Programowanie niskopoziomowe

Materiały uzupełniające

Labolatoria 8

„Koprocesor”

Cześć materiałów autorstwa prof. A. Timofiejewa

3.2.2. Koprocesor

Z punktu widzenia programista rejestry koprocesora dodawane są do rejestrów procesora. Koprocesor ma osiem 80-bitowych rejestrów ST(0) – ST(7) do rozmieszczenia danych zmiennoprzecinkowych oraz trzy 16 bitowych rejestrów Control Word, Status Word, Tag Word do sterowania koprocesorem.

Każdy z rejestrów ST(0) – ST(7) jest dostępny przez indeks. Rejestry są organizowane też w stos sprzętowy, wierzch którego jest zaznaczany jako rejestr ST(0).

Struktura rejestru Control Word:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
x	x	x	x	RC		PC		x	x	PM	UM	OM	ZM	DM	IM

W tym:

- RC (Rounding Control) – sterowanie zaokrągleniem (00 – zaokrąglenie do najbliższego lub parzystego, 01 – zaokrąglenie do mniejszego (w kierunku minus nieskończoność), 10 – zaokrąglenie do większego (w kierunku plus nieskończoność) , 11 – odrzucenie (w kierunku zera),
- PC (Precision Control) – sterowanie dokładnością (00 – 24 bity (Single Precision), 01 – rezerwa, 10 – 53 bity (Double Precision), 11 – 64 bity (Extended Precision)),
- PM (Precision Mask) – bit zezwolenia na wyjątek z powodu dokładności,
- UM (Stack Underflow Mask) – bit zezwolenia na wyjątek z powodu pustego stosu rejestrów,
- OM (Stack Overflow Mask) – bit zezwolenia na wyjątek z powodu przepełnienia stosu rejestrów,
- ZM (Zero Divisor Mask) – bit zezwolenia na wyjątek z powodu dzielnika równego zero,
- DM (Denormalized Operand Mask) – bit zezwolenia na wyjątek z powodu denormalizacji operandu,
- IM (Invalid Operation Mask) – bit zezwolenia na wyjątek z powodu niepoprawnej operacji.

Struktura rejestru Status Word:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	C3	TOP			C2	C1	C0	ES	SF	PE	UE	OE	ZE	DE	IE

W tym:

- B (Busy Bit) – bit zajętości operacją (jest ustawiany też, jeśli bit ES=1),
- C3, C2, C1, C0 (Condition Bits) – znaczniki operacji,
- TOP (Top Register) – indeks rejestru na wierzchu stosu,
- ES (Summary Error Bit) – bit wyjątku sumarycznego,
- SF (Stack Flagt) – znacznik informujący razem z C1 o błędzie stosu (C1=1 - przepełnienia stosu, C1=0– stos pusty),
- PE (Precision Exception Bit) – bit wyjątku z powodu dokładności,
- UE (Stack Underflow Exception Bit) – bit wyjątku z powodu pustego stosu rejestrów,
- OE (Stack Overflow Exception Bit) – bit wyjątku z powodu przepełnienia stosu rejestrów,
- ZE (Zero Divisor Exception Bit) – bit wyjątku z powodu dzielnika równego zero,
- DE (Denormalized Operand Exception Bit) – bit wyjątku z powodu denormalizacji operandu,
- IE (Invalid Operation Exception Bit) – bit wyjątku z powodu niepoprawnej operacji.

Struktura rejestru Tag Word:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Tag(7)		Tag(6)		Tag(5)		Tag(4)		Tag(3)		Tag(2)		Tag(1)		Tag(0)	

gdzie Tag(i) – para bitów informująca o stanie i-go rejestru stosu:

00 – poprawnie, 01 – w rejestrze zero, 10 – w rejestrze wartość QNaN, SNaN, Infinity, Denormal lub Unsupported Formats, 11 – rejestr jest pusty.

Do rozpatrywanych rejestrów należy dodać dwa 48-bitowych rejestry Instruction Pointer i Data Pointer, które przechowują adres rozkazu i operandu w czasie wykonania operacji przez koprocesor. W 32-bitowej architekturze Intel rejestry są rozmieszczone w procesorze bazowym, a nie w koprocesorze.

Procesor i koprocesor współdziałają przez wspólną pamięć. Nie można przesyłać dane bezpośrednio między rejestrami procesora i koprocesora. Przy ładowaniu danej do rejestru koprocesora procesor wystawia adres komórki pamięci na szynie adresu, a zawartość komórki przyjmuje z szyny danych koprocesor. Jeżeli dana jest przesyłana do pamięci, to procesor też wystawia adres komórki pamięci, a koprocesor wystawia na szynie danych zawartość komórki.

6.11. Operacje jednostki zmiennoprzecinkowej

Nazwy instrukcji jednostki zmiennoprzecinkowej (koprocesora) rozpoczynają się od litery F. Litera I na drugim miejscu informuje o operacji nad liczbą całkowitą, a litera B - o operacji nad liczbą w upakowanym kodzie BCD. Litera P na końcu nazwy oznacza zdjęcie argumentu operacji ze stosu.

Koprocesor przechowuje dane na stosie z ośmiu 80-bitowych rejestrów st(0), st(1),...,st(7), gdzie st(0) jest wierzchołkiem stosu. W instrukcjach można pisać krótko „st” zamiast „st(0)”. W roli rejestru st(0) może być każdy z rejestrów.

6.11.1. Formaty danych

Koprocesor operuje na danych z formatami przedstawionymi w tab. 6.22.

Rozkazy procesora są wykonywane nie na wszystkich rodzajach operandów. Dlatego w tabelach z instrukcjami są wskazane rodzaje operandów:

m32real - operand Real, Single Precision,
m64real - operand Real, Double Precision,
m80real - operand Real, Extended Precision,
m16int - operand Word Integer,
m32int - operand Short Integer,

m64int - operand Long Integer,
m80bcd - operand Packed BCD,
m2byte – operand 2-bajtowy,
m14/28byte – operand – tablica na 14 lub 28 bajtów,
m94/108byte - operand – tablica na 94 lub 108 bajtów

Tabela 6.22

Formaty danych zmiennoprzecinkowych

Nazwa	Typ	Dokładność	Kodowanie
Word Integer	**DW**	16 bitów	16-bitowy kod dopełniający
Short Integer	**DD**	32 bitów	32-bitowy kod dopełniający
Long Integer	**DQ**	64 bitów	64-bitowy kod dopełniający
Packed BCD	**DT**	18 cyfr dziesiętnych	10 bajtów: bajt znakowy i 9 bajtów z upakowanym kodem BCD
Single Precision	**DD**	24 bitów	bit 31 znakowy, bity 30..23 wykładnik (potęga – baza równa 7Fh), bity 22..0 mantysa z „niewidoczną 1”
Double Precision	**DQ**	53 bitów	bit 63 znakowy, bity 62...52 wykładnik (potęga – baza równa 3FFh), bity 51...0 mantysa z „niewidoczną 1”
Extended Precision	**DT**	64 bitów	bit 79 znakowy, bity 78...64 wykładnik (potęga – baza równa 3FFFh), bity 63...0 mantysa

Jeśli dana nie była wprowadzona lub wynik operacji znajduje się poza zakresem „normalnych” wartości, procesor inicjuje wyjątek.

6.11.2. Przesyłanie danych

Koprocesor może przesyłać dane warunkowo i bezwarunkowo, z przesunięciem stosu rejestrów i bez przesunięcia.

Bezwarunkowe przesyłanie danych bez przesunięcia stosu

Do bezwarunkowego przesyłania danych bez przesunięcia stosu służą instrukcje przedstawione w tab. 6.23.

Bezwarunkowe przesyłanie danych z przesunięciem stosu

Do bezwarunkowego przesyłania danych z przesunięciem stosu służą instrukcje przedstawione w tab. 6.24.

Tabela 6.23.

Przeznaczenie	Instrukcja

Ładowanie Integer na stos rejestrowy	`fild m16int`
	`fild m32int`
	`fild m64int`

Ładowanie Real na stos rejestrowy	`fld m32real`
	`fld m64real`
	`fld m80real`
	`fld ST(i)`

Ładowanie BCD na stos rejestrowy	`fbld m80bcd`

Kopiowanie ST(0) do operandu	`fst m32real`
	`fst m64real`
	`fst ST(i)`

Kopiowanie ST(0) do operandu	`fist m16int` `fist m32int`
Zamiana miejscami ST(0) i ST(1) Zamiana miejscami ST(0) i ST(i)	`fxch` `fxch ST(i)`

Tabela 6.24.

Instrukcje bezwarunkowego przesyłania danych z przesunięciem stosu

Przeznaczenie	Instrukcja

Przesyłanie Integer z ST(0) do operandu i zdejmowanie ST(0) ze stosu	`fistp m16int`
	`fistp m32int`
	`fistp m64int`

Przesyłanie Real z ST(0) do operandu i zdejmowanie ST(0) ze stosu	`fstp m32real`
	`fstp m64real`
	`fstp m80real`
	`fstp ST(i)`

Przesyłanie BCD z ST(0) do operandu i zdejmowanie ST(0) ze stosu	`fbstp m80bcd`

Warunkowe przesyłanie danych

Do warunkowego przesyłania danych służą instrukcje przedstawione w tab. 6.25.

Instrukcje warunkowego przesyłania danych FCMOVxx korzystają ze znaczników rejestru EFLAGS. Znaczniki C0, C2, C3 koprocesora można przenieść do rejestru EFLAGS następującymi instrukcjami: `fstsw AX ;przeniesienie znaczników C0,C2,C3 do rejestru AX` sahf ;przeniesienie rejestru AH do rejestru F (części EFLAGS)

Tabela 6.25.

Instrukcje warunkowego przesyłania danych

Przeznaczenie	Instrukcja
Przesyłanie, jeśli ZF=1	`fcmove ST(0),ST(i)`
Przesyłanie, jeśli ZF=0	`fcmovne ST(0),ST(i)`
Przesyłanie, jeśli CF=1	`fcmovb ST(0),ST(i)`
Przesyłanie, jeśli CF=0	`fcmovnb ST(0),ST(i)`
Przesyłanie, jeśli CF=1 lub ZF=1	`fcmovbe ST(0),ST(i)`
Przesyłanie, jeśli CF=0 i ZF=0	`fcmovnbe ST(0),ST(i)`
Przesyłanie, jeśli PF=1	`fcmovu ST(0),ST(i)`
Przesyłanie, jeśli PF=0	`fcmovnu ST(0),ST(i)`

6.11.3. Ładowanie stałych

W programach z operacjami na liczbach zmiennoprzecinkowych często są używane stałe „0.0", „1.0" i „pi".

Istnieją instrukcje (tab. 6.26) do ładowania tych wartości.

Jeszcze jedna grupa stałych jest związana z operacjami potęgowania i logarytmowania dla liczb zmiennoprzecinkowych (tab. 6.26).

Tabela 6.26

Instrukcje przesyłania stałych

Przeznaczenie	Instrukcja
Ładowanie 0.0 na stos rejestrowy	`fldz`
Ładowanie 1.0 na stos rejestrowy	`fld1`
Ładowanie „pi" na stos rejestrowy	`fldpi`
Ładowanie „log2(10)" na stos rejestrowy	`fldl2t`
Ładowanie „log2(e)" na stos rejestrowy	`fldl2e`
Ładowanie „log10(2)" na stos rejestrowy	`fldlg2`
Ładowanie „loge(2)" na stos rejestrowy	`fldln2`

6.11.4. Operacje arytmetyczne

Zawsze jeden z operandów operacji arytmetycznej jest wierzchołkiem stosu, tj. rejestrem st(0). Jeżeli w instrukcji jest napisany tylko jeden operand, to ten operand jest interpretowany jako drugi, a pierwszym operandem służy niejawny operand - wierzchołek stosu st(0). Wynik zapisuje się do pierwszego operandu. W przypadku instrukcji z dwoma operandami, jeden z nich musi być wierzchołkiem stosu, tj. rejestrem st(0).

Operacje arytmetyczne można podzielić na bazowe, do których należą sumowanie, odejmowanie, mnożenie i dzielenie, i operacje złożone.

Instrukcje arytmetyczne bazowe mogą być wykonywane z przesuwaniem lub bez przesuwania stosu rejestrowego.

Instrukcje arytmetyczne bazowe bez przesuwania stosu

Instrukcje arytmetyczne bez przesuwania stosu są przedstawione w tab. 6.27.

Instrukcje arytmetyczne bazowe z przesuwaniem stosu

Instrukcje arytmetyczne bazowe z przesuwaniem stosu są przedstawione w tab. 6.28.

Instrukcje arytmetyczne złożone

Instrukcje arytmetyczne złożone są przedstawione w tab. 6.29.

6.11.5. Operacje trygonometryczne

Jednostka zmiennoprzecinkowa wykonuje operacji trygonometryczne: sinus, cosinus, tangens i arkus tangens (tab. 6.30). Argument operacji jest mierzony w radianach.

6.11.6. Operacje porównania

Operacje porównania jednostki zmiennoprzecinkowej powodują ustawienie znaczników C0, C2, C3 koprocesora. Część instrukcji kończy się ustawieniem znaczników ZF, PF i CF procesora.

Do porównania liczb służą instrukcje przedstawione w tab. 6.31.

Tabela 6.27.

Instrukcje arytmetyczne bazowe bez przesuwania stosu

Przeznaczenie	Instrukcja

Dodawanie Real do ST(0)
Dodawanie Real do ST(0)
Dodawanie Real do ST(i)
	`fadd m32real`
	`fadd m64real`
	`fadd ST(0),ST(i)`
	`fadd ST(i),ST(0)`

Dodawanie Integer do ST(0)	`fiadd m16int` `fiadd m32int`

Odejmowanie Real od ST(0)
Odejmowanie Real od ST(0)
Odejmowanie Real od ST(i)
	`fsub m32real`
	`fsub m64real`
	`fsub ST(0),ST(i)`
	`fsub ST(i),ST(0)`

Odejmowanie Integer od ST(0)	`fisub m16int` `fisub m32int`

Odejmowanie Real ST(0), wynik w ST(0)
Odejmowanie Real ST(0), wynik w ST(0)
Odejmowanie Real ST(i), wynik w ST(i)
	`fsubr m32real`
	`fsubr m64real`
	`fsubr ST(0),ST(i)`
	`fsubr ST(i),ST(0)`

Odejmowanie Integer ST(0), wynik w ST(0)	`fisubr m16int` `fisubr m32int`

Mnożenie Real ST(0), wynik w ST(0)
Mnożenie Real ST(0), wynik w ST(0)
Mnożenie Real ST(i), wynik w ST(i)
	`fmul m32real`
	`fmul m64real`
	`fmul ST(0),ST(i)`
	`fmul ST(i),ST(0)`

Mnożenie Integer ST(0), wynik w ST(0) Mnożenie Integer ST(0), wynik w ST(0)	`fimul m16int` `fimul m32int`

Dzielenie Real ST(0), wynik w ST(0)
Dzielenie Real ST(0), wynik w ST(0)
Dzielenie Real ST(i), wynik w ST(i)
	`fdiv m32real`
	`fdiv m64real`
	`fdiv ST(0),ST(i)`
	`fdiv ST(i),ST(0)`

Dzielenie Integer ST(0), wynik w ST(0)	`fidiv m16int` `fidiv m32int`

Dzielenie Real na ST(0), wynik w ST(0)
Dzielenie Real na ST(0), wynik w ST(0)
Dzielenie Real na ST(i), wynik w ST(i)
	`fdivr m32real`
	`fdivr m64real`
	`fdivr ST(0),ST(i)`
	`fdivr ST(i),ST(0)`

Dzielenie Integer na ST(0), wynik w ST(0)	`fidivr m16int` `fidivr m32int`
Reszta ST(0)/ST(1), wynik w ST(0)	`fprem`
Reszta ST(0)/ST(1), wynik w ST(0) (standard IEEE)	`fprem1`

Tabela 6.28.

Instrukcje arytmetyczne bazowe z przesuwaniem stosu

Przeznaczenie	Instrukcja
Dodawanie ST(1) do ST(0) i przesuwanie stosu Dodawanie ST(i) do ST(0) i przesuwanie stosu	`faddp` `faddp ST(i),ST(0)`
Odejmowanie ST(1) od ST(0) i przesuwanie stosu Odejmowanie ST(i) od ST(0) i przesuwanie stosu	`fsubp` `fsubp ST(i),ST(0)`
Odejmowanie ST(1) od ST(0), wynik w ST(1), przesuwanie stosu Odejmowanie ST(i) od ST(0), wynik w ST(i), przesuwanie stosu	`fsubrp` `fsubrp ST(i),ST(0)`
Mnożenie ST(1) na ST(0), wynik w ST(1), przesuwanie stosu Mnożenie ST(i) na ST(0), wynik w ST(i), przesuwanie stosu	`fmulp` `fmulp ST(i),ST(0)`
Dzielenie ST(1) na ST(0), wynik w ST(1), przesuwanie stosu Dzielenie ST(i) na ST(0), wynik w ST(i), przesuwanie stosu	`fdivp` `fdivp ST(i),ST(0)`
Dzielenie ST(0) na ST(1), wynik w ST(1), przesuwanie stosu Dzielenie ST(0) na ST(i), wynik w ST(i), przesuwanie stosu	`fdivrp` `fdivrp ST(i),ST(0)`

Tabela 6.29.

Instrukcje arytmetyczne złożone

Przeznaczenie	Instrukcja
Zamiana ST(0) na moduł ST(0)	`fabs`
Zmiana znaku ST(0) na przeciwny	`fchs`
Zaokrąglenie ST(0) do wartości całkowitej	`frndint`
Podniesienie dwójki do potęgi całkowitej z ST(1) i mnożenie na ST(0)	`fscale`
Pierwiastek kwadratowy ST(0)	`fsqrt`
Przekształcenie ST(0) do postaci m * (2^c), gdzie mantysa – ułamek m znajduje się w ST(0), a potęga c - liczbą całkowitą w ST(1)	`fxtract`
Podniesienie 2 do potęgi ST(0) z zakresu [-1,+1] i odejmowanie 1. Wynik w ST(0)	`f2xm1`
Obliczenie ST(1)*log2(ST(0)), zapisywanie wyniku do ST(1) i przesuwanie stosu rejestrów	`fyl2x`
Obliczenie ST(1)*log2(ST(0)+1.0), zapisywanie wyniku do ST(1) i przesuwanie stosu rejestrów	`fyl2xp1`

Tabela 6.30

Instrukcje trygonometryczne

Obliczenie sinusa od argumentu w ST(0). Sinus w ST(0)	fsin
Obliczenie cosinusa od argumentu w ST(0). Kosinus w ST(0)	fcos
Obliczenie sinusa i cosinusa od argumentu w ST(0). Sinus w ST(0), kosinus w ST(1)	fsincos
Obliczenie tangensa od argumentu w ST(0). Tangens jest zapisywany najpierw do ST(0), a później na stos rejestrów jest odkładana liczba 1.0	fptan
Obliczenie arkustangensa od argumentu w ST(1) i dzielenie na ST(0). Wynik jest umieszczany w ST(1) i jest przesuwany stos rejestrów	fpatan

Tabela 6.31

Instrukcje do porównania liczb

Przeznaczenie	Instrukcja

Porównanie ST(1) z ST, ustawienie C0, C2 i C3
Porównanie operandu z ST, ustawienie C0, C2 i C3
 	
Porównanie ST(i) z ST, ustawienie C0, C2 i C3
	`fcom`
	`fcom m32real`
	`fcom m64real`
	`fcom ST(i)`

Porównanie ST(1) z ST, ustawienie C0, C2 i C3, przesuwanie stosu
Porównanie operandu z ST, ustawienie C0, C2 i C3, przesuwanie stosu
Porównanie ST(i) z ST, ustawienie C0, C2 i C3, przesuwanie stosu
	`fcomp`
	
	`fcomp m32real`
	`fcomp m64real`
	`fcomp ST(i)`

Porównanie ST(1) z ST, ustawienie C0, C2 i C3, przesuwanie stosu dwa razy	`fcompp`

Porównanie ST(1) z ST bez reakcji na QNaN, ustawienie C0, C2 i C3
Porównanie ST(i) z ST bez reakcji na QNaN, ustawienie C0, C2 i C3
	`fucom`
	
	`fucom ST(i)`

Porównanie ST(1) z ST bez reakcji na QNaN, ustawienie C0, C2 i C3, przesuwanie stosu
Porównanie ST(i) z ST bez reakcji na QNaN, ustawienie C0, C2 i C3, przesuwanie stosu
	`fucomp`
	
	`fucomp ST(i)`

Porównanie ST(1) z ST bez reakcji na QNaN, ustawienie C0, C2 i C3, przesuwanie stosu dwa razy	`fucompp`
Porównanie operandu z ST, ustawienie C0, C2 i C3	`ficom m16int` `ficom m32int`
Porównanie operandu z ST, ustawienie C0, C2 i C3, przesuwanie stosu	`ficomp m16int` `ficomp m32int`
Porównanie ST(i) z ST, ustawienie ZF, PF i CF	`fcomi ST,ST(i)`
Porównanie ST(i) z ST, ustawienie ZF, PF i CF, przesuwanie stosu	`fcomip ST,ST(i)`
Porównanie ST(i) z ST bez reakcji na QNaN, ustawienie ZF, PF i CF, przesuwanie stosu	`fucomi ST,ST(i)`
Porównanie ST(i) z ST bez reakcji na QNaN, ustawienie ZF, PF i CF, przesuwanie stosu	`fucomip ST,ST(i)`
Porównanie ST z 0.0, ustawienie znaczników C0, C2 i C3	`ftst`
Klasyfikacja danej z ST z ustawieniem C0, C2 i C3	`fxam`

Instrukcje FCOMxx, FICOMx, FTST, FXAM, FUCOMxx zapisują wynik porównania do znaczników C3, C2, C0:

(C3,C2,C0)=000, jeśli st > op,

(C3,C2,C0)=001, jeśli st < op,
(C3,C2,C0)=101, jeśli st == op,
(C3,C2,C0)=111, jeśli nie można porównać.

Instrukcje FCOMIxx zapisują wynik porównania do znaczników ZF, PF, CF rejestru E FLAGS:
(ZF,PF,CF)=000, jeśli st > op,
(ZF,PF,CF)=001, jeśli st < op,
(ZF,PF,CF)=101, jeśli st == op,
(ZF,PF,CF)=111, jeśli nie można porównać.

6.11.7. Operacje sterujące stanem koprocesora

Do sterowania stanem koprocesora służą instrukcje przedstawione w tab. 6.32.

Tabela 6.32

Instrukcje do sterowania stanem koprocesora

Przeznaczenie	Instrukcja
Zwiększenie o 1 indeksu rejestru na wierzchu stosu rejestrów, przy czym wartość 7 jest zamieniana na 0	`fincstp`
Zmniejszenie o 1 indeksu rejestru na wierzchu stosu rejestrów, przy czym wartość 0 jest zamieniana na 7	`fdecstp`
Zaznaczenie i-go rejestru jako pustego	`ffree ST(i)`
Najpierw obsługiwanie wyjątków oczekujących. Następnie ustawienie w stan początkowy rejestrów: sterowania na 037Fh (zaokrąglenie do najbliższego, wyjątki są dozwolone, dokładność 64-bitowa), stanu na 0, danych na „puste", wskaźników instrukcji i danej na 0	`finit`
Ustawienia taki jak `finit`, tylko bez obsługiwania wyjątków oczekujących	`fninit`
Najpierw obsługiwanie wyjątków oczekujących. Następnie ustawienie w stan początkowy znaczników: wyjątków PE, UE, OE, ZE, DE, IE i ES na 0, błędu stosu SF na 0, zajętości B na 0	`fclex`
Ustawienia taki jak `fclex`, tylko bez obsługiwania wyjątków oczekujących	`fnclex`
Najpierw obsługiwanie wyjątków oczekujących. Następnie kopiowanie 16-bitowego słowa z rejestru sterowania	`fstcw m2byte`
Kopiowanie 16-bitowego słowa z rejestru sterowania bez obsługiwania wyjątków oczekujących	`fnstcw m2byte`
Załadowanie 16-bitowego słowa do rejestru sterowania	`fldcw m2byte`
Najpierw obsługiwanie wyjątków oczekujących. Następnie kopiowanie rejestrów sterowania jednostki do 14- lub 28-bajtowej tablicy (tryb Real - 14, tryb Protected - 28)	`fstenv m14/28byte`
Kopiowanie rejestrów sterowania jednostki do 14- lub 28-bajtowej tablicy bez obsługiwania wyjątków oczekujących	`fnstenv m14/28byte`
Załadowanie rejestrów sterowania jednostki z 14- lub 28-bajtowej tablicy (tryb Real - 14, tryb Protected - 28)	`fldenv m14/28byte`
Najpierw obsługiwanie wyjątków oczekujących. Następnie kopiowanie stanu jednostki do 94- lub 108-bajtowej tablicy (tryb Real - 94, tryb Protected - 108). Inicjacja jednostki	`fsave m94/108byte`
Kopiowanie stanu jednostki do 94- lub 108-bajtowej tablicy bez obsługiwania wyjątków oczekujących. Inicjacja jednostki	`fnsave m94/108byte`
Załadowanie stanu jednostki z 94- lub 108-bajtowej tablicy (tryb Real - 94, tryb Protected - 108)	`frstor m94/108byte`

Najpierw obsługiwanie wyjątków oczekujących, a następnie kopiowanie do pamięci 16-bitowego słowa z rejestru stanu.
Najpierw obsługiwanie wyjątków oczekujących, a następnie kopiowanie do rejestru AX 16-bitowego słowa z rejestru stanu
	`fstsw m2byte`
	
	`fstsw AX`

Kopiowanie 16-bitowego słowa z rejestru stanu do pamięci bez obsługiwania wyjątków oczekujących. Kopiowanie 16-bitowego słowa z rejestru stanu do AX bez obsługiwania wyjątków oczekujących	`fnstsw m2byte` `fnstsw AX`

Zmniejszenie o 1 wskaźnika wierzchołka stosu	FDECSTR
Zwiększenie o 1 wskaźnika wierzchołka stosu	FINCSTR
Czekanie na obsługę wyjątków oczekujących	`wait/fwait`
Brak działań (przejście do następnej instrukcji)	`fnop`

Instrukcja FINIT (FNINIT) ustawia rejestr Control Word sterowania koprocesora na wartość 037Fh, co oznacza zaokrąglanie do najbliższej liczby, zezwolenie reakcji na wszystkie błędy i 64-bitowa precyzja, a także zeruje rejestr Status Word stanu koprocesora, rejestr wskaźnika rozkazu oraz rejestr wskaźnika operandu, ustawia w rejestrze Tag Word znaczników stanu rejestrów danych osiem 2-bitowych pola w stan „pusty” (11b)

6.11.8. Przykład

Obliczmy $y = a^b$, tj. potęgę b liczby a. Ponieważ koprocesor nie posiada instrukcji do podobnego obliczenia, zastosujemy wzór:

$a^b = 2^x$,

gdzie $x = \log_2 a^b = b*\log_2 a$.

Do obliczenia `x` pasuje instrukcja `FYL2X` (obliczenie ST(1)*log2(ST(0)), zapisywanie wyniku do ST(1) i podniesienie stosu rejestrów).

Wynik tego obliczenia to liczba rzeczywista.

Wśród instrukcji koprocesora brakuje instrukcję potęgowania dla dowolnego x rzeczywistego. Można zastosować instrukcje FSCALE i F2XM1.

Użycie instrukcji FSCALE (podniesienie dwójki do potęgi całkowitej z ST(1) i mnożenie na ST(0)) jest ograniczone przypadkiem całkowitej potęgi,

a użycie instrukcji F2XM1 (podniesienie 2 do potęgi ST(0) i odejmowanie 1; wynik w ST(0)) jest ograniczone przypadkiem potęgi od -1 do +1.

Aby zastosować instrukcje FSCALE i F2XM1, rozdzielimy "x" na część całkowitą "c" i ułamek "u", a "y" będziemy liczyć następująco:

$y = 2^x = 2^c * 2^u$.

```
.487
.MODEL small
STACK 100h
.DATA
    a DD 2.5
    b DD 1.5
    y DT (?)
.CODE
 Start:
     mov AX,@DATA
     mov DS,AX
    ;;sprawdzanie, czy jest koprocesor
    mov AL,14h ; adres komórki CMOS
    out 70h,AL ; zapisywanie adresu komórki do portu 70h
    in AL,71h ; odczyt bajta
    and AL,00000010b ;;sprawdzanie bitu 1
    jnz koproc ; skok do gałęzi „koprocesor istnieje”
    jmp koniec ; gałąź „koprocesor nie istnieje”
    koproc:
    ;;obliczenie x = log2a
    b
    fld b ; b ładujemy pierwszą, aby b znalazła się w st(1)
    fld a ; st = a
    fyl2x ; st = log2ststst(1) ; x znajduje się w st
    fld st ; przesuwanie x w st(1)
    frndint ; w st znajduje się cała część x zaokrąglona
    ; z rozszerzoną precyzją (64 bity)
    fsub st(1), st ; w st(1) ułamek x
    fxch st(1) ; w st ułamek u, w st(1) część całkowita
    f2xm1 ; podniesienie 2 do potęgi u i odejmowanie 1
    fld1 ; załadowanie 1 na stos
    faddp st(1), st ; w st 2 podniesiona do potęgi ułamka x
    fscale ; ostatnie obliczenie
    fst y ; przesyłanie wyniku
    koniec:
    mov AX, 4C00h
    int 21h
END Start
```

Przedstawiony program jest skierowany na “16-bitowe” programowanie z zastosowaniem funkcji przerwań MS DOS. W przypadku “32-bitowego” programowanie z zastosowaniem funkcji API Win32 fragment “sprawdzanie, czy jest koprocesor” należy zamienić na instrukcję `FSAVE`, która odczytuje stan koprocesora. Zmodyfikowany program jest przedstawiony niżej:

```
.586P
.MODEL flat, STDCALL
;--- funkcje API Win32 ---
...
ExitProcess PROTO :DWORD
;-------------
_DATA SEGMENT
    ...
    stan DB 108 DUP (0) ;; 108 = 7*4 bajty + 8*10 bajtów
    stan2 DB 108 DUP (0) ;; 108 = 7*4 bajty + 8*10 bajtów
    a REAL10 0.5
    ALIGN 4 ; przesuniecie do adresu podzielnego na 4
    b REAL10 0.25
    ...
_DATA ENDS
;------------
_TEXT SEGMENT
start:
     pushfd
     pushad
    ...
    ;;sprawdzanie, czy jest koprocesor
    fsave stan ;stan koprocesora (FPU)
    fwait ; tylko dla procesora 286 i 386 po każdej instrukcji
    finit ; ustawia Tag Word w "1111111111111111"
    fsave stan2 ;stan koprocesora (FPU)
    lea ESI,stan2
    lodsd ; Control Word ==> EAX
    lodsd ; Status Word ==> EAX
    lodsd ; Tag Word ==> EAX
    cmp AX,0FFFFh
    je koproc ; jest koprocesor
    jmp koniec
    koproc:
    ;;obliczenie y = a^b
    fld b ; b ładujemy pierwszą, aby b znalazła się w st(1)
    fld a ; st = a
    fyl2x ; st = st(1)*log2(st) ; x znajduje się w st
    fld st ; przesuwanie x w st(1)
    frndint ; w st znajduje się cała część x zaokrąglona
    ; z rozszerzoną precyzją (64 bity)
    fsub st(1), st ; w st(1) ułamek x
    fxch st(1) ; w st ułamek u, w st(1) część całkowita
    f2xm1 ; podniesienie 2 do potęgi u i odejmowanie 1
    fld1 ; załadowanie 1 na stos
    faddp st(1), st ; w st 2 podniesiona do potęgi ułamka x
    fscale ; ostatnie obliczenie
    koniec:
    ;--- zakończenie procesu ---------
    finit
    frstor stan
    fwait
    popad
    popfd
    ...
    ExitProcess, 0
_TEXT ENDS
END start
```