

Introduction to 8086 Assembly

Lecture 20

x86 floating point

Floating point operations



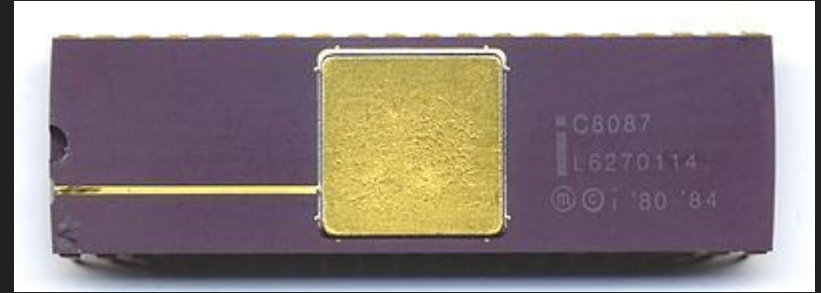
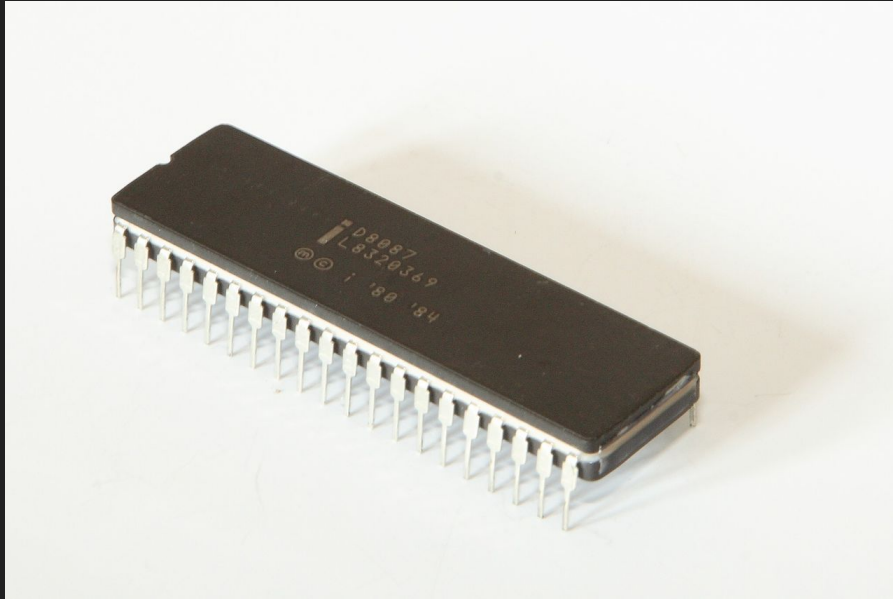
K. N. Toosi
University of Technology

- Floating point libraries
- Math coprocessor (floating point coprocessor, floating point unit FPU)
 - add-on
 - Integrated

Intel 8087 coprocessor



K. N. Toosi
University of Technology

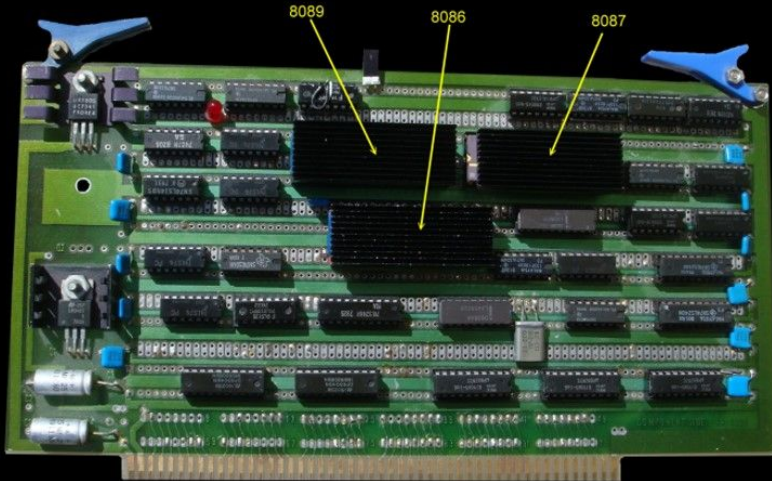


https://en.wikipedia.org/wiki/Intel_8087

Intel 8087 coprocessor



K. N. Toosi
University of Technology



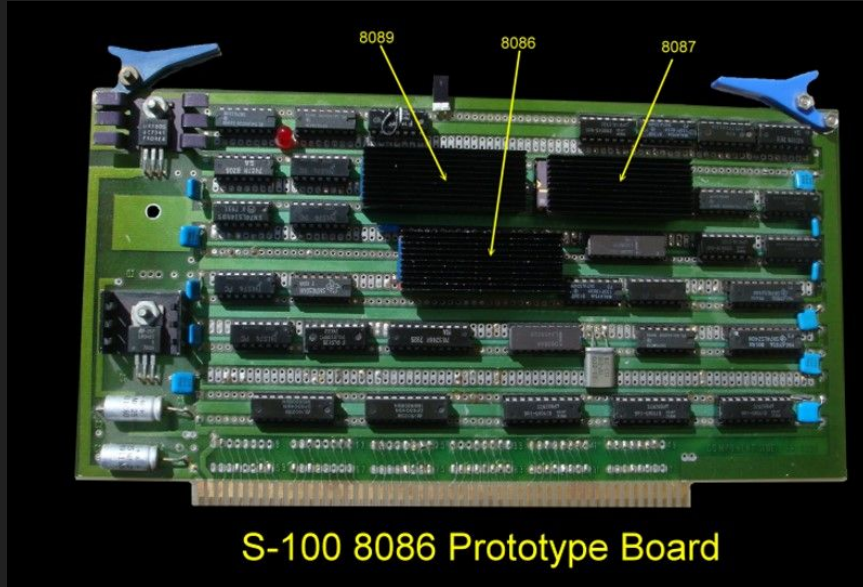
S-100 8086 Prototype Board

[http://www.s100computers.com/My%20System%20Pages/
8086%20Board/8086%20CPU%20Board.htm](http://www.s100computers.com/My%20System%20Pages/8086%20Board/8086%20CPU%20Board.htm)

Intel 8087 coprocessor



K. N. Toosi
University of Technology



S-100 8086 Prototype Board



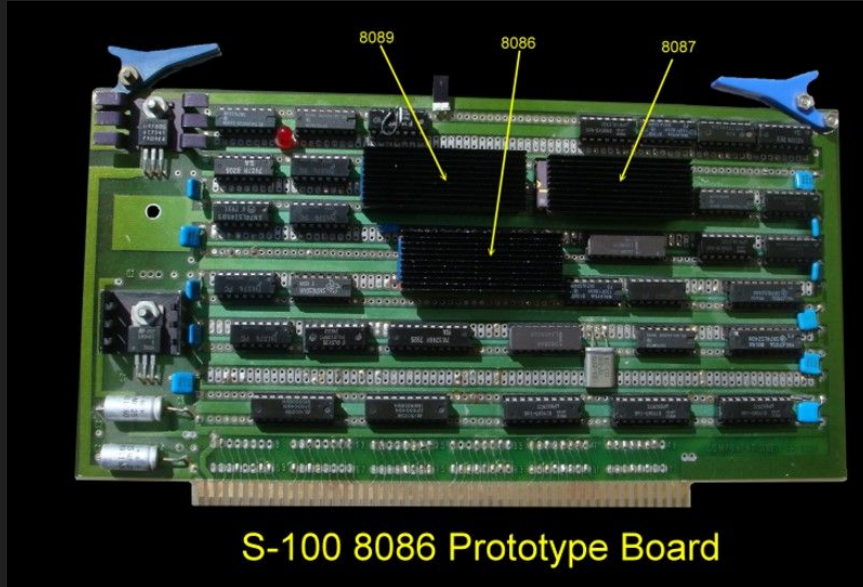
<http://7review.com/remembering-maths-coprocessor/>

<http://www.s100computers.com/My%20System%20Pages/8086%20Board/8086%20CPU%20Board.htm>

Intel 8087 coprocessor



K. N. Toosi
University of Technology



S-100 8086 Prototype Board



<http://7review.com/remembering-maths-coprocessor/>

<http://www.s100computers.com/My%20System%20Pages/8086%20Board/8086%20CPU%20Board.htm>

Intel 8087 coprocessor



K. N. Toosi
University of Technology



https://en.wikipedia.org/wiki/Intel_80486SX

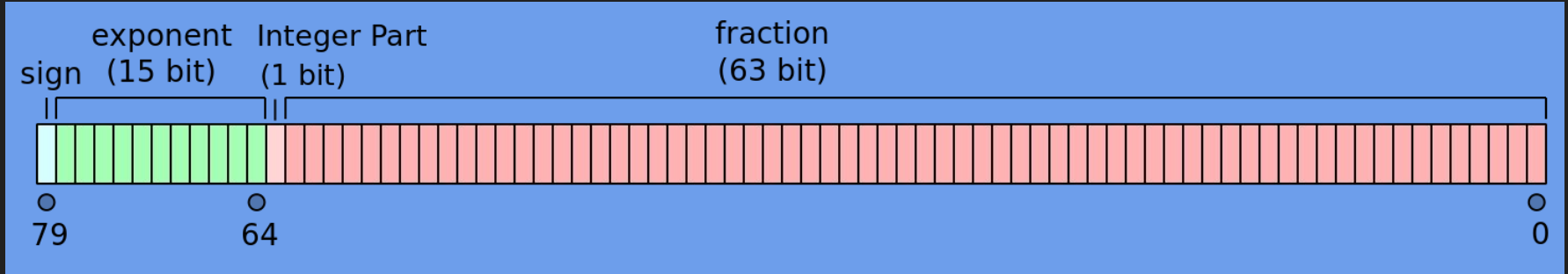


https://en.wikipedia.org/wiki/Intel_80486

Intel x87 extended precision



K. N. Toosi
University of Technology



https://en.wikipedia.org/wiki/Extended_precision

8087 register stack



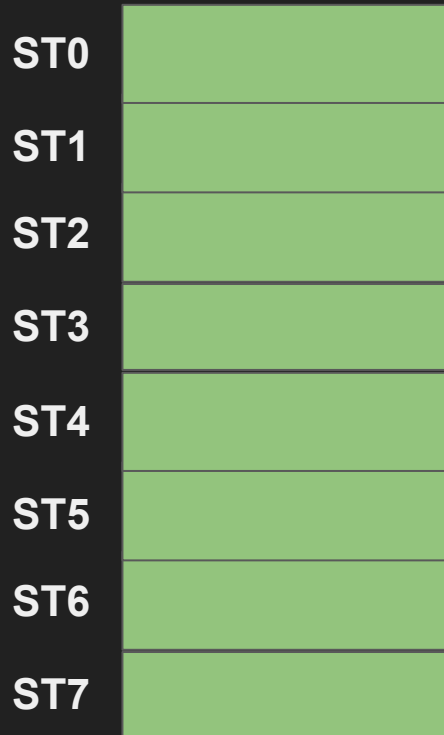
K. N. Toosi
University of Technology

- ST0, ST1, ST2, ST3, ST4, ST5, ST6, ST7

8087 register stack



K. N. Toosi
University of Technology



8087 register stack



K. N. Toosi
University of Technology

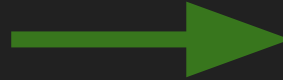
ST0	48.88
ST1	12.04
ST2	111.1
ST3	40.9
ST4	0.003
ST5	12.0
ST6	6.8
ST7	4.2

8087 register stack



ST0	48.88
ST1	12.04
ST2	111.1
ST3	40.9
ST4	0.003
ST5	12.0
ST6	6.8
ST7	4.2

push 72.0



ST0	72.0
ST1	48.88
ST2	12.04
ST3	111.1
ST4	40.9
ST5	0.003
ST6	12.0
ST7	6.8

8087 register stack



K. N. Toosi
University of Technology



Floating points in memory



K. N. Toosi
University of Technology

```
segment .data
l1:      dd  1.0, 18.9, 0.000001, 1e-14, 12.0
l2:      dq  1.0, 18.9, 0.000001, 1e-14, 12.0
```

single precision



double precision



Floating points in memory



K. N. Toosi
University of Technology

32 bit integer



```
l1:    dd    1.0, 18.9, 0.000001, 1e-14, 12
l2:    dq    1.0, 18.9, 0.000001, 1e-14, 12.0
```

double precision



Example



```
segment .data  
  
l1:    dd  1.0, 18.9, 0.00001, 1e-14, 12.0  
  
l2:    dq  1.0, 18.9, 0.00001, 1e-14, 12.0
```

```
fld dword [l1]           ; ST0 = [l1]  
fld dword [l1+4]        ; ST0 = [l1+4]  
  
fadd ST1                ; ST0 += ST1  
  
fist dword [l1]         ; [l1] = (int) ST0  
  
mov eax, [l1]  
call print_int  
call print_nl
```


Example



```
segment .data  
  
l1:    dd  1.0, 18.9, 0.00001, 1e-14, 12.0  
  
l2:    dq  1.0, 18.9, 0.00001, 1e-14, 12.0
```

```
fld dword [l1]           ; ST0 = [l1]  
fld dword [l1+4]        ; ST0 = [l1+4]  
  
fadd ST1                ; ST0 += ST1  
  
fist dword [l1]         ; [l1] = (int) ST0  
  
mov eax, [l1]  
call print_int  
call print_nl
```

```
b.nasihatkon@kntu:lecture20$ ./run.sh float1  
20
```

Loading (pushing)



K. N. Toosi
University of Technology

<code>FLD mem32 (/mem64/mem80)</code>	<code>push ST0 <- mem32 (so on)</code>
<code>FLD STi</code>	<code>push ST0 <- STi</code> <code>STi: ST0, ST1, ... or ST7</code>
<code>FILD mem16 (/mem32/mem64)</code>	<code>push ST0 <- int2float(mem32)</code>

Example:

```
FLD dword [11]
```

```
FLD qword [12]
```

Loading (pushing) constants



K. N. Toosi
University of Technology

FLD1	<code>push ST0 <- 1.0</code>
FLDZ	<code>push ST0 <- +0.0</code>
FLDPI	<code>push ST0 <- Π (the pi number)</code>
FLDL2T/FLDL2E FLDLG2/FLDLN2	

Storing



K. N. Toosi
University of Technology

<code>FST mem32/mem64</code>	<code>mem32 <- ST0</code>
<code>FST STi</code>	<code>STi <- ST0 (i=0,1,...,7)</code>
<code>FIST mem16/mem32/mem64</code>	<code>mem32 <- float2int(ST0)</code>
<code>FSTP dest</code> <code>FISTP dest</code>	similar to FSTP and FISTP but also pops top of stack

Example:

```
FST dword [11]
```

```
FST qword [12]
```

Exchange



K. N. Toosi
University of Technology

`FXCH STi`

`ST0 <-> STi (i=0,1,...,7)`

Math operations



K. N. Toosi
University of Technology

FADD	src	ST0 += src
FSUB	src	ST0 -= src
FMUL	src	ST0 *= src
FDIV	src	ST0 /= src
		src: STi/mem32/mem64

Math operations



FADD	src	ST0 += src
FSUB	src	ST0 -= src
FMUL	src	ST0 *= src
FDIV	src	ST0 /= src
FSUBR	src	ST0 = src - ST0
FDIVR	src	ST0 = src / ST0
		src: STi/mem32/mem64

Math operations



FADDP	STi	STi += ST0
FSUBP	STi	STi -= ST0
FMULP	STi	STi *= ST0
FDIVP	STi	STi /= ST0
FSUBRP	STi	STi = ST0 - STi
FDIVRP	STi	STi = ST0 / STi
		And pop top of stack STi: ST0, ST1, ... or ST7

Math operations



FADDP FADDP	STi STi, ST0	STi += ST0
FSUBP FSUBP	STi STi, ST0	STi -= ST0
FMULP FMULP	STi STi, ST0	STi *= ST0
FDIVP FDIVP	STi STi, ST0	STi /= ST0
FSUBRP FSUBRP	STi STi, ST0	STi = ST0 - STi
FDIVRP FDIVRP	STi STi, ST0	STi = ST0 / STi
		And pop top of stack

Math operations



FADD	ST _i , ST ₀	ST _i += ST ₀
FSUB	ST _i , ST ₀	ST _i -= ST ₀
FMUL	ST _i , ST ₀	ST _i *= ST ₀
FDIV	ST _i , ST ₀	ST _i /= ST ₀
FSUBR	ST _i , ST ₀	ST _i = ST ₀ - ST _i
FDIVR	ST _i , ST ₀	ST _i = ST ₀ / ST _i
		ST _i : ST ₀ , ST ₁ , ... or ST ₇

Math operations



FIADD src	ST0 += int2float(src)
FISUB src	ST0 -= int2float(src)
FIMUL src	ST0 *= int2float(src)
FIDIV src	ST0 /= int2float(src)
FISUBR src	ST0 = int2float(src) - ST0
FIDIVR src	ST0 = int2float(src) / ST0
	src: mem32/mem64

Example



```
segment .data

l1:    dd  1.0, 18.9, 0.00001, 1e-14, 12.0

l2:    dq  1.0, 18.9, 0.00001, 1e-14, 12.0
```

```
fld dword [l1]           ; ST0 = [l1]
fld dword [l1+4]         ; ST0 = [l1+4]

fadd ST1                 ; ST0 += ST1

fist dword [l1]          ; [l1] = (int) ST0

mov eax, [l1]
call print_int
call print_nl
```

Math operations



K. N. Toosi
University of Technology

FCFS	$ST0 = -ST0$
FABS	$ST0 = ST0 $
FSQRT	$ST0 = \text{sqrt}(ST0)$
FSCALE	$ST0 *= 2^{\text{floor}(ST1)}$
FRNDINT	$ST0 = \text{round}(ST0)$ still floating point

Math operations



K. N. Toosi
University of Technology

FSIN	<code>ST0 = sin(ST0)</code>
FCOS	<code>ST0 = cos(ST0)</code>
FSINCOS	<code>ST0 = cos(ST0)</code> <code>then ST0 <- push sin(ST0)</code>

x87 status register



K. N. Toosi
University of Technology

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	C ₃	TOP			C ₂	C ₁	C ₀	E _S	S _F	P _E	U _E	O _E	Z _E	D _E	I _E

<http://www.russinoff.com/libman/text/node48.html>

Making comparisons



FCOM src	compare ST0, src src: mem32/mem64/STi
FCOMP src	like FCOM src AND pop top of stack
FCOMPP	Compare ST0, ST1 AND pop twice
FICOM mem16/mem32	compare ST0, int2float(src) src: mem32/mem64/STi
FTST	compare ST0, 0

Making comparisons



K. N. Toosi
University of Technology

FSTSW	mem16	
FSTSW	AX	
SASF		FLAGS <- AX (not all bits)
LASF		AX <- FLAGS (not all bits)

Making comparisons



K. N. Toosi
University of Technology

FSTSW mem16	
FSTSW AX	
SASF	FLAGS <- AX (not all bits)
LASF	AX <- FLAGS (not all bits)

FSTSW AX

SASF

follows every comparison

Directly setting EFLAGS



K. N. Toosi
University of Technology

FCOMI STi	compare ST0, STi sets EFLAGS
FCOMIP STi	FCOMI STi pops top of stack

Practice: roots of a quadric



```
extern printf
segment .data
a: dq 1.0
b: dq -3.0
c: dq 2.0
minus4: dq -4.0
temp: dq 0.0
format: db "%f", 10, 0
no_roots_msg: db "No real roots", 10, 0

segment .text
:
```

Practice: roots of a quadric



quadroots.asm

```
extern printf
segment .data
a:    dq    1.0
b:    dq   -3.0
c:    dq    2.0
minus4: dq  -4.0
temp:  dq    0.0
format: db  "%f", 10, 0
no_roots_msg: db "No real roots", 10, 0

segment .text
:
fld  qword [b]
fld  qword [b]
fmulp ST1
```

Practice: roots of a quadric



```
extern printf
segment .data
a:    dq    1.0
b:    dq   -3.0
c:    dq    2.0
minus4: dq  -4.0
temp:  dq    0.0
format: db  "%f", 10, 0
no_roots_msg: db "No real roots",10, 0

segment .text
:
fld qword [b]
fld qword [b]
fmulp ST1

fld qword [a]
fld qword [c]
fld qword [minus4]
fmulp ST1
fmulp ST1
```

```
extern printf
```

```
segment .data
```

```
a: dq 1.0
```

```
b: dq -3.0
```

```
c: dq 2.0
```

```
minus4: dq -4.0
```

```
temp: dq 0.0
```

```
format: db "%f", 10, 0
```

```
no_roots_msg: db "No real roots", 10, 0
```

```
segment .text
```

```
:
```

```
fld qword [b]
```

```
fld qword [b]
```

```
fmulp ST1
```

```
fld qword [a]
```

```
fld qword [c]
```

```
fld qword [minus4]
```

```
fmulp ST1
```

```
fmulp ST1
```

```
faddp ST1
```

```
fldz
```

```
fcomip ST1
```

```
ja no_roots
```

```
quadroots.asm
```



K. N. Toosi
University of Technology

quadroots.asm

```

extern printf
segment .data
a:    dq    1.0
b:    dq   -3.0
c:    dq    2.0
minus4: dq  -4.0
temp:  dq    0.0
format: db "%f", 10, 0
no_roots_msg: db "No real roots", 10, 0
segment .text
:
    fld qword [b]
    fld qword [b]
    fmulp ST1

    fld qword [a]
    fld qword [c]
    fld qword [minus4]
    fmulp ST1
    fmulp ST1
    faddp ST1

    fldz
    fcomip ST1
    ja    no_roots

```

```

fsqrt

fld st0
fld qword [b]
faddp st1
fchs

fld qword [a]
fld1
fld1
faddp
fmulp
fdivp ST1      ; ST1 /= ST0

; print st0
fst qword [temp]
push dword [temp+4]
push dword [temp]
push format
call printf
add esp, 12

fcomp

```

```

fld qword [b]
fchs
faddp

fld qword [a]
fld1
fld1
faddp
fmulp

fdivp ST1      ; ST1 /= ST0

; print st0
fst qword [temp]
push dword [temp+4]
push dword [temp]
push format
call printf
add esp, 12

jmp endl

no_roots:
    mov eax, no_roots_msg
    call print_string

endl:

```


quadroots.asm

```
extern printf
segment .data
a: dq 1.0
b: dq -3.0
c: dq 2.0
minus4: dq -4.0
temp: dq 0.0
format: db "%f", 10, 0
no_roots_msg: db "No real
segment .text
:
fld qword [b]
fld qword [b]
fmulp ST1

fld qword [a]
fld qword [c]
fld qword [minus4]
fmulp ST1
fmulp ST1
faddp ST1

fldz
fcomip ST1
ja no_roots
```

```
fsqrt
fld st0
fld qword [b]
faddp st1
fchs

fld qword [a]
```

```
fld qword [b]
fchs
faddp

fld qword [a]
fld1
fld1
faddp
fmulp
```

This code is inefficient!
Go through the code and make it
more efficient.

```
push qword [temp]
push format
call printf
add esp, 12

fcomp
```

```
; ST1 /= ST0

[temp]
d [temp+4]
d [temp]
at

add esp, 12

jmp endl

no_roots:
mov eax, no_roots_msg
call print_string

endl:
```