
Lecture 9: Floating Point Arithmetic

Dragan Mirkovic
Department of Computer Science
University of Houston

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Announcements

- Today we will continue our discussion of floating point operations and the associated hardware
 - Chapter 19 in text
 - Quiz #6 Floating Point on Thursday 4/15/04

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

FP Arithmetic

- The stack machine implements Reverse Polish Notation (RPN)
 - RPN allows for simple and unambiguous coding of complex arithmetic expressions with minimum number of special symbols
 - Each operator pops the arguments it needs from the stack
 - The arguments need to be pushed onto the stack before executing the arithmetic operator
- Examples:
 - $a+b$ in RPN is $ab+$
 - $(a+b)/(d-f)$ in RPN is $ab+df-/$
 - The variable name v should be interpreted as **push v**

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Examples

- The order of the arguments is important
 - Example:
- In general:

Code	1	0
<code>fld A;</code>		A
<code>fld B;</code>	A	B
<code>fsub ;</code>		A-B

Code	1	0
<code>fld A;</code>		A
<code>fld B;</code>	A	B
<code>fop ;</code>		A op B

- Example:

Code	1	0
<code>fld A;</code>		A
<code>fld B;</code>	A	B
<code>fsubr ;</code>		B-A

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Examples

- For floating point numbers **A**, **B**, **C**, **D**, and **E** use the FPU to set **E** = **(A+B) * (C-D)** .
- The RPN form of the expression is: **AB+CD-***

• Solution:

	ST(2)	ST(1)	ST
<code>fld A;</code>			A
<code>fld B;</code>		A	B
<code>fadd ;</code>			A+B
<code>fld C;</code>		A+B	C
<code>fld D;</code>	A+B	C	D
<code>fsub ;</code>		A+B	C-D
<code>fmul ;</code>			(A+B) * (C-D)
<code>fstp E;</code>	Pop to clear	the stack	

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Simplifications

- The frequently occurring pairs of instructions
`f[i]ld mem`
`fop`
and
`fld ST(n)`
`fop`
- Can be abbreviated as
`f[i]op mem`
`fop ST, ST(n);`
- Register form of the FP instruction (both operands must be specified and one of them must be **ST**)
`fop ST, ST(n);`
`fop ST(n), ST;`

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example

- Solution: `ST(2) ST(1) ST`
`fld A; A`
`fadd B; A+B`
`fld C; A+B C`
`fsub D; A+B C-D`
`fmul ; (A+B) * (C-D)`
`fstp E; Pop to clear the stack`
- This method uses one less stack entry
 - Important when evaluation complicated expressions

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

FP Optimization

- In many applications the efficiency of the floating point operations is crucial.
- Requires careful optimization or a good optimizing compiler.
- Example:
 - Use FPU to compute $X = (A+1)/(A-1)$.

- Register version
`fld A`
`fld ST;`
`fldl ;`
`fadd ST(2), ST;`
`fsubp ST(1), ST;`
`fdiv ;`
`fstp X`

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

FPU Arithmetic Instructions

- In addition to the four basic arithmetic instructions all 80x86 processors include the following FP instructions:

<code>fabs</code>	Absolute value
<code>fchs</code>	Change sign
<code>fsqrt</code>	Square root
<code>fscale</code>	Scaling by power of 2
<code>frndint</code>	Rounds ST to an integer
<code>fprem</code>	Remainder

- Starting with 80387 FPU can compute transcendental functions like sin, cos and tan.

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Floating Point I/O

- The representation of FP numbers is more complex than the integer representation
- It is much more complicated to convert FP numbers than integers
- UTIL.LIB contains routines for conversion

<code>GetFP</code>	Input: none
	Output: Read ASCII real from the keyboard and store it in ST
<code>PutFP</code>	Input: ST on the FPU stack
	Output: the number is displayed and popped from the FPU stack

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example

- Read floating point numbers A, B, and C from the keyboard and compute and display the the solution to $Ax + B = C$.
- Solution: $x = (C - B)/A$.
- Code:

```
INCLUDE PCMAC.INC
        .MODEL    SMALL
        .586
        .STACK   100h
        .DATA
PromptA DB    'Type A: $'
PromptB DB    'Type B: $'
PromptC DB    'Type C: $'
OutMsg  DB    'The solution of Ax + B = C is x = $'
        .CODE
        ...
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example...

```
Linear  PROC
        EXTRN   GetFP : NEAR, PutFP : NEAR
        _Begin
        _PutStr PromptA ;          ST(2)  ST(1)  ST
        call   GetFP ;                      A
        _PutStr PromptB
        call   GetFP ;                      A    B
        _PutStr PromptC
        call   GetFP ;                      A    B    C
        fsubrp ST(1), ST ;                A    C-B
        fdivr  ;                          (C-B)/A
        _PutStr OutMsg
        call   PutFP
        _PutCh 13, 10
        _Exit  0
Linear  ENDP
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example...

- A typical run:

```
C:\Src\Asm>linear
Type A: 2
Type B: 3
Type C: 4
The solution of Ax + B = C is x = 0.5
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Floating Point Exceptions

- We have seen that the floating point numbers have special representations for $\pm\infty$ and NaN.
- Example: What would be the output of the following code?

```
OutMsg DB      '1/0 = $'
        ...
        _Begin
        _PutStr OutMsg
        fldl
        fldz
        fdiv
        call    PutFP
        _PutCh 13, 10
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Results

- The output is:

```
C:\Src\Asm>tstfp
1/0 = ∞
```

- (PutFP displays $\pm\infty$ for all floating point exceptions.
The result of $-1/0$ is:

```
C:\Src\Asm>tstfp
-1/0 = -∞
```

- The code is

```
fld1
fchs ; sign change
fldz
fdiv
```

- The NaN example:

```
fld1
fchs ; sign change
fsqrt
```

- This code returns NaN but the output routine PutFP displays only the infinity sign.

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

FPU Comparing and Branching

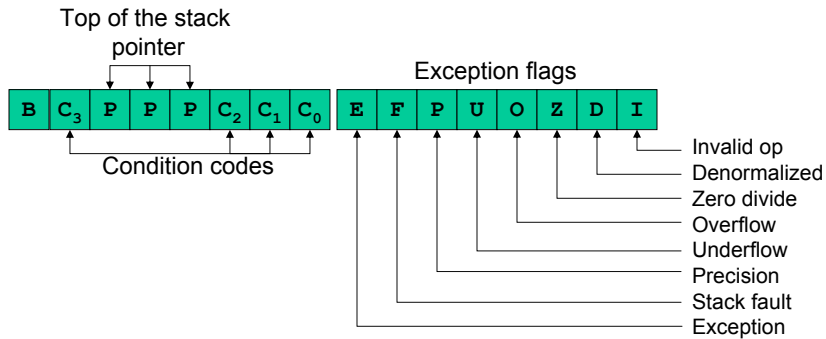
- Problems:
 - The comparing part takes place on the FPU
 - The branching uses flags of the CPU
 - Requires special sequence of instructions
- FPU Compare instructions:
 - All have a pop option

```
fcom[p] mem32 or mem64 ; compare ST to the operand
fcom[p] ST(n) ; compare ST to ST(n)
fcom[p] ; compare ST to ST(1)
fcompp ; compare ST to ST(1) and pop twice
ficom[p] mem16 or mem32 ; compare ST to int operand
ftst ; compare ST to 0.0
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

FPU Status Word

- Contains the status bits of the FPU
- The status word layout is



D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

FPU Condition Codes

After <code>fcom</code>	After <code>ftst</code>	C_3	C_2	C_0
ST > source	ST is positive	0	0	0
ST < source	ST is negative	0	0	1
ST = source	ST is 0	1	0	0
Not comparable	ST is NsN or Inf	1	1	1

- These need to be transferred to the CPU flags
- The instructions for transfer are

`fstsw mem16/ax ;` Store status word to ax or mem
`sahf ;` Transfer ah to the flags register

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

FPU Branching

- The `fstsw/sahf` instruction sequence sets the CPU flag bits for the use of the **unsigned** jumps
- Example: $A = \max(B, C)$

```

                                ST(1)  ST
fld  B;                          B
fld  C;                          B    C
fcom ;                          B    C
fstsw StatWd ;                   B    C    These can be replaced by
mov  ax, StatWd ;                B    C    fstsw  ax (286+)
sahf ;                          B    C
jbe  OrderOK
fxch ;                          C    B
OrderOK:
fstp A;
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

The transfer of the flags

- C_0 overwrites the carry flag
- C_2 overwrites the parity flag
- C_3 overwrites the zero flag
- C_1 overwrites an undefined bit and it cannot be used directly



D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Miscellaneous FPU Instructions

- Content exchange instructions
 - `fxch` exchange contents of `ST` and `ST(1)`
 - `fxch ST(n)` exchange contents of `ST` and `ST(n)`
- The `fwait` Instruction
 - CPU and FPU may execute instructions in parallel
 - `fwait` is used to avoid processing of FPU results before the FPU instructions have completed.
 - `fstsw` is the most common situation that requires `fwait`.
 - `fstsw` issues `fwait` automatically.
 - Other occurrences: floating-point to ASCII Conversion.

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Transcendental Functions

- FPU has instructions to compute
 - `sin`, `cos`, `sincos`, and `arctan` of numbers on the stack
 - One can compute all of the trig functions
 - 2^x and $\log_2 x$ are also available
 - That allow computation of all logarithmic and exponential functions

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example

```
.DATA
A    REAL4    10.35    ;Sides of the rectangle
B    REAL4    13.07    ;
D    REAL4    12.93    ; Diameter of the circle

.CODE
TstFP3 PROC
    _Begin
;Calculate the area of the rectangle
fld    A
fmul   B
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example

```
;Calculate the area of the circle
fldl                    ;Load 1 and
fadd    st, st          ; double it to get 2
fdivr   D               ; D/2
fmul    st, st          ; (D/2)^2
fldpi                   ; Load pi
fmul    st, st          ; A = Pi* (D/2)^2
```

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Example

```
fcompp      ; Compare and discard both
fstsw      ax      ;
sahf       ; Transfer FPU flags to the CPU
jp  NoComp
jz  same
jc  Rect
jmp  Circ
NoComp:
    ...
Same:
    ...
Rect:
    ...
Circ:
```