Keyword	Full Name	Space Allocated
DB	Define Byte	Allocates 1 byte (8 bits)
DW	Define Word	Allocates 2 bytes (16 bits)
DD	Define Doubleword	Allocates 4 bytes (32 bits)
DQ	Define Quadword	Allocates 8 bytes (64 bits)
DT	Define Ten Bytes	Allocates 10 bytes (80 bits)

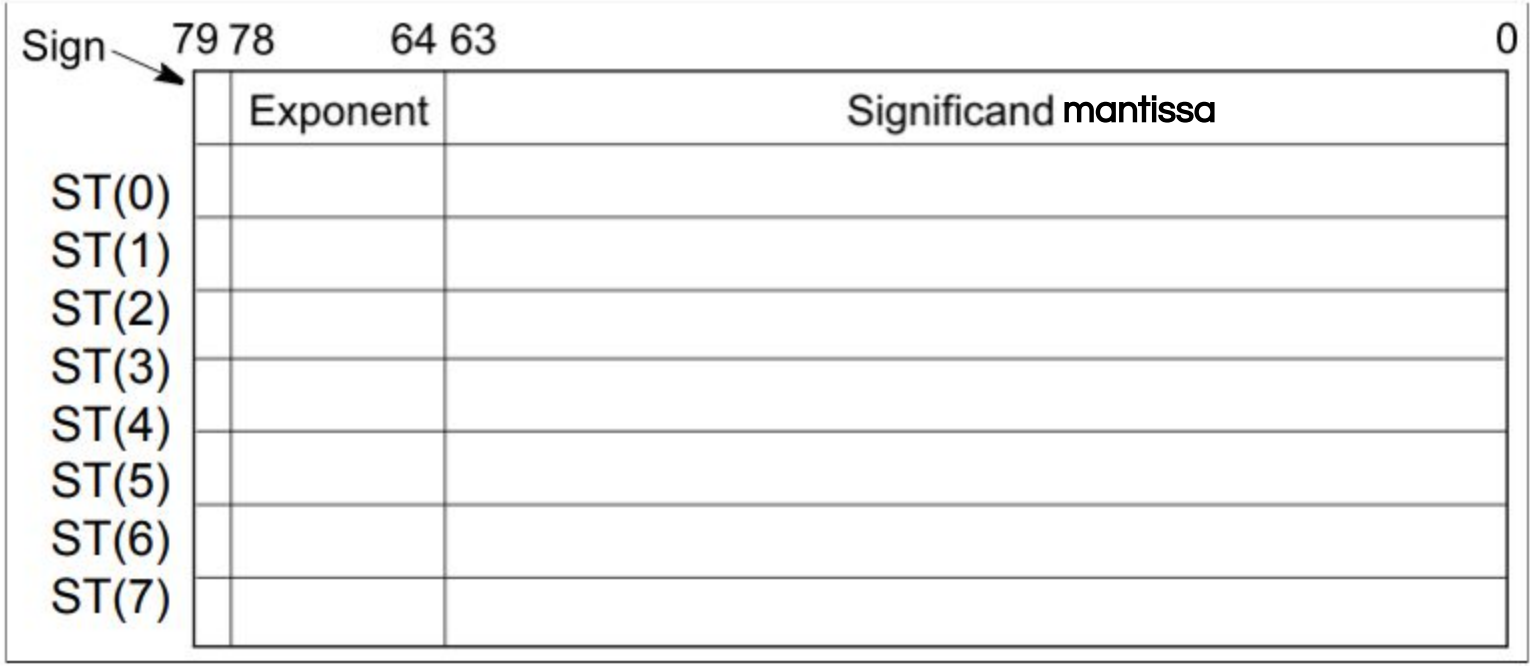
FPU

Os processadores atuais da família x86 possuem uma FPU (Unidade de Ponto Flutuante) dedicada para efetuar operações de ponto flutuante

– No hardware, algumas arquiteturas podem compartilhar elementos entre a FPU e a CPU (ex.: Pentium)

- Números em ponto flutuante adotam o formato IEEE754

- A FPU possui 8 registradores de dados de 80 bits cada para armazenar números em ponto flutuante


Sign
79 78
64 63
0
Exponent
Significand mantissa
ST(0)
ST(1)
ST(2)
ST(3)
ST(4)
ST(5)
ST(6)
ST(7)

Programação utlizando a FPU

Utilizando Operandos Implícitos (Formato pilha)

- A forma clássica de endereçamento acessa os registradores via o formato “pilha”.

- Os operandos não são especificados. O (primeiro) operando é o registrador **ST(0)** – fonte – e o segundo operando (se houver) é o registrador **ST(1)**.

Movimentação

Instrução FLD

• Carrega um número de 32, 64 ou 80 bits na pilha.

• De fato, FLD, inicialmente decrementa o TOS (bits 11,12 e 13 do registrador de status) e posteriormente armazena a informação no registrador apontado por TOS.

Esta instrução converte o número do formato de memória para o formato de ponto flutuante.

Instrução FST e FSTP

- Copia valor do topo da pilha em um registrador ST(i) ou posição de memória
- A instrução FSTP desempilha o valor do topo da pilha

```
section .data

n1              dq      4.5
n2              dq      3.2
result          dq      0.0
s:              db      "%f"

section .text
global main
extern printf
main:
        push    rbp
        mov     rbp, rsp

        fld     qword [n1]
        fld     qword [n2]
        fadd    st0, st1
        fstp    qword [result]

        mov     rdi, s
        movq    xmm0, [result]
        call    printf

        leave
        mov     rax,0
        ret

```

x86-64 Reference Sheet

David Broman, KTH Royal Institute of Technology
Version 0.2, December 1, 2020

Calling Conventions

- Integer arguments: rdi, rsi, rdx, rcx, r8, r9
- Floating-point arguments: xmm0 - xmm7
- Additional arguments pushed on stack, right to left, removed by caller
- Callee saved registers: rbp, rbx, r12, r13, r14, r15
- Integer return register(s): rax or rdx:rax
- Floating-point number return register(s): xmm0 or xmm1:xmm0

Registradores de ponto flutuante

Arquitetura x86-64

Código de ponto flutuante baseado no conjunto de registradores e operações
SSE (Streaming SIMD Extensions)

- SSE é um processo ou tecnologia que habilita instrução simples de múltiplos dados. Processadores mais antigos processam apenas um único elemento de dados por instrução.
- A SSE habilita a instrução de gerenciar vários elementos de dados.
- Ela é utilizada em aplicações de uso intensivo, como gráficos 3D, para um processamento mais rápido.
- SSE2, SSE3S e SSE4.

Fonte: https://www.intel.com.br/content/www/br/pt/support/articles/000005779/processors.html

Extensão SSE

SSE3: 16 registradores (%xmm0 a %xmm15) de 128 bits

https://docs.oracle.com/cd/E19120-01/open.solaris/817-5477/eojde/index.html

Chamadas de procedimentos

Até 8 argumentos de ponto flutuante passados em registradores (%xmm0 a %xmm7)

Todos os registradores %xmm podem ser sobrescritos pela função chamada!

Valor de retorno de ponto flutuante em **%xmm0**

Operações básicas

registrador , registrador
memória , registrador

mul**sd** xmm0, xmm1
div**sd** xmm0, xmm1
mul**sd** xmm0, xmm1
sub**sd** xmm0, xmm1

s- float
d-double

```
section .data
    n1      dq 9.5
    n2      dq 2.0

    fmt     dq  "%f "

section .text
    global main
    extern printf

    main:
    push    rbp
    mov     rbp, rsp

    mov     rdi, fmt
    movq    xmm0, [n1]
    call    printf

    mov    rdi, fmt
    movq   xmm0, [n2]
    call   printf

    leave
    ret
```

```
section .data
    n1      dq 9.5
    n2      dq 2.0

    fmt     dq "%f "

section .text
    global main
    extern printf

    main:
    push    rbp
    mov     rbp, rsp

    mov     rdi, fmt
    movq    xmm0, [n1]
    call    printf

    mov    rdi, fmt
    movq   xmm0, [n2]
    call   printf

    leave
    ret
```

```
section .data
    n1      dq 9.5
    n2      dq 2.0

    fmt     dq "%f "

section .text
    global main
    extern printf

    main:
    push    rbp
    mov     rbp, rsp

    mov     rdi, fmt
    movq    xmm0, [n1]
    call    printf

    mov     rdi, fmt
    movq    xmm0, [n2]
    call    printf

    leave
    ret
```

Somando dois valores

```
1  section .data
2      n1      dq 9.5
3      n2      dq 2.3
4
5
6      fmt     dq  "%f"
7
8  section .text
9      global main
10     extern printf
11
12     main:
13     push    rbp
14     mov     rbp, rsp
15
16
17     movq  xmm0, [n1]
18     movq  xmm1, [n2]
19     addsd xmm0, xmm1
20
21     mov     rdi, fmt
22     call    printf
23
24     leave
25     ret
26
```