Architektura komputerów

Wykład 7
Koprocesor numeryczny i SEE

Wojciech Kordecki

Collegium Witelona
Wydział Nauk Technicznych i Ekonomicznych
Zakład Informatyki

Semestr letni 2023/24

Typy zmiennoprzecinkowe (1)

Liczby zmiennoprzecinkowe są podzbiorem liczb wymiernych postaci

$$z = (-1)^s m p^w,$$

- p – podstawa (liczba naturalna > 1),
- m – mantysa (nieujemna liczba wymierna o skończonym rozwinięciu przy podstawie p),
- s – znak ($s = 0$ lub $s = 1$),
- w – wykładnik (liczba całkowita).

Typy zmiennoprzecinkowe (2)

Jeżeli

$$1 \leq m < p,$$

to przedstawienie liczby $z \neq 0$ jest jednoznaczne.
W komputerze zawsze jest $p = 2$ (notacja binarna) natomiast do „normalnego" użytku jest $p = 10$ (notacja dziesiętna). W notacji dziesiętniej stosuje się zapis $z = x\,\mathrm{E}\,y = x10^y$, gdzie y jest liczbą całkowitą, a x jest liczbą wymierną o skończonym rozwinięciu dziesiętnym.

Typy zmiennoprzecinkowe (3)

Dla $p = 2$, jeżeli spełniony jest warunek $1 \leq m < p$, to część całkowita mantysy jest równa 1. Można wtedy przyjąć ją za domyślną i nie zapamiętywać, a zapamiętywać tylko część ułamkową. Dla takich m jest to liczba znormalizowana.
Wykładnik spełnia nierówność $w_{\min} < w < w_{\max}$.
Liczby

$$w_{\min} + 1 \text{ oraz } w_{\max} - 1$$

określają najmniejszy i największy wykładnik liczby.

Typy zmiennoprzecinkowe (4)

Dla liczb znormalizowanych z

$$2^{w_{\min}+1} \leq |z| < 2 \cdot 2^{w_{\max}-1} = 2^{\cdot \max}$$

Wartość $w_{\max}$ jest zarezerwowana dla reprezentacji symboli, a $w_{\min}$ – dla zera i liczb zdenormalizowanych.

Zero i liczby zdenormalizowane

Jeżeli $w = w_{\min}$, to z założenia $m < 1$ i wtedy $z = (-1)^s m 2^{w_{\min}+1}$. Wobec tego

$$z = 0 \iff m = 0 \wedge w = w_{\min},$$

oraz dla $|z| < 2^{w_{\min}+1}$

$$z = (-1)^s m \cdot 2^{w_{\min}+1} \iff m < 1 \wedge w = w_{\min}.$$

Symbole

Nieskończoność

$$+\infty \iff m = 1.0 \wedge w = w_{\max} \wedge s = 0,$$
$$+\infty \iff m = 1.0 \wedge w = w_{\max} \wedge s = 1.$$

Nieliczby NaN (ang. *Not a Number*):

$$m \neq 1 \wedge w = w_{\max}.$$

Formaty zmiennoprzecinkowe (1)

Format:

- pojedynczy, 32 bity,
- podwójny, 64 bity,
- chwilowy, 80 bitów.

Zapis liczby w trzech polach od najstarszego do najmłodszego bitu: pole znaku s, pole wykładnika w, pole mantysy m.
Pole znaku ma zawsze 1 bit.

Formaty zmiennoprzecinkowe (2)

Pole wykładnika zawiera wykładnik w dwójkowym kodzie spolaryzowanym w_B (polaryzacja=ang. *bias*), wg wzoru:

$$w = w_B - B$$

gdzie B jest polaryzacją wyznaczoną tak, że

$$w_{\text{min}} = w_B - B \text{ dla } w_B = 000\ldots000\text{B},$$
$$w_{\text{max}} = w_B - B \text{ dla } w_B = 111\ldots111\text{B}.$$

Pole mantysy m zawiera część ułamkową m' mantysy w naturalnym kodzie dwójkowym. W formacie chwilowym również jeden bit części całkowitej.

Formaty zmiennoprzecinkowe (3)

Format	s	Wykładnik w_B	l	Część ułamk. m'
32 bity	31	30 23		22 0
64 bity	63	62 52		51 0
80 bitów	79	78 64	63	62 0

Daje to następujące rozmiary pól:

Format	Wykładnik	Część ułamk.
32 bity	8	23
64 bity	11	52
80 bitów	15	63

Typy całkowite

Koprocesor obsługuje też typy całkowite: dwu- cztero i ośmiobajtowe w kodzie uzupełnień do dwóch oraz liczby w kodzie BDC – dziewięć bajtów (18 cyfr dziesiętnych) oraz bajt znaku (najstarszy bit w najstarszym bajcie).

Typy całkowite

Koprocesor obsługuje też typy całkowite: dwu- cztero i ośmiobajtowe w kodzie uzupełnień do dwóch oraz liczby w kodzie BDC – dziewięć bajtów (18 cyfr dziesiętnych) oraz bajt znaku (najstarszy bit w najstarszym bajcie).

Kod uzupełnień do dwóch: negujemy wszystkie bity i dodajemy 1.
Przykład. Zmiana znaku:

$$10101101 \implies 01010010 + 1 \implies 01010011$$

Typy całkowite

Koprocesor obsługuje też typy całkowite: dwu- cztero i ośmiobajtowe w kodzie uzupełnień do dwóch oraz liczby w kodzie BDC – dziewięć bajtów (18 cyfr dziesiętnych) oraz bajt znaku (najstarszy bit w najstarszym bajcie).

Kod uzupełnień do dwóch: negujemy wszystkie bity i dodajemy 1.
Przykład. Zmiana znaku:

$$10101101 \implies 01010010 + 1 \implies 01010011$$

$$\begin{array}{r} 10101101 \\ 01010011 \\ \hline 00000000 \end{array}$$

Typy całkowite

Koprocesor obsługuje też typy całkowite: dwu- cztero i ośmiobajtowe w kodzie uzupełnień do dwóch oraz liczby w kodzie BDC – dziewięć bajtów (18 cyfr dziesiętnych) oraz bajt znaku (najstarszy bit w najstarszym bajcie).

Kod uzupełnień do dwóch: negujemy wszystkie bity i dodajemy 1.
Przykład. Zmiana znaku:

$$10101101 \Longrightarrow 01010010 + 1 \Longrightarrow 01010011$$

$$\begin{array}{r} 10101101 \\ 01010011 \\ \hline 00000000 \end{array}$$

Dwukrotna zmiana znaku:

$$01010011 \Longrightarrow 10101100 + 1 \Longrightarrow 10101101$$

Odwrotna notacja polska

Notacja polska, zapis przedrostkowy, notacja Łukasiewicza – sposób zapisu wyrażeń logicznych (a później arytmetycznych), podający najpierw operator, a potem operandy (argumenty), który został przedstawiony w 1920 przez polskiego filozofa i logika Jana Łukasiewicza.

Odwrotna notacja polska ONP (ang. reverse Polish notation, RPN) – znak wykonywanej operacji umieszczony jest po operandach (zapis postfiksowy), a nie pomiędzy nimi jak w konwencjonalnym zapisie algebraicznym (zapis infiksowy) lub przed operandami jak w zwykłej notacji polskiej (zapis prefiksowy).

Odwrotna notacja polska została opracowana przez australijskiego naukowca Charlesa Hamblina jako „odwrócenie” beznawiasowej notacji polskiej Jana Łukasiewicza.

Przykład ONP

$$a + b \equiv ab+$$
$$(a + b)/c \equiv ab + c/$$
$$a + b/c \equiv abc/+$$
$$a/c + b \equiv ac/b+$$

Przykład realizacji ONP

Założenia:

- działania na dwuargumentowe $+ - */$ wykonywane rozkazami `add, sub, mul, div,`
- argumenty na stosie – lewy na wierzchołku stosu, prawy poniżej,
- wynik na wierzchołku stosu, po zdjęciu obu argumentów,
- push – położenie na wierzchołek stosu z pamięci,
- pop zdjęcie z wierzchołka stosu do pamięci.

Przykład realizacji ONP

Obliczenie

$$w = x + y\,(z \cdot t + u/v) \equiv xyzt * uv/ + *+$$

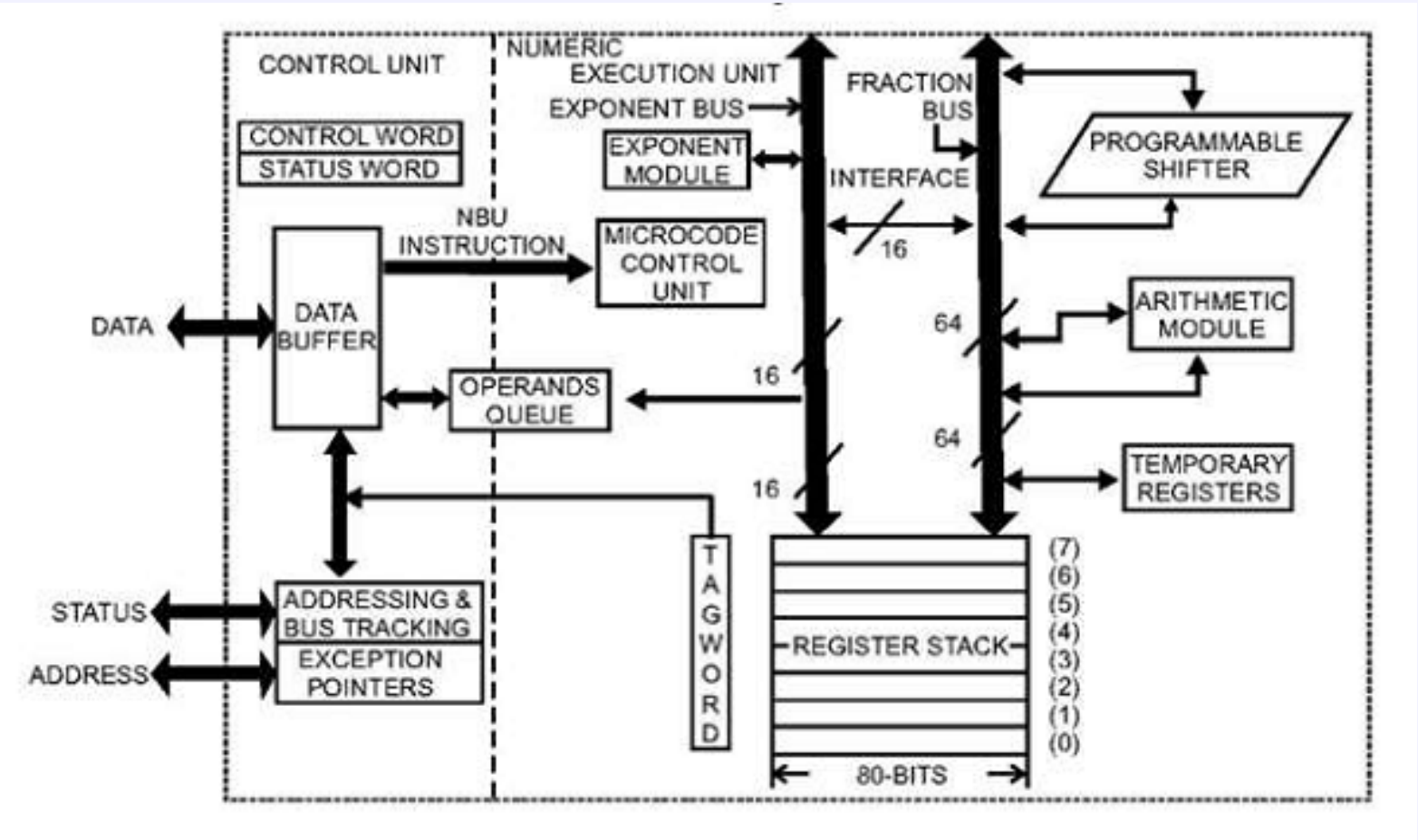
```
push x
push y
push z
push t    % x y z t
mul       % x y zt
push u    % x y zt u
push v    % x y zt u v
div       % x y zt u/v
add       % x y zt+u/v
mul       % x y(zt+u/v)
add       % x+y(zt+u/v)
pop w     % wynik zapamiętany
```

Literatura

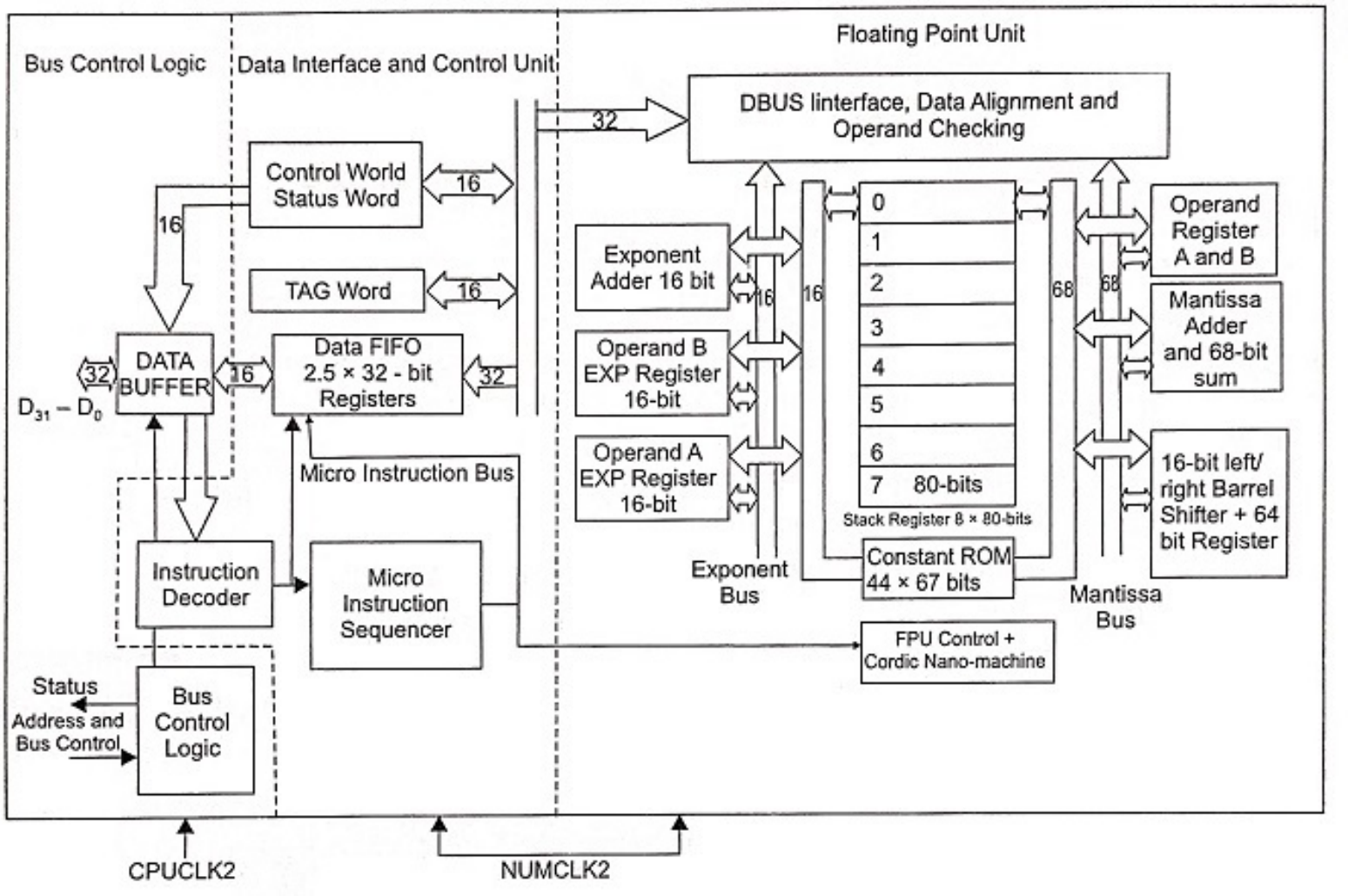
Książka wydana bardzo, bardzo dawno temu, ale prawie nic się nie przestarzała:

S. Kruk. *Asembler w koprocesorze*. Warszawa, Mikom 2003.

Schemat blokowy koprocesora 8087



Schemat blokowy koprocesora 80387



Architektura koprocesora 80x87 (1)

Koprocesor 80x87 zawiera stos ośmiu rejestrów 80 bitowych, rejestry sterowania, rejestry stanu koprocesora, stanu stosu oraz rejestry instrukcji i argumentu.
Stos rejestrów R0 – R7 rośnie dół, adres wierzchołka stosu zajmuje jako wskaźnik stosu rejestrów ST, trzy bity rejestrze stanu koprocesora. Rejestry adresowane są wyłącznie względem wierzchołka stosu. Rejestr na wierzchołku stosu $\mathrm{ST}(0)$. Przez $\mathrm{ST}(n)$ oznaczony jest rejestr leżący o n pod wierzchołkiem, czyli mający adres rejestru o n większy.
Po położeniu na stos nowej wartości, rejestr będący poprzednio $\mathrm{ST}(n)$ staje się rejestrem $\mathrm{ST}(n+1 \mod 8)$.

Architektura koprocesora 80x87 (2)

Po inicjacji, wskaźnik stosu rejestrów ma wartość 0, czyli wartość zostanie wpisana do R7.
Z każdym rejestrem stosu związane jest dwubitowe pole stanu rejestru Rn w rejestrze stanu stosu koprocesora.

- $Rn = 00$ – liczba poprawna,
- $Rn = 01$ – zero,
- $Rn = 10$ – liczba niepoprawna lub nieskończoność,
- $Rn = 11$ – rejestr pusty.

Definicje liczb w formacie koprocesora (1)

DW – liczba całkowita dwubajtowa,
DD – liczba całkowita czterobajtowa,
DQ – liczba całkowita ośmiobajtowa,
DT – liczba całkowita dziesięciobajtowa w BDC,
DD – liczba zmiennoprzecinkowa w formacie pojedynczym,
DQ – liczba zmiennoprzecinkowa w formacie podwójnym,
DT – liczba zmiennoprzecinkowa w formacie chwilowym.

Definicje liczb w formacie koprocesora (2)

zmp-poj – liczba zmiennoprzecinkowa w formacie pojedynczym
zmp-pod – liczba zmiennoprzecinkowa w formacie podwójnym
zmp-chw – liczba zmiennoprzecinkowa w formacie chwilowym
zmp = zmp-poj|zmp-pod|zmp-chw
calk2 – liczba całkowita (binarna) dwubajtowa
calk4 – liczba całkowita (binarna) czterobajtowa
calk8 – liczba całkowita (binarna) ośmiobajtowa
calk = calk2|calk4|calk8

Instrukcje przesłania.

FLD ST(i)|zmp
umieszcza na stosie argument i zmniejsza wskaźnik stosu.
FST ST(i)|zmp-poj|zmp-podw
zapisuje wartość z wierzchołka stosu w argumencie.
FSTP ST(i)|zmp
zapisuje wartość z wierzchołka stosu w argumencie i zdejmuje ją ze stosu (zwiększa wskaźnik stosu).
FXCH ST(i)
wymienia wartość z wierzchołka stosu z zawartością rejestru.
FILD całk
umieszcza na stosie argument i zmniejsza wskaźnik stosu.
FIST calk2|calk4
zapisuje wartość z wiechołka stosu w argumencie.
FISTP calk
zapisuje wartość z wiechołka stosu w argumencie i zdejmuje ją ze stosu (zwiększa wskaźnik stosu).

Instrukcje arytmetyczne.

Format stosowy: Fop – wykonuje operacje na argumentach ST i ST(1), wynik w ST(1), zdejmuje ST ze stosu.
Format rejestrowy: Fop ST(i),ST|Fop ST,ST(i) – wykonuje operacje na argumentach ST i ST(i), wynik w pierwszym, nie ma zdjęcia ze stosu.
Format rejestrowy ze zdjęciem ze stosu: FopP ST(i),ST – wykonuje operacje na argumentach ST i ST(i), zdejmuje ST ze stosu.
Format z argumentem zmiennoprzecinkowym: Fop zmp-poj|zmp-pod – wykonuje operacje na argumentach ST i zmp-poj|zmp-pod.
Format z argumentem całkowitym: FIop calk2|calk4 – wykonuje operacje na argumentach ST i calk2|calk4.
Operacjami op mogą być ADD,SUB,SUBR, MUL, DIV, DIVR.

Przykład

FADD
dodaje ST do ST(1) i umieszcza w ST(1), zdejmuje ST ze stosu.
FADD ST(i),ST|ST,ST(i)
dodaje argumenty, wynik w pierwszym.
itd.
FSQRT – oblicza pierwiastek kwadratowy z wartości na wierzchołku stosu i wpisuje tam wynik.
FRNDINT – zaokrągla liczbę z wierzchołka stosu do całkowitej i wpisuje tam wynik.

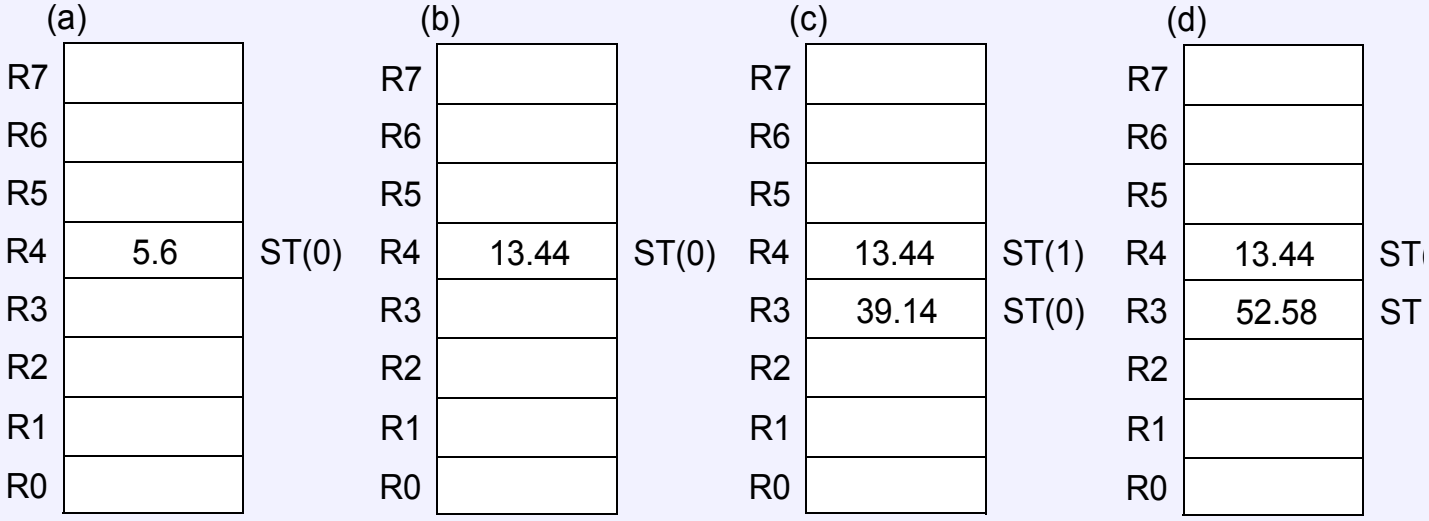
Przykład

http://www.infophysics.net/x87.pdf

Computation
Dot Product = (5.6 x 2.4) + (3.8 x 10.3)

Code:

```
FLD  value1 ;(a) value1=5.6
FMUL value2 ;(b) value2=2.4
FLD  value3 ; value3=3.8
FMUL value4 ;(c)value4=10.3
FADD ST(1)  ;(d)
```



Instrukcje obliczania funkcji przestępnych

FPTAN – oblicza tangens z liczby ST(0) w postaci y/x, umieszcza y w ST(0), po czym kładzie x na stos.
Inne: FPATAN, FSIN, FCOS, FSINCOS i.in.
Na przykład FYL2X oblicza wartość wyrażenia $y \log_2 x$, gdzie y to ST(1), x to ST(0). Następnie x jest zdejmowane ze stosu, a wynik do ST.

Przykład

$$y = ax^3 + bx^2 + cx + d$$

```
fld d ; // d
fld x ; // x ; d
fld st ; // x ; x ; d
fmul st , st (1) ; // xx ; x ; d
fld st (1) ; // x ; xx ; x ; d
fmul st , st (1) ; // xxx ; xx ; x ; d
fmul a ; // axxx ; xx ; x ; d
faddp st (3) , st ; // xx ; x ; axxx+d
fmul b ; // b* xx ; x ; axxx+d
faddp st (2) , st ; // x ; axxx+b* xx+d
fmul c ; // c*x ; axxx+b* xx+d
fadd ; // axxx+b* xx+c*x+d
fstp y
```

Koprocesor ARM

Elektronika B2B
16 maja 2012

https://elektronikab2b.pl/technika/16653-koprocesor-arytmetyczny-w-stm32f4

Rozszerzenie MMX (1)

MMX – **M**anager **M**emory E**X**tended.

SIMD – Single Instruction Multiple Data
MMX – rejestry 64 bitowe, dane spakowane po 8 bajtów, 4 słowa, 2 podwójne słowa. Rejestry mm0 – mm7.
64 bity: 4 słowa (dwubajtowe). Operacje na argumencie (argumentach) – równocześnie na czterech słowach.
Mnemoniki operacji poprzedzone literą P (packed), np. PADDW – dodanie czterech słów równocześnie.

Rozszerzenie MMX (2)

Rejestry MMX to fizycznie rejestry FPU.
Pakowanie: jednostki numerowane od 0 – najmniej znaczące bity.
Lista rozkazów MMX ma 57 rozkazów [**?**]:

- rozkazy przenoszenia danych
- rozkazy arytmetyczne
- rozkazy porównania
- rozkazy konwersji
- rozkazy logiczne
- rozkazy przesunięcia
- rozkaz EMMS

Rozszerzenie MMX (3)

Przy przepełnieniu lub niedopełnieniu nie jest ustawiany żaden bit przeniesienia lub pożyczki i nie ma przeniesienia lub pożyczki.
Zawijanie (wrap around): pomija się przeniesienie, czyli po 0FFFFh następuje 0000h.
Nasycenie (saturation): wynik dochodzi do wartości maksymalnej (minimalnej) i taki pozostaje.
Przy operacjach z nasyceniem występuje dodatkowa litera S, np. PADDSW.

Rozszerzenie MMX (4)

- Kopiowanie danych MOVQ, MOVD
- Konwersja – pakowanie rozpakowanie.
- Operacje dodawania, odejmowania, mnożenia na typach BYTE, WORD, DWORD.
- Porównanie – bajtami, słowami, podwójnymi słowami. Prawda – same jedynki, fałsz – same zera.
- Operacje logiczne na bitach – alternatywa, koniunkcja, różnica symetryczna, negacja.
- Przesunięcia i obroty.
- EMMS – zawsze wtedy, gdy po użycie MMX będzie się wykonywać instrukcje koprocesora.

Nie można równocześnie używać koprocesora i MMX. W razie konieczności przerwania zadania i jego ponownego podjęcia, trzeba rejestry zapamiętać i odtworzyć instrukcjami FXSAVE i FXRTOR.

Użycie MMX w programach

Rozkaz EMMS ustawia pola rejestru stanu zawartości rejestrów stosu koprocesora na 1. Wszystkie inne rozkazy MMX zerują pola tego rejestru. Rozkaz EMMS umożliwia koprocesorowi uruchomienie innych programów.
11 – oznacza pusty rejestr: rejestr stanu koprocesora.
Przed uruchomieniem programu używającego kodów MMX należy sprawdzić, czy maszyna ma te kody zaimplementowane. Do tego używa się rozkazu CPUID z argumentem w EAX

```
MOV EAX,1
CPUID
TEST EDX,00800000h ;MMX? bit 23 w EDX
JNZ jest_MMX
; .........
jest_MMX:   ;dalej
```

Przykład

```
liczby word 0D04h,65AEh,00FEh,1379h,
            1004h,0FD23h,2500h,3FD8h
lea esi,liczby          ;adres pierwszej liczby
movq mm0,[esi]      ;MM0 = 1379 00FE 65AE 0D04
movq mm1, [esi+8]           ;MM1 = 3DF8 2500 FD23 1004
movq mm2,mm0
pmullw mm0,mm1              ;MM0 = 3518 B600 DCCA 7410
pmulhw mm1,mm2              ;MM1 = 04DB 0024 FEDC 00D0
movq mm2,mm0
punpcklwd mm0,mm1           ;MM0 = FEDC DCCA 00D0 7410
punpckhwd mm2,mm1           ;MM2 = 04DB 3518 0024 B600
```

Rozszerzenie SSE – rejestry

SSE – Streaming SIMD Extensions
SSE – rejestry 128 bitowe, dane spakowane.
Rejestry xmm0 – xmm7. Rejestry XMM są niezależne od koprocesora. Instrukcje można mieszać ze sobą.
128 bitów: 4 liczby zmienooprzecinkowe (czterobajtowe) – float, real4 – 2 liczby zmienooprzecinkowe (ośmiobajtowe) – double, 16 liczb stałoprzecinkowych (jednobajtowe), analogicznie – 8, 4, 2 słowa

Rozszerzenie SSE – instrukcje

Instrukcje SSE dzielą się na cztery grupy.

- Instrukcje operujące na czterech liczbach typu float.
- Dodatkowe instrukcje na 64 bitowych liczbach całkowitych, np. średnia.
- Instrukcje sterujące stanem SSE.
- Instrukcje związane z pamięcią podręczną, pobraniem wstepnym, komunikowaniem się procesora z pamięcią.

Instrukcja CPUID – identyfikuje procesor, rozpoznaje SSE – dokumentacja na stronie producenta (ponad 70 stron!).
Instrukcje FXSAVE FXSTOR kopiują i odtwarzają również rejestry XMM.

Rozszerzenie SSE – instrukcje dla danych 128 bitowych

Instrukcje rozszerzenia SSE dla danych 128 bitowych.

- Kopiowanie danych między MMX/XMM i pamięcią. Wymaga wyrównania adresów do 16 – inaczej wyjątek.
- Instrukcje arytmetyczne, np. pierwiastek, odwrotność.
- Porównywanie par argumentów.
- Operacje logiczne.
- Operacje tasujące i rozpakowujące.
- Różne konwersje między formatami.

Przykład

Skalowanie figury geometrycznej płaskiej. Dane (x, y) jako dwie liczby pojedynczej precyzji (float czyli real4).

```
.data
ws real4 2.0    ;początkowo razy 2
tablica real4 100 dup (0.0)
liczba_wierzch = lengthof tablica/2
```

Przykład c.d.

```
.code
(...)
mov ecx,liczba_wierzch
mov esi, offset tablica
mov ebx,ecx
shr ecx,1    ;po 2 wierzchołki
mov edx,ecx
movss xmm0,ws                ; ?  ?  ? ws
shufps xmm0,xmm0, 0   ;ws ws ws ws
(...)
```

SSE – ewolucja

1999	SSE
2001	SSE2
2003	SSE3
2006	SSE4
2007	SSE4a
2008	SSE4.2
2009	SSE5
2011	AVX
2013	AVX2
...	

Wikipedia

https://pl.wikipedia.org/wiki/MMX_(zestaw_instrukcji)

https://pl.wikipedia.org/wiki/3DNow!

https://pl.wikipedia.org/wiki/Streaming_SIMD_Extensions

AVX

AVX – Advanced Vector Extensions.

Szesnaście 256 bitowych rejestrów.

Każdy po 8 liczb 32 bitowych (float) lub po 4 liczby 64 bitowe (double).

Dokumentacja – 595 stron!
https://www.intel.com/content/dam/develop/external/us/en/documents/36945

Warto też zajrzeć na stronę:
https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html#

Obszerne omówienie w Wikipedii

https://pl.wikipedia.org/wiki/Advanced_Vector_Extensions

- 256-bitowe rejestry – w asemblerze YMM0...YMM15. W dalszych wersjach AVX mogą pojawić się jeszcze większe rejestry, 512-, 1024-bitowe.
- Kilka rozkazów działających wyłącznie na rejestrach YMM (19 rozkazów).
- Dodane czteroargumentowe rozkazy akumulujące wyniki mnożenia wektorów liczb zmiennoprzecinkowych, to znaczy wykonujące obliczenia według schematu $w = \pm z + (\pm x \cdot y)\, w = \pm z + (\pm x \cdot y)$ (12 rozkazów).
- Dodane specjalizowane instrukcje wspomagające szyfrowanie AES (6 rozkazów).

Obszerne omówienie w Wikipedii c.d.

- Część rozkazów SSE, może wykonywać działania na rejestrach YMM (88 rozkazów).
- Rozszerzone kodowanie rozkazów, dzięki któremu możliwe stało się wykonywanie niektórych instrukcji SSE w wariancie trójargumentowym lub czteroargumentowym.