecinkowych krótkich dla rozwiązania tradycyjnego (tylko normale, E = e + 128) i przyjętego w bibliotece „P" (denormale, a dla normali E = e + 127).



Rozwiązanie tradycyjne (PDP–11)

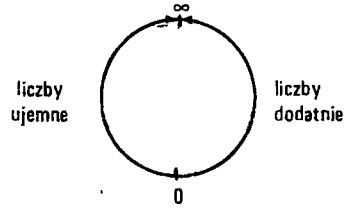


Biblioteka „P"

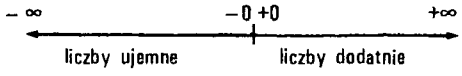
Reasumując: jeżeli liczba jest skończona i różna od zera, to jest bądź normalem (N = 1), bądź nieznormalizowana (N = 0). Liczby nieznormalizowane dzielimy na anormale (o wykładnikach jak dla normali) i denormale (o wykładniku rzeczywistym eden równym dolnej granicy wykładników liczb znormalizowanych, a wykładniku przebazowanym E równym 0).

NIESKOŃCZONOŚĆ ZE ZNAKIEM, dla której zarezerwowano maksymalną wartość wykładnika przy zerowym ułamku (E = maxE, N = 1, f = 0) należy interpretować jako przedstawienie liczby o dowolnie dużym module. Nieskończoność pojawia się w programie zwykle jako odpowiedź zamaskowana na błąd nadmiaru lub dzielenia przez zero.

Arytmetykę nieskończonych argumentów w niniejszej bibliotece realizuje się w trybie projekcyjnym (projective), który jest domyślny dla koprocesora 8087. Norma i koprocesor dają ponadto możliwość wyboru trybu afinicznego (affine). W trybie afinicznym: $-\infty <$ wartość skończona $< +\infty$, zaś w trybie projekcyjnym: $-\infty = +\infty = \infty$ i nie ma uporządkowania między ∞ i wartością skończoną (ponieważ znak nieskończoności uważa się za nieznany). Ilustruje to rysunek:



Tryb projekcyjny



Tryb afiniczny

W trybie projekcyjnym obie wartości $+\infty$ i $-\infty$ należy traktować jako jedną nieoznakowaną nieskończoność (podobnie do traktowania +0 i –0). Tryb afiniczny zawiera więcej informacji, ale może wprowadzić złą informację, gdy np. wynik obliczenia x wynosi +0 lub –0 (ta sama wartość liczbowa). W trybie projekcyjnym $1/x = \infty$, zaś w trybie afinicznym $+\infty$ lub $-\infty$, czyli dwie różne wartości.

W trybie afinicznym, tak jak w zwykłej algebrze, nieokreślona jest różnica nieskończoności o równych znakach. W trybie projekcyjnym nieokreślona jest zarówno różnica jak i suma dwóch nieskończoności, gdyż znaki argumentów uznaje się za niepewne.

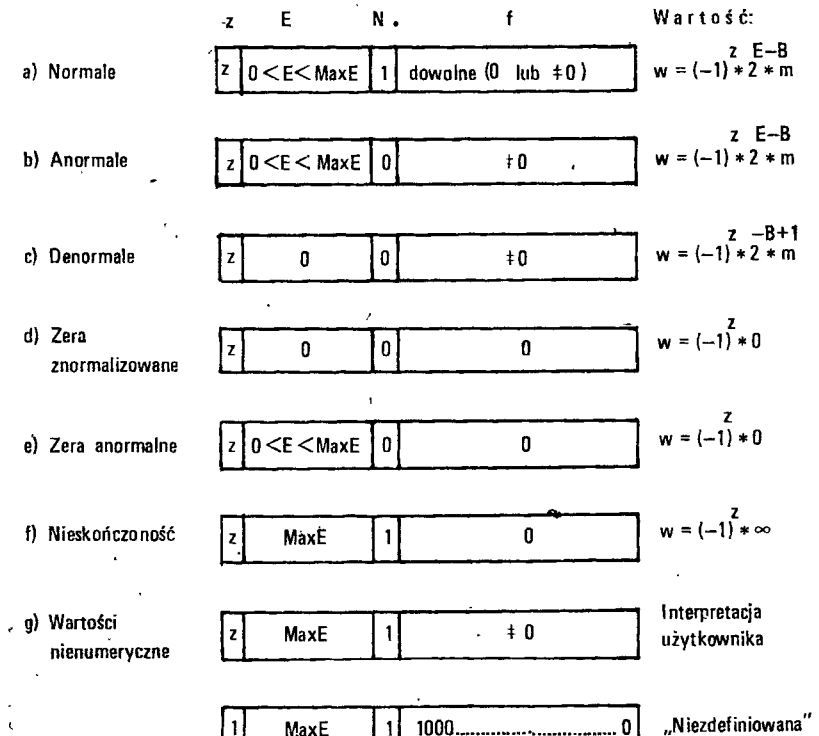
Niniejsza biblioteka realizuje tryb projekcyjny dość „łagodnie". Mianowicie uwzględnia się znaki zera i nieskończoności w operacjach arytmetycznych (np. $1/(+0)=+\infty$, $1/(-0)=-\infty$, $+0/(-\infty)=-0$), zachowując maksimum informacji. Pozostawia się programowi użytkowemu jednakowe traktowanie –0 i +0 oraz $-\infty$ i ∞. Faktycznie więc tryb projekcyjny narzucony przez bibliotekę objawia się tylko w traktowaniu operacji typu $\infty+\infty$ i $\infty-\infty$ jako niedozwolonych, niezależnie od znaków argumentów.

WARTOŚĆ NIENUMERYCZNA (Not–a–Number, NaN) może być przypisana liczbie zmiennoprzecinkowej upakowanej przez nadanie bazowanemu wykładnikowi E maksymalnej wartości dla danego formatu, MaxE, zaś ułamkowi f dowolnej niezerowej wartości. Znak liczby może być dowolny. Bit N, podobnie jak dla nieskończoności, przyjmuje się przy rozpakowaniu za równy 1. Podstawowe operacje arytmetyczne można wywoływać z argumentami nienumerycznymi. Przy zamaskowanym błędzie I (zob. dalej) argument nienumeryczny przenosi się do wyniku. Jeżeli dwa argumenty są

nienumeryczne – za wynik podstawia się ten, który ma większy moduł (a więc f). Ten mechanizm może być użyty do wykrycia wartości niezainicjowanych, jeżeli wszystkim zmiennym nada się pierwotnie wartość NaN.

WARTOŚĆ NIEZDEFINIOWANA (indefinite) jest wybraną wartością nienumeryczną, którą biblioteka podstawia za wynik operacji, kiedy nie można dostarczyć wyniku liczbowego ani o wartości nieskończonej (np. za ∞–∞ lub 0*∞). Wartość nienumeryczna niezdefiniowana jest reprezentowana przez z = 1, E = MaxE dla danego formatu, m = 1.1000...0 (tj. f = .1000...0). Zaleca się nie używać tej postaci dla innych NaN–ów.

Poniższy rysunek zestawia wszystkie rodzaje danych zmiennoprzecinkowych. Pola z,E,f występują faktycznie w postaci upakowanej, pole N nie występuje i podano je tylko dla informacji. Mantysa m wynosi m = N.f, gdzie f jest pamiętanym ułamkiem. MaxE wynosi 255 dla formatu krótkiego i 2047 dla średniego, B wynosi odpowiednio 127 i 1023. Znak z jest dowolny (0 oznacza +, 1 oznacza –).



Dane anormalne (b, e) istnieją tylko w formacie wewnętrznym (Temporary Real) koprocesora 8087; biblioteka „P" ich nie używa. W formatach upakowanych (bez bitu N) nie mogą być one pamiętane, gdyż nie różniłyby się od wartości typu a.

Zaokrąglenie

Podstawowe operacje arytmetyczne są wykonywane tak, jak by się to działo z pełną dokładnością, a następnie, po normalizacji, jeżeli ułamek wyniku jest dłuższy od przeznaczonego dla niego pola, to liczba podlega zaokrągleniu. Niniejsza biblioteka wykonuje zaokrąglenie liczby według algorytmu „do najbliższej lub parzystej" (round to nearest or even), który jest domyślny dla procesora 8087. Zaokrąglenie wykonuje się na module liczby, tak samo dla liczb dodatnich i ujemnych. Jeżeli część odrzucana jest mniejsza od połowy wagi najmniej znaczącego bitu, to moduł liczby zaokrągla się w dół, jeżeli większa, to w górę, przez dodanie 1 do najmniej znaczącego bitu. Jeżeli część odrzucana jest równa dokładnie 1/2 ostatniego pamiętanego bitu, to zaokrągla się do liczby parzystej, tzn. jeżeli ostatnia pamiętana cyfra jest 0, to pozostaje bez zmian, a jeżeli 1, to dodaje się 1. Przykładowo, przy zaokrąglaniu do całości w systemie dziesiętnym: 13.5≈14, 14.5≈14. Ten sposób zaokrąglania liczb zakończonych przez 1/2 jest bardziej sprawiedliwy od tradycyjnego (w górę), co można przewidzieć przy sumowaniu dużej liczby zaokrąglonych wartości.

Norma i koprocesor umożliwiają ponadto inne sposoby zaokrąglenia, tu nie uwzględnione. Podobnie nie umożliwia się wyboru precyzji obliczeń wewnętrznych innej niż 32 bity (koprocesor umożliwia różne precyzje).

4. Określenie błędów

W ślad za procesorem 8087 biblioteka „P" wykrywa i obsługuje 6 rodzajów błędów (exceptions):

- niedozwolona operacja (invalid operation, I)
- nadmiar (overflow, O)
- dzielenie przez zero (zerodivide, Z)
- niedomiar (underflow, U)
- argument zdenormalizowany (denormalized, D)
- precyzja (precision, P) czyli niedokładny wynik (inexact result).

Błąd „niedozwolona operacja" jest zwykle fatalny, natomiast wiele obliczeń może dać wyniki możliwe do przyjęcia pomimo wystąpienia pozostałych błędów.

Omówimy je kolejno.

1. NIEDOZWOLONA OPERACJA (I) ma miejsce, gdy argument lub układ argumentów jest niedozwolony dla danej operacji, argument jest nienumeryczny, argument rozpakowany jest nieznormalizowany lub ma niedozwolone wskaźniki, albo gdy wynik całkowity nie może być zapamiętany w danym formacie. Jeżeli ma być wydany wynik — podstawia się wartość nienumeryczną. Przykładowo: operacje 0/0, $\infty-\infty$, $\sqrt{-5.2}$, NaN + 3.0 są niedozwolone. Argument nienumeryczny (NaN) w razie zamaskowania błędu I przechodzi do wyniku. Dla dwóch argumentów NaN — ten, który ma większy moduł. Jeżeli żaden z argumentów nie jest NaN, to w przypadku zamaskowanego błędu I za wynik podstawia się wartość nienumeryczną „niezdefiniowaną". W przypadku wyniku całkowitego operacja jest niedozwolona, gdy wewnętrzna postać zmiennoprzecinkowa wyniku jest denormalem, anormalem, nieskończonością, NaN lub przekracza zakres przeznaczenia. W razie zamaskowania błędu I wydaje się wartość całkowitą „nienumeryczną niezdefiniowaną".

2. NADMIAR (O) występuje, gdy wykładnik zmiennoprzecinkowego wyniku jest za duży, by zmieścił się w formacie przeznaczenia. Obsługa wewnętrzna tego błędu polega na podstawieniu za wynik nieskończoności z zachowaniem znaku. Sygnalizuje się przy tym błędy O i P, gdyż wartość skończona zastępowana jest przez nieskończoną.
   Uwaga: Przy przekroczeniu zakresu wyniku stałoprzecinkowego sygnalizuje się błąd I,nie O.

3. DZIELENIE PRZEZ ZERO (Z) oznacza próbę dzielenia skończonej, różnej od zera liczby przez zero. Obsługa wewnętrzna (zamaskowana) tego błędu podstawia za wynik nieskończoność o znaku będącym różnicą symetryczną znaków dzielnej i dzielnika (jak dla prawidłowych argumentów).
   Uwaga: Dzielenie 0/0 uważa się za błąd I,nie Z.

4. NIEDOMIAR (U) występuje, gdy różny od zera znormalizowany wynik ma wykładnik za mały, by mógł być zapamiętany w formacie przeznaczenia, tj. e<eden (eden = — 126 dla formatu krótkiego, a —1022 dla średniego). Obsługa wewnętrzna tego błędu polega na denormalizacji wyniku (w razie potrzeby aż do zera). Otrzymuje się denormal lub znormalizowane zero, z sygnalizacją błędu U i zwykle P (gdy mantysa traci z prawej strony bity niezerowe). Denormalizacja realizuje tzw. stopniowy niedomiar (gradual underflow). Zamiast tradycyjnego zredukowania wyniku do zera, pozwala się dalej prowadzić obliczenia, choć z mniejszą dokładnością, gdyż przesunięta w prawo mantysa ma mniej cyfr znaczących.
   Uwaga: W formacie rozpakowanym gdy e<—32 k (tzw. superniedomiar), nie stosuje się denormalizacji, ale przy obsłudze wewnętrznej podstawia się za wynik 0 z sygnalizacją błędów U i P.

5. ARGUMENT ZDENORMALIZOWANY (D) oznacza, że przynajmniej jeden z argumentów jest denormałem lub (dla argumentów rozpakowanych powstał z rozpakowania denormala. Odpowiedź zamaskowana na ten błąd polega na normalizacji denormalnego argumentu w czasie rozpakowania. Danej się nie zmienia.

6. PRECYZJA (P) czyli niedokładny wynik ma miejsce wtedy, gdy wyniku nie można przedstawić dokładnie w formacie przeznaczenia. To znaczy dla formatów całkowitych — gdy wynik nie jest całkowity, zaś dla zmiennoprzecinkowych — gdy znormalizowana mantysa ma ponad 24 lub 32 bity znaczące. Wynik podlega zawsze zaokrągleniu według podanego uprzednio algorytmu. Z reguły przy działaniu na liczbach zmiennoprzecinkowych, które z natury przedstawiają prawidłową wartość w przybliżeniu, jest to zjawisko normalne.

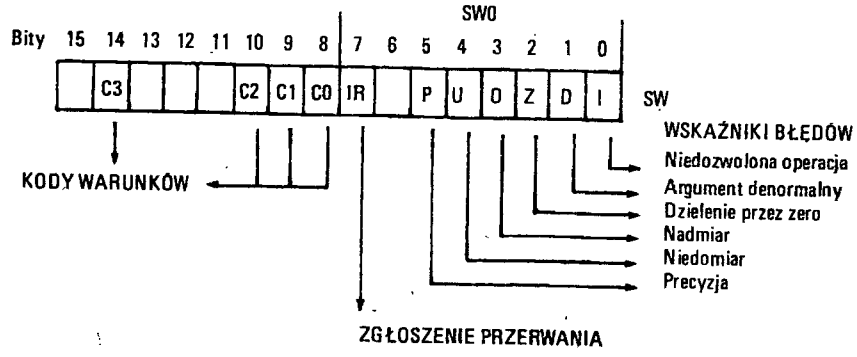
5. Wewnętrzne rejestry biblioteki

Użytkownik ma dostęp do następujących wewnętrznych rejestrów biblioteki: słowa stanu, słowa sterującego i akumulatora wewnętrznego.

Słowo stanu

Słowo stanu (Status Word, SW) wzorowane jest na słowie stanu koprocesora. Pewne procedury, na razie nie zrealizowane, mają ustawiać w nim bity C0—C3 warunków (stanów) określających wynik operacji. W mniej znaczącym bajcie słowa, SW0, ustawiane są wskaźniki (flagi) błędów, które mogą wystąpić w operacjach. Pojawienie się błędu powoduje ustawienie odpowiedniego bitu na 1. Bit 7 — zgłoszenie przerwania (Interrupt Request, IR) jest ustawiany łącznie z bitem konkretnego błędu, gdy ten jest odmaskowany (zob. dalej). Wiąże się to z obsługą błędu przez przerwanie.

Postać słowa stanu jest następująca:



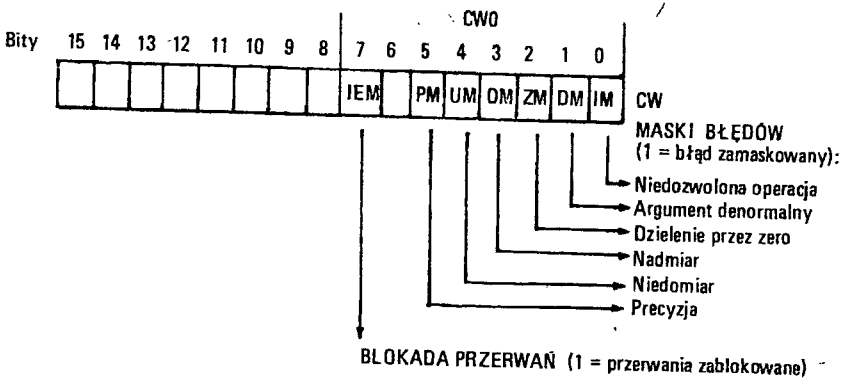
Uwaga:

Bity C0–C3 oraz bity nie opisane nie są aktualnie wykorzystane. Nie można zakładać, że mają np. wartość 0, gdyż w przyszłości mogą mieć znaczenie.

Użytkownik może odczytać słowo stanu operacją PSTSW i zerować jego młodszy bajt (SW0) operacją PCLEX.

Słowo sterujące

Słowo sterujące (Control Word, CW) jest wzorowane na słowie sterującym procesora 8087. Aktualnie ma znaczenie tylko młodszy bajt, CW0, w którym można ustawiać bity 0–6 maskujące poszczególne rodzaje błędów w SW0. Bit 7 (Interrupt Enable Mask, IEM) jest maską dopuszczającą lub blokującą obsługę wszystkich przerwań od błędów.
Postać słowa sterującego:



Użytkownik może odczytywać słowo sterujące procedurą PSTCW i nadawać mu wartość procedurą PLDCW. Może też ustawiać maskę IEM procedurą PDISI i zerować ją procedurą PENIN.

Wewnętrzny akumulator biblioteki

Wewnętrzny akumulator biblioteki, WRL (Wynik Rozpakowany Lokalny) jest tablicą, w której podstawowe i pomocnicze procedury matematyczne składają wynik operacji rozpakowanej przed wywołaniem handlera obsługi błędu (zob. dalej). Użytkownik może przy pomocy procedury PSTWRL odczytać zawartość akumulatora jako liczbę zmiennoprzecinkową rozpakowaną. Koprocesor 8087 nie używa takiego rejestru.

6. Obsługa błędów

Wykrycie przez procedurę biblioteczną któregoś z błędów opisanych w rozdziale 4 powoduje ustawienie w słowie stanu wskaźnika tego błędu na 1. Odpowiadający bit maski w słowie sterującym decyduje o sposobie obsługi błędu. Jeżeli maska ma wartość 1, to mówimy, że błąd jest zamaskowany (od programu użytkowego), jeżeli 0 — błąd jest odmaskowany. W pierwszym przypadku biblioteka daje na błąd odpowiedź zamaskowaną (masked response) i operacja jest kontynuowana. W drugim przypadku daje się odpowiedź odmaskowaną, sprowadzającą się (poza przypadkiem P — zob. dalej) do przerwania (trap) obsługiwanego przez program użytkownika (handler). Odpowiedź zamaskowaną będziemy nazywać obsługą wewnętrzną błędu, zaś wykonanie programu obsługi przerwania — obsługą zewnętrzną.

Maska blokady przerwań, IEM w CW, ma znaczenie tylko przy odpowiedzi odmaskowanej i decyduje o wykonaniu lub nie obsługi zewnętrznej. Mianowicie dla IEM = 0 przerwania są dopuszczone (enabled) i w razie ustawienia IR w SW z powodu któregokolwiek błędu odmaskowanego następuje skok do handlera instrukcją przerwania programowego. Natomiast dla IEM = 1 przerwania są zablokowane (disabled). Przechodzi się wtedy do następnej instrukcji za wywołaniem procedury, pomijając wykonanie handlera. Błąd powoduje tylko zakończenie operacji (z wyjątkiem błędu P, kiedy wynik się zaokrągla), gdyż nie ma ani obsługi wewnętrznej, ani zewnętrznej. Użytkownik może jednak stwierdzić błąd badając SW0 i wykonać własną obsługę bez mechanizmu przerwań. Działanie IEM jest analogiczne do blokady przerwań procesora Intel 8086 przez flagę IF (przy odwróconej konwencji, gdyż IF = 1 dopuszcza, a IF = 0 blokuje przerwania). Nawiasem mówiąc, flaga IF nie ma wpływu na przerwanie programowe wykorzystywane przez bibliotekę. Natomiast przerwanie z rzeczywistego procesora 8087 podłączone na wejście INTR przez układ 8259A może być zablokowane flagą IF.

Podsumujmy. Jeżeli indywidualna maska błędu w CW0 wynosi 1, to odpowiedź na błąd jest wewnętrzna i operacja jest kontynuowana. Jeżeli maska indywidualna wynosi 0, to błąd powoduje przerwanie operacji (tylko w przypadku P kończy się operację) i następuje skok do handlera, ale pod warunkiem, że IEM = 0. W przeciwnym wypadku (IEM = 1) handler jest pomijany i następuje powrót z wywołanej procedury.

Po wykonaniu handlera biblioteka natychmiast kończy operację tzn. wykonuje powrót z wywołanej procedury. Nie ma mowy o kontynuowaniu operacji po obsłudze zewnętrznej.

Wskaźniki błędów ustawiane w słowie stanu zachowują swą wartość do czasu ich wyzerowania przez użytkownika procedurą PCLEX, a zatem SW0 zawiera zbiór błędów wykrytych od ostatniego zerowania.

Użytkownik może pewną grupę błędów obsługiwać handlerem, a inne pozostawić obsłudze wewnętrznej. Z kolei błędy obsługiwane wewnętrznie można badać w SW0 po każdej operacji lub po

ciągu operacji zapoczątkowanym zerowaniem SW0. Można też, przy blokadzie przerwań przez IEM = 1, po zbadaniu SW0 po powrocie z procedury, zrealizować indywidualną obsługę zewnętrzną błędów jakiegoś wywołania.

W jednej procedurze może wystąpić więcej niż jeden błąd, jeżeli po jego wykryciu operacja jest kontynuowana. Załóżmy, że przed daną operacją wyzerowano błędy, a następnie badamy SW0 w handlerze w ramach obsługi zewnętrznej jakiegoś błędu. Mechanizm przerwania, a tym samym zakończenia operacji, powoduje, że w SW0 może być tylko jeden błąd odmaskowany (który właśnie spowodował obsługę zewnętrzną) i wszystkie błędy zamaskowane wcześniejsze, które obsłużono wewnętrznie.

Ponadto obowiązuje zasada, że niezależnie od tego, czy przed daną operacją zerowano błędy, czy nie, w SW0 nie może być błędów odmaskowanych pochodzących z wcześniejszych operacji. Takie błędy bowiem powodują wejście do handlera w chwili powstania błędu lub w chwili jego odmaskowania. Z handlera zaś nie ma powrotu do przerwanego programu przed wyzerowaniem wszystkich odmaskowanych błędów (zob. rozdział 7 poz. 7).

Kolejność sygnalizacji błędów w ramach jednej operacji jest następująca:

1) D – gdy odmaskowany 2) I 3) Z 4) D – gdy zamaskowany	Przed obliczeniem wyniku
5) (P) O/U [6) I] 7)	Po obliczeniu wyniku

Błędy „przed" wykrywane są na podstawie argumentów. W chwili wywołania handlera dla obsługi takiego błędu wynik w postaci rozpakowanej nie jest jeszcze obliczony i zmienna przeznaczenia nie jest jeszcze zapisana. Błędy „po" sygnalizuje się po obliczeniu wyniku rozpakowanego. Przy obsłudze zewnętrznej można uzyskać ten wynik procedurą PSTWRL. Zmienna przeznaczenia także nie jest wtedy zapisana, z wyjątkiem przypadku błędu P (pozycja 7), który jest sygnalizowany po przesłaniu wyniku do argumentu przeznaczenia.

Dla błędów obsługiwanych wewnętrznie kolejność sygnalizacji nie jest specjalnie istotna, gdyż za wynik podstawia się wartość przewidzianą w odpowiedzi zamaskowanej i operacja jest kontynuowana.

Obsługę wewnętrzną (zamaskowaną) poszczególnych błędów określono, w ślad za procesorem 8087 w sposób odpowiadający, jak się przypuszcza, większości użytkowników. Intencja twórców procesora było, aby z tej obsługi korzystać jak najczęściej, może z wyjątkiem błędu I, który uważa się za fatalny. Obydwa rodzaje odpowiedzi na błędy zebrano w poniższej tablicy:

Bł	Odpowiedź zamaskowana	Odpowiedź odmaskowana
I	1 argument NaN: Wynik = ten sam NaN 2 argumenty NaN: Wynik = NaN o większym module Żaden argument NaN: Wynik = wartość nienumeryczna „niezdefiniowana" /1/	Przerwanie
Z	Wynik = ∞ o znaku będącym różnicą symetryczną znaków dzielnej i dzielnika	
D	Argument denormalny upakowany normalizuje się przy rozpakowaniu /2/	
O	∞ o znaku wyniku	
U	Wynik się denormalizuje, być może do 0, zachowując znak. Dla superniedomiaru /3/ wynik = 0 z zachowaniem znaku	Przerwanie /3/
P	Wynik się zaokrągla.	Wynik się zaokrągla. Przerwanie po przesłaniu wyniku do miejsca przeznaczenia

Uwagi:

/1/ Błąd I sygnalizowany jest zwykle przed obliczeniem wyniku (pozycja 2 na liście kolejności). W przypadku obu argumentów numerycznych może to być błąd np. typu 0/0 lub 0*∞. Sygnalizacja I po obliczeniu wyniku (pozycja 6 listy kolejności) oznacza stwierdzenie nieznormalizowanego wyniku w postaci rozpakowanej. Aktualnie ten błąd nie powinien nigdy wystąpić w operacji arytmetycznej, gdyż jest on skutkiem denormalnych lub anormalnych argumentów. Otóż argumenty upakowane są w razie potrzeby normalizowane przed operacją, zaś argumenty rozpakowane, jeżeli powstały z uprzednich obliczeń — są znormalizowane. Normalność tę sprawdza się przed każdą operacją arytmetyczną i przy jej braku sygnalizuje się błąd I typu „przed".

/2/ Zamaskowana obsługa błędu D jest niezgodna z rozwiązaniem Intela (ale uważana za perspektywiczną przez jednego z twórców normy IEEE na operacje zmiennoprzecinkowe [4]). Koprocesor przekształca argument na równoważny anormal, co może prowadzić do wyniku anormalnego. Niniejsza biblioteka normalizuje argument przy rozpakowaniu, ustawiając w słowie wskaźników postaci rozpakowanej bit „den". Każde użycie tej liczby powoduje sygnalizację błędu D, jakkolwiek liczba jest znormalizowana, a stąd i wynik rozpakowany znormalizowany. Takie rozwiązanie jest prostsze, gdyż w postaci rozpakowanej, w której realizuje się operacje arytmetyczne, działa się wyłącznie na normalach. Należy jednak pamiętać, że dokładność wyniku jest określona przez dokładność denormalnego argumentu, który może mieć mało cyfr znaczących.

/3/ Rozróżnia się „zwykły" nadmiar i niedomiar, wykładnika formatów upakowanych, od nadmiaru i niedomiaru wykładnika formatu rozpakowanego. Te ostatnie określamy jako supernadmiar (gdy $e \geqslant 32$ k) i superniedomiar (gdy $e < -32$ k). Mogą one wystąpić tylko dla operacji o argumentach rozpakowanych. W czasie obsługi zewnętrznej zwykłego nadmiaru/niedomiaru wykład-

nik e w WRL jest bez zmian: dla nadmiaru e ⩾ ensk formatu wyniku >0, zaś dla niedomiaru e < eden <0. Natomiast dla supernadmiaru e w WRL jest za mały o 64 k (dla procedur o wynikach upakowanych e < ensk), a dla superniedomiaru e jest większy o 64 k od wartości prawidłowej (dla procedur pakujących badać e ⩾ eden formatu). Nawiasem mówiąc, koprocesor inaczej obsługuje nadmiar/niedomiar postaci wewnętrznej, gdyż tam używa się bazowanego wykładnika na 15 bitach.

Dla wszystkich nadmiarów i niedomiarów mantysa m w WRL jest zaokrąglona do długości przeznaczenia: do 24 lub 32 bitów. Przy tym dla 24 bitów mniej znaczący bajt m2 jest wyzerowany. Z powodu tego zaokrąglenia przy skoku do handlera obsługi nadmiaru/niedomiaru bit P może już być ustawiony w SW (niezależnie od tego, czy P jest zamaskowane, czy nie), żeby było wiadomo, że m jest ewentualnie niedokładne. Jest to niewielka modyfikacja oryginalnej kolejności sygnalizacji błędów określonej w opisie koprocesora, co zaznaczono w pozycji 5 podanej tutaj kolejności przez (P).

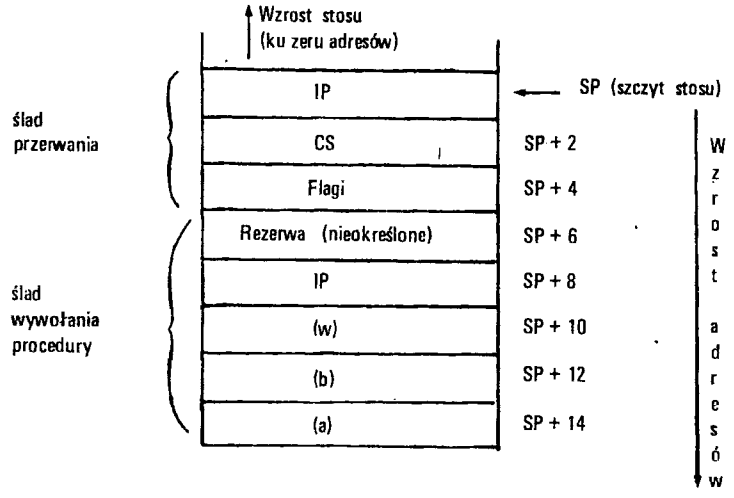
7. Handler obsługi błędów

Jeżeli istnieje możliwość wystąpienia chociaż jednego błędu niezamaskowanego, użytkownik powinien dostarczyć handler obsługi błędów. Typowa decyzja wyboru błędów do własnej obsługi polega na zamaskowaniu wszystkich błędów z wyjątkiem I (niedozwolona operacja). Maska błędów w słowie sterującym powinna mieć wtedy wartość FEH.

Handler jest procedurą obsługi przerwania programowego wywoływanego przez bibliotekę. Programowy emulator koprocesora w wykonaniu Intela używa przerwania nr 16. Niestety, nie można beztrosko przyjąć tej wartości dla biblioteki „P", ponieważ np. IBM PC wykorzystuje to przerwanie do innych celów. Można tylko zadeklarować, że użytkownik będzie poinformowany (lub sam zadecyduje) o używanym przez bibliotekę numerze przerwania. Obowiązkiem użytkownika jest wpisanie adresu handlera do wektora przerwań procesora (offset i baza odpowiednio w słowach n*4 i n*4 + 2, gdzie n jest numerem przerwania). W danej chwili jeden handler obsługuje wszystkie błędy operacji zmiennoprzecinkowych, gdyż wszystkie błędy powodują to samo przerwanie.

Typowy program handlera powinien kolejno wykonać następujące czynności:

1. Przechować na stosie używane rejestry (jak wiadomo flagi, CS, i IP już są złożone). Własne zmienne robocze również zaleca się rezerwować na stosie.
2. Odczytać słowo stanu operacją PSTSW (n1) i, jeżeli handler może być użyty rekursyjnie, zapamiętać wynik operacji na stosie. Podobnie, jeżeli zachodzi potrzeba, można odczytać słowo sterujące operacją PSTCW.
3. Wyzerować błędy i bit przerwania w słowie stanu operacją PCLEX.
4. Odblokować przerwania mikroprocesora instrukcją STI, gdyż zablokował je mechanizm przerwania.
5. Ustalić błąd. Słowo stanu, odczytane w p. 2 obsługi, zawiera bity wszystkich błędów (łącznie z ostatnim) operacji wywołanych przez program użytkownika od ostatniego zerowania. Przerwanie spowodował ten błąd, który jest odmaskowany w aktualnym CW.
   Wywołanie, które spowodowało błąd, ustala się na podstawie stosu, który przy wejściu do handlera ma postać:



Rejestr IP śladu wywołania zawiera adres następnego bajtu za instrukcją CALL, zaś (a), (b) i (w) oznaczają adresy argumentów w segmencie danych użytkownika w kolejności składania na stos. Liczba adresów jest równa liczbie argumentów wywołania. Słowo rezerwy na stosie przeznaczone jest dla ewentualnego wykorzystania w późniejszej wersji biblioteki, gdy instrukcja CALL typu „FAR" będzie składać kolejno („od dołu") CS i IP.
Wynik w postaci rozpakowanej, jeżeli już został obliczony, znajduje się w wewnętrznym akumulatorze biblioteki, z którego może być odczytany podprogramem PSTWRL. Przy obsłudze odmaskowanych błędów, z wyjątkiem P, zmienna przeznaczona na wynik nie jest zapisana.

6. Obsłużyć błąd (błędy) według indywidualnego algorytmu. Można zliczać błędy, drukować lub wyświetlać komunikat diagnostyczny, próbować naprawić błąd przez podstawienie za wynik (w, nie WRL!) jakiejś wartości, lub zastąpić wynik przez określoną wartość nienumeryczną itp. Można po prostu zakończyć obliczenia, co zaleca się zrobić dopiero po wyjściu z handlera, dla zwolnienia stosu.

7. Wyjść z handlera. Należy zwolnić obszary zajęte na stosie, odtworzyć rejestry i wyjść z programu obsługi przerwania instrukcją IRET. Jeżeli po tej operacji w SW nie ma ustawionych bitów niezamaskowanych błędów, to nastąpi powrót do programu użytkowego tuż za instrukcją CALL wywołania operacji błędnej. W przeciwnym wypadku nastąpi powtórne wywołanie handlera, co może doprowadzić do stałej pętli.

Uwagi dodatkowe

Przechowywanie i odtwarzanie rejestrów (p.1 i 7) przez program obsługi przerwania jest zwykle konieczne w obsłudze przerwania sprzętowego (np. pochodzącego od realnego koprocesora Intel 8087), zdarzającego się asynchronicznie w stosunku do pracy jakiegoś programu. Niniejsza biblioteka zaś wywołuje handler synchronicznie z pracą programu, w określonym punkcie programu, za pomocą instrukcji INT n. Takie wywołanie jest co do istoty równoważne instrukcji CALL, toteż handler powinien przechowywać lub zachowywać tylko te rejestry, które muszą pozostać niezmienione. W danym przypadku istotne jest zachowanie następujących rejestrów: CS, DS, ES, SS, SP, BP, IP

i bitów sterujących w rejestrze flag. W podrozdziale 8.1 zadeklarowano, że ich wartości biblioteka nie niszczy, więc nie powinien ich także zniszczyć handler, ze względu na pracę zarówno biblioteki, jak i programu użytkowego. Rejestry CS, IP i flagi są przechowywane przez mechanizm przerwania. Jeżeli więc handler nie zmienia rejestrów DS, ES i SS, oraz odtwarza rejestr SP przez zdjęcie ze stosu swoich zmiennych, to może nie przechowywać i nie odtwarzać innych rejestrów niż BP dla poprawności pracy biblioteki i programu przerwanego. Handler, oczywiście, może dla własnych celów, np. pomiędzy wywołaniami lub wewnątrz jakichś procedur, przechowywać pozostałe rejestry (AX, BX, CX, DX, SI i DI). Ponadto zakaz równoległego używania biblioteki pozwala przenieść p.4 (odblokowanie przerwań) na początek obsługi.

Handler może sam wywoływać poprogramy biblioteki zmiennoprzecinkowej. Trzeba przy tym pamiętać o generalnej zasadzie nieprzechowywania przez bibliotekę rejestrów procesora, z wyjątkiem podanych wyżej. Oczywiście, wewnętrzne wywołanie zapisuje lokalne obszary biblioteki. Dlatego np. analizę WRL (p.5) trzeba zakończyć przed pierwszym wywołaniem procedury zmiennoprzecinkowej w handlerze.

Na czas swojej pracy handler może nadać słowu sterującemu nową wartość dla zabezpieczenia się przed niezamaskowanymi błędami instrukcji wewnętrznych i np. ograniczyć się do analizy SW0 po każdej instrukcji. Trzeba pamiętać, że wystąpienie niezamaskowanego błędu spowoduje przerwanie pracy handlera i powtórne (rekursyjne) wejście do niego, po złożeniu na stosie nowej warstwy opisanej w p.5. Dla prawidłowej obsługi takiej sytuacji może być konieczne przechowanie na początku i odtworzenie na końcu rejestrów procesora oraz rezerwacja zmiennych lokalnych na stosie.

Program użytkowy może niektóre błędy obsługiwać handlerem a inne — w głównej linii wykonania, po analizie SW po wykonaniu jednej lub ciągu operacji zmiennoprzecinkowych. Należy przy tym pamiętać, że biblioteka tylko raz odtwarza rejestr BP — przy wyjściu z operacji lub przy wejściu do handlera. Nie zmienia się też zawartości SW i CW wprowadzonych przez handler.

8. Procedury

8.1. Wprowadzenie

Podprogramy wchodzące w skład biblioteki można podzielić na 4 grupy: podstawowe operacje arytmetyczne, pomocnicze procedury matematyczne, złożone operacje matematyczne i procedury organizacyjne dla procesu obliczenia.

Nazwy procedur

Nazwy procedur biblioteki rozpoczynają się literą P (od Point), przez analogię do nazw instrukcji koprocesora, zaczynających się od F (Floating).

Drugi znak nazwy w operacjach matematycznych wskazuje na format argumentów wejściowych według kodu:

S — argument(y) formatu „Short Real"
M — argument(y) formatu „Middle Real"
U — argument(y) formatu „Unpacked Real"
1 — argument całkowity 1—słowowy
2 — argument całkowity 2—słowowy

Następne litery nazwy (zwykle trzy) określają krótko funkcje procedury w języku angielskim, np. ADD oznacza dodawanie, NIN jest skrótem od „Nearest Integer" tj. „najbliższa całkowita", zaś UNP znaczy „rozpakuj".

W nazwach procedur matematycznych, jeżeli wynik ma inny format niż argumenty wejściowe, to za ciągiem znaków określających funkcję podprogramu występuje jeszcze jeden znak sygnalizujący format wyniku według podanego wyżej kodu. Np. PUADD oznacza dodawanie o wszystkich argumentach rozpakowanych, zaś PUADDS — o argumentach rozpakowanych, a wyniku zmiennoprzecinkowym krótkim. P1FLTS — oznacza konwersję liczby całkowitej 1—słowowej na zmiennoprzecinkową krótką, zaś PSNIN1 — odwrotnie.

Argumenty procedur

Argumenty procedur w niniejszym opisie z reguły są oznaczone dwuznakowo: mała litera i cyfra, np. a2, n1, w4. Początkowe litery alfabetu (a, b ...), jeżeli nie podano inaczej, oznaczają argumenty zmiennoprzecinkowe wejściowe, zaś końcowe (w...) — argumenty zmiennoprzecinkowe wyjściowe. Dla argumentów całkowitych wejściowych przeznaczono litery i,j, k, l zaś dla całkowitych wyjściowych — litery m, n. Cyfra oznacza długość argumentu w słowach. Argumentów o nieparzystej liczbie bajtów biblioteka aktualnie nie używa. Przewiduje się w razie potrzeby oznaczanie argumentów jednobajtowych tylko literą, bez cyfry.

Przykłady:

a2 — argument wejściowy „Short Real"
i2 — argument wejściowy „Short Integer"
n1 — argument wyjściowy „Word Integer"
w4 — argument wyjściowy „Unpacked Real"

Jeżeli jakiś argument nie będzie spełniał powyższej konwencji, będzie on określony oddzielnie.

Postać biblioteki i programu użytkowego

Biblioteka składa się z szeregu podprogramów napisanych w języku assemblera mikroprocesora Intel 8086. W dalszej części pracy opisano procedury dostępne dla użytkownika, które będą nazywane zewnętrznymi. Aktualnie biblioteka stanowi dwa moduły wynikowe assemblera, spełniające warunki tzw. „małego modelu obliczeń" języka PLM—86. W przyszłości możliwa jest zmiana tej postaci. Definicje użytych w dalszym opisie pojęć Czytelnik znajdzie w podręczniku języka ASM—86.

Moduły biblioteki zawierają trzy segmenty: danych zmiennych o nazwie DATA, danych stałych o nazwie CONST i segment kodu o nazwie CODE. Wszystkie segmenty mają atrybuty PARA i PUBLIC tj. są lokowane w pamięci od adresów podzielnych przez 16 i mogą być łączone z fragmentów zawartych w różnych modułach kompilacji. Segment zmiennych zadeklarowano jako należący do klasy o nazwie 'DATA' i grupy DGROUP, segment stałych do klasy 'CONST' i grupy DGROUP, a segment kodu — do klasy 'CODE' i grupy CGROUP. Segment zmiennych powinien być ładowany do pamięci typu RAM, zaś stałe i kod mogą być w pamięci typu RAM lub ROM. Program użytkowy również powinien deklarować grupy DGROUP i CGROUP, obejmujące jego segmenty odpowiednio: danych i kodu. Powoduje to ograniczenie wielkości programu (lub części programu) współpracującego z biblioteką do 64 kB w obszarze danych i 64 kB w obszarze kodu. Ponadto program użytkowy powinien zadeklarować segment stosu, którego biblioteka używa poprzez aktualną zawartość rejestrów SS i SP. Dla potrzeb biblioteki trzeba przewidzieć na stosie ok. 30 słów.

Wszystkie procedury biblioteki są typu NEAR.

Wywołania procedur

Przy wejściu do biblioteki rejestr CS musi zawierać adres (ściśle – numer 16–bajtowego paragrafu) początku grupy CGROUP, zaś rejestr IP – adres procedury względem początku tej grupy. Rejestry DS i ES muszą zawierać adres DGROUP. Tu uwaga praktyczna: łatwo się zapomina o nadaniu wartości ES.

Argumenty procedur zewnętrznych, poza szczególnymi, opisanymi przypadkami, leżą w pamięci, a ich adresy są przekazywane przez stos. Adresy są 16–bitowymi offsetami w grupie DGROUP. Dla wywołania procedury należy złożyć na stos adresy argumentów w ilości i kolejności podanej w opisie procedury. Argument wyjściowy, jeżeli występuje, jest zawsze ostatni. Następnie należy wywołać procedurę instrukcją CALL z nazwą procedury, która powinna być zadeklarowana w dyrektywie EXTRN jako typu NEAR. Przy powrocie z procedury adresy i ślad wywołania są zdjęte ze stosu.

Przykład:

Dla obliczenia w postaci rozpakowanej różnicy X=A–B można użyć sekwencji:

```
MOV BX, OFFSET DGROUP:A
PUSH BX
MOV BX, OFFSET DGROUP:B
PUSH BX
MOV BX, OFFSET DGROUP:X
PUSH BX
CALL PUSUB
```

Taka sekwencja instrukcji (lub równoważna ze względu na zawartość stosu) będzie w dalszym opisie oznaczona przez PUSUB (A,B,X).

Zarówno stos, jak i argumenty powinny mieć parzyste adresy (word aligned), aby nie spowolnić pracy mikroprocesora Intel 8086.

Argumenty wejściowe rozpakowane wszystkich podprogramów powinny być znormalizowane. Wtedy wyniki numeryczne rozpakowane też będą znormalizowane (także dla argumentów upakowanych denormalnych). Dla argumentów rozpakowanych nieznormalizowanych sygnalizuje się błąd I i w razie zamaskowania – za wynik podstawia się wartość „niezdefiniowaną". Złożone operacje matematyczne wymagają wszystkich argumentów znormalizowanych. Dla argumentów nieznormalizowanych wydają one jako wynik wartość nienumeryczną „niezdefiniowaną", bez sygnalizacji żadnych błędów.

Aby nie dopuścić do konfliktu nazw używanych przez program użytkowy i bibliotekę, nazwy wszystkich obiektów biblioteki niedostępnych dla użytkownika zaczynają się od @. Program użytkowy nie powinien zawierać nazw zaczynających się tym znakiem.

Biblioteka nie zmienia argumentów wejściowych i dlatego mogą być one stałymi np. w pamięci ROM. Jednak dla oszczędności miejsca można określić ten sam argument jako wejściowy i wyjściowy, gdyż przesłanie wyniku wykonuje się po zakończeniu obliczenia. Np. wywołanie PSMUL (X,X,X) daje kwadrat X w miejscu X.

Biblioteka zachowuje rejestry CS, DS, ES, SS, BP i SP oraz w rejestrze flag bity sterujące: TF, DF i IF. Przy powrocie z procedury w programie głównym (nie w handlerze) rejestr SP wskazuje szczyt stosu przed złożeniem adresów argumentów. Pozostałych rejestrów i flag nie przechowuje się, i jeżeli nie określono inaczej, ich zawartość przy powrocie z wywołania trzeba uznać za nieokreśloną.

Zalecenia co do wykorzystania danych o różnych formatach

W obecnym zakresie biblioteka jest przeznaczona zasadniczo do obliczeń w formacie krótkim (Short Real), o dokładności ok. 7 cyfr dziesiętnych. Format rozpakowany o dokładności ok. 9 cyfr został wprowadzony dla realizacji wewnętrznych obliczeń złożonych operacji matematycznych o danych i wynikach krótkich. Format średni jest „produktem ubocznym" formatu rozpakowanego i ma tę samą dokładność.

Operacja arytmetyczna składa się z rozpakowania argumentów, wykonania obliczenia w formacie rozpakowanym i zapakowania wyniku do formatu przeznaczenia. Dla oszczędności czasu użytkownik może ograniczyć wpływ pierwszego i ostatniego etapu przez jednorazowe rozpakowanie danych, pracę w formacie rozpakowanym i wyjście z algorytmu podprogramem pakującym (typu PUADDS). Biblioteka nie zawiera oddzielnych podprogramów pakujących dla uniknięcia błędów podwójnego zaokrąglenia — po raz pierwszy wyniku rozpakowanego, a następnie krótkiego. W dziedzinie liczb dziesiętnych można to zilustrować na przykładzie zaokrąglania liczby 127.48 do 3 cyfr według algorytmu podanego w rozdziale 3. Prawidłowym wynikiem jest 127, podczas gdy kolejne zaokrąglenia do 4 i 3 cyfr dadzą 127.5 i 128. Podprogramy o argumentach rozpakowanych a wyniku upakowanym zaokrąglają, naturalnie, tylko raz.

Format średni może być wykorzystany dla zapamiętania dużej liczby danych z dokładnością 32 bitów. Konwersję z postaci rozpakowanej do średniej wykonuje się podprogramem wspomnianego wyżej typu. Jeżeli ostatnia operacja nie wynika z algorytmu, to można ją zrealizować przez dodanie do danej liczby zera rozpakowanego.

Niech AU oznacza daną rozpakowaną. Wtedy wywołanie:

PUADDM (AU, ZER, AM)

gdzie określono:

ZER DW 0,0,0,10H; kolejno: m2, m1, e, WSK, a w WSK bit mz (patrz p. 2.5)

pakuje AU do AM w formacie średnim.

Konwersja z formatu rozpakowanego do średniego powyższą metodą przebiega bez straty dokładności. Analogiczny sposób dla zapakowania w formacie krótkim może spowodować wspomniany wyżej błąd wynikający z powtórnego zaokrąglenia.

8.2. Podstawowe operacje arytmetyczne

W poniższej tablicy zebrano wywołania operacji: dodawania, odejmowania, mnożenia i dzielenia zmiennoprzecinkowego. Daną operację można wywoływać pod różnymi nazwami, w zależności od formatów argumentów.

Argumenty wejściowe: (a2, b2) (a3, b3) (a4, b4) — liczby zmiennoprzecinkowe odpowiednio: krótkie, średnie, rozpakowane.

Argumenty wyjściowe: (w2, w3, w4) — liczby zmiennoprzecinkowe odpowiednio: krótkie, średnie, rozpakowane.

Wywołania procedur			Operacja
PSADD (a2, b2, w2) PUADDS (a4,b4,w2)	PMADD (a3,b3,w3) PUADDM (a4,b4,w3)	PUADD (a4,b4,w4)	w = a + b
PSSUB (a2,b2,w2) PUSUBS (a4,b4,w2)	PMSUB (a3,b3,w3) PUSUBM (a4,b4,w3)	PUSUB (a4,b4,w4)	w = a − b

PSMUL (a2,b2,w2) PUMULS (a4,b4,w2)	PMMUL (a3,b3,w3) PUMULM (a4,b4,w3)	PUMUL (a4,b4,w4)	w = a * b
PSDIV (a2,b2,w2) PUDIVS (a4,b4,w2)	PMDIV (a3,b3,w3) PUDIVM (a4,b4,w3)	PUDIV (a4,b4,w4)	w = a / b

Błędy:

Dla wszystkich wymienionych operacji: I, D, O, U, P.
Ponadto dla dzielenia: Z.

8.3. Pomocnicze procedury matematyczne

1. Konwersje z postaci stałoprzecinkowej na zmiennoprzecinkową (Float).

Argumenty:

i1 – dana stałoprzecinkowa, 1–słowowa (Word Integer)
i2 – dana stałoprzecinkowa, 2–słowowa (Short Integer)
w2 – wynik zmiennoprzecinkowy w formacie krótkim
w3 – wynik zmiennoprzecinkowy w formacie średnim
w4 – wynik zmiennoprzecinkowy w formacie rozpakowanym

Wywołania dla danej 1–słowowej:

P1FLTS (i1, w2) P1FLTM (i1, w3) P1FLTU (i1, w4)

Wywołania dla danej 2–słowowej:

P2FLTS (i2, w2) P2FLTM (i2, w3) P2FLTU (i2, w4)

Błędy: P (tylko P2FLTS) – wynik zaokrąglony.

2. Konwersje z postaci zmiennoprzecinkowej na najbliższą liczbę całkowitą stałoprzecinkową (Nearest Integer).

Podane niżej procedury zaokrąglają argument wejściowy do najbliższej liczby całkowitej według algorytmu podanego w rozdziale 3, dając w wyniku liczbę stałoprzecinkową.

Argumenty:

a2 – dana zmiennoprzecinkowa krótka (Short Real)
a3 – dana zmiennoprzecinkowa średnia (Middle Real)
a4 – dana zmiennoprzecinkowa rozpakowana (Unpacked Real)
n1 – wynik stałoprzecinkowy w 1 słowie
n2 – wynik stałoprzecinkowy w 2 słowach

Wywołania dla wyniku w 1 słowie:

PSNIN1 (a2, n1) PMNIN1 (a3, n1) PUNIN1 (a4, n1)

Wywołania dla wyniku w 2 słowach:

PSNIN2 (a2, n2) PMNIN2 (a3, n2) PUNIN2 (a4, n2)

Błędy:

I – argument NaN, niemożliwość zapakowania liczby stałoprzecinkowej (wynik nie mieści się w zakresie argumentu przeznaczenia, dana ∞). Podstawia się wartość „niezdefiniowaną całkowitą".

D – argument denormalny lub powstały z rozpakowania denormala

P – wynik zaokrąglony.

W czasie obsługi zewnętrznej powyższych błędów WRL zawiera daną zmiennoprzecinkową w postaci rozpakowanej.

3. Rozpakowanie liczby zmiennoprzecinkowej (Unpack).

Argumenty:

a2 – dana zmiennoprzecinkowa w formacie krótkim
a3 – dana zmiennoprzecinkowa w formacie średnim
w4 – wynik zmiennoprzecinkowy w formacie rozpakowanym

Wywołania:

PSUNPU (a2, w4) PMUNPU (a3, w4)

Błędy:

Błędów się nie sygnalizuje, ale w słowie wskaźników wyniku mogą ustawić się bity: „den" lub „NaN", które przy użyciu liczby jako argumentu operacji spowodują błędy odpowiednio D lub I.

8.4. Złożone operacje matematyczne

Biblioteka realizuje następujące złożone operacje matematyczne; pierwiastek kwadratowy, sinus, cosinus i atan2. Procedury realizujące wymienione operacje wykorzystują (wywołują) opisane uprzednio proste operacje arytmetyczne wykonywane przez procedury: PUADD, PUADDS, PUSUB, PUSUBS, PUMUL oraz procedurę rozpakowującą PSUNPU.

Metoda zastosowana do wyznaczania wartości funkcji sinus, cosinus i atan2 została oparta na rozwinięciu tych funkcji w szereg Taylora i ograniczeniu obliczeń do dwóch pierwszych wyrazów tego szeregu:

$$(*) \quad w = f(a) = f(a0 + h) = f(a0) + h * f'(a0)$$

gdzie:
w – to wartość funkcji f(x) dla argumentu a,
a – to wartość argumentu,
a0 – to najbliższa, stablicowana wartość, taka, że:

$$a0 \leqslant [\text{wartość bezwzględna (a)}] = |a|$$

f(a0) – to stablicowana wartość funkcji f(x) w punkcie a0,
f'(a0) – to stablicowana wartość pochodnej f'(x) funkcji f(x) w punkcie a0,

$h = a - a0$

Procedury wyznaczające funkcje: sinus, cosinus i atan2 wykorzystują zatem do obliczeń trzy tablice pomocnicze (umieszczone w segmencie danych – CONST):

– tablicę @ALFA, zawierającą stablicowane wartości kątów od 0 do $\pi/2$ radianów, zmieniających się co $[(0.3/180.0) * \pi]$ radianów,

- tablicę @SINUS, zawierającą stablicowane wartości funkcji sinus dla argumentów zawartych w tablicy @ALFA,
- tablicę @POCHOD (tylko dla procedur obliczających atan2), zawierającą stablicowane wartości pochodnych funkcji arcus sinus(x) dla argumentów zawartych w pierwszych 151 elementach tablicy pomocniczej @SINUS.

Dodatkowo, w tablicy @ALFA stablicowano kąty: π, $3*\pi/2$ i $2*\pi$ radianów. Wszystkie elementy tablic pomocniczych zakodowano w formacie rozpakowanym.

Algorytm wyznaczania funkcji sinus, cosinus i atan2 sprowadza się zatem do określenia takiego najbliższego argumentu a0, który spełnia warunek:

$$a0 \leqslant [\text{wartość bezwzględna (a)}] = |a|,$$

wykorzystania wzoru (*) i odpowiedniej modyfikacji uzyskanego wyniku. Do wykonania tych operacji potrzebne są cztery proste zmiennoprzecinkowe operacje arytmetyczne. W przypadku funkcji sinus i cosinus, jeżeli wartość bezwzględna argumentu wykracza poza przedział $(0, +2*\pi)$ radianów, procedury dodatkowo tyle razy będą odejmować od tej wartości bezwzględnej liczbę $2*\pi$, aż ją „sprowadzą" do tego przedziału. Omówiona metoda wyznaczania funkcji trygonometrycznych i funkcji atan2 jest metodą przybliżoną, zapewniającą jednak dość dużą dokładność obliczeń. W przypadku funkcji trygonometrycznych maksymalny błąd względny zmienia się od ok. 0.0000077 (PUSIN, PUCOS) do ok. 0.000016 (PSSIN, PSCOS). W przypadku funkcji atan2 wartość bezwzględna różnicy pomiędzy wartościami dokładnymi a wartościami uzyskanymi procedurami biblioteki „P" dla tych samych argumentów, zmienia się od ok. 0.0000077 radiana (PUATAN2) do ok. 0.000016 radiana (PSATAN2).

Procedury poszukujące wartość pierwiastka kwadratowego wyznaczają, w pierwszym etapie obliczeń, wartość wykładnika wyniku, a następnie, traktując mantysę argumentu jako liczbę stałoprzecinkową, pierwiastek kwadratowy z mantysy argumentu. Należy zaznaczyć, że procedury obliczania pierwiastka kwadratowego wykorzystują dodatkowo tylko procedurę PSUNPU.

Złożone procedury matematyczne nie używają wymienionych w rozdziale 5 wewnętrznych rejestrów biblioteki. We wszystkich przypadkach wykrycie błędu danych lub błędu wynikającego z działania samej procedury (np. w przypadku procedury PUSQRTS — błędu nadmiaru) powoduje przerwanie obliczeń i podstawienie za wynik wartości nienumerycznej niezdefiniowanej (8001FFFFC0000000H — dla wyniku w formacie rozpakowanym lub FFC00000H — dla wyniku w formacie krótkim).

Wszystkie procedury realizujące złożone operacje matematyczne wymagają, aby argumenty wejściowe upakowane lub rozpakowane były znormalizowane (niespełnienie tego warunku zawsze traktowane jest jako błąd). Szczegółowe wymagania co do argumentów wejściowych podano przy opisie każdej zrealizowanej operacji.

1. Pierwiastek kwadratowy (Square Root):

$$w = \sqrt{a}, \quad a \geqslant 0$$

Argumenty:
a2 – dana zmiennoprzecinkowa krótka,
a4 – dana zmiennoprzecinkowa rozpakowana,
w2 – wynik zmiennoprzecinkowy krótki,
w4 – wynik zmiennoprzecinkowy rozpakowany.

Wywołania procedur:
PSSQRT (a2, w2) PSSQRTU (a2, w4) PUSQRT (a4,w4) PUSQRTS (a4,w2)

Powyższe procedury sygnalizują błąd (tzn. podstawiają za wynik wartość nienumeryczną niezdefiniowaną) w następujących przypadkach:

- argument wejściowy jest ujemny. Dotyczy to także przypadku $\sqrt{-0}$,
- argument wejściowy jest wartością nienumeryczną lub wartością nienumeryczną niezdefiniowaną,
- argument wejściowy ma wartość ∞,
- argument wejściowy jest liczbą nieznormalizowaną.

Ponadto, w przypadku procedury PUSQRTS (a4, w2), dodatkowo w taki sam sposób sygnalizowany jest błąd nadmiaru lub niedomiar .

2. Funkcja sinus:

$$w = \text{sinus}(a), \quad -\infty < a < +\infty$$

Argumenty:
a2 – dana zmiennoprzecinkowa krótka (kąt podany w radianach),
a4 – dana zmiennoprzecinkowa rozpakowana (kąt podany w radianach),
w2 – wynik zmiennoprzecinkowy krótki,
w4 – wynik zmiennoprzecinkowy rozpakowany.

Wywołania procedur:

PSSIN(a2, w2) PSSINU(a2, w4) PUSIN(a4, w4) PUSINS(a4, w2)

Powyższe procedury realizują funkcję trygonometryczną sinus dla argumentu a z przedziału: $-\infty < a < +\infty$.
Błąd zostaje zasygnalizowany (tzn. za wynik jest podstawiona wartość nienumeryczna niezdefiniowana) w następujących przypadkach:

- argument wejściowy jest wartością nienumeryczną lub wartością nienumeryczną niezdefiniowaną,
- argument wejściowy ma wartość ∞,
- argument wejściowy jest liczbą nieznormalizowaną.

3. Funkcja cosinus:

$$w = \text{cosinus}(a), \quad -\infty < a < +\infty$$

Argumenty:

a2 – dana zmiennoprzecinkowa krótka (kąt podany w radianach),
a4 – dana zmiennoprzecinkowa rozpakowana (kąt podany w radianach),
w2 – wynik zmiennoprzecinkowy krótki,
w4 – wynik zmiennoprzecinkowy rozpakowany.

Wywołania procedur:

PSCOS(a2, w2) PSCOSU(a2, w4) PUCOS(a4, w4) PUCOSS(a4, w2)

Powyższe procedury realizują funkcję trygonometryczną cosinus dla argumentu a z przedziału: $-\infty < a < +\infty$.

Błąd zostaje zasygnalizowany (tzn. za wynik jest podstawiona wartość nienumeryczna niezdefiniowana) w następujących przypadkach:

– argument wejściowy jest wartością nienumeryczną lub wartością nienumeryczną niezdefiniowaną,

– argument wejściowy jest nieskończonością ze znakiem,

– argument wejściowy jest liczbą nieznormalizowaną.

4. Funkcja atan2:

$$w = \text{atan2}(a,b), \qquad -1 \leqslant a \leqslant +1, \quad -1 \leqslant b \leqslant +1,$$

Argumenty:

a2 – dana zmiennoprzecinkowa krótka,
b2 – dana zmiennoprzecinkowa krótka,
a4 – dana zmiennoprzecinkowa rozpakowana,
b4 – dana zmiennoprzecinkowa rozpakowana,
w2 – wynik zmiennoprzecinkowy krótki (kąt w radianach),
w2 – wynik zmiennoprzecinkowy rozpakowany (kąt w radianach).

Wywołania procedur:

PSATAN2(a2,b2,w2) PSATAN2U(a2,b2,w4) PUATAN2(a4,b4,w4) PUATAN2S(a4,b4,w2)

Uwaga:

Funkcja atan2(a,b,w) jest funkcją polegającą na wyznaczeniu kąta „w" na podstawie znajomości wartości argumentów „a" i „b" takich, że: a = sinus(w), b = cosinus(w). Wynika stąd, że oba argumenty: „a" i „b" muszą należeć do przedziału $<-1, +1>$ i poza przypadkiem, gdy ich wartości bezwzględne są równe $\sqrt{2}/2$, nie mogą być sobie równe.

Powyższe procedury wyznaczają funkcję atan2 dla argumentów „a" i „b" należących do przedziału $<-1, +1>$. Realizacja tych procedur została tak pomyślana, że do obliczeń zostaje wyznaczona wartość bezwzględna tego argumentu, która jest mniejsza: jeśli jest to argument „a" to zostaje wyznaczona wartość funkcji arcus sinus(a), jeśli jest to argument „b", zostaje wyznaczona wartość funkcji arcus cosinus(b). W obu przypadkach uzyskany wynik jest kątem należącym do przedziału $<0, \pi/2>$ radianów i na podstawie zapamiętanych wcześniej znaków obu argumentów wejściowych zostaje zmodyfikowany do właściwej ćwiartki układu współrzędnych kartezjańskich. Wyjaś-

nienie to jest potrzebne dla zrozumienia pewnej tolerancji w wartościach obu argumentów wejściowych: „a" i „b". Procedury wygenerują wynik różny od wartości nienumerycznej niezdefiniowanej (sygnalizującej błąd) nawet wówczas, gdy suma kwadratów obu argumentów nie będzie równa 1. Założeniem jest tutaj, że poprawny wynik uzyskuje się na podstawie argumentu o mniejszej wartości bezwzględnej, która jest przy tym mniejsza lub równa $\sqrt{2}/2$. (sinus($\pi/4$) = cosinus($\pi/4$) = $\sqrt{2}/2$). W przypadku, gdy oba argumenty mają wartości bezwzględne większe od $\sqrt{2}/2$ (ale jeszcze „nie tak duże" jak w wymienionych poniżej przypadkach takich kombinacji danych wejściowych, które sygnalizują błąd) powyższe procedury jako wynik podadzą kąt $\pi/4$ radianów, zmodyfikowany następnie na podstawie znaków obu argumentów do właściwej ćwiartki układu współrzędnych kartezjańskich.

Błąd zostaje zasygnalizowany (tzn. za wynik jest podstawiona wartość nienumeryczna niezdefiniowana) w następujących przypadkach:

— którykolwiek z argumentów wejściowych jest wartością nienumeryczną lub wartością nienumeryczną niezdefiniowaną,

— którykolwiek z argumentów wejściowych ma wartość ∞.

— którykolwiek z argumentów wejściowych jest liczbą nieznormalizowaną,

— wartość bezwzględna któregokolwiek z argumentów wejściowych jest większa lub równa 2,

— oba argumenty wejściowe są równe zero,

— wartości bezwzględne obu argumentów wejściowych są większe lub równe 1.

Uwaga:

Procedury obliczające wartość funkcji atan2 mogą dawać wyniki zawarte w przedziale:
a. albo $<-\pi, +\pi>$ radianów

b. albo $<0, +2 * \pi>$ radianów

Aktualnie biblioteka realizuje wariant a. Istnieje jednak możliwość wyboru jednego z przedstawionych wariantów, odpowiednio uaktywniając i traktując jako komentarz, przed etapem assemblacji, te instrukcje w programie źródłowym biblioteki, które opatrzono komentarzem:

```
;**WYKONUJĄC TE INSTRUKCJE UZYSKA SIĘ
;**WYNIK Z PRZEDZIAŁU <0,2 * π>
```

lub:

```
;‡‡WYKONUJĄC TE INSTRUKCJE UZYSKA SIĘ
;‡‡WYNIK Z PRZEDZIAŁU <-π, +π>
```

Ewentualne zmiany należy przeprowadzić w czterech miejscach:

— pomiędzy etykietami: @ATAN10 a @ATAN11,

— pomiędzy etykietami: @ATAN25 a @ATAN26,

— pomiędzy etykietami: @ATAN45 a @ATAN46,

— pomiędzy etykietami: @ATAN47 a @ATAN48.

8.5. Podprogramy organizacyjne

Podprogramy organizacyjne umożliwiają kierowanie procesem obliczeń przez odczyt i nadawanie wartości wewnętrznym rejestrom biblioteki. Dla tych procedur nie ma wyboru rodzaju argumentów i dlatego w nazwach nie koduje się ich formatów. Nazwy tych procedur wzorowane są na nazwach instrukcji koprocesora. Składają się one z litery P i skrótu określenia wykonywanej funkcji.

W opisach procedur podano niszczone rejestry w wersji NEAR (aktualnie dostępnej) i FAR (ewentualnie w przyszłości).

1. Inicjacja biblioteki – PINIT
   Podprogram PINIT (bez argumentów) ustawia początkowy stan biblioteki. Zeruje się słowo stanu (SW) tzn. wskaźniki błędów i zgłoszenie przerwania. Słowo sterujące (CW) przyjmuje wartość 3FFH tzn. wszystkie błędy mają obsługę zamaskowaną i przerwania są zablokowane.
   Rejestry niszczone w wersji NEAR: AX
   Rejestry niszczone w wersji FAR: AX, DX, DI

2. Odczyt słowa stanu (Store Status Word) – PSTSW
   Podprogram PSTSW(n1) przepisuje słowo stanu pod adres określony argumentem n1 i jednocześnie do rejestru Ax. n1 jest zmienną całkowitą 1–słowową.
   Rejestry niszczone w wersji NEAR: AX, BX
   Rejestry niszczone w wersji FAR: AX, BX, DX, DI

3. Zerowanie błędów (Clear Exceptions) – PCLEX
   Podprogram PCLEX (bez argumentów) zeruje młodszy bajt słowa stanu tzn. wskaźniki błędów i bit zgłoszenia przerwania IR.
   Rejestry niszczone w wersji NEAR: AL
   Rejestry niszczone w wersji FAR: AL, DX, DI

4. Odczyt słowa sterującego (Store Control Word) – PSTCW
   Podprogram PSTCW(n1) przepisuje aktualne słowo sterujące (CW) pod adres określony argumentem n1 i jednocześnie do rejestru AX. n1 jest zmienną całkowitą 1–słowową.
   Rejestry niszczone w wersji NEAR: AX, BX
   Rejestry niszczone w wersji FAR: AX, BX, DX, DI

5. Nadanie wartości słowu sterującemu (Load Control Word) – PLDCW
   Podprogram PLDCW(i1) wpisuje do słowa sterującego (CW) słowo, którego adres określa i1. Aktualnie tylko młodszy bajt słowa sterującego ma znaczenie. Można w ten sposób zmienić indywidualne maski błędów i maskę blokady przerwań z biblioteki (IEM). Jeżeli w wyniku tego zostanie odmaskowany błąd aktualnie ustawiony w SW, to przy IEM = 0 nastąpi przy wyjściu z PLDCW przerwanie i wejście do handlera. Zalecaną praktyką jest wyzerowanie błędów w słowie stanu przed zmianą CW. Rejestry podano tylko dla przypadku braku odmaskowanych błędów w SW0.
   Rejestry niszczone w wersji NEAR: AX, BX
   Rejestry niszczone w wersji FAR: AX, BX, DX, DI

6. Odblokowanie przerwań z biblioteki (Enable Interrupts) – PENIN
   Podprogram PENIN (bez argumentów) zeruje w słowie sterującym bit IEM, dopuszczając wystąpienie przerwania od niezamaskowanego indywidualnie błędu ustawionego w SW. Tak jak dla PLDCW, zaleca się uprzednie wyzerowanie w SW dotychczasowych błędów. Rejestry podano tylko dla przypadku braku odmaskowanych błędów w SW0.
   Rejestry niszczone w wersji NEAR: AL
   Rejestry niszczone w wersji FAR: AL, DX, DI

7. Blokada przerwań z biblioteki (Disable Interrupts) — PDISI
   Podprogram PDISI (bez argumentów) ustawia w słowie sterującym maskę blokady przerwań IEM na wartość 1, nie dopuszczając do zgłoszenia przerwania nawet dla błędów, dla których maski indywidualne przewidują odpowiedź odmaskowaną. Przez zerowanie SW0 przed operacją i badanie go po operacji można, przy IEM = 1, zrealizować własną obsługę błędów, nie korzystając z mechanizmu przerwań.
   Rejestry niszczone w wersji NEAR: — (rejestry bez zmian)
   Rejestry niszczone w wersji FAR: DX, DI

8. Odczyt wewnętrznego akumulatora biblioteki (Store WRL) — PSTWRL
   Przed wejściem do handlera obsługi błędów typu „po" (patrz rozdział 6) podprogramy arytmetyczne przygotowują wynik operacji w postaci rozpakowanej w wewnętrznym akumulatorze WRL. Można odczytać zawartość tego akumulatora przy pomocy wywołania podprogramu PSTWRL(w4), gdzie w4 jest adresem 4—słowowej zmiennej dla wyniku w formacie rozpakowanym. Podprogram przeznaczony jest zasadniczo do analizy wyniku w postaci rozpakowanej przez handler obsługi nadmiaru lub niedomiaru. Poza handlerem i dla błędów typu „przed" zawartość WRL, jeżeli nie podano inaczej, jest nieokreślona.
   Rejestry niszczone w wersji NEAR: BX, CX, SI, DI
   Rejestry niszczone w wersji FAR: BX, CX, SI, DI, DX, ES.

9. Czasy wykonania niektórych procedur

W poniższej tabeli podano obliczone teoretycznie czasy wykonania 4 operacji arytmetycznych dla zegara 5 MHz (cykl = 0,2 mikrosekunty). Czasy wyrażono w mikrosekundach, po zaokrągleniu w górę do pełnych dziesiątek. Założono, że argumenty są znormalizowane i nie o szczególnych wartościach (0, ∞, NaN itp.) oraz, że operacje są bezbłędne, najwyżej poza zamaskowanym błędem precyzji.

Operacja		Argumenty				
		Rozpakow.	Krótkie	Średnie	Rozp./Krót.	Rozp./Śred.
Dodawanie/ odejmowanie	ZZ /1/	150–210	250–310	280–350	190–250	200–260
	ZZ /2/	160–350	260–450	290–480	200–380	200–390
Mnożenie		250–270	350–370	380–410	290–310	300–320
Dzielenie		290–420	390–520	430–560	330–460	340–470

Uwagi:

(1) Dodawanie liczb o znakach zgodnych lub odejmowanie liczb o znakach przeciwnych.
(2) Dodawanie liczb o znakach przeciwnych lub odejmowanie liczb o znakach zgodnych.

Jeżeli znaki są nieznane, trzeba przyjąć dolną granicę jak dla znaków zgodnych, a górną jak dla przeciwnych.

Czasy operacji konwersji (w mikrosekundach) przy założeniach jak wyżej:

P1FLTU	P1FLTS	P1FLMT	P2FLTU	P2FLTS	P2FLTM
60–130	80–150	100–160	70–200	120–260	100–240
PUNIN1	PSNIN1	PMNIN1	PUNIN2	PSNIN2	PMNIN2
110–140	120–150	140–160	110–140	120–150	130–160

Czasy złożonych operacji matematycznych (w mikrosekundach):

Podane w poniższych tablicach czasy oszacowano przy założeniach jak dla procedur realizujących 4 proste funkcje arytmetyczne. Dodatkowo przyjęto, że każda procedura wykonuje obliczenia dla takich argumentów, które powodują skrajnie niekorzystne ze względów czasowych jej działanie. Praktycznie, średnie teoretyczne czasy wykonywania każdej z podanych procedur są krótsze o około 15–20%.

Procedury obliczające funkcje trygonometryczne:

sinus				cosinus			
PUSIN	PUSINS	PSSIN	PSSINU	PUCOS	PUCOSS	PSCOS	PSCOSU
1560	1580	1640	1610	1550	1580	1630	1610

Procedury obliczające funkcję atan2:

PUATAN2	PUATAN2S	PSATAN2	PSATAN2U
1530	1550	1580	1570

Procedury obliczające pierwiastek kwadratowy:

PUSQRT	PUSQRTS	PSSQRT	PSSQRTU
1200	790	840	1250

Rozpakowanie (w mikrosekundach):

PSUNPU	PMUNPU
46.2	58.8

Niektóre podprogramy organizacyjne (w mikrosekundach):

PSTSW	PSTWRL	PCLEX
14.4	29.6	4.6

Podane czasy nie obejmują sekwencji wywołania procedur. Typowe wywołanie procedury 3–argumentowej ma postać (w komentarzach podano liczbę cykli przypadającą na każdą instrukcję):

```
MOV  BX OFFSET DGROUP:A      ; 4
PUSH BX                      ;11
MOV  BX OFFSET DGROUP:B      ; 4
PUSH BX                      ;11
MOV  BX, OFFSET DGROUP:C     ; 4
PUSH BX                      ;11
CALL PSDIV                   ;19
```

Sekwencja ta trwa 64 cykle tj. 12.8 mikrosekundy dla zegara 5 MHz.

Ponadto może być konieczne przechowanie rejestrów, których zawartość biblioteka niszczy – po 3.8 mikrosekundy na rejestr składany i zdejmowany ze stosu instrukcjami PUSH i POP.

Niżej podano tablicę porównującą czasy niektórych operacji zmiennoprzecinkowych dla procesora Intel–8087, formowego emulatora programowego (według [3]) i naszej biblioteki „P". Dane dla koprocesora i emulatora dotyczą formatu „temporary real" o bazowanym wykładniku 15–bitowym i 64–bitowej mantysie. Dla biblioteki dane dotyczą formatów rozpakowanych, bez czasów wywołań i przechowywania rejestrów.

Operacja	Przybliżony czas w mikrosekundach dla zegara 5 Mhz		
	8087	Emulator na 8086	Biblioteka "P" na 8086
Dodawanie/odejmowanie	17	1600	250
Mnożenie	27	2100	260
Dzielenie	39	3200	360
Pierwiastek kwadratowy	36	19600	1200

Biblioteka jest znacznie szybsza od emulatora, ale za cenę następujących ograniczeń:

1. Mniejsza dokładność.
   Mantysa biblioteki ma 32 bity (dokładność 9 cyfr dziesiętnych), zaś koprocesor i emulator używają mantys 64–bitowych (19 cyfr dziesiętnych).

2. Ustalenie przy realizacji biblioteki pewnych parametrów, których wybór umożliwiają koprocesor i emulator: sposobu zaokrąglenia, typu ∞ i precyzji obliczeń. Ponadto biblioteka wyklucza pracę bez normalizacji argumentów i wyników.

3. Bardziej uciążliwe programowanie.
   Operacje koprocesora i emulatora są wywoływane instrukcjami assemblera 8086 przy użyciu stosu danych zmiennoprzecinkowych. Mogą to być instrukcje 1–argumentowe (dopuszczalne są wszystkie sposoby adresowania) lub nawet bezargumentowe – działające wyłącznie na stosie. Daje to zwartość programu, nie ogranicza liczby i wielkości bloków danych i nie niszczy rejestrów uniwersalnych.
   Te możliwości, naturalne dla rozwiązania sprzętowego (8087) zajmują dużo czasu przy realizacji programowej, toteż w bibliotece „P" zrezygnowano z nich.

10. Rezerwacja pamięcie przez bibliotekę „P"

W obecnym wykonaniu procedury wymienione w rozdziale 8 zajmują łącznie obszar 60H = 96 bajtów w segmencie DATA, 17C0H = 6080 bajtów w segmencie CONST i 19C0H = 6592 bajtów w segmencie CODE. Ponadto, jak uprzednio wspomniano, biblioteka używa stosu programu wywołującego, gdzie jej zapotrzebowanie (z pewnym marginesem) wynosi 30 słów. Wartość ta nie uwzględnia rezerwacji dokonywanych przez handler obsługi błędów, ani przypadku rekursyjnego wywołania handlera.

11. Uwagi końcowe

Przy realizacji biblioteki za najważniejsze kryterium przyjęto minimalny czas wykonania, przy założonej dokładności, wynoszącej ok. 7 lub 9 cyfr dziesiętnych. Uzyskane czasy (zob. rozdz. 9) pozwalają na wykorzystanie biblioteki przez program sterujący pracą złożonego robota przemysłowego.

Ogólnie rzecz biorąc – biblioteka „P" przeznaczona jest do wykorzystania w urządzeniach pracujących w czasie rzeczywistym, sterowanych przez mikroprocesory typu Intel 8086. Może być zastosowana w typowych układach rejestracji i sterowania, w których aktualnie używa się przetworników analogowo/cyfrowych i cyfrowo/analogowych o dokładności ok. 4 cyfr dziesiętnych. Oczywiście, można jej użyć do wszelkich obliczeń, dla których oferowane parametry są wystarczające. Biblioteka może w pewnym zakresie zastąpić obecnie trudno dostępny w kraju koprocesor Intel 8087, który jest oczywiście szybszy i dokładniejszy (zob. rozdz. 9). Bibioteka „P" nie jest zasadniczo przeznaczona do złożonych obliczeń matematycznych o dużej precyzji, tak ze względu na czas wykonania, jak i dokładność.

Informacje na temat biblioteki „P" można uzyskać w Ośrodku Automatyzacji Procesów Produkcji (OAP) w PIAP, tel. 23 84 89.

12. Literatura (dostępna w PIAP)

[1] „8086/8087/8087 MACRO ASSEMBLY LANGUAGE REFERENCE MANUAL for 8080/8085-Based Development Systems"
Chapter 6: The 8087 Instruction Set.
Intel 1980, Manual Order No. 121623-001 Rev.A.

[2] „Microprocessor and Peripheral Handbook" – katalog Intela z 1983r.
Manual Order No. 210844-001.

[3] „iAPX 86/88, 186/188 User's Manual. Programmer's Reference" Vol. 1, Chapter 6: „The 8087 Numeric Processor Extension".
May 1983. Wydawnictwo Intela. Manual Order No. 210911-001.

[4] Jerome T. Connen: „Underflow and the Denormalized Numbers". COMPUTER, March 1981, s. 75-87.

[5] Banasik Z.: „Projekt normy IEEE – arytmetyka zmiennoprzecinkowa". INFORMATYKA Nr 1/1985.

[6] Startz R.: „8087 Applications and Programming for the IBM PC and Other PCs".
Wyd. w 1983 r. przez Robert J. Brady Co., Prentice-Hall, Bowie, Maryland 20715,USA.
ISBN 0-89303-420-7.

[7] „A Proposed Standard for Binary Floating-Point Arithmetic, Draft 8.0 of IEEE Task P754".
COMPUTER, March 1981, s.51-62.