Uso del Coprocesador Matemático

El coprocesador o unidad de control de punto flotante es un circuito integrado especializado o parte del procesador principal, que se caracteriza por tener un conjunto de instrucciones orientadas a la resolución de cálculos matemáticos complejos y con alta precisión (funciones básicas, trigonométricas, logarítmicas, etc.)

El coprocesador puede manejar valores en distintas precisiones, ***operando internamente con 80 bits***:

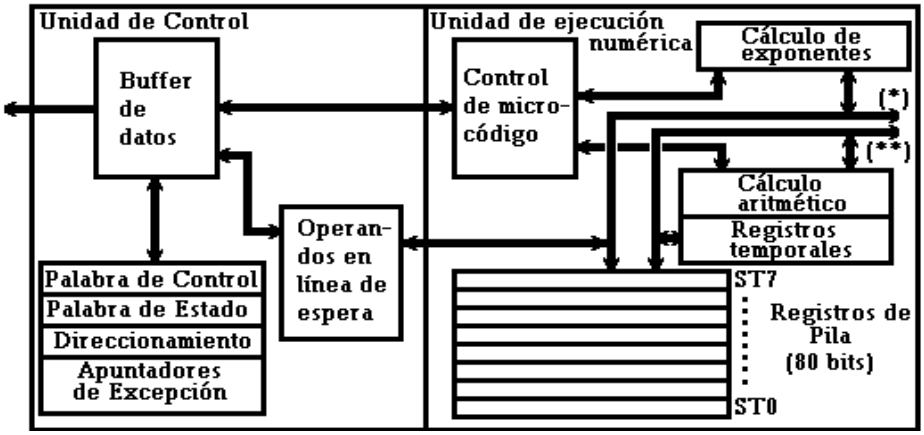
Words, 16 bits = 2 bytes, (DD) definen enteros con signo (-32768 a +32767).

Dwords, 32 bits = 4 bytes, (DD) definen enteros con signo (-2.147.483.648 a +2.147.483.647), y valores de punto flotante con precisión simple.

Qwords, 64 bits = 8 bytes, (DQ) definen valores en el formato de punto flotante de doble precisión.

80 bits = 10 bytes, (DT) definen cantidades en el formato de punto flotante con precisión plena ("full precision")

(Existen algunas instrucciones del coprocesador que reconocen el formato BCD en 10 bytes)



El coprocesador matemático posee básicamente una unidad de control y una unidad de ejecución que posee 8 registros de punto flotante que pueden almacenar 80 bits y están organizados como una pila LIFO.
Cada registro se nombra como ST0, ST1, ST2,……,ST7 .
ST0 se refiere siempre al valor en el tope de la pila y todos aquellos valores nuevos se añaden al tope.

Al cargar el primer registro (ST0) con un dato se produce automáticamente un desplazamiento de los datos contenidos en los otros registros. Así, si se escribe en ST0 un dato que se encuentra en memoria (instrucción FLD → load), entonces el dato que está en ST0 se transfiere a ST1 dejando disponible ST0, el dato que está en ST1 se transfiere a ST2, el de ST2 a ST3, etc., y se pierde el dato en ST7.

De igual manera, cada vez que se extrae de la pila a la memoria, siempre se hace desde el tope. Los cálculos en general, se hacen sobre el elemento 1 y luego se desplazan hacia arriba (pop) , de manera que el ST1 pasa a estar en ST0 lo que quiere decir que el cálculo siempre quedará en el tope.

El coprocesador y el procesador se comunican directamente haciendo uso de las instrucciones propias del primero. Para distinguir dichas instrucciones, en general se les coloca una F delante de cada una de ellas.

Las instrucciones para la carga desde memoria al tope de la pila y viceversa, son:

FLD n	Carga un nº de punto flotante (n) de la memoria a la pila del coprocesador (Puede ser un DD, DQ o un DT)
FILD n	Ídem FLD, pero con un entero (n) convirtiéndolo a flotante
FST *dest*	copia de la pila ST(0) a la memoria (*dest*). El *dest*, puede ser precisión simple o doble
FIST *dest*	Ídem FST, para enteros.
FSTP *dest*	Ídem FST, pero además lo quita de la pila
FISTP *dest*	Ídem FIST, pero lo quita de la pila, para enteros

Para las comparaciones de números en punto flotante, el coprocesador utiliza en la palabra de estado, entre otras, cuatro banderas: c0, c1, c2 y c3 (bits 8 9 10 y 14) que mantienen información que corresponde a la indicación del código de condición para las instrucciones de comparación.

Por ejemplo, para las instrucciones de comparación, la condición es anunciada a través de los bits C3 y C0. (si son 00, entonces ST0 es mayor que el operando; si son 01, ST0 es menor que el operando; cuando resulta 10, entonces ST0 es igual al operando y si son 11, los operandos son incomparables.)

Estos bits no pueden ser accedidos directamente desde la CPU, de manera que las instrucciones de salto consultan el registro *FLAGS*, pero no los registros de estado del coprocesador. Para resolverlo se deben utilizar nuevas instrucciones para transferir los bits de la palabra de estado del coprocesador a los correspondientes bits del registro *FLAGS*:

FSTSW dest	Almacena la palabra de estado del coprocesador en memoria o en dest. (generalmente registro AX)
SAHF	Almacena el registro AH en el registro *FLAGS*
LAHF	Carga el registro AH con los bits del registro *FLAGS*

Para el truncado y el redondeo, el coprocesador utiliza la Palabra de Control, la cual posee algunas banderas para indicar Operación Invalidada (bit 0), Operando desnormalizado (bit 1), División por Cero (bit 2), Overflow y Underflow (bits 3 y 4 respectivamente), Precisión (bit 5), Control de Precisión (PC, bits 8 y 9) y Control de Redondeo (RC, bits 10 y 11). Estos dos últimos bits permiten fijar las reglas para el redondeo que realiza la instrucción FRNDINT: 00, redondea al entero más próximo; 01, redondea al entero inferior; 10, redondea al entero superior y 11 redondea a 0.

FLDCW mem2i	Carga en la Palabra de control lo que se almacena en mem2i.
FSTCW mem2i	Almacena la palabra de control del coprocesador en memoria o en dest. (2 bytes con signo)

Ejemplo

Si deseo efectuar **if (x >y)**

```
        fld     y           ;  ST0 = y
        fld     x           ;  ST0 = x   ST1 = y
        fcomp               ; compara ST0 con ST1 ( x – y )
        fstsw   ax          ; mueve los bits C a FLAGS
        fwait
        sahf
        jbe else_part       ; si x no es mayor que y, vaya a else_part

then_part:
        ; .... código para el “then”
        jmp     end_if
else_part:
        ;código para parte “else”
end_if:
```

Ejemplo de suma de dos reales

Si deseo efectuar

```
        var2 := 5.3;
        var3 := 2.6;
        var1 := var2 + var3;

.MODEL          LARGE   ; tipo del modelo de memoria usado.
.386
.STACK 200h     ; bytes en el stack

.DATA
var1 dd 0
var2 dd 0
var3 dd 0
_cte_1 dd 5.300000
_cte_2 dd 2.600000

.CODE
                mov AX,@DATA ; inicializa el segmento de datos
                 mov DS,AX ;

                FINIT   ; Inicializacion del co-procesador

                fld     _cte_1          (Coloca cte1 en el tope)
                fstp    var2            (Mueve el tope a var2 y lo quita de la pila)
                fld     _cte_2          (Coloca cte2 en el tope)
                fstp    var3            (Mueve el tope a var3 y lo quita de la pila)
                fld     var2            (Coloca var2 en el tope  ST0 = var2)
                fld     var3            (Coloca var3 en el tope  ST0 = var3 ,ST1 = var2)
                fadd                    (Calcula ST1 = var2+var3 y desplaza hacia el tope
                                         quedando ST0 = var2+var3)
                fstp    var1            (Mueve el tope a var1 y lo quita de la pila, quedando
                                          la suma en var1 y la pila vacía)

                int 21h
                mov ax, 4C00h                   ;fin de ejecución
end;
```

Ejemplo de resta de dos reales

Si deseo efectuar

var1 := 9-8;

```
.MODEL          LARGE   ; tipo del modelo de memoria usado.
.386
.STACK 200h     ; bytes en el stack

.DATA
var1 dd 0
_cte_1 dd 9.000000
_cte_2 dd 8.000000

.CODE
                mov AX,@DATA ; inicializa el segmento de datos
                 mov DS,AX ;

                FINIT   ; Inicializacion del co-procesador

                fld     _cte_1          (Coloca cte1 en el tope)
                fld     _cte_2          (Coloca cte2 en el tope  ST0 = cte2 ,ST1 = cte1)
                fsub                    (Calcula ST1 = cte2-cte1 y desplaza hacia el tope
                                         quedando ST0 = cte2-cte1)
                fstp    var1            (Mueve el tope a var1 y lo quita de la pila, quedando
                                          la resta en var1 y la pila vacía)

                int 21h
                mov ax, 4C00h                   ;fin de ejecución
end;
```

Ejemplo de redondeo

```
                fld cte         ;5.5    (Coloca cte en el tope)
                frndint                 (Redondea y el resultado queda en el tope)
                fistp aux000            (Mueve el tope a aux (valor 6) y vacía la pila)
```

Ejemplo de truncado

```
                 fld cte        ;5.110000
                 fstp var

                 xor eax,eax
                 fstcw truncv           ; toma la palabra de control actual
                 fwait
                 mov ax,truncv
                 or ax,0c00h            ; fija el redondeo de los bits a 11 (truncar)
                 mov truncn,ax
                 fldcw truncn           ; restaura la palabra de control

                 fld var
                 frndint
                 fistp aux   (Aca queda el truncado)
                 fldcw   truncv
```

```
aux     dd      ?
var     dd      ?
truncv  dw      ?
truncn  dw      ?
```

Apéndice

Set completo de instrucciones del coprocesador matemático (*2):

Instrucción	Descripción
F2XM1	0 := (2.0 ** 0) - 1.0
FABS	0 := |0|
FADD	1 := 1 + 0, pop
FADD i	0 := i + 0
FADD i,0	i := i + 0
FADD 0,i	0 := i + 0
FADD mem4r	0 := 0 + mem4r
FADD mem8r	0 := 0 + mem8r
FADDP i,0	i := i + 0, pop
FBLD mem10d	push, 0 := mem10d (dato en BCD)
FBSTP mem10d	mem10d := 0, pop (dato en BCD)
FCHS	0 := -0
FCLEX	borrar excepciones
FCOM	comparar, 0 - 1
FCOM 0,i	comparar, 0 - i
FCOM i	comparar, 0 - i
FCOM mem4r	comparar, 0 - mem4r
FCOM mem8r	comparar, 0 - mem8r
FCOMP	comparar, 0 - 1, pop
FCOMP 0,i	comparar, 0 - i, pop
FCOMP i	comparar, 0 - i, pop
FCOMP mem4r	comparar, 0 - mem4r, pop
FCOMP mem8r	comparar, 0 - mem8r, pop
FCOMPP	comparar, 0 - 1, ambos pop
FCOS	sólo 387: push, 1/0 := coseno(ant. 0)
FDECSTP	decrementar el stack pointer
FDISI	deshabilitar interrupciones(ignora .287)
FDIV	1 := 1 / 0, pop
FDIV i	0 := 0 / i
FDIV i,0	i := i / 0
FDIV 0,i	0 := 0 / i
FDIV mem4r	0 := 0 / mem4r
FDIV mem8r	0 := 0 / mem8r
FDIVP i,0	i := i / 0, pop
FDIVR	1 := 0 / 1, pop
FDIVR i	0 := i / 0
FDIVR i,0	i := 0 / i
FDIVR 0,i	0 := i / 0
FDIVR mem4r	0 := mem4r / 0
FDIVR mem8r	0 := mem8r / 0
FDIVRP i,0	i := 0 / i, pop
FENI	habilitar interrupciones (ignora .287)
FFREE i	i vacío
FIADD mem2i	0 := 0 + mem4i
FIADD mem4i	0 := 0 + mem2i
FICOM mem2i	comparar, 0 - mem2i
FICOM mem4i	comparar, 0 - mem4i
FICOMP mem2i	comparar, 0 - mem2i, pop
FICOMP mem4i	comparar, 0 - mem4i, pop
FIDIV mem2i	0 := 0 / mem2i
FIDIV mem4i	0 := 0 / mem4i

FIDIVR mem2i	0 := mem2i / 0
FIDIVR mem4i	0 := mem4i / 0
FILD mem2i	push, 0 := mem2i
FILD mem4i	push, 0 := mem4i
FILD mem8i	push, 0 := mem8i
FIMUL mem2i	0 := 0 * mem2i
FIMUL mem4i	0 := 0 * mem4i
FINCSTP	incrementar stack pointer
FINIT	inicializar el 80x87
FIST mem2i	mem2i := 0
FIST mem4i	mem4i := 0
FISTP mem2i	mem2i := 0, pop
FISTP mem4i	mem4i := 0, pop
FISTP mem8i	mem8i := 0, pop
FISUB mem2i	0 := 0 - mem2i
FISUB mem4i	0 := 0 - mem4i
FISUBR mem2	0 := mem2i - 0
FISUBR mem4	0 := mem4i - 0
FLD i	push, 0 := old i
FLD mem10r	push, 0 := mem10r
FLD mem4r	push, 0 := mem4r
FLD mem8r	push, 0 := mem8r
FLD1	push, 0 := 1.0
FLDCW mem2i	Palabra de Control:= mem2i
FLDL2E	push, 0 := log base 2.0 de e
FLDL2T	push, 0 := log base 2.0 de 10.0
FLDLG2	push, 0 := log base 10.0 de 2.0
FLDLN2	ush, 0 := log base e de 2.0
FLDPI	push, 0 := Pi
FLDZ	push, 0 := +0.0
FMUL	1 := 1 * 0, pop
FMUL i	0 := 0 * i
FMUL i,0	i := i * 0
FMUL 0,i	0 := 0 * i
FMUL mem4r	0 := 0 * mem4r
FMUL mem8r	0 := 0 * mem8r
FMULP i,0	i := i * 0, pop
FNCLEX	borrar excepciones sin Wait
FNSTCW mem2i	mem2i := palabra de control
FNSTSW AX	AX := palabra de estado
FNSTSW mem2i	mem2i := palabra de estado
FPATAN	0 := arctan(1/0), pop
FPREM	0 := REPITE(0 - 1)
FPREM1	387 sólo: 0 := REPITE(0 - 1) IEEE compat.
FPTAN	push, 1/0 := tan(ant.0)
FRNDINT	0 := redondear(0)
FSCALE	0 := 0 * 2.0 ** 1
FSETPM	setear modo de protección
FSIN	387 sólo: push, 1/0 := seno(ant.0)
FSINCOS	387 sólo: push, 1 := seno, 0 := cos(ant.0)
FSQRT	0 := raíz cuadrada de 0
FST i	i := 0
FST mem4r	mem4r := 0
FST mem8r	mem8r := 0
FSTCW mem2i	mem2i := palabra de control
FSTP i	i := 0, pop
FSTP mem10r	mem10r := 0, pop

FSTP mem4r	mem4r := 0, pop
FSTP mem8r	mem8r := 0, pop
FSTSW AX	AX := palabra de estado
FSTSW mem2i	mem2i := palabra de estado
FSUB	1 := 1 - 0, pop
FSUB i	0 := 0 - i
FSUB i,0	i := i - 0
FSUB 0,i	0 := 0 - i
FSUB mem4r	0 := 0 - mem4r
FSUB mem8r	0 := 0 - mem8r
FSUBP i,0	i := i - 0, pop
FSUBR	1 := 0 - 1, pop
FSUBR i	0 := i - 0
FSUBR i,0	i := 0 - i
FSUBR 0,i	0 := i - 0
FSUBR mem4r	0 := mem4r - 0
FSUBR mem8r	0 := mem8r - 0
FSUBRP i,0	i := 0 - i, pop
FTST	comparar 0 - 0.0
FWAIT	esperar para 87 listo (sólo 8088(86))
FXAM	C3 -- C0 := tipo de 0
FXCH	intercambio 0 y 1
FXCH 0,i	intercambio 0 y i
FXCH i	intercambio 0 y i
FXCH i,0	ntercambio 0 y i
FXTRACT	push, 1 := exponente, 0 := signifcando
FYL2X	0 := 1 * log base 2.0 de 0, pop
FYL2XP1	0 := 1 * log base 2.0 de (0+1.0), pop

***(2) Referencias**

ant.	anterior
mem4r	dirección u "offset" de memoria con un dato de 4 bytes (DobleWord, definido con la directiva "dd").
mem8r	dirección u "offset" de memoria con un dato de 8 bytes (QuadWord, definido con la directiva "dq").
mem10r	dirección u "offset" de memoria con un dato de 10 bytes, definido con la directiva "dt".
mem10d	dirección u "offset" de memoria con dato en BCD, el que será reconocido por las instrucciones FBLD y FBSTP.
mem4i, mem2i	corresponde con números enteros de 4 y 2 bytes respectivamente, con signo.
mem14 y mem94	buffers de 14 y 94 bytes que contienen el estado de la máquina 80x87.
0, 1, 2...	Posición en la pila (STn)