

Conceitos básicos

Programação I

2022/23

Salvador Abreu

spa@uevora.pt

Teresa Gonçalves

tcg@uevora.pt

Departamento de Informática, ECT-UÉ

Sumário

Arquitetura de Computadores

Linguagem C

Valores

Variáveis

Operadores, Operandos e Expressões

Instruções

Comentários



Arquitetura de Computadores

Modelo de von Neumann

Modelo dos computadores

Meados do século XX

Baseada no conceito de "programa armazenado"

Modelo von Neumann

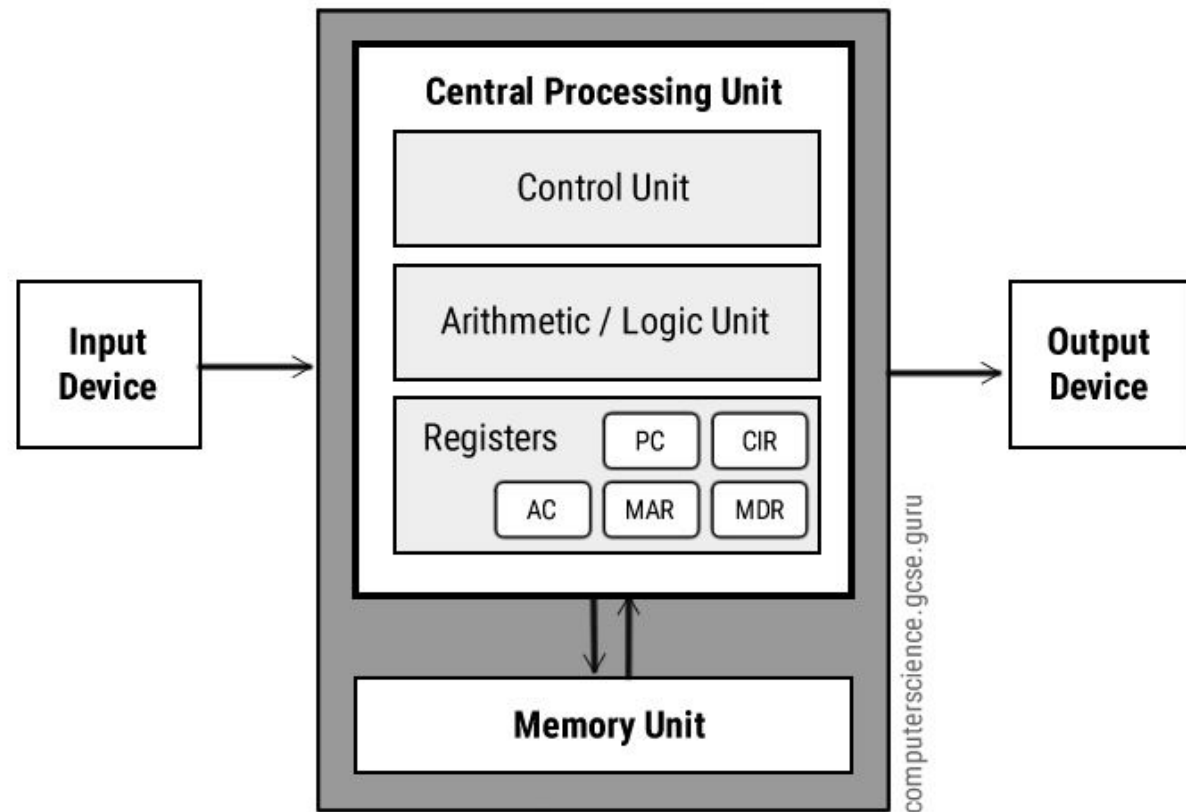
Componentes

Unidade central de processamento (CPU, central processing unit)

Unidade de memória

Dispositivos de armazenamento

Dispositivos de entrada e de saída



CPU – central processing unit

O que é?

Circuito eletrónico responsável pela execução das instruções de um programa

Também conhecido como

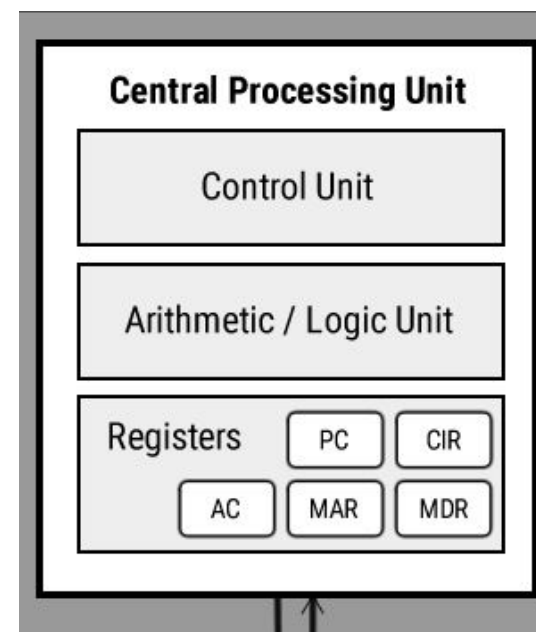
Microprocessador, processador

Contém diversos elementos

Unidade de controlo

Unidade de aritmética e lógica (ALU)

Registos



Elementos da CPU

Unidade de Aritmética e Lógica

Permite a execução de operações

Aritméticas (soma, subtração, etc...)

Lógicas (and, or, not, etc)

Unidade de controlo

Controla a operação da ALU, memória e dispositivos de entrada/saída

Fornece sinais de sincronização e de controlo necessários para os outros componentes

Registos

Áreas de armazenamento locais de alta velocidade

Unidade de memória

Memória principal ou primária (RAM)

Memória rápida e diretamente acessível pelo CPU

Está **dividida** em "fatias" (palavras) com

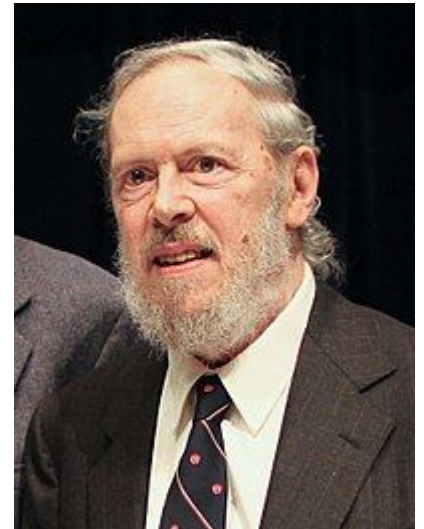
- **tamanho fixo**
 - 8 bits (valores de 0 a 255) ~ **BYTE**
 - 4 ou 8 bytes ~ **palavra** ("word")
- **endereço**
 - todos têm **endereço de BYTE**
(quer seja byte, palavra ou outro)

Linguagem C

História

Informação

Criada em 1972
nos AT&T Bell Labs
por Dennis Ritchie
para o desenvolvimento do SO Unix



Versões

1973: K&R (ver livro)
1989: ANSI C (comissão de normalização)
Atualmente: C18 standard revision (junho 2018)

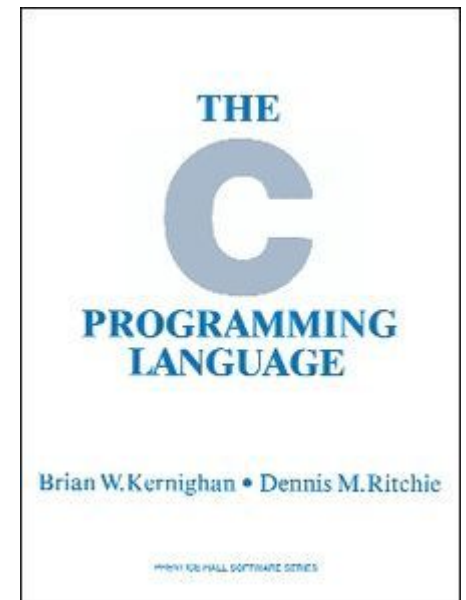
Livro de referência

The C Programming Language

Brian Kernighan e Dennis Ritchie (1978)

serviu durante muitos anos como a especificação informal da linguagem (K&R C)

continua a ser a referência principal para a linguagem C



Valores

Valor

Exemplos

1

-20

3.141519

'c'

Tem um tipo

int

float

char



Variáveis

Variável

Designa um local na memória

Permite armazenar um valor

Referenciado por um nome (identifica uma posição de memória)

Tem um tipo (*)

int

float

char

Nome da variável

Qualquer comprimento

Qualquer identificador válido, exceto

Palavras reservadas

Dígito no início

Case sensitive

Deve

ser **sugestivo**

porque facilita a leitura e compreensão do código

procurar coerência na sua escolha de nomes

Declaração

Antes de utilizar a variável é preciso declará-la

Declaração de variável

Indicar **tipo** e **nome**

Exemplos

```
int num;
```

```
float pi;
```

```
char letra;
```



Afetação

variavel = expressão;

Instrução que associa um valor (direita) à variável (esquerda)

Exemplos

```
num = 5;
```

```
pi = 3.1415926535897931;
```

```
letra = 'u';
```



Operadores, Operandos e Expressões

Operadores, operandos e expressões

Operador

Símbolo que representa uma operação

Operando

Valor/variável à qual se aplicam os operadores

Expressão

Combinação de valores, variáveis e operadores

Operadores aritméticos

- + soma
- subtração
- / divisão
- * multiplicação
- % resto da divisão inteira

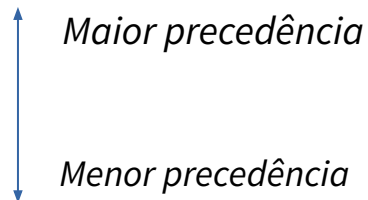
Precedência

Regras que definem a ordem de avaliação das expressões

Parêntesis

Multiplicação e divisão

Soma e subtração



Operadores relacionais

== igual

!= diferente

< menor que

<= menor que ou igual

> maior que


>= maior que ou igual



Operadores lógicos

&&	e
	ou
!	não (negação)

Precedência

!		<i>Maior precedência</i>
&&		
		<i>Menor precedência</i>

Expressão

Uma expressão com operadores numéricos é do **tipo dos seus operandos**

Exemplos

$3+5*6$

$3.16/2.0$

$a-5$



Tipos numéricos

Tipos numéricos compostos têm uma hierarquia

double

float

long

int

short

char

Mistura de tipos «puxa» para cima



Programa e instruções

Instrução e programa

Instrução

Uma unidade de código que é executada

As instruções **terminam com ;**

Exemplos

`a = 3;`

`y = x*a;`

Programa

Uma sequência de instruções

A execução do programa começa pela 1ª instrução e continua com a 2ª, etc

Comentários

`/*` início de comentário

`*/` fim de comentário

ou

`//` comentário até ao fim da linha

São anotações em linguagem natural

ajudam a entender o código fonte



Escrita de dados

printf

Escrita (formatada) de dados no *standard output*

Exemplos

```
printf ("%d %f\n", x, y);
```

Escreve no stdout o valor de x seguido do de y; x é inteiro, y é em vírgula flutuante, e passa à linha seguinte.

```
printf ("número: %d\n", num );
```

Escreve no stdout o texto «número: » seguido do valor da variável num, e passa à linha seguinte.

%d é um especificador de formato



Especificadores de formato

%d **int (em decimal)**

%f **float (em vírgula flutuante, sem expoente)**

%c **char (entendido como carácter)**



Pré-processador

Antes de ser "visto" pelo compilador, o programa em C é pré-processado pelo CPP (C Preprocessor), que faz como que uma "edição de texto" antes de compilar.

Elementos basicos do CPP:

```
#define NOME <definição>
```

```
#define NOME(A1,A2, ...) <definição que use A1, A2, ...>
```

```
#include <ficheiro.h>
```

```
#include "ficheiro.h"
```

e finalmente, o uso das definições feitas com `#define`, i.e. usos de `NOME` ou `NOME(...)` para todos os `NOME` que tiverem sido `#define`-idos

Este uso do CPP está documentado no livro principal (K&R)

Leitura de dados (linha de comando)

O programa principal em C é dado pela **função main**, que se escreve sempre assim:

```
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[]) {
    ...
}
```

Que introduz duas variáveis (**argc** e **argv**) que, quando o programa estiver a ser executado, vão indicar respetivamente:

- quantos argumentos é que foram colocados depois do nome do programa (**argc**)
- quais os valores textuais dos referidos argumentos (**argv[1]**, **argv[2]**, etc...)

As variáveis **argc** e **argv[??]** podem (e devem!) ser usadas no programa.

Fazer programas em C (versão simplificada)

Para construir, compilar e executar programas em C, no âmbito da UC Programação I, deverá

- 1) copiar o fichero defs.h disponibilizado no Moodle
- 2) construir o programa de forma a usar as definições aí contidas

Os programas tomarão então a seguinte forma:

```
#include "defs.h"

MAIN() {
    ...
}
```

Dentro do código da função MAIN poderemos usar as definições **ARG1**, **ARG2**... **ARG4** para designar os textos dos 1º a 4º argumentos colocados na linha de comando, em tempo de execução.

Em geral, podemos usar **ARG(i)** para o i-ésimo argumento.

Conversão de texto para valores numéricos

Para decodificar as representações textuais de valores numéricos, há sobretudo duas funções / operações, definidas na biblioteca <stdlib.h> (já incluída pela instrução **#include "defs.h"**):

atoi converte de texto para inteiro

atof converte de texto para número em vírgula flutuante

Exemplos

`atoi ("123")` dá o número inteiro 123

`atoi (ARG3)` dá o número inteiro indicado como 3º argumento na linha de comando, quando o programa for executado

`atof ("12.34")` dá o número em vírgula flutuante 12.34 (ou aproximado)

Leitura de dados

scanf

Leitura (formatada) de dados do standard input

Exemplo

```
scanf ("%d %f", &inteiro, &real);
```

Lê um inteiro e um valor real do teclado e coloca-o nas variáveis `inteiro` e `real`, respectivamente

`&` é um operador que indica o endereço/posição de memória do que se encontra à sua direita (neste caso, o nome da variável); nessa posição será guardado o valor introduzido

Lembrete

Variável

Designa uma posição de memória; permite armazenar um valor

Identificada por um nome

Afetação

variavel = expressão;

A expressão pode conter valores, operadores e/ou variáveis

nome da variável

À esquerda do = refere a posição de memória (L-value)

À direita do = refere o valor (R-value)

Esquema geral dum Programa

```
#include "defs.h"
```

```
MAIN() {
```

Declarar variáveis

Ler dados

Processar dados

Escrever resultados

```
return 0;
```

```
}
```



Conversão euros → ienes

Dado um valor em euros (EUR), mostrar o correspondente em ienes (JPY).

Considere 1 EUR = 134.5 JPY

Obter o valor em ienes e guardar na variável ienes

Calcular o valor correspondente em euros

Mostrar o valor calculado



Conversão euros → ienes

```
#include "defs.h"
```

```
MAIN() {
```

```
    float euros, ienes;
```

```
    printf ("Qual o valor em euros? ");  
    scanf ("%f", &euros);
```

```
    ienes = euros * 134.5;
```

```
    printf ("%f\n", ienes);
```

```
    return 0;
```

```
}
```



Conversão euros → ienes (versão 2)

```
#include "defs.h"
```

```
MAIN() {
```

```
    float euros, ienes;
```

```
    euros = atof (ARG1);
```

```
    ienes = euros * 134.5;
```

```
    printf ("%f\n", ienes);
```

```
    return 0;
```

```
}
```


Troca de valores

Trocar o valor de duas variáveis x e y

Pedir os valores ao utilizador e colocá-los nas variáveis x e y

Trocar os valores das variáveis

Mostrar que os valores estão trocados



Troca de valores

```
#include "defs.h"
```

```
MAIN() {
```

```
    int x, y;
```

```
    printf ("Introduza os valores inteiros para x e y: ");  
    scanf ("%d %d", &x, &y);
```

```
    // TROCAR VALORES DE x E y
```

```
    printf ("x=%d y=%d", x, y);
```

```
    return 0;
```

```
}
```



Troca de valores (versão 2)

```
#include "defs.h"
```

```
MAIN() {
```

```
    int x, y;
```

```
    x = atoi (ARG1);  
    y = atoi (ARG2);
```

```
    // TROCAR VALORES DE x E y
```

```
    printf ("x=%d y=%d", x, y);
```

```
    return 0;
```

```
}
```



Trocar valores de x e y: tentativa 1

Tentemos o código seguinte:

x = y;

y = x;

Supondo que, inicialmente, temos x=23 e y=45, o resultado seria:

Depois de	x	y	
(início)	23	45	
x = y;	45	45	
y = x;	45	45	

Trocar valores de x e y: tentativa 2

Tentemos o código seguinte:

```
z = x;           // guardar valor de x em temporário
x = y;           // copiar y -> x
y = z;           // copiar temporário (antigo x) -> y
```

De novo, se os valores iniciais forem x=23 e y=45, o resultado será:

Depois de	x	y	z
(início)	23	45	0
z = x;	23	45	23
x = y;	45	45	23
y = z;	45	23	23



Troca de valores (versão 2)

```
#include "defs.h"
```

```
MAIN() {
```

```
    int x, y, z;
```

```
    x = atoi (ARG1);  
    y = atoi (ARG2);
```

```
    z = x;           // TROCAR VALORES DE x E y  
    x = y;  
    y = z;
```

```
    printf ("x=%d y=%d", x, y);
```

```
    return 0;
```

```
}
```

ficheiro defs.h

```
// == defs.h =====  
//  
// Definicoes standard para uso do C em P1  
//  
// Versao de 2022/23  
  
#include <stdio.h>  
#include <stdlib.h>  
  
#define MAIN() int main (int argc, char *argv[])  
  
#define ARG(n) (((n)<=argc)? argv[n]: "inexistente")  
  
#define ARG1 ARG(1)  
#define ARG2 ARG(2)  
#define ARG3 ARG(3)  
#define ARG4 ARG(4)
```