REPRESENTAÇÃO DE NÚMEROS EM VÍRGULA FLUTUANTE

TeSP de Aplicações Móveis

André Martins Pereira

INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

REPRESENTAÇÃO EM VÍRGULA FIXA

- Valores “reais” (fracionários) são representados com um nº de *bits* **fixo** antes e depois da vírgula/ponto:
  - $4.5_{10} = 1*2^2+0*2^1+0*2^0+1*2^{-1}+0*2^{-2} = 100.10_2$
- Problema:
  - Como representar números muito próximos de zero? 4.00001, por exemplo?
    - Muitos bits depois da vírgula/ponto…
    - $4.000001_{10} = 100.???_2$
  - E um número muito grande?
    - $4*2^{365}{}_{10} = 01000000000…0_2$ (“100” seguido de 365 zeros)
- Existe forma mais eficiente de representar estes casos?

REPRESENTAÇÃO EM VÍRGULA FLUTUANTE

- Solução: usar uma norma, uma notação científica
  - 20 000 000 = $2*10^7$,
  - 400 000 000 000 = 4E11
- Representação de valores na seguinte forma: **$V = (-1)^s * M * Radix^E$**
  - RADIX = 2 -> binário ; RADIX = 10 -> decimal
- **S** -> Bit do sinal
  - S = 0 -> V > 0 ; S = 1 -> V < 0
- **M** -> Mantissa (ou parte fracionária **F**)
  - Valor fracionário em binário ($1 \leq M < 2$, ou $0 \leq M < 1$).
- **E** -> Expoente
  - Usado para aumentar a amplitude do valor

FLOATING POINT - NORMALIZAÇÃO

- Notação científica permite representar o mesmo nº de várias formas
  - $43.789*10^{12} = 0.43789*10^{14}$, $43789*10^{9}$
- Objetivo: normalizar!
  - Impedir que o mesmo número tenha representações diferentes
- Um número está normalizado se a Mantissa (*M*) se encontra no intervalo **]Radix , 1]**
  - Ou seja, existe sempre um dígito diferente de 0 à esquerda do ponto decimal
  - $1.4*10^{5}$ -> **Normalizado!**
  - $0.14*10^{6}$-> **Desnormalizado!**
- E em binário? Qual o valor de **M** para que esteja normalizado?
  - **2 > M >= 1**

FLOATING POINT – BIT "ESCONDIDO"

- Valor **normalizado** tem sempre um dígito diferente de zero
  - À esquerda do ponto decimal
- Se um valor é **normalizado**, não faz sentido representar um valor que é sempre igual!
- Só é necessário para efetuar as operações
- Logo, aquando da representação, **não se representa a parte inteira**

FLOATING POINT – EXPOENTE

- Representação: **Excesso 2^{n-1}-1**
- Porquê?
  - É uma representação contínua
  - Mais fácil o *hardware* comparar grandezas
  - Exemplo: comparar dois números
    - 0 01011011 10110011010010101101101
    - 0 10000100 01101010110010111101101

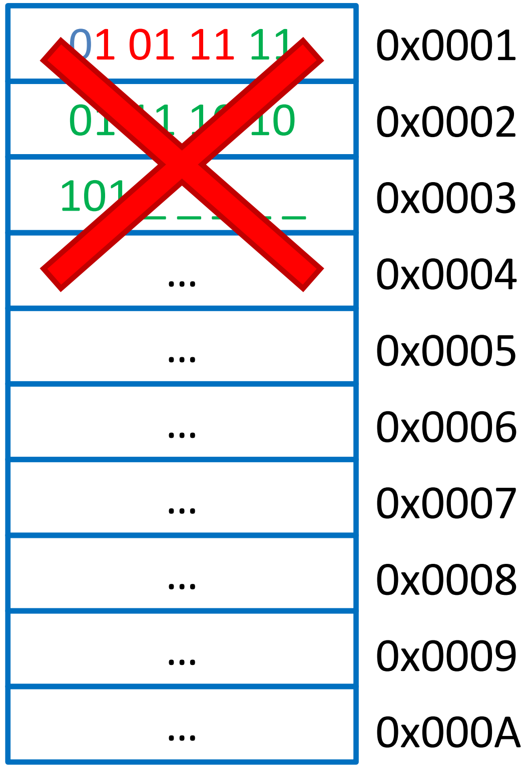
FLOATING POINT - NORMALIZAÇÃO

- Representação normalizada: **$V = (-1)^S * 1.M * 2^E$**
  - E = Exp - Excesso
- **Problema**: Números muito próximos do zero não estão abrangidos!
- **Solução**: Considerar menor valor possível do expoente para representação **APENAS** de valores **desnormalizados**
- Todos as outras representações designam-se por **não normalizadas**

- Representação desnormalizada: **$V = (-1)^S * 0.M * 2^E$**
  - E = - (Excesso - 1)

FLOATING POINT – INTERVALO VALORES REPRESENTÁVEIS

- O objetivo passa **sempre** por:
  - Obter o maior intervalo de representação possível: representar o maior número possível de valores
  - Conseguir melhor precisão: diminuir distância entre 2 valores consecutivos
- Número limitado de *bits* para **M** e **Exp**
  - O que acontece ao aumentar um e outro?
  - Intervalo depende de **Exp**; Precisão depende de **N**
- Número total de *bits* **tem de ser** um múltiplo de 8
  - Uma célula de memória tem… 8 bits!
  - Se nº bits não for múltiplo de 8 vão ser desperdiçados *bits* em memória
  - Exemplo: S -> 1 *bit;* M -> 5 *bits*; Exp -> 13 *bits*



FLOATING POINT – INTERVALO VALORES REPRESENTÁVEIS

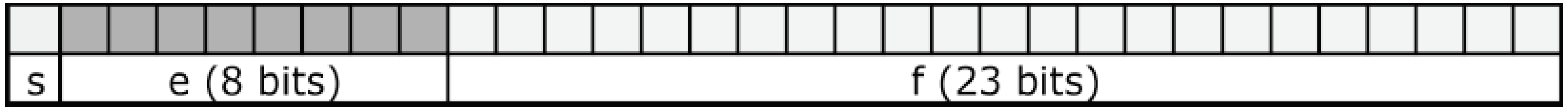
- Número total de *bits* deve ser pelo menos 32
  - Com 16 -> 1 *bit* para sinal, 15 *bits* para **M** + **Exp**
  - 15 *bits* são **insuficientes** apenas para **M**
  - Precisão seria apenas de 4 algarismos
- Assim, com 32 *bits* usamos:
  - **8** para **Exp**: permite representar uma gama da ordem de grandeza dos 10^{39}
  - **23** para **M**: permite uma precisão equivalente a 7 algarismos decimais

INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

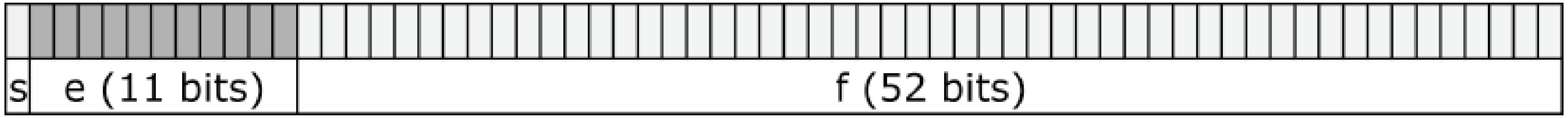
FLOATING POINT – FORMATO BINÁRIO

- Sinal **S**:
  - Ficando mais à esquerda, permite usar o mesmo hardware (que trabalha com valores inteiros) para testar o sinal de um valor em fp;
- Expoente, **E**:
  - Ficando logo a seguir vai permitir fazer comparações quanto à grandeza relativa entre valores absolutos em *fp*: basta comparar os valores como se de valores meramente binários se tratassem;
- Parte fraccionária, **F**:
  - É o campo mais à direita.

Float em C



Double em C



FLOATING POINT – NORMA IEEE 754 (1985)

- Representação com o formato definido até aqui ainda tem imprecisões
    - Várias combinações para representar o mesmo número
    - Como representar valores **desnormalizados**?
    - E valores fora do intervalo permitido com a notação normalizada?
- Norma IEEE 754 define claramente estas imprecisões
- Representação do sinal e parte fracionária
    - Formato definido anteriormente
- Representação do expoente
    - Excesso 127
    - Varia entre -127 e 128

FLOATING POINT – NORMA IEEE 754 (1985)

- Valor decimal de um *fp* em binário (normalizado):
  - $V = (-1)^S * (1.F) * 2^{E-127}$
- Representação de valores desnormalizados
  - $V = (-1)^S * (0.F) * 2^{-126}$
  - Norma IEEE reserva o valor de $E = 0000\ 0000_2$ para representar valores desnormalizados
- Representação do zero
  - $E = 0$ e $F = 0$
- Representação de $\pm\infty$
  - $E = 1111\ 1111_2$ e $F = 0$
- Representação de n.º não real
  - $E = 1111\ 1111_2$ e $F \neq 0$

FLOATING POINT – NORMA IEEE 754 (1985)

Normalized	±	0 < Exp < Max	Any bit pattern
Denormalized	±	0	Any nonzero bit pattern
Zero	±	0	0
Infinity	±	1 1 1…1	0
Not a number	±	1 1 1…1	Any nonzero bit pattern

Sign bit

FLOATING POINT - EXERCÍCIOS

- Pequeno 1
  - $V= (-1)^S * 1.F * 2^{E-7}$
    - Expoente: 4 bits
    - Mantissa: 3 bits
- Pequeno 2
  - $V= (-1)^S * 1.F * 2^{E-3}$
    - Expoente: 3 bits
    - Mantissa: 4 bits

REPRESENTAÇÃO DE NÚMERO EM VÍRGULA FLUTUANTE