MATA49
Programação de Software Básico

Unidade de ponto flutuante

Leandro Andrade
leandrojsa@ufba.br

Unidade de ponto flutuante

- Os números não inteiros também podem ser representados em bases binárias
  - Por exemplo: podemos representar em decimal:

    $0{,}123 = 1 \times 10^{-1} + 2 \times 10^{-2} + 3 \times 10^{-3}$

    Já em binário:

    $0{,}101_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 0{,}625$

Unidade de ponto flutuante

- Representação binária de ponto flutuante:
  - $0{,}101_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 0{,}625$
  - $0{,}101 = 1 \times 1/2 + 0 \times 1/4 + 1 \times 1/8 = 0{,}625$
  - $0{,}101 = 0{,}5 + 0{,}125 = 0{,}625$

Unidade de ponto flutuante

- Representação binária de ponto flutuante:

Binary	Decimal Fraction	Decimal Value
.1	1/2	.5
.01	1/4	.25
.001	1/8	.125
.0001	1/16	.0625
.00001	1/32	.03125

Unidade de ponto flutuante

- Representação binária de ponto flutuante:

Binary Floating-Point	Base-10 Fraction
11.11	3 3/4
101.0011	5 3/16
1101.100101	13 37/64
0.00101	5/32
1.011	1 3/8
0.000000000000000000000001	1/8388608

Unidade de ponto flutuante

- Representação binária de ponto flutuante:
  - Para converter 0,85 temos a seguinte situação:

    $0,85 = 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$

    $0,85 = 0,5 + 0,25 + 0,0625 = 0,7875$
    (precisamos de mais bits)

    $0,85 = 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-6} + 1 \times 2^{-7} + 1 \times 2^{-8} \ldots$

    (nunca alcançará o valor exato)
  - O computador armazena um valor aproximado

Ponto flutuante: Representação IEEE

- A IEEE projetou padrão de formatação binária para representar os números com ponto flutuante
- Usado nas principais arquiteturas de processadores
- Dois tipos de formatos de precisão
  - Single: 32 bits
  - Double: 64 bits

Ponto flutuante: Representação IEEE

- O Número é dividido em 3 partes:
  - Sinal
  - Significando
  - Expoente

Ponto flutuante: Representação IEEE

- Sinal
  - 1 = negativo, 0 = positivo
- Significando
  - Dígitos decimais à esquerda e à direita do ponto decimal
  - Exemplo:

    $123.154 = (1 \times 10^{2}) + (2 \times 10^{1}) + (3 \times 10^{0}) + (1 \times 10^{-1})$
    $+ (5 \times 10^{-2}) + (4 \times 10^{-3})$

Ponto flutuante: Representação IEEE

- Expoente
  - Armazena o valor da potência de dois que será aplicado no significando
  - O número binário armazenado é polarizado através da adição de 127
    - Para precisão simples

Ponto flutuante: Representação IEEE

- Expoente:
  - Exemplos:

Exponent (E)	Biased (E + 127)	Binary
+5	132	10000100
0	127	01111111
-10	117	01110101
+127	254	11111110
-126	1	00000001
-1	126	01111110

Ponto flutuante: Representação IEEE

- Normalização
  - A maioria dos números flutuantes são armazenados normalizados para maximizar a precisão do significante
  - A normalização ocorre quando um simples “1” aparece a esquerda do “.”
  - O número de deslocamentos são expressados no expoente
    - Positivo: deslocamento para esquerda
    - Negativo: deslocamento para direita

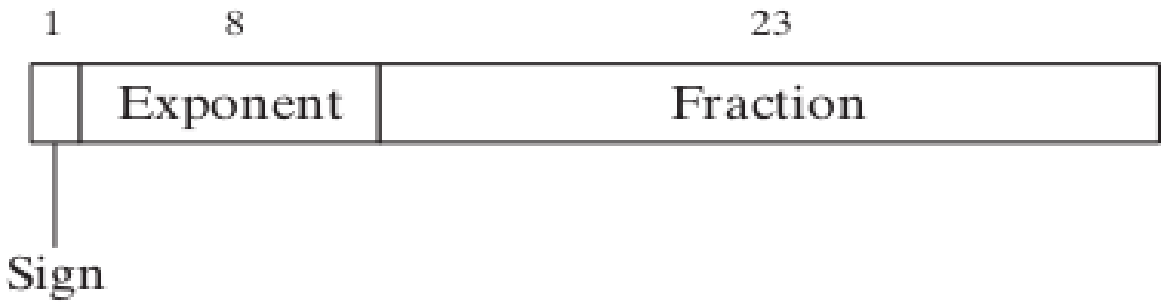
Ponto flutuante: Representação IEEE

- Normalização:
  - Exemplos:

Unnormalized	Normalized
1110.1	1.1101×2^{3}
.000101	1.01×2^{-4}
1010001.	1.010001×2^{6}

Ponto flutuante: Representação IEEE

- Precisão simples:
  - Usa 32 bits para codificar o número



Sinal: bit 1 para negativo, bit 0 para positivo

Ponto flutuante: Representação IEEE

- Precisão simples:
  - Codificação do número flutuante
    - Número são armazenados normalizados

    Exemplo:

    1.101×2^0

    Bit de sinal: 0

    Expoente: 01111111 (127)

    Significando: 10100000000000000000000

Ponto flutuante: Representação IEEE

- Exemplos:

Binary Value	Biased Exponent	Sign, Exponent, Fraction
-1.11	127	1 01111111 11000000000000000000000
+1101.101	130	0 10000010 10110100000000000000000
-.00101	124	1 01111100 01000000000000000000000
+100111.0	132	0 10000100 00111000000000000000000
+.0000001101011	120	0 01111000 10101100000000000000000

Ponto flutuante: Representação IEEE

- Representação dos infinitos:
  - Positivo (+∞):
    - Representa o maior número real positivo possível
  - Negativo (-∞):
    - Representa o menor número real negativo possível

Unidade de ponto flutuante

- O processador Intel 8086 foi projetado para realizar operações aritméticas somente em números inteiros
  - Resolver a aritmética de números decimais via software é bastante custo
    - Por exemplo programas como AutoCad realizam pesados cálculos com números float

Unidade de ponto flutuante

- A Intel integrou na sua arquitetura um coprocessador para operar números flutuantes
  - Foi integrado a CPU principal a partir do Intel 486
  - Chamado de FPU (Floating-Point Unit)

Unidade de ponto flutuante

- A FPU não usa os registradores de uso geral como eax, ebx, etc;
- Em contrapartida existe uma série de registradores para esse tipo de operação chamados registradores de pilha
  - Usados para carregar, armazenar e calcular números float

Unidade de ponto flutuante

- Aplica as operações através da notação posfixa

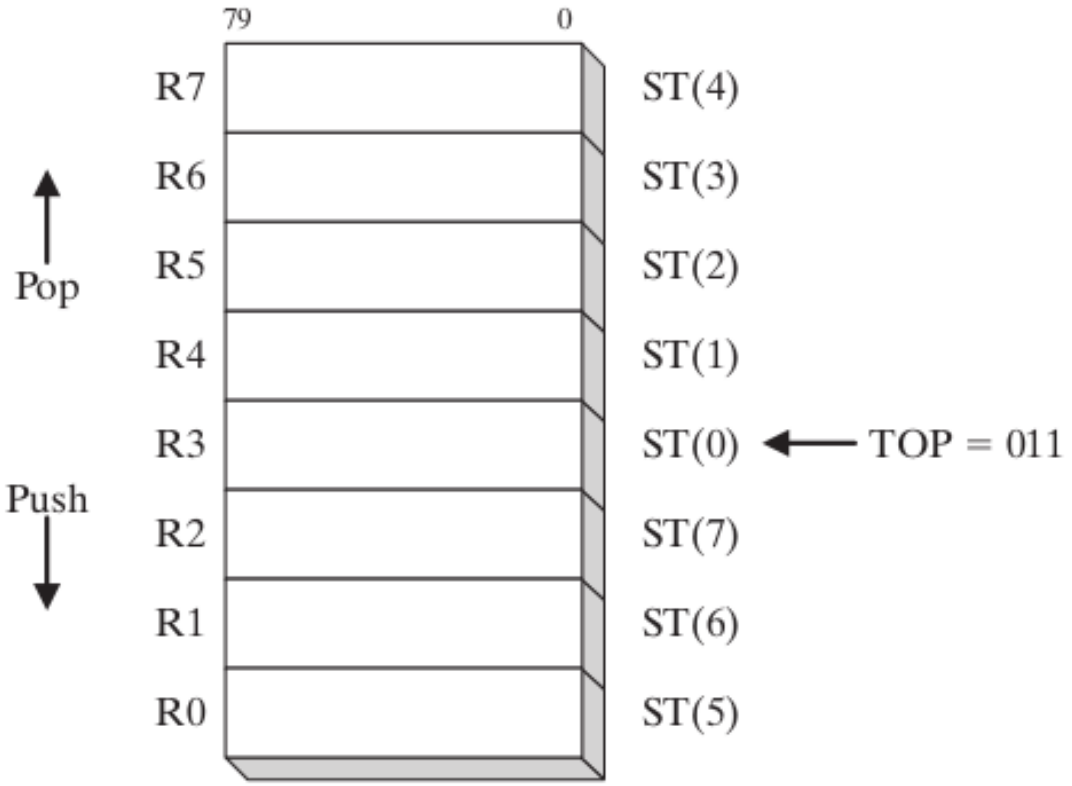
Infix	Postfix
A + B	A B +
(A - B) / D	A B - D /
(A + B) * (C + D)	A B + C D + *
((A + B) / C) * (E - F)	A B + C / E F - *

Unidade de ponto flutuante

- Registradores de dados da FPU
  - 8 registradores de 80 bits (ST0,..., ST7) interligados em uma pilha
  - Endereçados individualmente (R0,...,R7)
  - Um campo chamado TOP (de 3 bits) aponta para o topo da pilha
    - Operação para empilhar reduz em 1 o valor de TOP
    - Operação para desempilhar incrementa em 1 o valor de TOP

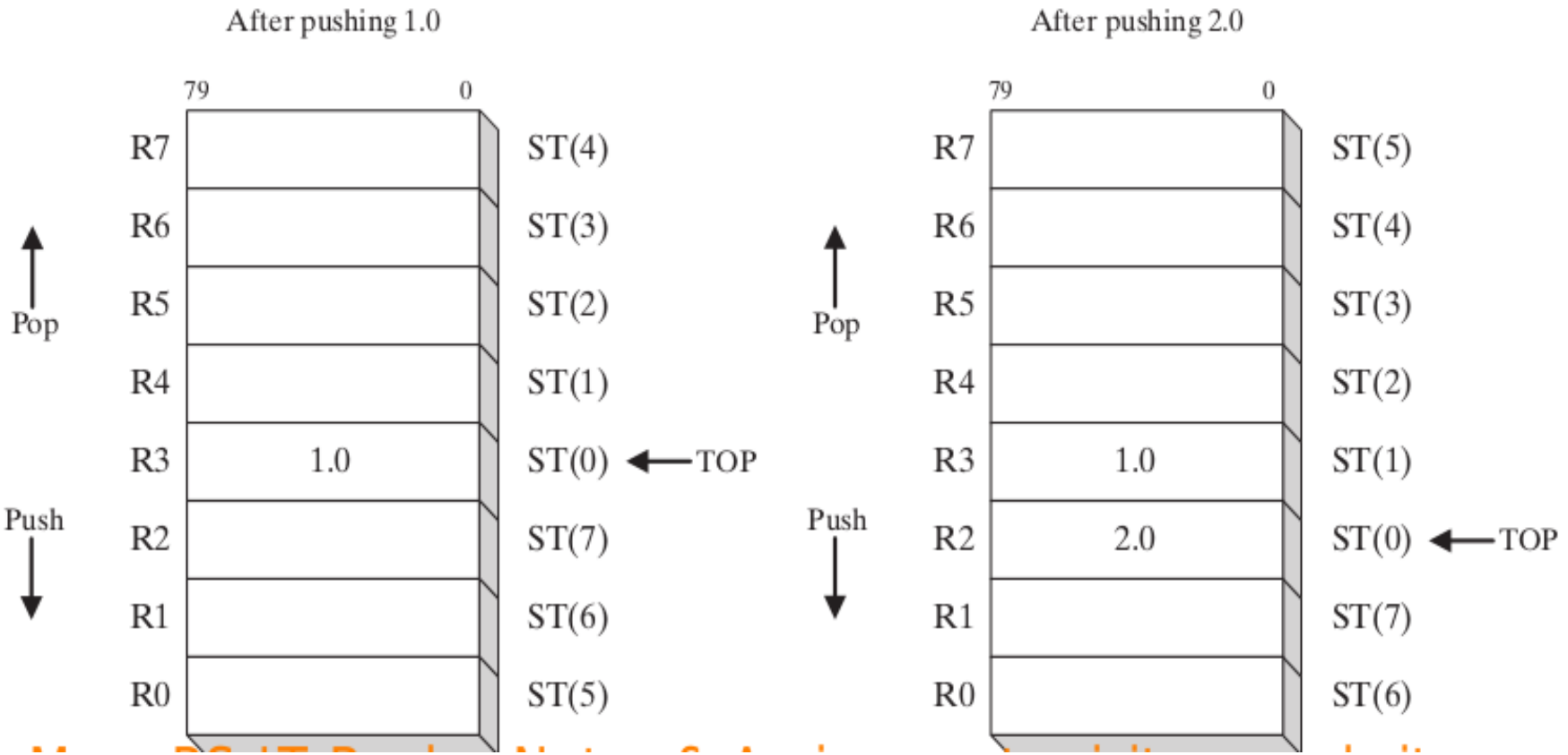
Unidade de ponto flutuante

- Registradores de dados da FPU



Unidade de ponto flutuante

- Registradores de dados da FPU



Instrução FLD

- Carrega o número float passado como parâmetro para o topo da pilha da FPU.
- Sintaxe: FLD <fonte>
  - O operando deve ser uma posição de memória ou registradores da FPU

Instrução FILD

- Lê um inteiro da fonte e converte para float armazenando no topo da pilha da FPU
- Sintaxe: FILD <fonte>
  - O operando deve ser uma posição de memória

Outras instruções de armazenamento

- FLD1
  - Armazena o 1.0 em float no topo da pilha da FPU
  - Sintaxe: FLD1
    - Sem operando
- FLDZ
  - Armazena o Zero no topo da pilha da FPU
  - Sintaxe: FLDZ
    - Sem operando

Instrução FST

- Lê o valor do topo da pilha do FPU e armazena no operando passada como parâmetro
- O valor **não** é retirado do topo pilha
- Sintaxe: FST <destino>
  - Operando memória
  - Ou registradores da FPU

Instrução FSTP

- Lê o valor do topo da pilha do FPU e armazena no operando passada como parâmetro
- O valor é retirado do topo pilha **(desempilha)**
- Sintaxe: FSTP <destino>
  - Operando memória
  - Ou registradores da FPU

Instrução FIST

- Lê o valor do topo da pilha e armazena seu valor inteiro correspondente no operando passado por parâmetro
- Não desempilha o valor do topo
- A aproximação é feita para o inteiro mais próximo
  - Isso pode ser modificado
- Sintaxe: FIST <destino>
  - Memória ou registradores da FPU

Instrução FISTP

- Tem o mesmo funcionamento da instrução FIST porém desempilha o valor do topo da pilha
- Só permite parâmetro de 64bits (quad word)

Instrução FFREE

- Limpa a a pilha
- Torna a pilha nula fazendo que seja equivalente a uma pilha não usada ou vazia

Instruções de adição e subtração

- Por padrão as instruções de adição e subtração operam sobre o valor armazenado do registrador ST0 com outro passado por parâmetro

Instruções de adição e subtração

- FADD
  - Soma
  - Forma1:
    - Sintaxe: FADD <fonte>
      - ST0 = ST0 + <fonte>
      - A fonte pode ser uma memória ou um registrador FPU
  - Forma2:
    - Sintaxe FADD <destino>, ST0
      - <destino> = <destino> + ST0
      - Destino só pode ser um registrador FPU

Instruções de adição e subtração

- Instrução FADDP
  - Faz o mesmo que a FADD porém desempilha a FPU
  - Sintaxe: FADDP <destino> ou FADDP <destino>, ST0
    - <destino> = <destino> + ST0
- Instrução FIADD
  - Adiciona um inteiro ao valor do ST0
  - Sintaxe: FIADD <fonte>
    - ST0 = ST0 + <fonte>
    - Operando do tipo memória

Instruções de adição e subtração

```
1  segment .bss
2  array         resq SIZE
3  sum           resq 1
4
5  segment .text
6        mov     ecx, SIZE
7        mov     esi, array
8        fldz                    ; ST0 = 0
9  lp:
10       fadd    qword [esi]     ; ST0 += *(esi)
11       add     esi, 8          ; move to next double
12       loop    lp
13       fstp    qword sum       ; store result into sum
```

Instruções de adição e subtração

- Instrução FSUB
  - Similar ao FADD, porém operando uma subtração
  - Sintaxe:
    - Forma1:
      - FSUB <fonte>
      - ST0 = ST0 - <fonte>
    - Forma2:
      - FSUB <destino>, ST0
      - <destino> = <destino> - ST0
    - A fonte pode de ser memória ou registrador float

Instruções de adição e subtração

- Instrução FSUBR
  - Realiza: Operando menos ST0
  - Sintaxe:
    - Forma1:
      - FSUBR <fonte>
      - ST0 = <fonte> - ST0
    - Forma2:
      - FSUBR <destino>, ST0
      - <destino> = ST0 - <destino>

Instruções de adição e subtração

- Instrução FSUBP
  - Faz o mesmo que a FSUB porém desempilha a FPU
  - Sintaxe: FSUBP <destino> ou FSUBP <destino>, ST0
    - <destino> = <destino> - ST0
- Instrução FISUB
  - Subtrai um inteiro ao valor do ST0
  - Sintaxe: FISUB <fonte>
    - ST0 = ST0 - <fonte>
    - Operando do tipo memória

Instruções de adição e subtração

- Instrução FISUBR
  - Subtrai do valor de ST0 um número inteiro
  - Sintaxe: FISUBR <fonte>
    - ST0 = <fonte> - ST0
    - Operando do tipo memória

Instruções de multiplicação e divisão

- FMUL
  - Multiplicação
  - Forma1:
    - Sintaxe: FMUL <fonte>
      - ST0 = ST0 * <fonte>
      - A fonte pode ser uma memória ou um registrador FPU
  - Forma2:
    - Sintaxe FMUL <destino>, ST0
      - <destino> = <destino> * ST0
      - Destino só pode ser um registrador FPU

Instruções de multiplicação e divisão

- Instrução FMULP
  - Faz o mesmo que a FMUL porém desempilha a FPU
  - Sintaxe: FMULP <destino> ou FMULP <destino>, ST0
    - <destino> = <destino> * ST0
- Instrução FIMUL
  - Multiplica um inteiro ao valor do ST0
  - Sintaxe: FIMUL <fonte>
    - ST0 = ST0 * <fonte>
    - Operando do tipo memória

Instruções de multiplicação e divisão

- Instrução FDIV
  - Similar ao FMUL, porém operando uma divisão
  - Sintaxe:
    - Forma1:
      - FDIV <fonte>
      - ST0 = ST0 / <fonte>
    - Forma2:
      - FDIV <destino>, ST0
      - <destino> = <destino> / ST0
    - A fonte pode de ser memória ou registrador float

Instruções de multiplicação e divisão

- Instrução FDIVR
  - Realiza: Operando dividido ST0
  - Sintaxe:
    - Forma1:
      - FDIVR <fonte>
      - ST0 = <fonte> / ST0
    - Forma2:
      - FDIVR <destino>, ST0
      - <destino> = ST0 / <destino>

Instruções de multiplicação e divisão

- Instrução FDIVP
  - Faz o mesmo que a FDIV porém desempilha a FPU
  - Sintaxe: FDIVP <destino> ou FDIVP <destino>, ST0
    - <destino> = <destino> / ST0
- Instrução FIDIV
  - Divide um inteiro ao valor do ST0
  - Sintaxe: FIDIV <fonte>
    - ST0 = ST0 / <fonte>
    - Operando do tipo memória

Instruções de multiplicação e divisão

- Instrução FIDIVR
  - Divide do valor de ST0 um número inteiro
  - Sintaxe: FIDIVR <fonte>
    - ST0 = <fonte> / ST0
    - Operando do tipo memória

Comparações

- As instruções de comparações com elementos float não modifica diretamente o registrador FLAGS
- Elas modificam C0 , C1 , C2 e C3, que são FLAGS da FPU
- Para usar saltos condicionais é necessário mover os valores desses dados para o FLAGS

Comparações

- Instrução FSTSW
  - Copia os status do FPU (C0, C1...) para um destino
  - Sintaxe: FSTSW <destino>
    - Destino deve ser do tipo WORD
- Instruções SAHF e LAHF
  - SAHF: copia o valor do AH para o registrador FLAGS
  - LAHF: Carrega o valor do FLAGS no AH

Comparações

- Instrução FCOM
  - Compara ST0 com o operando passado como parâmetro
  - Sintaxe FCOM <fonte>
    - <fonte> memória ou registrador da FPU
- Instrução FCOMP
  - Idêntica a FCOM porém desempilha o topo da pilha de registradores
- Instrução FCOMPP
  - Compara o ST0 e o ST1 e desempilha

Comparações

- Instrução FICOM
  - Compara ST0 com um inteiro passado como parâmetro
  - Sintaxe: FICOM <fonte>
    - Fonte só pode ser memória
- Instrução FICOMP
  - Idêntica a FICOM, com acréscimo desempilha o topo da pilha de registradores

Outras Instruções

- FCHS: Inverte o sinal do ST0
  - ST0 = - ST0
- FABS: Calcula o módulo de ST0
  - ST0 = módulo ST0
- FSQRT: Calcula a raiz quadrada de ST0
- FSCALE
  - Calcula ST0 = ST0 x 2 ST1(parte inteira)