PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
MERA-PIAP
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81

Ośrodek Automatyzacji Procesów Produkcji (OAP)

4410 BE1

Główny Wykonawca mgr inż. Kazimierz Maliszewski

Wykonawcy

Konsultant

Nr zlecenia S 1237

Wybór oprogramowania do obsługi prac laboratoryjnych, naukowo-badawczych oraz wybór i opracowanie wzorów dokumentów stosowanych do tych prac w PIAP. Wybór oprogramowania dla ochrony komputerów przed wirusami.
Etap 8: Opracowanie biblioteki matematycznej w wersji do systemu operacyjnego czasu rzeczywistego.

Zleceniodawca

Pracę rozpoczęto dnia 1991.06.01 zakończono dnia 1991.12.15

Z-ca Dyrektora d/s Badawczo-Rozwojowych
dr inż. J. Jabłkowski

Kierownik Ośrodka
dr inż. Marian Wrzesień

Praca zawiera:
stron 10
rysunków
fotografii
tabel
tablic
załączników 1

Nr rejestr. 6793

Rozdzielnik - ilość egz: 4

Egz. 1 Archiwum
Egz. 2 ORC
Egz. 3 ZSS
Egz. 4 OAP
Egz. 5
Egz. 6

Analiza deskryptorowa

Analiza dokumentacyjna

Tytuły poprzednich sprawozdań

UKD

PIAP 41/88 10000

2

SPIS TREŚCI

UWAGA: ZASTRZEGA SIĘ MOŻLIWOŚĆ WPROWADZANIA ZMIAN

1. WPROWADZENIE

Przedmiotem niniejszej pracy jest opracowanie minimalnej biblioteki zmiennoprzecinkowych procedur matematycznych, które mogą być wykorzystane w programach pracujących pod systemem operacyjnym czasu rzeczywistego SIRTOS (zob. Biuletyn PIAP nr 4/144 z 1989 r). W tym celu opracowano drugą wersję biblioteki procedur matematycznych "P". Opis pierwszej wersji tej biblioteki podano w Biuletynie PIAP nr 3/116 z 1986 r, z późniejszą erratą. Określono tam szereg używanych tutaj terminów. Znajomość tego artykułu jest praktycznie niezbędna dla pełnego wykorzystania możliwości biblioteki. Niniejszy opis skupia się głównie na zmianach wprowadzonych do pierwszej wersji.

Ogólnie rzecz biorąc, zmiany te mają dwa cele:
1) przystosowanie biblioteki do wywołania w języku C, a ściśle AZTEC C, czyli w języku aktualnie stosowanym w systemie SIRTOS;
2) przystosowanie biblioteki do pracy wielowejściowej w sensie wykorzystania tego samego programu biblioteki przez różne zadania, przerywane i wznawiane zgodnie ze strategią systemu operacyjnego.

Zakres pracy obejmuje opisane w wersji 1 podstawowe operacje arytmetyczne, (cztery działania), pomocnicze operacje matematyczne (konwersje stało - zmiennoprzecinkowe i rozpakowanie) oraz procedury organizacyjne (kontrola sposobu obsługi błędów). Praca nie obejmuje złożonych operacji matematycznych wersji 1 (pierwiastka i funkcji trygonometrycznych).

2. PRZYSTOSOWANIE BIBLIOTEKI "P" DO JĘZYKA AZTEC C

2.1. Nowy zbiór wywołań procedur

Dla odróżnienia procedur wersji 2 od 1, zmieniono pierwszą literę wszystkich nazw z P na C. "P" w nazwach pierwszej wersji można teraz interpretować jako "pierwsza" lub "do języka PL/M", zaś "C" drugiej wersji oznacza "przystosowana do języka (AZTEC) C". Ponadto występująca w nazwach pierwszej wersji litera M, oznaczająca format Middle, została zastąpiona przez D (Double). Po trzecie - wszystkie nazwy zostały zakończone znakiem podkreślenia (_), którego wymaga AZTEC C. Ten znak jest niewidoczny w języku C, tzn. nie należy go dopisywać, ale jest konieczny przy wywołaniach procedur w asemblerze.

W związku z nową postacią wywołania handlera błędów, procedura PSTWRL z wersji 1 została usunięta. Pełną listę wywołań podano w p.4.

Argumenty zmiennoprzecinkowe są tak jak poprzednio przekazywane przez adresy na stosie. W przypadku par segment-offset najpierw składa się segment, a potem (pod niższym adresem) offset.

Argumenty stałoprzecinkowe są przekazywane wprost przez wartość na stosie, a nie przez adresy, jak w wersji pierwszej.
W przypadku liczb dwusłowowych, zgodnie z konwencją firmy Intel, najpierw składa się słowo bardziej znaczące, a potem (pod niższy adres) słowo mniej znaczące.

Jeżeli wynik procedury jest całkowity, to jest on przekazywany w języku AZTEC C przez nazwę funkcji, a w asemblerze przez rejestr AX (wynik 1-słowowy) lub przez parę DX, AX (wynik 2-słowowy ze starszą częścią w rejestrze DX).

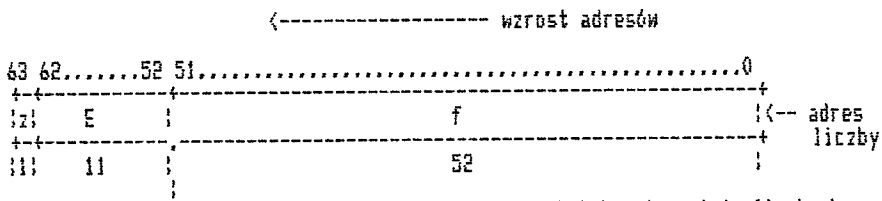
Zgodnie z konwencją języka C, argumenty powinny być składane na stosie w kolejności odwrotnej do istniejącej w wywołaniu, przez co na stosie tworzy się sekwencja (ze wzrostem adresów) zgodna z kolejnością istniejącą w wywołaniu. Jest to kolejność odwrotna niż w wersji 1, zgodnej z PL/M.

Przy wyjściu z procedury, zgodnie z wymogiem języka C, argumenty pozostają na stosie i muszą być usunięte przez program wywołujący. W wersji 1 argumenty były usuwane przez bibliotekę.

2.2. Liczba zmiennoprzecinkowa długa (Double czyli Long Real)

Zmiennoprzecinkowy format średni (Middle Real) został w wersji 2 zastąpiony przez format długi (Double według nomenklatury C, Long Real według nazewnictwa Intela).

Liczba zmiennoprzecinkowa długa ma postać upakowaną w czterech słowach:



z - oznacza znak całej liczby: 0 oznacza +, 1 oznacza -.
Liczba przeciwna do danej różni się od niej tylko znakiem.
E - pamiętany wykładnik
f - ułamek
Zakresy wartości pól (takie same, jak dla formatu średniego):

$$0 \le E \le MaxE = 2047$$
$$0 \le f \le 1 - 2^{-31}$$

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
	1≤E≤2046	-1022≤e≤1023	$0 \le f \le 1-2^{-31}$	$(-1)^z * 2^{E-1023} * 1.f$	Znormalizowana (normal)
D	0	/1/ -1022	$0 < f \le 1-2^{-31}$	$(-1)^z * 2^{-1022} * 0.f$	Denormal
0	0	-	0	$(-1)^z * 0$	Zero ze znakiem
n	2047 (111...1)	/2/ 1024	0	$(-1)^z * \infty$	Nieskończoność ze znakiem
	2047	1024	$\neq 0$	-	Wartość nienumeryczna
1	2047	1024	1/2 (1000...0)	-	W. niemumeryczna "niezdefiniowana"

Uwagi:

/1/ Jest to eden (wykładnik denormali i najniższy wykładnik normali) formatu długiego.
/2/ Jest to ensk (symboliczny wykładnik nieskończoności i NaN-ów) formatu długiego, ensk = MaxE - 1023.

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

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

$$\text{denormale: } 2^{-1074} \le |x| \le 2^{-1022} * (1-2^{-31}) < 2^{-1022} \quad \text{; dokładność od } 10^{-9} \text{ do } 0.5$$

Format długi niniejszej wersji biblioteki można by słuszniej nazwać formatem "pseudo-długim", gdyż nie jest realizowana dokładność formatu długiego. Mianowicie, 52-bitowy ułamek tego formatu jest na wejściu (w rejestrach biblioteki) zaokrąglany do 31 bitów, obliczenia są wykonywane na 32 bitach, a na wyjściu 31-bitowy ułamek wyniku jest dopełniany z prawej zerami do 52 bitów. Powód jest prosty - biblioteka wykonuje obliczenia w formacie rozpakowanym o 32-bitowej mantysie. Tak więc dokładność formatu długiego jest tylko ok. 100 razy większa niż krótkiego i równa dokładności formatu rozpakowanego.

Jeżeli rozpakowany wynik ma wykładnik mniejszy od eden, to przy pakowaniu następuje denormalizacja mantysy na 1+52 bitach. Wtedy biblioteka może zapisać ostatnie 21 bitów f zgodnie z wynikiem, ale zawsze jest najwyżej 31 bitów znaczących, a poprzedzające i dopełniające są zerowe.

Format długi (Double), choć tak ograniczony, jest ważny dla programów pisanych w AZTEC C, gdyż kompilator używa go dla parametrów procedur nawet deklarowanych jako pojedyńczej długości (Float w C czyli Short Real Intela). Niniejsza wersja rezygnuje z formatu średniego, gdyż jest on nieznany dla kompilatorów.

Zauważymy na koniec, że dość oryginalne zaokrąglenie już w fazie rozpakowania argumentów powoduje z reguły ustawienie w słowie stanu błędu P jeszcze przed obliczeniem wyniku. Błąd ten będzie widoczny w słowie stanu przy obsłudze przy pomocy handlera ewentualnych dalszych błędów.

2.3. Modele pamięci

Wersja 2 biblioteki P stosuje się do modeli pamięci tj. segmentacji programu przyjętej przez kompilator języka AZTEC C. Program po kompilacji może mieć "małe dane" (Small Data) lub "duże dane" (Large Data) i "mały" lub "duży" kod (Small Code lub Large Code). Wiąże się to ze sposobem adresacji danych i kodu.

Dla "małych danych" rejestry segmentowe DS i SS przy wejściu do biblioteki powinny mieć tę samą wartość, równą bazie segmentu o nazwie dataseg. Biblioteka nie zmienia tej wartości (a ponadto nie używa rejestru ES).

Adresy argumentów są w tej wersji są 1-słowowymi offsetami w segmencie dataseg.

Dla "dużych danych" nie zakłada się, że DS i SS mają równe wartości. Rejestry DS i ES są z reguły (bo niekiedy nie potrzeba) przechowywane na wejściu i odtwarzane na wyjściu z procedury.

Adresy argumentów są w tej wersji parami słów segment-offset.

Kod biblioteki mieści się w segmencie o nazwie codeseg.

Dla "małego kodu" wszystkie procedury biblioteki są typu "Near". Kod wywołujący bibliotekę powinien mieć w rejestrze CS bazę segmentu codeseg, a wywołująca instrukcja CALL powinna składać na stosie tylko rejestr IP.

Dla "dużego kodu" zewnętrzne procedury biblioteki są typu "Far". Wywołująca instrukcja CALL powinna składać na stosie rejestry CS i IP.

Wybór modelu danych i kodu dokonuje się w fazie asemblacji biblioteki przez nadanie wartości symbolom @LD i @LC.

"Małe dane" deklaruje się przez nadanie @LD wartości 0, "duże dane" - przez nadani @LD wartości 2 (nie 1!).

Analogicznie "mały kod" wymaga nadania symbolowi @LC wartości 0, zaś "duży kod" wymaga, by @LC miał wartość 2.

2.4. Wykorzystanie rejestrów procesora

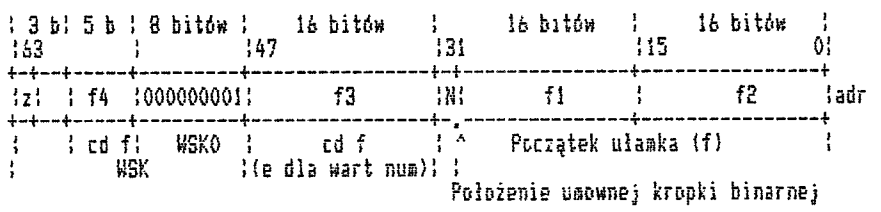
Zgodnie z wymaganiami kompilatora cc języka AZTEC C biblioteka P zawsze zachowuje lub przechowuje rejestry segmentowe (CS, DS, ES i SS), a ponadto rejestry BP, SI i DI oraz w rejestrze flag bity sterujące: TF, DF i IF. Zawartość pozostałych rejestrów i flag na wyjściu z biblioteki trzeba uważać za zniszczoną. Jedynie w przypadku procedur organizacyjnych i to jeżeli nie było wejścia do handlera błędów, zachowana jest także zawartość rejestru CX.

2.5. Nowa postać rozpakowanej wartości nienumerycznej

Wprowadzenie upakowanego formatu długiego daje potencjalną możliwość uzycia danej nienumerycznej (NaN) w tym formacie (jakkolwiek autorowi pracy nie wiadomo o używaniu takich danych przez AZTEC C).

52-bitowy ułamek długiej wartości nienumerycznej właśnie z powodu "nienumeryczności" nie powinien być zaokrąglany. W związku z tym rozszerzono definicję rozpakowanej wartości nienumerycznej.

Rozpakowana wartość nienumeryczna (NaN) ma teraz ułamek 52-bitowy, który jest rozmieszczony w 4 słowach liczby w sposób pokazany na rysunku:



Opis pól:

z - znak liczby: 0 oznacza "+", 1 oznacza "-" .

WSKO=1 - ustawiony bit NaN określa, ze liczba ma wartość nienumeryczną.

N - wiodący bit mantysy - jest przy rozpakowaniu upakowanej wartości nienumerycznej umownie ustawiany na wartość 1.

f1 (15 bitów), f2 (16 bitów), f3 (16 bitów) i f4 (5 bitów) - w tej kolejności składają 52-bitowy ułamek liczby, który powinien być różny od zera.

Wartość nienumeryczna "niezdefiniowana" jest określona przez: z=1, N=1, f1=1/2 (100...0), f2=0, f3=0, f4=0, a więc przez wartość szesnastkową 8001H, 0, C000H, 0. W pamięci obowiązuje kolejność słów (ze wzrostem adresów): 0, C000H, 0, 8001H.

Jak wiadomo, biblioteka P udostępnia operacje na danych rozpakowanych, a wyniku upakowanym. Może stąd powstać konieczność zapakowania wartości nienumerycznej do innego formatu, niż ten, w którym została zdefiniowana.

Jeżeli NaN ma być wydany w postaci upakowanej, to ułamek podlega obcięciu do długości przewidzianej przez format przeznaczenia, a jeśli po tej operacji jest zerowy, to (uwaga !) jest zastępowany przez 1 na najmniej znaczącym bicie formatu upakowanego, aby wartość różniła się od nieskończoności. Jeżeli obcinana część jest różna od zera, jak i w przypadku zastąpienia f=0 przez 1 na ostatniej pozycji, jest sygnalizowany błąd P.

3. PRZYSTOSOWANIE BIBLIOTEKI "P" DO SYSTEMU OPERACYJNEGO "SIRTOS"

3.1. Lokalizacja zmiennych wewnętrznych biblioteki

Wielowejściowość 2 wersji biblioteki wyklucza używanie statycznych zmiennych lokalnych, które istnieją w wersji 1.

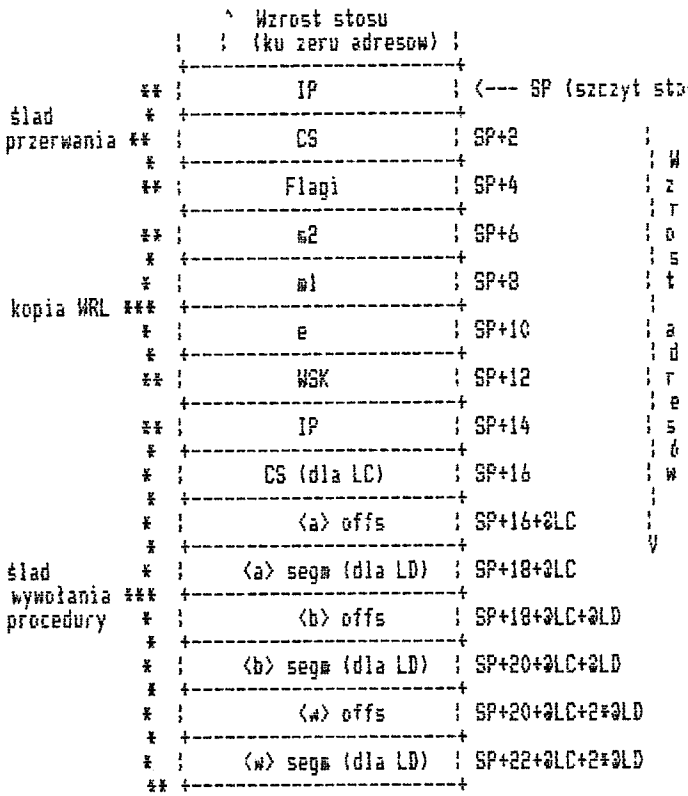
Zmienne wewnętrzne biblioteki, które są potrzebne do obsługi jednego wywołania, są przy wejściu do biblioteki rezerwowane na szczycie stosu w postaci obszaru o długości 14 słów, zwanego w projekcie biblioteki ENV. Przy wyjściu obszar jest zwalniany. W stosunku do wersji 1 jest to dla użytkownika zmiana niezauważalna, poza niewielkim wydłużeniem czasu obliczeń i większym zapotrzebowaniem na pamięć stosu.

Zmienne biblioteki, które muszą być zachowane pomiędzy wywołaniami (słowo stanu, słowo sterujące i maska otrzymywana na podstawie słowa sterującego) są przechowywane na dnie stosu każdego zadania. Adres tych 3 słów musi być przy starcie zadania umieszczony przez system operacyjny w stałym miejscu pamięci, mianowicie na ustalonej pozycji wektora przerwań. Program użytkowy również nie widzi tej zmiany, poza narzutem czasu związanym z dostępem do tych danych (zwłaszcza w przypadku błędu).

3.2. Zmiana postaci wywołania handlera błędów

Wewnętrzny akumulator biblioteki (WRL) jest w wersji 2 rezerwowany wśród innych zmiennych na szczycie stosu. Przed wejściem do handlera błędów wszystkie zmienne biblioteki muszą być zdjęte ze stosu. W związku z tym zawartość WRL jest przekazywana handlerowi w postaci 4 słów umieszczanych na stosie tuż pod śladem przerwania (na większych adresach). Handler ma do nich dostęp przez ich deklarację w postaci 4 argumentów całkowitych. Wobec tego zlikwidowano procedurę PSTWRL (wraz z możliwością jej niewłaściwego użycia).

Poniższy rysunek przedstawia postać stosu przy wywołaniu handlera dla obsługi błędu procedury o argumentach zmiennoprzecinkowych (a, b, w). Adresy argumentów zaznaczono ostrymi nawiasami; LC oznacza "Large Code" (wtedy @LC=2, a inaczej 0); LD znaczy "Large Data" (wtedy @LD=2 a inaczej 0).



Przy założeniu "małego kodu" handler (umownie nazwany HANBL) mógłby być procedurą o parametrach:

HANBL (cs, fl, m2, m1, e, wsk, &proc, &a, &b, &w) ;

gdzie cs, fl są nieistotnymi dla handlera argumentami typu całkowitego, (dla modelu "dużego kodu" cs nie występuje, gdyż wchodzi do śladu wywołania);

m2, m1, e, wsk (typu całkowitego) są kolejnymi słowami tablicy WRL;

&proc jest adresem, który przybiera wartość śladu (miejsca) wywołania procedury bibliotecznej;

&a, &b, &w są adresami argumentów tej procedury.

Wśród argumentów nie ma kodu rodzaju procedury, co poważnie ogranicza możliwości handlera. Wprowadzenie takiego parametru (co wiąże się z nieznacznym spowolnieniem wszystkich procedur zewnętrznych) jest możliwe w późniejszych wersjach lub na życzenie klienta.

Powrót z handlera powinien być wykonany w C instrukcją return(), zaś w asemblerze, dla zgodności z C, instrukcją RET (nie IRET !), odpowiednią dla modelu kodu, w którym była asemblowana biblioteka, a więc temu, w którym wywoływane są jej procedury.

4. LISTA WYWOŁAŃ PROCEDUR BIBLIOTEKI P

4.1. Wprowadzenie

Podprogramy wchodzące w skład 2 wersji biblioteki P można podzielić na trzy grupy: podstawowe operacje arytmetyczne, pomocnicze procedury matematyczne i procedury organizacyjne dla procesu obliczenia.

Nazwy procedur

Pierwszy znak nazwy jest literą C.

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

S - argument(y) formatu "Short Real"
D - argument(y) formatu "Double" czyli "Long Real"
U - argument(y) formatu "Unpacked Real"
1 - argument całkowity 1-słowowy
2 - argument całkowity 2-słowowy

Następne litery nazwy (zwykle trzy) określają krótko funkcję 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. CUADD oznacza dodawanie o wszystkich argumentach rozpakowanych, zaś CUADDS - o argumentach rozpakowanych, a wyniku zmiennoprzecinkowym krótkim. C1FLTS - oznacza konwersję liczby całkowitej 1-słowowej na zmiennoprzecinkową krótką, zaś CSNINi - odwrotnie.

Argumenty procedur

Argumenty procedur w niniejszym opisie są oznaczone dwu- lub trzyznakowo. Przykłady: &as, &bd, &wu, i1, i2, n1, n2.
W oznaczeniach trzyznakowych pierwszym znakiem jest zawsze &, który oznacza, że przekazuje się adres właściwego argumentu. W oznaczeniach dwuznakowych przekazuje się sam argument.
Pomijając znak &, pierwszy znak, jeżeli jest początkową literą alfabetu (a,b), określa, że argument jest zmiennoprzecinkowy wejściowy, jeżeli zaś jest "w", oznacza argument zmiennoprzecinkowy wyjściowy. Dla argumentów całkowitych wejściowych pierwszy znak jest literą i, zaś dla całkowitych wyjściowych - jest literą n.
Drugi znak (znów nie licząc &) określa długość i format argumentu. Litera "s" oznacza argument zmiennoprzecinkowy Short Real, "d" - zmiennoprzecinkowy Double, a "u" zmiennoprzecinkowy rozpakowany (Unpacked). Cyfra 1 lub 2 oznacza długość argumentu całkowitego w słowach.

Przykłady:
&as - adres argumentu wejściowego zmiennoprzecinkowego "Short Real"
&bd - adres argumentu wejściowego zmiennoprzecinkowego "Double"
i2 - argument wejściowy całkowity 2-słowowy ("Short Integer")
n1 - argument wyjściowy całkowity 1-słowowy ("Word Integer")
&wu - adres argumentu wyjściowego rozpakowanego ("Unpacked Real")

4.2. Podstawowe operacje arytmetyczne

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

Argumenty wejściowe (poprzedzone ponadto znakiem &):
(as,bs) (ad,bd) (au,bu) - liczby zmiennoprzecinkowe odpowiednio: krótkie, długie, rozpakowane.

Argumenty wyjściowe (poprzedzone ponadto znakiem &):
(ws,wd,wu) - liczby zmiennoprzecinkowe odpowiednio: krótkie, długie, rozpakowane.

Wywołania procedur			Operacja
CSADD (&as,&bs,&ws) CUADDS(&au,&bu,&ws)	CDADD (&ad,&bd,&wd) CUADDD(&au,&bu,&wd)	CUADD (&au,&bu,&wu)	w = a + b
CSSUB (&as,&bs,&ws) CUSUBS(&au,&bu,&ws)	CDSUB (&ad,&bd,&wd) CUSUBD(&au,&bu,&wd)	CUSUB (&au,&bu,&wu)	w = a - b
CSMUL (&as,&bs,&ws) CUMULS(&au,&bu,&ws)	CDMUL (&ad,&bd,&wd) CUMULD(&au,&bu,&wd)	CUMUL (&au,&bu,&wu)	w = a * b
CSDIV (&as,&bs,&ws) CUDIVS(&au,&bu,&ws)	CDDIV (&ad,&bd,&wd) CUDIVD(&au,&bu,&wd)	CUDIV (&au,&bu,&wu)	w = a / b

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

4.3. Pomocnicze procedury matematyczne

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

   Argumenty wejściowe:
   i1 - dana stałoprzecinkowa, 1-słowowa (Word Integer)
   i2 - dana stałoprzecinkowa, 2-słowowa (Short Integer)

   Argumenty wyjściowe (poprzedzone ponadto znakiem &):
   ws - wynik zmiennoprzecinkowy w formacie krótkim
   wd - wynik zmiennoprzecinkowy w formacie długim
   wu - wynik zmiennoprzecinkowy w formacie rozpakowanym

   Wywołania dla danej 1-słowowej:
   C1FLTS(i1,&ws) C1FLTD(i1,&wd) C1FLTU(i1,&wu)

   Wywołania dla danej 2-słowowej:
   C2FLTS(i2,&ws) C2FLTD(i2,&wd) C2FLTU(i2,&wu)

   Błędy: P (tylko C2FLTS) - wynik zaokrąglony.

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

   Podane niżej procedury zaokrąglają argument wejściowy do najbliższej liczby całkowitej, dając w wyniku liczbę stałoprzecinkową.

   Argumenty wejściowe (poprzedzone ponadto znakiem &):
   as - dana zmiennoprzecinkowa krótka (Short Real)
   ad - dana zmiennoprzecinkowa długa (Double)
   au - dana zmiennoprzecinkowa rozpakowana (Unpacked Real)

Argumenty wyjściowe:
n1 - wynik stałoprzecinkowy w 1 słowie
n2 - wynik stałoprzecinkowy w 2 słowach

Wywołania dla wyniku w 1 słowie:
n1 = CSNIN1 (&as) n1 = CDNIN1 (&ad) n1 = CUNIN1 (&au)

Wywołania dla wyniku w 2 słowach:
n2 = CSNIN2 (&as) n2 = CDNIN2 (&ad) n2 = CUNIN2 (&au)

Błędy: I - argument NaN, niemożliwość zapakowania liczby stałoprzecinkowej (wynik nie mieści się w zakresie argumentu przeznaczenia, dana oo). 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 (poprzedzone ponadto znakiem &):
as - dana zmiennoprzecinkowa w formacie krótkim
ad - dana zmiennoprzecinkowa w formacie długim
wu - wynik zmiennoprzecinkowy w formacie rozpakowanym

Wywołania:
CSUNPU (&as,wu) CDUNPU (&ad,wu)

Błędy: P - zaokrąglony argument Double.
Ponadto w słowie wskaźników wyniku mogą ustawić się bity "den" lub "NaN", które przy użyciu liczby jako argumentu operacji spowodują błędy odpowiednio D lub I.

4.4. Podprogramy organizacyjne

Podprogramy organizacyjne umożliwiają różne sposoby obsługi błędów 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 Intel 8087. Składają się one z litery C i skrótu określenia wykonywanej funkcji.
W opisach procedur podano rejestry, których zawartość jest niszczona. W każdym przypadku mogą też być zapisane flagi, z wyjątkiem TF, DF i IF.

1. Inicjacja biblioteki - CINIT
Podprogram CINIT (bez argumentów) ustawia początkowy stan biblioteki. Zeruje się słowo stanu (SW) tzn. wskaźniki błędów i zgłoszenie przerwania. Słowo sterujące (CW) przyjmuje wartość 1FFH tzn. wszystkie błędy mają obsługę zamaskowaną i przerwania są zablokowane. Niszczy się zawartość rejestrów AX i BX.

2. Odczyt słowa stanu (Store Status Word) - CSTSW
Bezargumentowa funkcja o wartościach słowowych CSTSW przekazuje przez swą nazwę (tj. przez rejestr AX) zawartość słowa stanu biblioteki. Niszczy się zawartość rejestrów AX, BX i DX.

3. Zerowanie błędów (Clear Exceptions) - CCLEX
Podprogram CCLEX (bez argumentów) zeruje młodszy bajt słowa stanu tzn. wskaźniki błędów i bit zgłoszenia przerwania IR. Niszczy się zawartość rejestrów AX i BX.

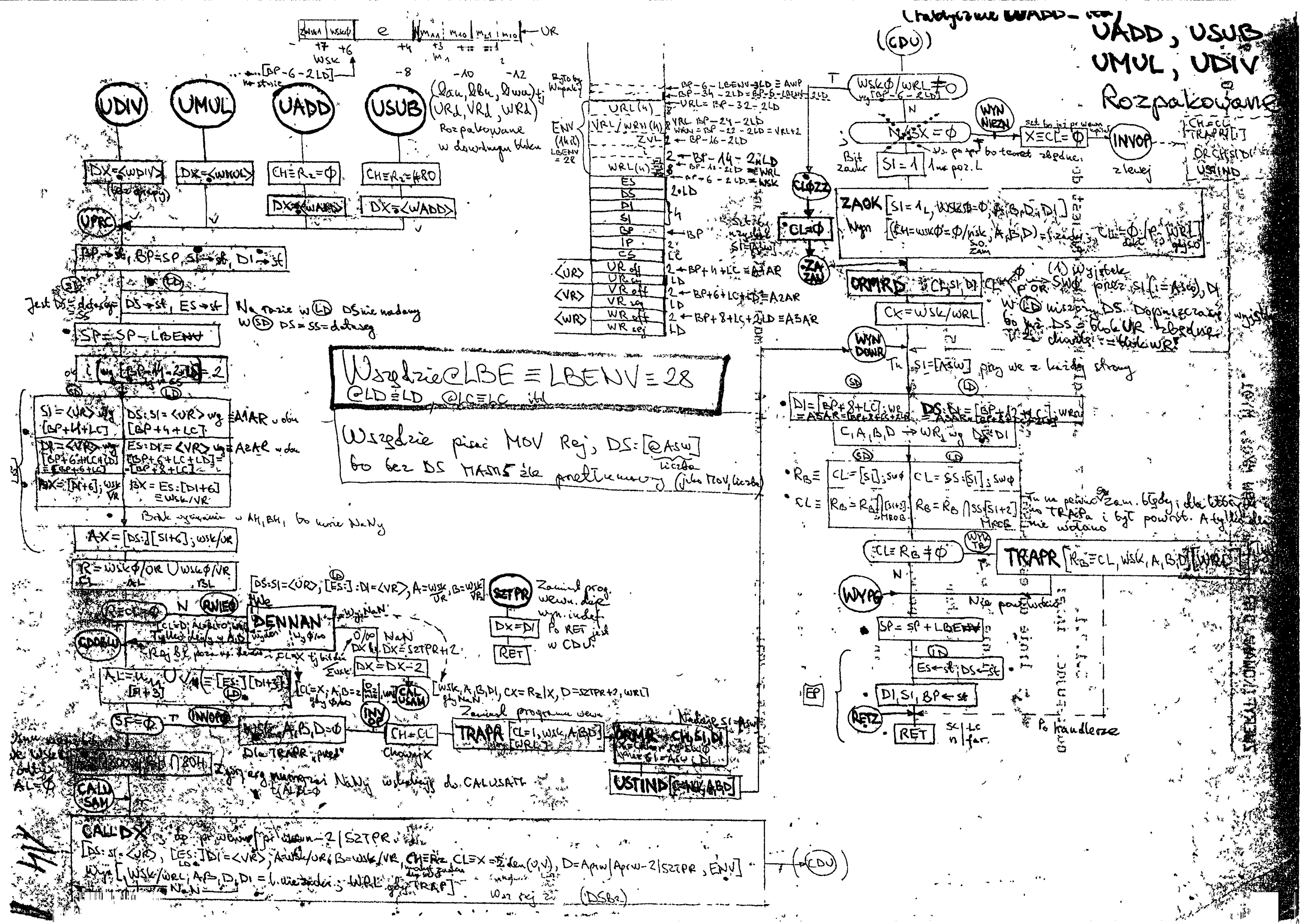
4. Odczyt słowa sterującego (Store Control Word) - CSTCW
   Bezargumentowa funkcja o wartościach słowowych CSTCW przekazuje przez swą nazwę (tj. przez rejestr AX) zawartość słowa sterującego biblioteki.
   Niszczy się zawartość rejestrów AX, BX i DX.

5. Nadanie wartości słowu sterującemu (Load Control Word) - CLDCW.
   Podprogram CLDCW(i1) wpisuje do słowa sterującego (CW) słowowy argument 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 CLDCW przerwanie i wejście do handlera błędów.
   Zalecaną praktyką jest wyzerowanie błędów w SW przed zmianą CW.
   W przypadku braku odmaskowanych błędów w SWO, niszczy się zawartość rejestrów AX, BX i DX.
   Jeżeli są odmaskowane błędy indywidualne, ale IEM=1, niszczy się zawartość rejestrów AX, BX, CX i DX (jak dla procedur obliczeniowych).
   Jeżeli są odmaskowane błędy indywidualne i IEM=0, nastąpi wejście do handlera i za ewentualne zniszczenie dalszych rejestrów poza AX, BX, CX i DX odpowiada handler.

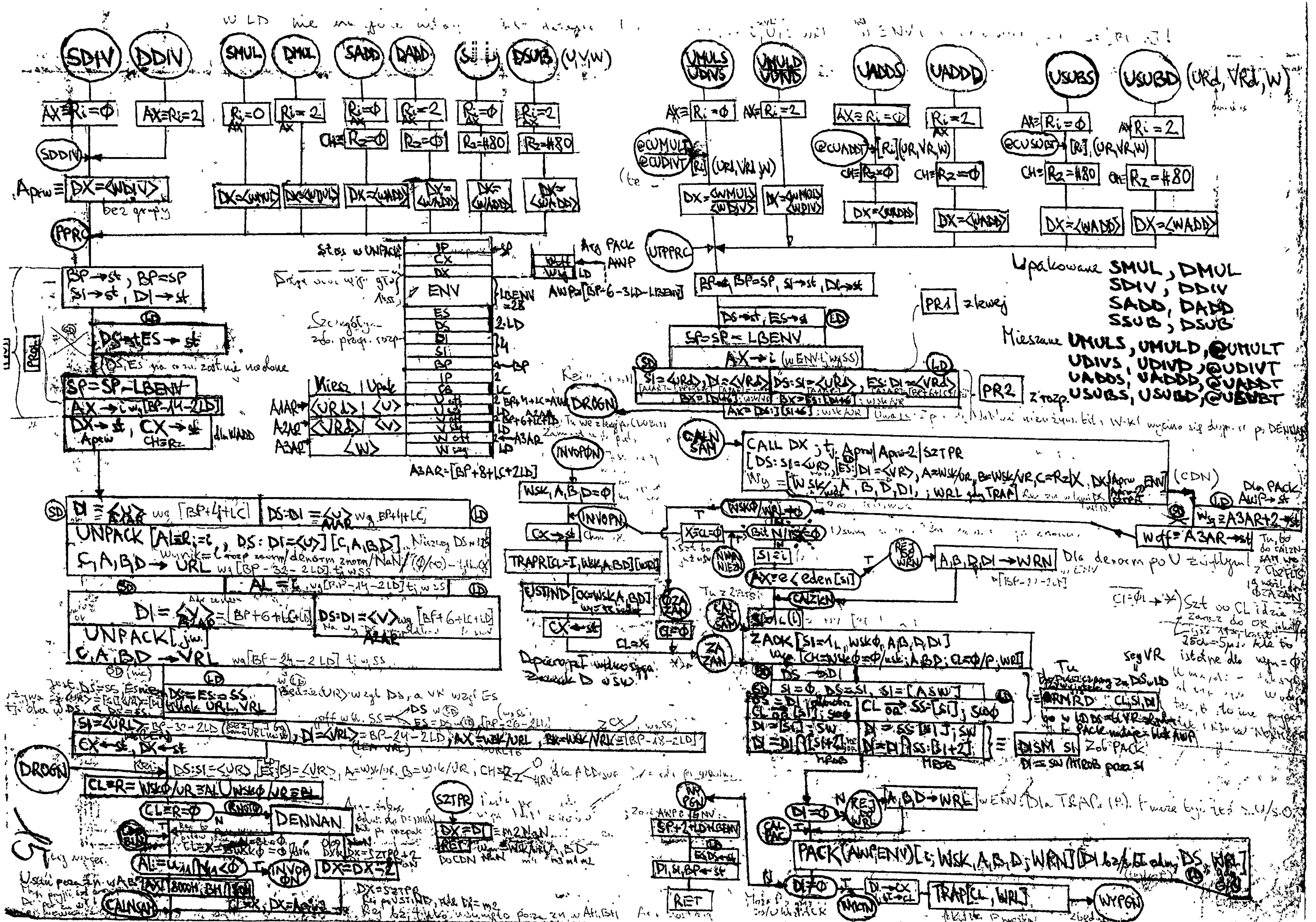
6. Odblokowanie przerwań z biblioteki (Enable Interrupts) - CENIN
   Podprogram CENIN (bez argumentow) zeruje w słowie sterującym bit IEM, dopuszczając wystąpienie przerwania od niezamaskowanego indywidualnie błędu ustawionego w SW. Tak jak dla CLDCW, zaleca się uprzednie wyzerowanie w SW dotychczasowych błędów.
   W przypadku braku odmaskowanych błędów w SWO, niszczy się zawartość rejestrów AX i BX.
   Jeżeli są odmaskowane błędy indywidualne, biblioteka niszczy zawartość rejestrów AX, BX, CX i DX, a za ewentualne dalsze odpowiada handler.

7. Blokada przerwań z biblioteki (Disable Interrupts) - CDISI
   Podprogram CDISI (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 SWO przed operacją i badanie go po operacji można, przy IEM=1, zrealizować własną obsługę błędów, nie korzystając z mechanizmu przerwań.
   Podprogram niszczy zawartość rejestrów AX i BX.

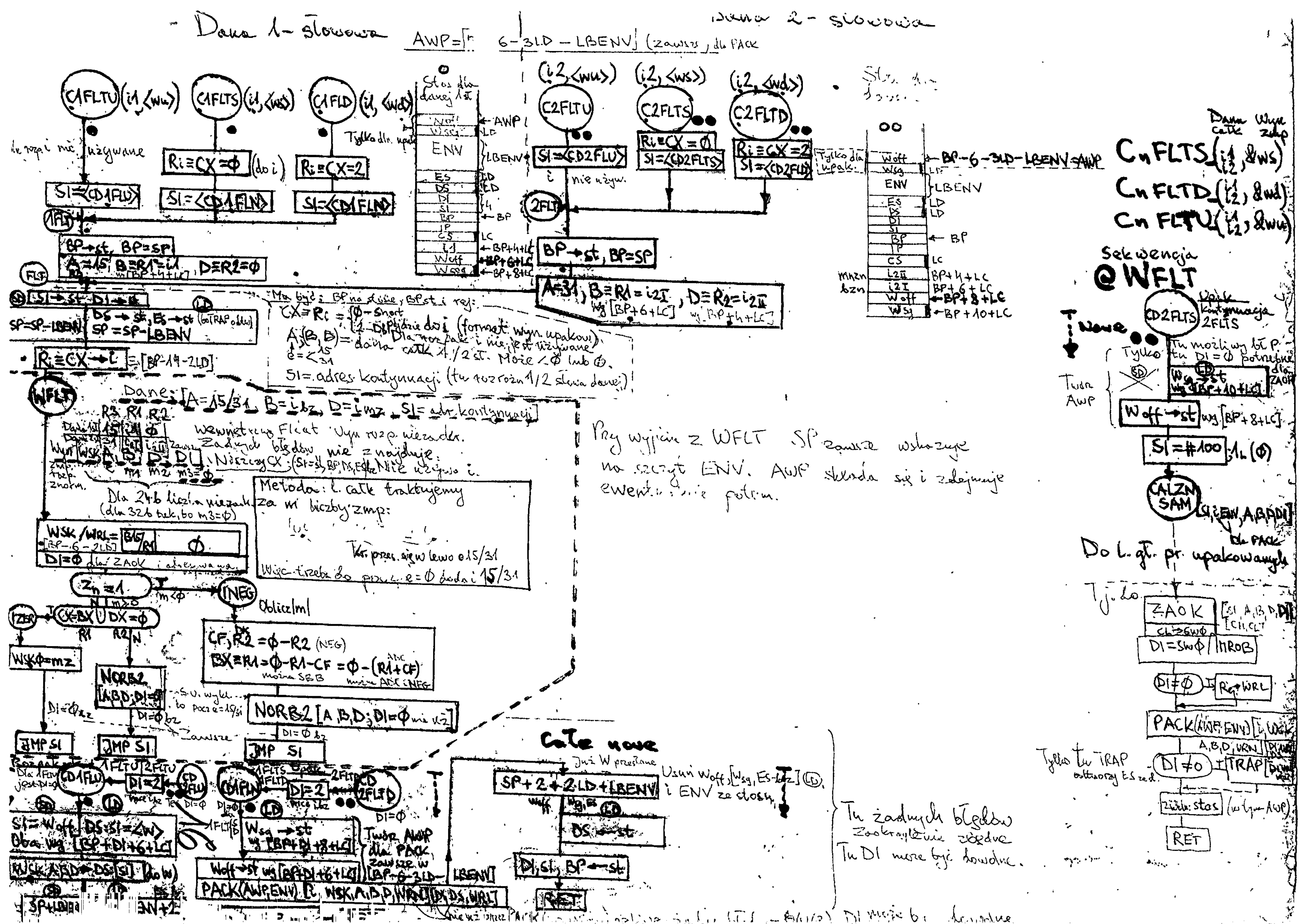
5. ZAŁĄCZNIK (POUFNY) - SIECI DZIAŁAŃ 2 WERSJI BIBLIOTEKI "P"



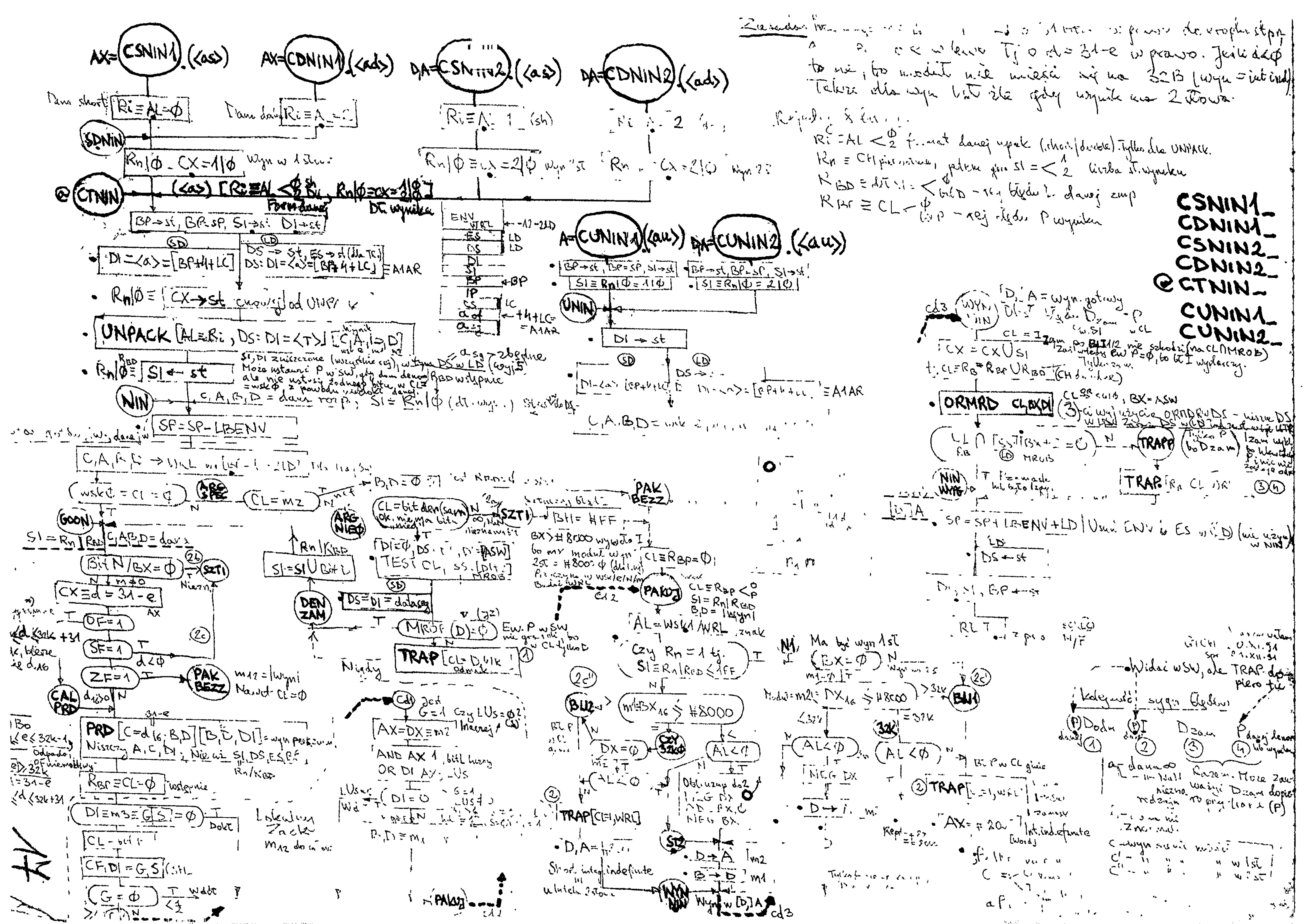
UADD, USUB
UMUL, UDIV
Rozpakowane
UDIV
UMUL
UADD
USUB
(CDU)
Wszędzie @LBE ≡ LBENV ≡ 28
@LD ≡ LD @LC ≡ LC itd
Wszędzie pisać MOV Rej, DS:[@ASW]
bo bez DS MASM5 źle przetłumaczy
SP = SP − LBENV
SP = SP + LBENV
DENNAN
TRAPR
ORMR
USTIND
ZAOK
ORMRD
CALL DX
CALUSAM
INVOP
SZTPR
DX = DI
RET
DI, SI, BP ← st
Po handlerze



SDIV
DDIV
SMUL
DMUL
SADD
DADD
DSUB
(U,V,W)
UMULS UDIVS
UMULD UDIVD
UADDS
UADDD
USUBS
USUBD
(URd, VRd, W)
AX=Ri=Φ
AX=Ri=2
SDDIV
Apr=DX=<WDIV>
bez grupy
PPRC
BP→st, BP=SP
SI→st, DI→st
DS→st ES→st
SP=SP-LBENV
DX→st, CX→st
UTPPRC
@CUMULT
@CUDIVT
@CUADDT
@CUSUBT
Ri=Φ
Ri=2
R2=Φ
R2=#80
DX=<WADD>
BP→st, BP=SP, SI→st, DI→st
DS→st, ES→st
SP=SP-LBENV
ENV
LBENV
Upakowane SMUL, DMUL
SDIV, DDIV
SADD, DADD
SSUB, DSUB
Mieszane UMULS, UMULD, @UMULT
UDIVS, UDIVD, @UDIVT
UADDS, UADDD, @UADDT
USUBS, USUBD, @USUBT
PR1 z lewej
PR2 z rozp.
DROGN
INVOPON
CALN SAM
CALL DX
UNPACK
URL
VRL
DS=ES=SS
CX←st, DX←st
DRDGN
CL=R=Φ
DENNAN
SZTPR
DX=DI
RET
INVOPON
CALNSAM
WSK,A,B,D=Φ
TRAPR
USTIND
ZAZAN
CALZKN
ZACK
ARMRD
DISM SI
RWRN
PACK (AWP,ENV)
TRAP[CL, WRL]
WYPGN
SP+2+LD+LBENV
DI,SI,BP←st
RET

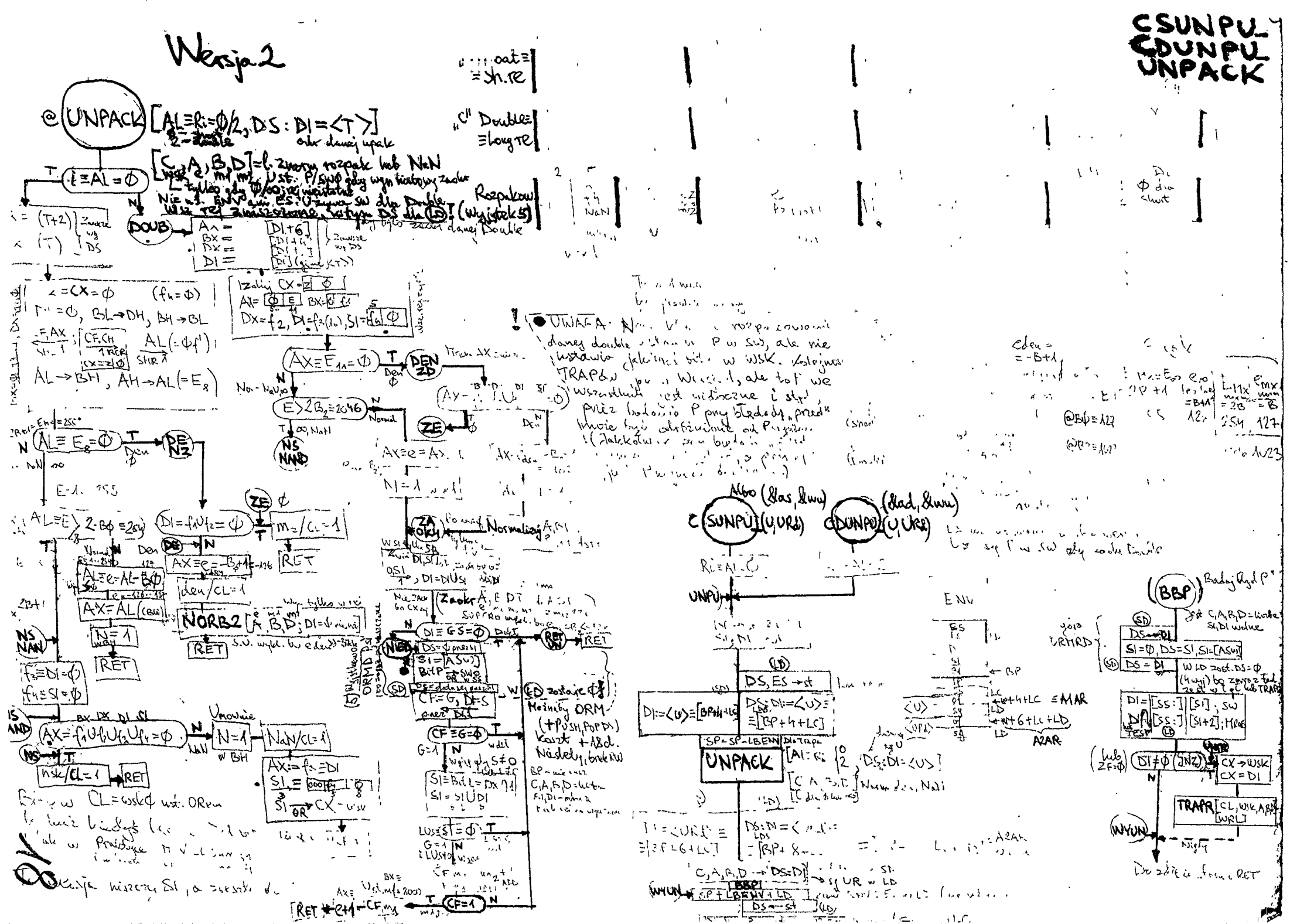


Dana 1-słowowa
AWP = [BP - 6 - 3LD - LBENV] (zawsze, dla PACK)
Dana 2-słowowa
C1FLTU (i1, <wu>)
C1FLTS (i1, <ws>)
C1FLD (i1, <wd>)
Ri = CX = Φ (do i)
Ri = CX = 2
SI = <CD1FLU>
SI = <CD1FLN>
1FLT
BP → st, BP = SP
A = 15, B = R1 = i1, D = R2 = Φ
FLT
SP = SP - LBENV
Ri = CX → i
WFLT
Dane: [A = 15/31, B = i bz, D = i mz, SI = adr. kontynuacji]
Wewnętrzny Float. Wyn. rozp. nieznorm.
Żadnych błędów nie znajduje.
Niszczy CX.
Metoda: l. całk. traktujemy za m liczby zmp.
Więc trzeba do pocz. e = Φ dodać 15/31
WSK/WRL
DI = Φ dla ZAOK
Zn = 1
INEG
Oblicz |m|
IZER
CX = BX ∪ DX = Φ
WSKΦ = mz
NORB2 [A, B, D; DI = Φ]
CF, R2 = Φ - R2 (NEG)
BX = R1 = Φ - R1 - CF = Φ - (R1 + CF)
NORB2 [A, B, D; DI = Φ nie uż.]
JMP SI
CD1FLU
DI = 2
CD1FLN
CD2FLTD
SI = Woff, DS:SI = <w>
Wsg → st
Woff → st wg [BP + DI + 6 + LC]
PACK (AWP, ENV) [L, WSK, A, B, D, WRN] [DX, DS, WRL]
(i2, <wu>)
(i2, <ws>)
(i2, <wd>)
C2FLTU
C2FLTS
C2FLTD
SI = <CD2FLU>
Ri = CX = Φ
SI = <CD2FLTS>
Ri = CX = 2
SI = <CD2FLD>
2FLT
BP → st, BP = SP
A = 31, B = R1 = i2I, D = R2 = i2II
Stos dla danej 1 sł.
Woff
Wsg
ENV
ES
DS
DI
SI
BP
IP
CS
i1
LBENV
BP - 6 - 3LD - LBENV = AWP
BP + 4 + LC
BP + 6 + LC
BP + 8 + LC
BP + 10 + LC
Cn FLTS (i1, i2, &ws)
Cn FLTD (i1, i2, &wd)
Cn FLTU (i1, i2, &wu)
Sekwencja @WFLT
CD2FLTS
Kontynuacja 2FLTS
Woff → st wg [BP + 8 + LC]
SI = #100
CALZN SAM
Do l. gł. pr. upakowanych
ZAOK
DI = SwΦ / MROB
DI ≠ Φ
PACK (AWP, ENV)
DI ≠ 0
TRAP
Zdejm. stos
RET
Całe nowe
SP + 2 + 2LD + LBENV
DS → st
DI, SI, BP ← st
RET
Usuń Woff, [Wsg, ES - bz] i ENV ze stosu
Tu żadnych błędów
Zaokrąglenie zwykłe
Tu DI może być dowolne.
Przy wyjściu z WFLT SP zawsze wskazuje na szczyt ENV. AWP składa się i zdejmuje ewentualnie potem.

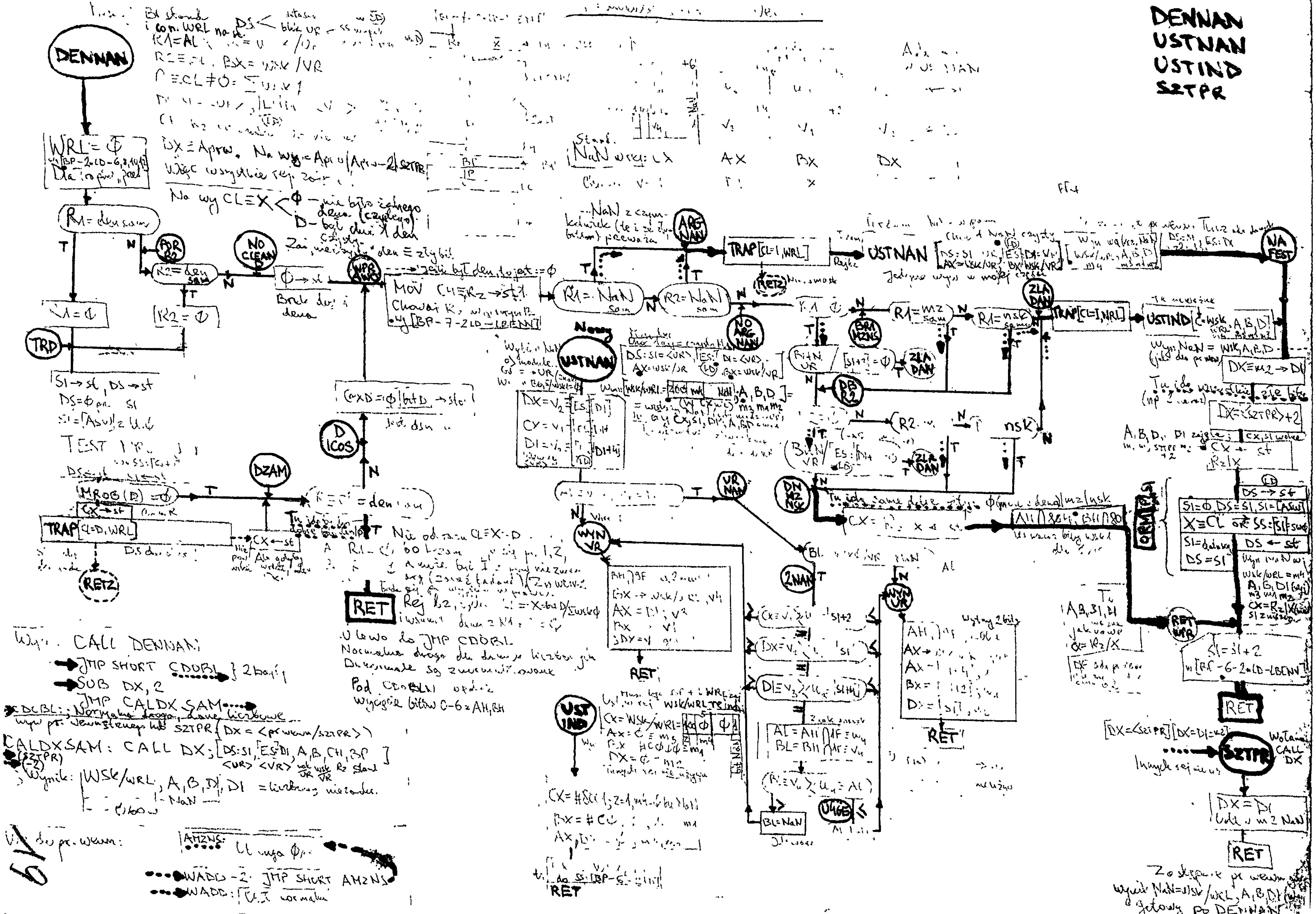


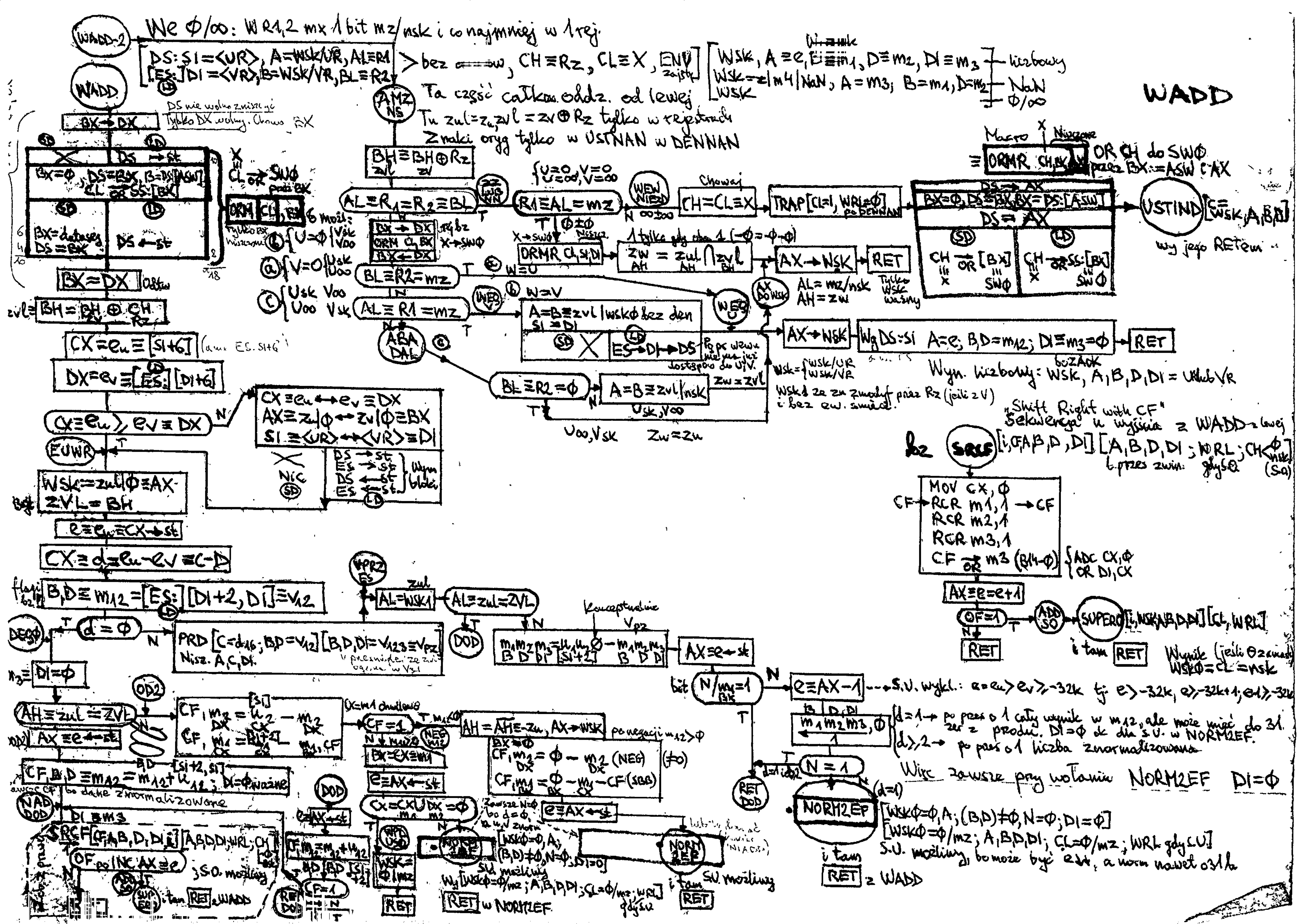
AX=CSNIN1.(<as>)
AX=CDNIN1.(<ad>)
DA=CSNIN2.(<as>)
DA=CDNIN2.(<ad>)
SDNIN
@CTNIN
A=CUNIN1.(<au>)
DA=CUNIN2.(<au>)
UNIN
CSNIN1_
CDNIN1_
CSNIN2_
CDNIN2_
@CTNIN_
CUNIN1_
CUNIN2_
UNPACK [AL=Ri, DS:DI=<T>]
SI ← st
NIN
SP=SP-LBENV
GOON
ARG SPEC
ARG NIEØ
PAK BEZZ
SZTI
PAKUJ
CX=d=31-e
DF=1
SF=1
ZF=1
CAL PRD
PRD
DEN ZAM
TRAP [CL=D,bit]
TRAP [CEI,WRL]
ORMRD CL,BX,DI
TRAPP
WYN NIN
BLI1
BLI2
CZY 32K
32K
NEG DX
AX=#20
Int. indefinite

Wersja 2

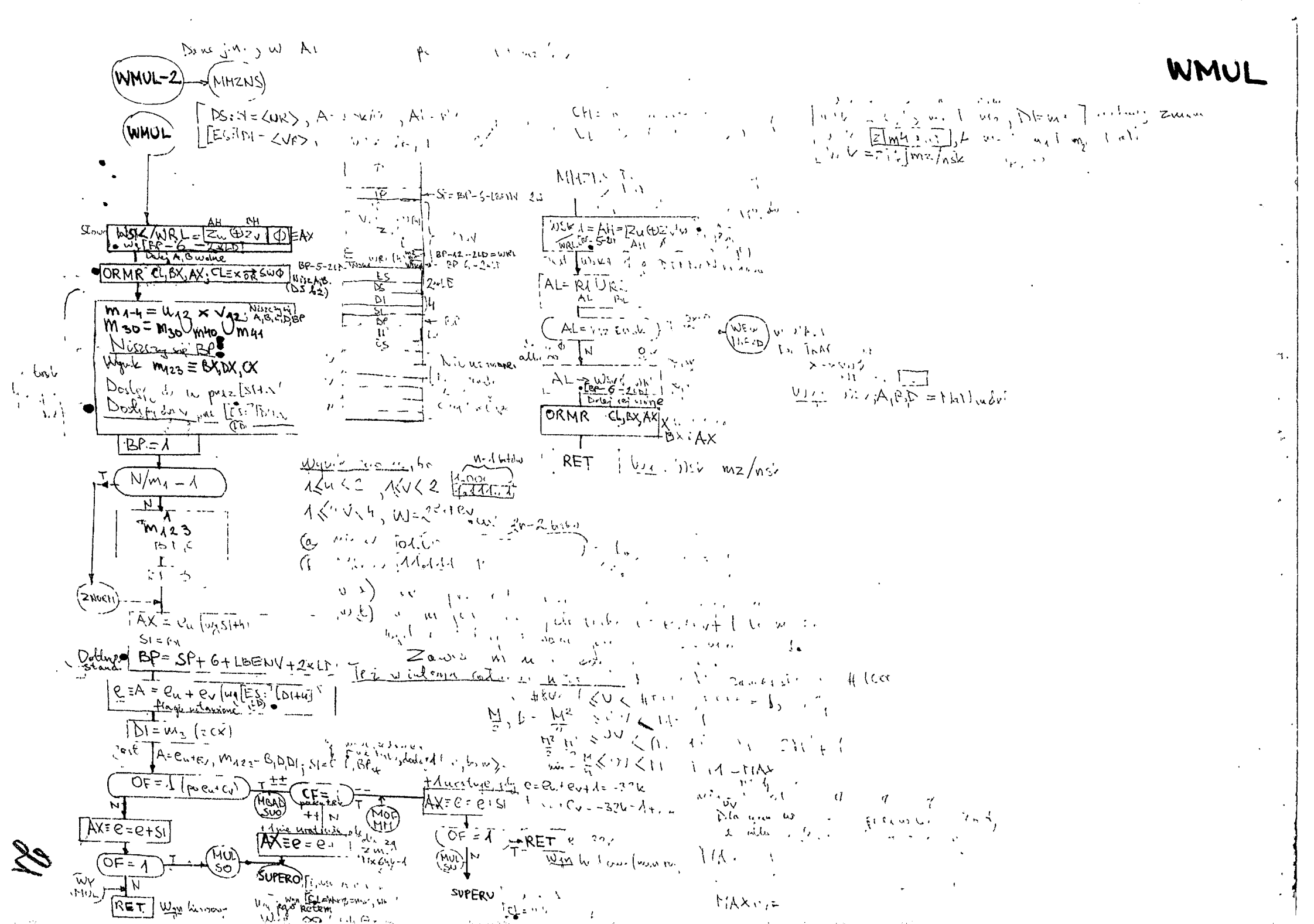

CSUNPU
CDUNPU
UNPACK
@UNPACK [AL=Ri=0/2, DS:DI=<T>]
DOUB
DEN
NORB2
RET
UNPACK
BBP

DENNAN
USTNAN
USTIND
SZTPR

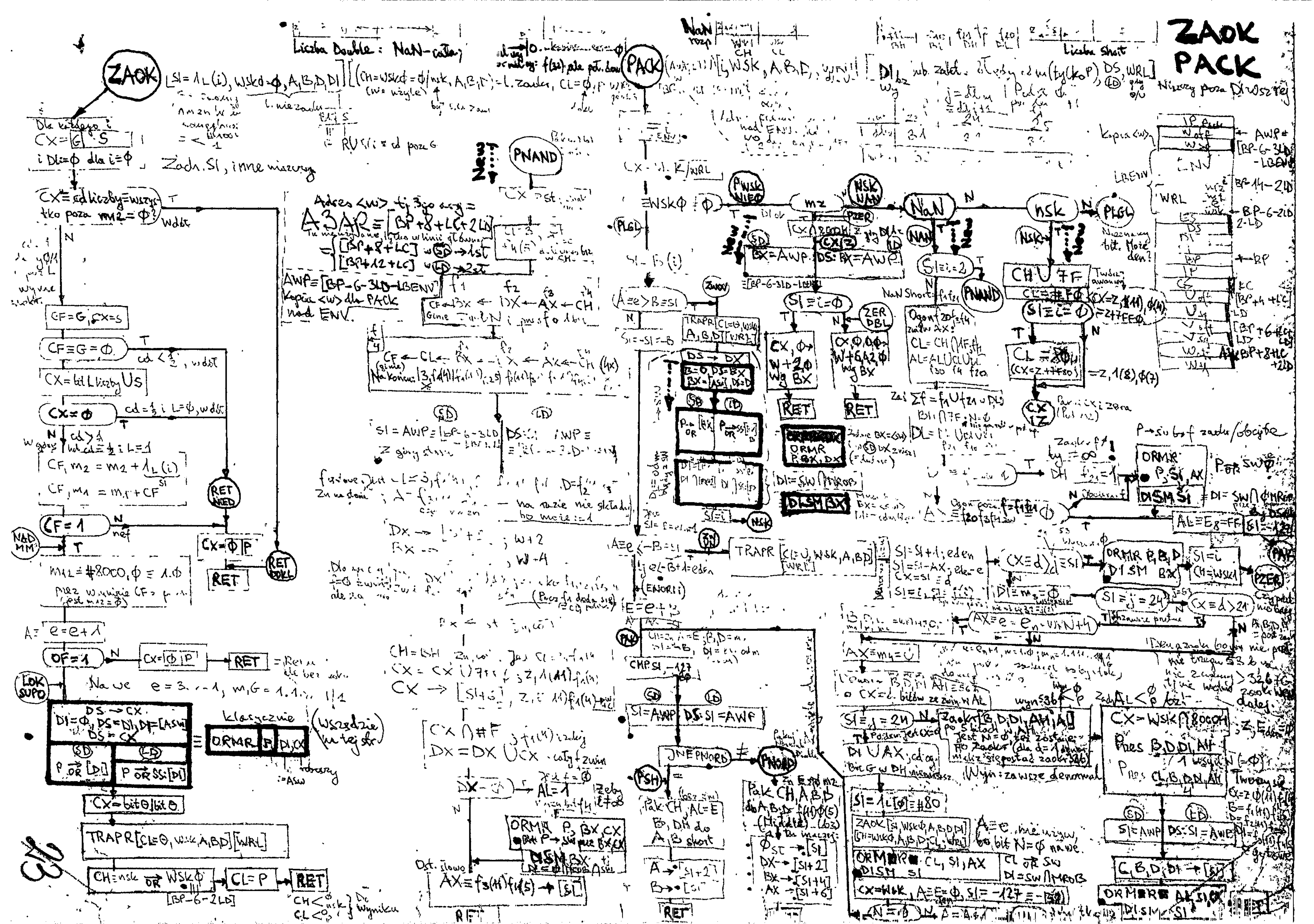



WADD
WADD-2
We Φ/∞: W R1,2 mx 1 bit mz/nsk i co najmniej w 1 rej.
[DS:SI = <UR>, A=WSK/UR, AL≡R1] [ES:]DI = <VR>, B=WSK/VR, BL≡R2] > bez ... w, CH≡Rz, CL≡X, ENV
[WSK, A≡e, ... m1, D≡m2, DI≡m3 — liczbowy
WSK=z|m4|NaN, A=m3; B=m1, D=m2 — NaN
WSK — Φ/∞
Ta część całkow. oddz. od lewej
Tu zul = zu, zvl = zv ⊕ Rz tylko w rejestrach
Znaki oryg tylko w USTNAN w DENNAN
BX→DX
DS nie wolno zniszczyć
Tylko DX wolny. Chowa BX
BH≡BH⊕Rz
AMZ NS
AL≡R1=R2≡BL
R1≡AL=mz
WEW NIED
CH=CL≡X
Chowaj
TRAP [CL=1, WRL=Φ] po DENNAN
Macro
ORMR CH,BX
OR CH do SWΦ przez BX:=ASW ∩ AX
USTIND [C=WSK, A,B,D]
wy jego RETem
DS=AX
BX=Φ, DS=BX, BX=DS:[ASW]
ORMR CH,SI,DI
ZW = zul ∩ zvl
AX→WSK
RET
AL=mz/nsk
AH=zw
BL≡R2=mz
AL≡R1=mz
A=B≡zvl|wskΦ bez den
SI≡DI
ES→DI→DS
AX→WSK
Wg DS:SI A=e; B,D=m12; DI≡m3=Φ
RET
Wyn. liczbowy: WSK, A,B,D,DI = U lub V R
ABA DAL
BL≡R2=Φ
A=B≡zvl|nsk
Usk, Voo
Voo, Vsk   Zw=Zu
BX≡DX
BH=BH⊕CH
CX=eu≡[SI+6]
DX=ev≡[ES:][DI+6]
(CX≡eu) ≥ ev≡DX
CX≡eu ↔ ev≡DX
AX≡zu|Φ ↔ zv|Φ≡BX
SI≡<UR> ↔ <VR>≡DI
EUWR
WSK=zul|Φ≡AX
ZVL=BH
e≡eu≡CX→st
CX≡d=eu−ev≡C−D
B,D≡m12=[ES:][DI+2, DI]≡V12
d=Φ
DEGΦ
PRD [C=d16; B,D=V12][B,D,DI=V123≡Vpz]
Nisz. A,C,DI
DI=Φ
AH≡zul=ZVL
AX≡e←st
CF, m2 = u2 − m2
CF, m1 = u1 − m1, CF
CF=1
AH=AH≡zu, AX→WSK
NEG m12
BX≡EX≡m1
e≡AX←st
CX=CX∪DX=Φ
CF,m2=Φ−m2 (NEG)
CF,m1=Φ−m1−CF (SBB)
CF,B,D≡m12=m12+u12; DI=Φ ważne
bo dane znormalizowane
NAD DOD
DI≡m3
SRCF [CF,A,B,D,DI,i] [A,B,D,DI,WRL,CH]
OF po INC AX≡e
S.O. możliwy
SUPERO
i tam RET z WADD
VPRZ ES
AL=WSK1
AL=zul=ZVL
DOD
Koncepcyjnie Vpz
m1m2m3=u1u2Φ − m1m2m3
AX≡e←st
bit N/m1=1
e≡AX−1
S.U. wykl.: e=eu>ev≥−32k
m1m2m3,Φ
N=1
NORM2EF
[WSKΦ=Φ, A; (B,D)≠Φ, N=Φ; DI=Φ]
[WSKΦ=Φ/mz; A,B,D,DI; CL=Φ/mz; WRL gdy S.U.]
S.U. możliwy, bo może być e+1, a norm nawet o 31 b
i tam RET z WADD
RET DOD
d=1 → po przes o 1 cały wynik w m12, ale może mieć do 31 zer z produ. DI=Φ ok dla S.U. w NORM2EF
d≥2 → po przes o 1 liczba znormalizowana
Więc zawsze przy wołaniu NORM2EF DI=Φ
„Shift Right with CF"
Sekwencja w wyjściu z WADD z lewej
SRCF [i,CF,A,B,D,DI] [A,B,D,DI; WRL; CH]
MOV CX,Φ
RCR m1,1 → CF
RCR m2,1
RCR m3,1
CF OR m3
ADC CX,Φ
OR DI,CX
AX≡e=e+1
OF=1
ADD SO
SUPERO [i,WSK,A,B,D,DI] [CL, WRL]
i tam RET
Wynik (jeśli O zamaskowany) WSKΦ=CL=nsk
RET w NORM2EF
SU możliwy

WMUL


WMUL-2
MNZNS
WMUL
ORMR CL,BX,AX; CL×DR SWØ
BP:=1
N/m1 − 1
m123
ZNUCH
AX = eu
SI = ev
BP = SP + 6 + LBENV + 2×LP
e = A = eu + ev
DI = m3 (=CX)
A = eu+ev, m123 - B,D,DI; SI = C
OF = 1 (eu+cv)
CF =
AX = e = e + SI
OF = 1
MUL SO
SUPERO
RET
AX = e = e1 SI
OF = 1
RET
MUL SO
SUPERU
AL = RL UR
ORMR CL,BX,AX
BX:AX
RET

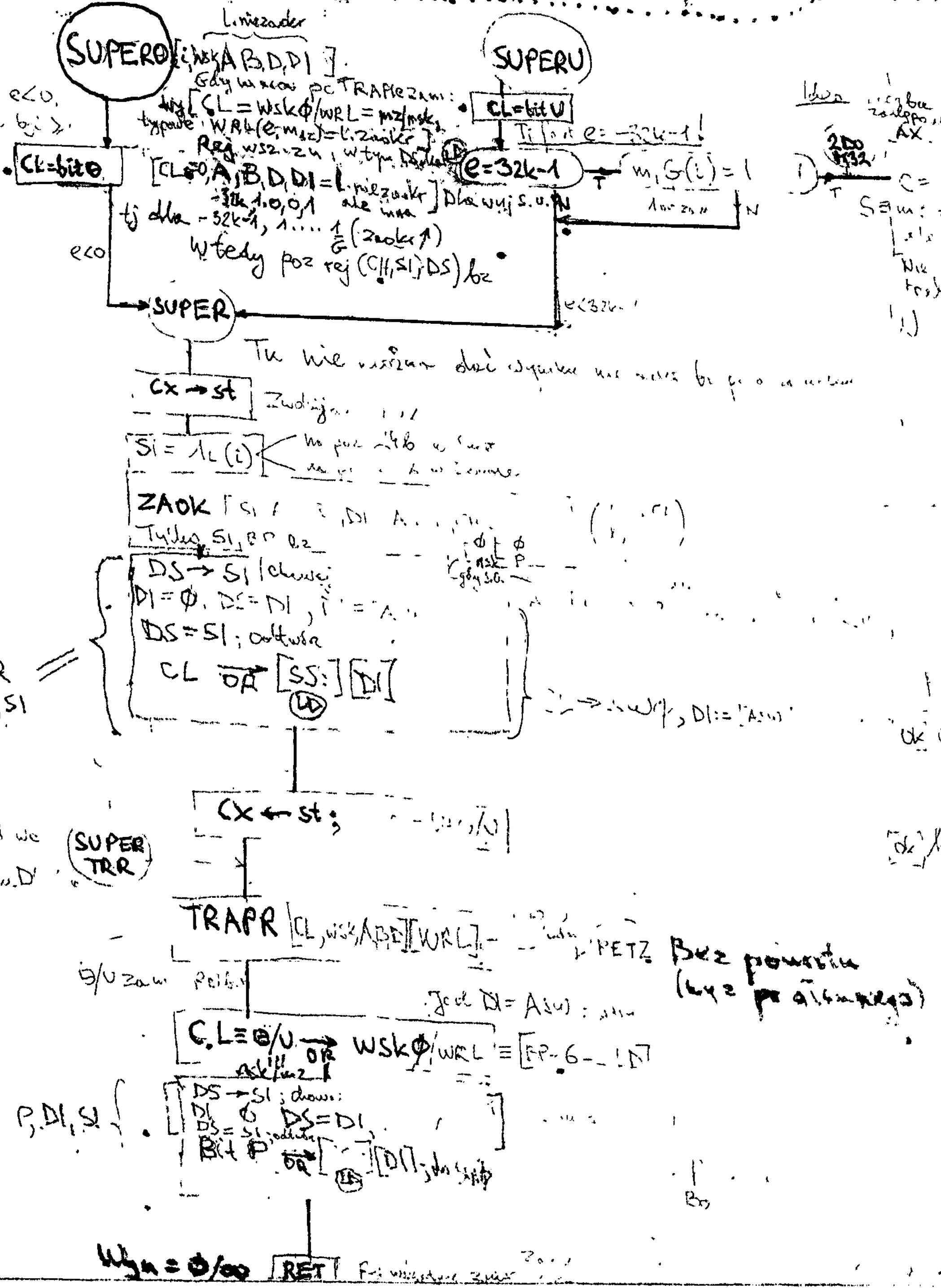

WDIV
WDIV-2
DMZ NS
WSK1/WRL = Zu ⊕ Zv
AL AH BH
WEW NIEZD
TRAP
USTIND
RET
XDO
ORMR CL, BX, AX
Dane do części
[DS:SI, [ES:]DI, A, B, CL, ENV]
WSK = A ⊕ B
[WSK WRL, A, B, D, DI]
ALE R1 = Φ
DBAD R2
ZERO DIV
CL = CH
WSK Φ = nsk
TRAP
W WRL pamiętamy tylko wsk, reszta na stosie
Stos na końcu
ES = DS
Deltae
SI = UR
DI = VR
URL
VRL/WRN
ZVL
ENV
ES
DS
DI
SI
BP
IP
CS
Nie używane
Krok D3P-D4
Oblicz B1S/q
RESZ
B, C = V12
D, A = RII, stos jw
BP = 0
2RLTV
2RGTV
2REQV
UGEV
FOR U1V2
UL
D3
D3P
Oblicz
ES = SI = JD
AX = uj+1 → st
SP = SP + 2
MGOT
ZMN QH
CF = 1
D4
Oblicz: (Φ) RI, RII = (uj) uj+1 uj+2 − q̂·v1v2
Stara reszta q̂ podniesiona na poziom b i powiększona o uj+2
ma starczyć na przyrost od iloczynu q̂·v2. Porównuje się liczbę ze słownik
DBAD SUO
OF = 1
OVFM
RET
Główny
DIV SO
DIV SU
SUPERO
SUPERU
RES2
AX = DI = RII = uj+1 nowe
DI = Φ = uj+2 nowe
SI = ES = JD
JD = SI = SI + 1
JD = SI = Φ
Jest: B, C = V12, A = RII = uj+1 nowe
DI = Φ = uj+2 nowe, BP = qj ost i na szcz. 5.



ZAOK
PACK
ZAOK
PACK
PNAND
Liczba Double : NaN-całą
Liczba Short
Dla każdego i
CX=[G][S]
i DI=Φ dla i=Φ
Zach. SI, inne niszczy
CX=cd liczby=wszystko poza m2=Φ?
CF=G, CX=S
CF=G=Φ
CX=bit L liczby U S
CX=Φ
CF, m2 = m2 + 1L(i)
CF, m1 = m1 + CF
RET NIED
CF=1
NAD MM
CX=Φ|P
RET
RET DOKL
m12 = #8000, Φ = 1.Φ
e=e+1
OF=1
CX=Φ|P
RET
LOK SUPO
Na we e=3, -1, m, G = 1.1..
DS → CX
klasycznie
ORMR P DI,CX
P OR [DI]
P OR SS:[DI]
CX=bitΘ|bitΘ
TRAPR [CL=Θ, WSK, A,B,D] [WRL]
CH=nsk OR WSKΦ
CL=P
RET
[BP-6-2LD]
Adres <w> tj. 3go arg =
A3AR=[BP+8+LC+2LD]
=[BP+8+LC] w SD → 1st
[BP+12+LC] w LD → 2st
AWP=[BP-6-3LD-LBENV]
Kopia <w> dla PACK nad ENV.
SI = AWP = [BP-6-3LD]
DS:SI AWP
Na razie nie składa, bo może = 1
CH=BH
CX → [SI+6]
CX ∩ #F
DX = DX ∪ CX
AL=1
ORMR P, BX, CX
DISM BX
AX=f3(4)f4(5) → [SI]
RET
Ost. słowo
WSKΦ = Φ
Pwsk NIEΦ
m2
NaN
nsk
PLGL
NSK NAN
PZER
BX=AWP. DS:BX=AWP
SI=i=2
PNAND
CH U 7F
CL=#FΦ
SI=i=Φ
SI=i=Φ
ZER DBL
ZWOV
A=e>B=SI
TRAPR [CL=Θ, WSK, A,B,D] [WRL]
DS → DX
CX, Φ→ W+2Φ wg BX
CX,Φ,Φ,Φ→ W+6,4,2,Φ wg BX
RET
RET
ORMR P, BX, DX
DISM BX
NSK
Kopia <w>
AWP
BP-6-3LD-LBENV
BP-14-2LD
BP-6-2LD
BP
LC
ORMR P, SI, AX
DISM SI
AL=E8=FF
TRAPR [CL=Θ, WSK, A,B,D] [WRL]
SI=SI+1; eden
(CX=d)
ORMR P,B,D
DISM BX
SI=i
PZER
SI=i=24
(CX=d>21)
AX=e-en-WYN+4
AX=m4=Φ
E=e+1
CMP SI, -127
SI=AWP DS:SI=AWP
JNE PNORD
PNORD
PSH
Pak CH, AL=E
Pak CH,A,B,D
A → [SI+2]
B → [SI]
RET
SI=i=24
ZAOK [B,D,DI,AH,A]
SI=1L[Φ]=#80
ZAOK [SI, WSKΦ, A,B,D,DI]
ORMR CL, SI, AX
DISM SI
CX=WSK ∩ 8000H
SI=AWP DS:SI=AWP
C,B,D,DI → [SI]
ORMR AL,SI,CX
DISM
Wyn: zawsze denormal

SUPERO
SUPERU

Cel: Zaokrąglij RN, wdaj TRAPR[%](e



C = 32k, m = 1.Ø

CL = ... → RET Wyn

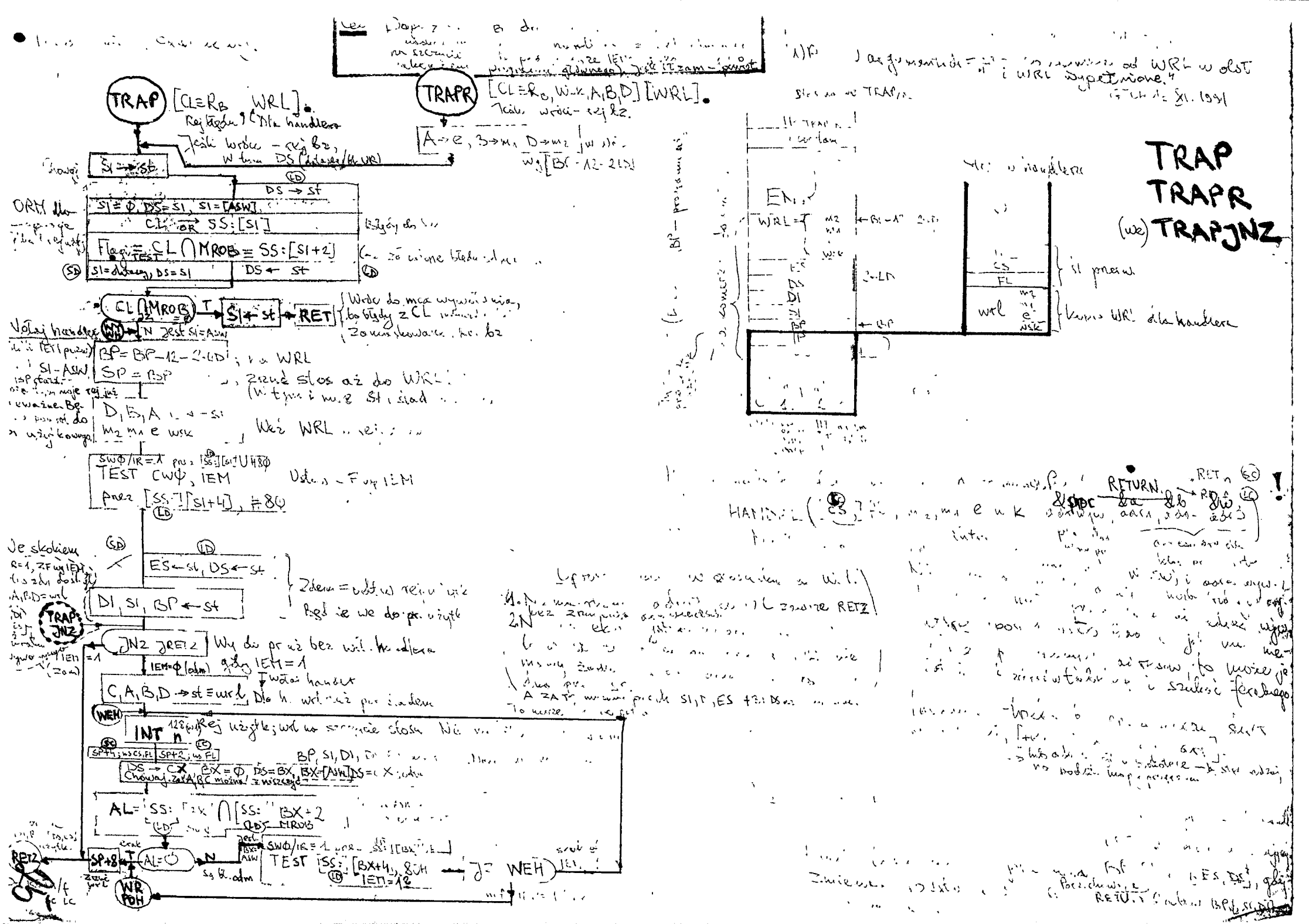
Powinien: dokładny wynik jest
-32k-1, 1.1111... $\frac{1}{G(i)}$
do -32k, 1.000...

do minreprezentacji

Vodm - wyjątkowe

do min repr w dół

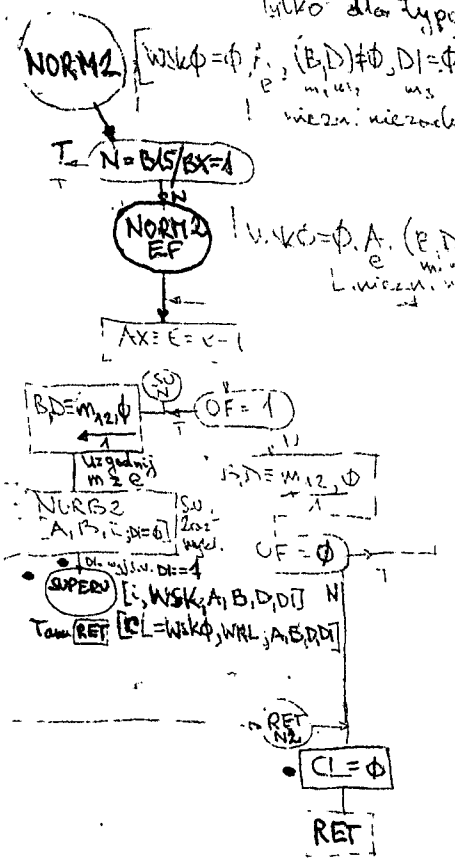
C.L = WRL

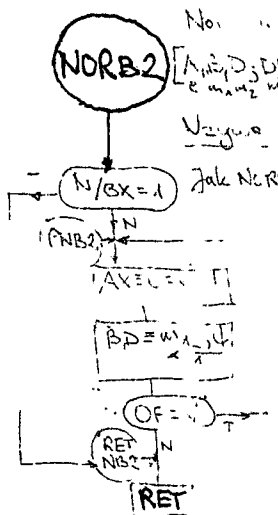

TRAP
TRAPR
(use) TRAPJNZ
RETURN
INT n
JNZ JRETZ
RETZ
WR POH

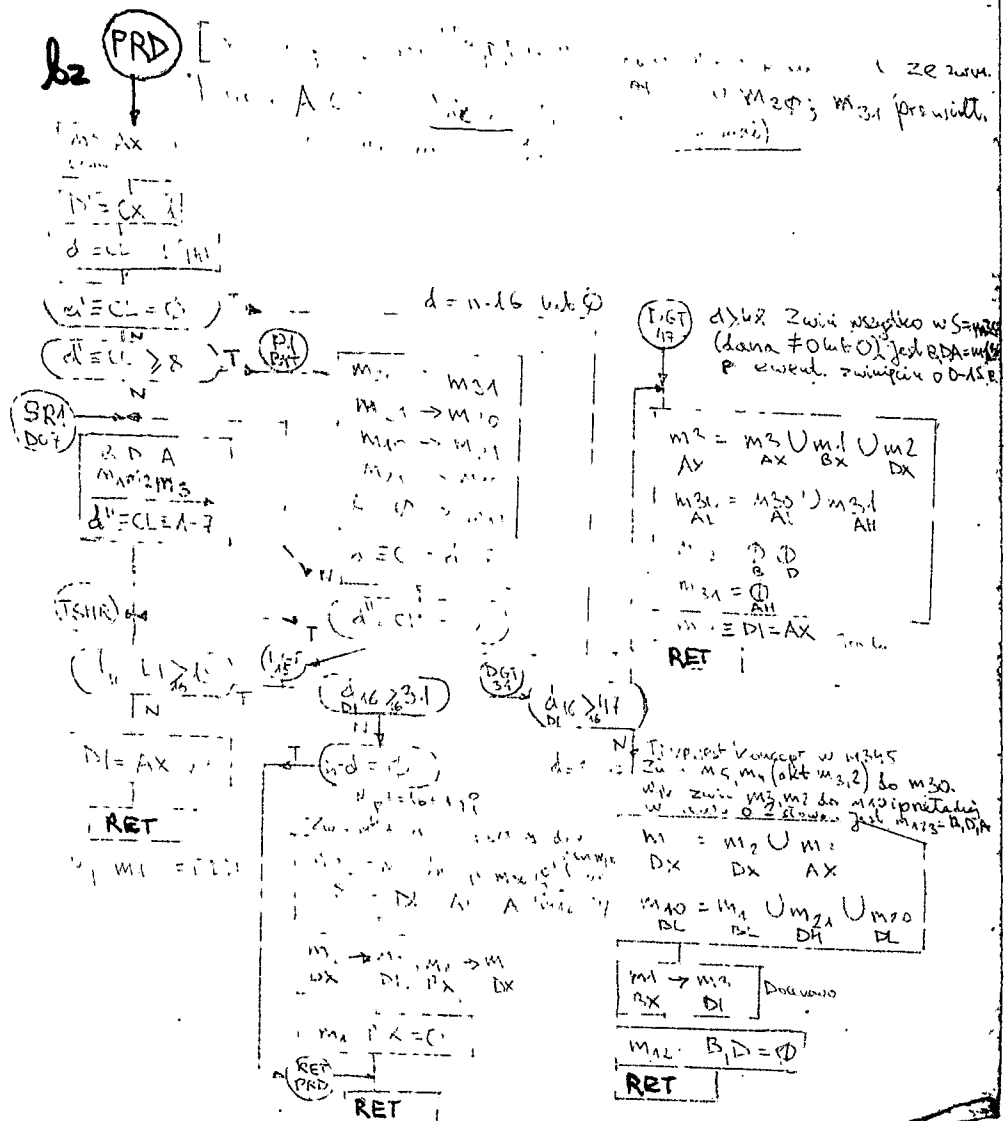
NORM2EF

NORB2

PRD







Organizacyjne