Ciências ULisboa | Informática

Arquiteturas de Sistemas Computacionais

Guião de laboratório
Sistemas de Numeração – Vírgula flutuante

1 Exercícios fundamentais

Descarregue o arquivo de suporte a esta aula, `Lab_SN_VFlutuante.tar.gz`, que se encontra no moodle junto a este Guião. Guarde o arquivo no diretório de `Downloads` (no caso de o Linux estar em português, o diretório chama-se `Transferências`).

Execute os seguintes comandos no terminal para criar o diretório de trabalho para esta aula, mover o arquivo comprimido para o diretório pretendido e obter os ficheiros de suporte a este guião:

```
$ mkdir -p ~/2023-24/ASC/07-vflutuante
$ mv ~/Downloads/Lab_SN_VFlutuante.tar.gz ~/2023-24/ASC/07-vflutuante
$ cd ~/2023-24/ASC/07-vflutuante
$ tar xvzf Lab_SN_VFlutuante.tar.gz
$ rm Lab_SN_VFlutuante.tar.gz
```

1.1 Precisão na representação em IEEE 754 de 32 bits usando C

Considere o programa `vf.c`. Este programa permite apresentar um número de vírgula flutuante (p.ex. 36,5) no formato IEEE 754 com 32 bits, tanto em hexadecimal como em binário.

Num terminal, **compile o programa utilizando o compilador de C da Gnu,** `gcc`, com a seguinte linha de comando que compila o código fonte `vf.c` e gera como saída o executável `vf`

```
$ gcc vf.c -o vf
```

Execute-o com a linha de comando (sem espaços em branco e começando por `./`):

```
$ ./vf
```

Anote o resultado. Este coincide com o que esperava?

❑ Sim
❑ Não

Experimente atribuir outros valores a `f.singleP` e verifique os resultados da execução do programa. Sugestões de valores:

- 3E9
- 3E100
- 3E-6
- 3E-100
- Negativos dos valores anteriores

Estude o programa e tente compreendê-lo.

Union na linguagem C
A *union* é uma construção especial do C que permite tratar um conjunto de bytes de diferentes formas, neste caso um conjunto de 4 bytes pode ser acedido como um número em vírgula flutuante representado em IEEE 754 de 32 bits (*float*) ou pode ser acedido como um inteiro de 32 bits o que é útil para se observar o conteúdo dos 4 bytes em binário ou hexadecimal.

1.2 Precisão na aritmética em IEEE 754 de 32 bits usando C

Considere agora uma variação do programa anterior, `precisao32.c`, com as seguintes modificações:

- Definição de duas constantes `BIG` ($1x10^7$) e `TOOBIG` ($1x10^8$)
- Função `main` inicializa com zero a variável `f.singleP`
- Depois repete 10 vezes: a escrita do valor de `f.singleP` e o seu incremento
- Finalmente adiciona `BIG` ao conteúdo de `f.singleP` e escreve o seu valor final

Compile e execute o programa e **valide** os resultados. Estes coincidem com o que esperava?

❑ Sim
❑ Não

Modifique a função `main` do programa anterior de modo a

- Inicializar a variável `f.singleP` com o valor `BIG` (em vez do valor zero),
- Comentar a linha que adiciona `BIG` ao conteúdo de `f.singleP` (para que não seja executada) inserindo, nessa linha, os caracteres `//` antes da instrução respetiva.

Compile e execute o programa e **valide** os resultados. Estes coincidem com o que esperava?

❑ Sim
❑ Não

Modifique novamente a função `main` do programa anterior de modo a

- Inicializar a variável `f.singleP` com o valor `TOOBIG` (em vez do valor `BIG`),

Compile e execute o programa e **valide** os resultados. Estes coincidem com o que esperava?

❑ Sim
❑ Não

Caso não tenha encontrado uma explicação válida, **converta** 1e7 (i.e. 10 milhões) da base 10 para a base 2 no Wolfram Alpha (https://www.wolframalpha.com/) e **observe** o resultado em binário. Faça o mesmo para 1e8 (i.e. 100 milhões).

Recorde que o IEEE 754 com 32 bits utiliza 23 bits para guardar a mantissa normalizada, o que resulta num total de 24 bits com o *hidden bit*.

Agora que já percebeu o que aconteceu na execução anterior, **encontre** o menor número `x` que deve ser utilizado para incrementar a variável `f.singleP` no ciclo de tal forma que todas as linhas escritas na consola voltem a ser diferentes. Por exemplo, `x=128` já faz os incrementos voltarem a funcionar, mas não é o menor valor que consegue esse resultado.

Modifique novamente a função `main` do programa anterior de modo a

- Alterar o incremento da variável `f.singleP` do valor `1` para o valor `x` escolhido.

Compile e execute o programa e valide os resultados.

1.3 Precisão na aritmética em IEEE 754 de 64 bits usando C

Considere uma nova variação dos programas anteriores, `precisao64.c`, a qual utiliza números de vírgula flutuante com precisão dupla (64 bits) para resolver as limitações dos programas anteriores.

Identifique as alterações face ao programa anterior. Note particularmente a utilização de `double` em vez de `float` e, a consequente, alteração do número de repetições do ciclo dentro da função `print_IEEE_64`. Observe também a utilização da sequência especial `%g` (*double*) no `printf`.

Compile e execute o programa e comente o resultado.

1.4 Conversor entre base 10 e IEEE 754

Considere o programa `conversor_vf.c`. Este programa permite converter de decimal para IEEE754 e vice-versa, tanto em 32bits como em 64bits.

Compile o programa `conversor_vf.c`, gerando um executável com o nome `conversor_vf`

O programa `vf.c` usado anteriormente, define o valor a representar diretamente no código do programa. Assim, quando se pretende alterar esse valor é necessário voltar a gerar o executável. No caso do programa `conversor_vf.c`, o **sentido da conversão**, o **tipo de precisão** a usar e o **valor** a converter não se encontram definidos diretamente no código. Estas três configurações são passadas ao programa através da linha de comando na seguinte ordem:

```
conversor_vf sentido_da_conversao tipo_de_precisao valor
```

Os três campos do programa `conversor_vf` são obrigatórios. Cada um deles deve ser substituído por um valor válido. Os valores válidos para cada campo podem ser identificados observando o código.

Em aulas posteriores, iremos estudar este mecanismo de passagem de parâmetros a partir da linha de comando.

Argumentos de comando de linha para um programa
A passagem de valores a um programa através dos argumentos de comando de linha permite controlar um programa a partir do exterior, sem necessidade de alterar o código fonte e consequente recompilação. Para poder aceder a estes valores passados no comando de linha, o programa em C vai usar parâmetros especiais que serão recebidos pela função principal, `main`, que se designam como `argc` e `argv`. `argc` vai conter o número de strings do comando de linha enquanto `argv` dá acesso a cada uma dessas strings, sendo que a 1ª será o próprio executável e as restantes os valores passados por parâmetro.

Exemplo de execução: `./conversor_vf num2bytes single 50.5`

Analise o código do programa para descobrir quais os parâmetros de entrada que o programa pode receber. Experimente as várias possibilidades de entrada

2 Exercícios complementares

Considere o programa `gamavf.c`. Este programa explora a fronteira da zona normalizada mais próxima de zero.

Compile o programa e execute-o. Concorda com a expressão que define o valor de `x` no programa e, consequentemente, com as afirmações apresentadas com a execução do programa?

❑ Sim
❑ Não

Considere o valor de `x` utilizado neste programa. Tome especial atenção à expressão que o definiu em linguagem C. Pode verificar que este é o menor número, positivo, normalizado.

Partindo dessa expressão, **crie** outra que defina `x2` como sendo o menor número representável que seja maior que `x`. Por outra palavras, `x2` deverá ser o segundo menor número, positivo, normalizado.

O programa deverá apresentar os valores correspondentes a `x2` e a `x2-x`.

Função powf
A função powf, é definida na biblioteca matemática de C e importada com a inclusão de `math.h`.
Esta função tem como resultado um valor em vírgula flutuante que resulta de elevar o seu 1º parâmetro ao expoente definido pelo seu 2º parâmetro. Neste caso o valor resultante da utilização desta função será a representação em IEEE 754 de 2^{-126}.