Introdução à Arquitetura de Computadores

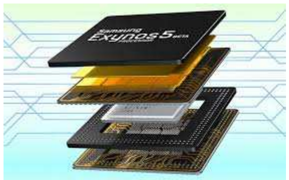
Pedro M. Lavrador

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro
plavrador@ua.pt

Indice

Introdução

- Os microprocessadores são o componente que mais contribuiu para a revolução tecnológica do mundo em que vivemos.



23-02-2017 PML – IAC - 2017 3

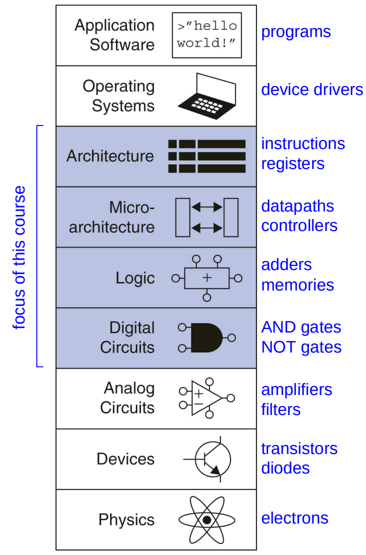
Introdução

- Hardware ou Software?
  - Não vivem isolados.
  - Um é escrito em função do outro.

- Para ser especialista num dos domínios é preciso ter uma visão das capacidades e limitações do outro.

23-02-2017 PML – IAC - 2017 4

Introdução



23-02-2017 PML – IAC - 2017 5

Objetivos da Disciplina

- Conhecer as formas de representação da informação nos computadores digitais, com relevo para a representação da informação numérica e as operações aritméticas básicas.
- Conhecer as operações lógicas e as componentes eletrónicas que as realizam.
- Compreender o funcionamento dos sistemas com memória e o funcionamento dos principais dispositivos de armazenamento de informação.

23-02-2017 PML – IAC - 2017 6

Objetivos da Disciplina

- Compreender a organização interna dos computadores digitais.
- Compreender os mecanismos de comunicação do computador com o exterior.
- Adquirir familiariedade com a arquitectura de processadores através da programação em *assembly.*

Programa

- I – Introdução: sistemas de computação de uso geral. Microprocessadores.
- II – Representação da informação e operações básicas
  - Tipos de dados e sua representação
  - O bit como unidade de informação
  - Aritmética Binária
  - Operações lógicas e Álgebra de Boole
- III – Circuitos Lógicos
  - Portas Lógicas: NOT, OR, NOR, AND, NAND
  - Blocos Combinatórios
  - Dispositivos de Armazenamento de Informação. Registos.
  - Memórias.

Programa

- IV – O Modelo de Von Neumann
  - Estrutura de um processador
- V – A arquitetura MIPS
  - Modelo de programação; tipos de instruções
- VI – A linguagem assembly e o *assembler*
- VII – Organização interna do processador
  - Unidades Operativas e de Controlo.
  - Implementação *Single* ou *Multi Cycle*
- VIII – Comunicação com o exterior: entrada e saída de dados

Bibliografia

- D.M.Harris and S.L.Harris, *Digital Design and Computer* Architecture, 2nd. Edition, Morgan Kaufmann, 2013.

- D.A.Patterson, J.Hennessy, *Computer Organization and Design – the hardware/software interface*, Elsevier.

- Y.N. Patt, S.J.Patel, *Introduction to Computing Systems – from bits & gates to C & beyond*, 2nd edition, McGraw_Hill Education, Indian edition.

Avaliação

- A nota final obtém-se da média ponderada de dois elementos de avaliação:
  - Um teste escrito presencial (T1) com um peso de 50% na nota final, a realizar na aula teórica de **6 de Abril** de 2017
  - Um teste escrito presencial (T2) com um peso de 50% na nota final, na época de exames.

Nota_Final = Nota_T1 x 0.5 + Nota_T2 x 0.5

- **Nota mínima**
  - Em cada um dos dois testes é necessário obter pelo menos 7,0 valores para obter aprovação a IAC.

Regime de Faltas

- Os estudantes que, não usufruindo do estatuto de trabalhador-estudante no corrente ano letivo, faltem injustificadamente a mais de **20% das aulas práticas** ou a mais de 30% das aulas teórico-práticas reprovam automaticamente à disciplina ficando impedidos de se apresentar a qualquer prova da mesma durante o corrente ano letivo.

Princípios Básicos da Arquitetura de Computadores

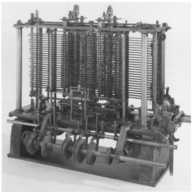
- Abstração:
  - Esconder os detalhes sempre que não são necessários;
- Disciplina:
  - Restringir intencionalmente as liberdades (por exemplo usando tensões digitais e não contínuas)
- Hierarquia
  - Dividir um sistema em módulos e submódulos
- Modularidade
  - Cada um dos módulos tem interfaces bem definidos
- Regularidade
  - Sempre que possível reutilizar os módulos já disponíveis.

A abstração Digital

- A maioria das grandezas físicas são contínuas:
  - A temperatura da sala;
  - Uma frequência de oscilação;
  - A tensão num circuito;
  - A posição de um corpo;
- A abstração digital leva-nos a considerar apenas um subconjunto discreto de todos os valores possíveis.
  - Se considerarmos um conjunto suficientemente grande podemos ter aproximações boas. (vídeo e áudio).

Um pouco de história…

- O primeiro computador…
- … foi projetado por Charles Babbage entre 1834 e 1871.
- Foi o primeiro computador digital, construído com engrenagem mecânicas onde cada “roda” representava um valor discreto entre 0 e 9.
- A máquina consegue calcular 25 dígitos.



23-02-2017 PML – IAC - 2017 15

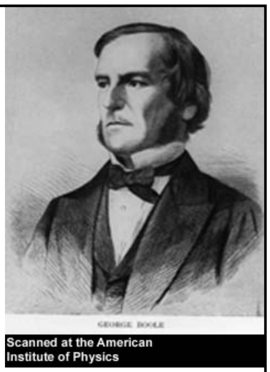
Um pouco de história…

- Ao contrário da máquina de Babbage, a maioria dos computadores atuais usa uma representação binária.
- A quantidade de informação numa variável discreta com N níveis é medida em número de bits de acordo com:

$$D = \log_2 N \text{ bits}$$

23-02-2017 PML – IAC - 2017 16

Um pouco de história



- George Boole (1815-1864)
- Filho de pais operários, aprendeu por si matemática.
- Juntou-se depois ao Queen College na Irlanda.
- Introduziu as variáveis binárias e os três operadores lógicos fundamentais: AND, OR e NOT
- Foi o percursor da lógica binária na qual se baseiam atualmente os sistemas digitais.

23-02-2017 PML – IAC - 2017 17

Indice

23-02-2017 PML – IAC - 2017 18

Sistemas de Numeração

- O que é um número?
  - Número é uma *coleção de unidades*, Tales de Mileto, sec. VI a.c.
  - Número é a *relação entre a quantidade e a unidade*, Isaac Newton. sec. XVII.
- Importa distinguir o número da sua representação.
  - 12 é sempre "uma dúzia"
  - Quer seja, 12, XII, doze, twelve, C_{16}, 14_8, 1100_2, etc

Sistemas de Numeração: Decimal

- Sempre trabalhámos com números decimais
  - (porque temos 10 dedos!)
- Em sistemas digitais (com 0's e 1's) o sistema binário ou hexadecimal é mais conveniente.
- No sistema decimal temos 10 dígitos: 0, 1, 2, …, 9 e o valor de cada um deles depende da posição que ocupa:

1 2 3

centenas dezenas unidades

Sistemas de Numeração: Decimal

- Qual a **gama de representação** que podemos ter num número decimal com N algarismos?
  - Por exemplo:
    - Com 2 algarismos podemos escrever 100 números diferentes: 0, 1, 2, …, 99.
- Com N algarismos podemos escrever 10^N números distintos entre 0 e 10^N-1.
- Ao conjunto de números que podemos escrever com N algarismos chamamos **gama de representação**.

Sistemas de Numeração: Binário

- Os bits representam dois valores possíveis e podem ser agrupados para formar números binários. (em base 2).
- Um número binário com 4 bits na forma:

$$b_3 b_2 b_1 b_0$$

- Tem o valor:

$$b_3 * 2^3 + b_2 * 2^2 + b_1 * 2^1 + b_0 * 2^0$$

Sistemas de Numeração: Binário

- Um número binário geral com N bits na forma:

$$b_{N-1} \ldots b_3 b_2 b_1 b_0$$

- Tem o valor:

$$\sum_{n=0}^{N-1} b_n 2^n$$

- A **gama de representação** de um número binário com N bits é 0, 1, …, 2^N-1.

Sistemas de Numeração: Binário/Decimal

- Os números decimais:

1000's 100's 10's 1's

$5\ 3\ 7\ 4_{10}$

- Os números binários:

8's 4's 2's 1's

$1\ 0\ 1\ 1_2$

Sistemas de Numeração: Binário

- Para trabalhar fluentemente em binário é útil conhecer as potências de 2:

$$
\begin{aligned}
2^0 &= 1\\
2^1 &= 2\\
2^2 &= 4\\
2^3 &= 8\\
2^4 &= 16\\
2^5 &= 32\\
2^6 &= 64\\
2^7 &= 128\\
2^8 &= 256\\
2^9 &= 512\\
2^{10} &= 1024\\
2^{11} &= 2048
\end{aligned}
$$

Sistemas de Numeração: Conversão entre bases

- Converter para Decimal:
  - Fazer a soma de cada digito b_n multiplicado pela potência de 2 correspondente:

$$valor = \sum_{n=0}^{N-1} b_n 2^n$$

- Converter para Binário:
  - Fazer divisões sucessivas por 2 até obter quociente 0.
  - O número em binário é o conjunto dos restos das divisões dispostos pela ordem inversa.

Sistemas de Numeração: Conversão entre bases

- Converter para Decimal:
  - 10011_2
  - 01101_2
- Converter para Binário:
  - 47_{10}
  - 31_{10}

Sistemas de Numeração: Hexadecimal

- Escrever longas sequências binárias é chato e muito suscetível a erros.
- Um grupo de 4 bits, representa uma de 2^4=16 possibilidades.
- Assim é possível e mais cómodo trabalhar num sistema de base 16, chamado hexadecimal.
- Os algarismos hexadecimais são:
  – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Sistemas de Numeração: Hexadecimal

Hexadecimal	Decimal	Binário
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Sistemas de Numeração: Hexadecimal

- Conversão binária para hexadecimal:
  - 1111010_2=
  - 0111101_2=
- Conversão decimal para hexadecimal e binário:
  - $333_{10} = ?_2 = ?_{16}$

Sistemas de Numeração: Jargão

- Um byte é um conjunto de 8 bits:
  - Pode representar 2^8 = 256 valores diferentes.
- Um *nibble* é um meio byte (4 bits):
  - Pode representar 2^4 = 16 valores diferentes.
- Um processador opera sobre conjuntos de bits chamados **palavras** (word):
  - O tamanho da palavra depende da arquitectura. (Atualmente a maioria dos processadores são de 64 bits o que significa que operam sobre palavras de 64 bits.)

Sistemas de Numeração: Jargão

- Dentro de um grupo de bits chama-se **bit menos significativo** (*lsb*) ao da direita e **bit mais significativo** ao mais à esquerda (*msb*).
- Dentro de uma palavra chama-se **byte menos significativo** (*LSB*) ao da direita e **byte mais significativo** ao mais à esquerda (*MSB*).
- O prefixo grego *kilo*, que designa 10^3, é reutilizado em binário para designar 2^{10}=1024.
  - Mega designa 2^{20}, Giga 2^{30}, e assim sucessivamente.

Adição Binária

- A adição binária funciona do mesmo modo que a adição decimal.
  - É até mais simples de efetuar porque a tabela de somas é menor:
    - 0+0 = 0
    - 0+1 = 1+0 = 1
    - 1+1 = 0 e vai 1 para a esquerda (*carry*).

carry

Decimal	Binário
11	11
4277	1011
+5499	+0011
9776	1110

Adição Binária

- Calcule a soma binária:
  - $0110_2 + 0101_2 =$
  - $1001_2 + 0011_2 =$
  - $1101_2 + 0101_2 =$
- Quantos bits são necessários para representar o resultado?
  - Poderá ocorrer *overflow*?

Adição Binária: Overflow

- As máquinas digitais operam com um número fixo de bits.
- Se o resultado da soma exceder o número de bits disponível dizemos que ocorreu *overflow*.
- Por exemplo num sistema de 4 bits $1100_2 + 0110_2$

```
  11
   1100
  +0110
  11010
```

- Não é representável com 4 bits, logo origina um *overflow*.

Indice

Representação de Números com Sinal

- Até aqui apenas vimos a representação de números positivos (ou sem sinal).
- Se quisermos tratar números positivos e/ou negativos precisamos de um sistema de numeração diferente.
- Uma representação em Sinal e Módulo é para nós a mais intuitiva.
  - Porque é a que estamos habituados.
- Nesta representação o bit mais significativo representa o sinal e os restantes o módulo.
- Implicações:
  - Uma vez que bit mais significativo representa o sinal é preciso fixar o número de bits

Representação de Números com Sinal

- Representação sinal e módulo:
  - Um número com N bits em sinal e módulo tem 1 bit (o da esquerda) para representar o sinal e N-1 bits para representar o módulo.
- Qual a representação de +3 e -3, em sinal e módulo com 4 bits?
  - +3 = 0011 -3 = 1011
- Qual o valor mínimo é máximo que se pode representar em sinal e módulo com N bits?
  - Máximo = 2^{N-1}-1
  - Mínimo = -2^{N-1}-1

Representação de Números com Sinal

- Representação sinal e módulo:
- Vantagens:
  - Intuitiva
- Desvantagens:
  - Dupla representação para zero (+0 e -0)
  - A realização de somas não é imediata, por exemplo +3 + (-3), não faz sentido (implica cálculos auxiliares)!

Representação de Números com Sinal

- Representação em complemento para 2:
- A representação de um número em complemento para 2 é idêntica à representação de números sem sinal, exceto que o bit mais significativo em vez de valer 2^{N-1} vale -2^{N-1}.
- Por exemplo, em complemento para 2, com 4 bits:
  - Zero escreve-se: 0000 $0*(-2^3)+0*2^2+0*2^1+0*2^0$
  - O mais positivo é: 0111 $0*(-2^3)+1*2^2+1*2^1+1*2^0$
  - O mais negativo é: 1000 $1*(-2^3)+0*2^2+0*2^1+0*2^0$
  - -1 escreve-se: 1111 $1*(-2^3)+1*2^2+1*2^1+1*2^0$

Representação de Números com Sinal

- Representação em complemento para 2:
- Em complemento para 2, com N bits:
  - O mais positivo é 2^{N-1}-1
  - O mais negativo é -2^{N-1}
- O bit mais significativo em complemento para dois pode ser visto como o sinal do número:
  - Os números positivos em complemento para dois têm o bit mais significativo 0 e os negativos têm bit mais significativo 1.

Representação de Números com Sinal

- Representação em complemento para 2:
- O valor de um número negativo em complemento para 2 pode ser obtido num processo que chamamos calcular o complemento para 2 que consiste em inverter todos os bits e somar 1.
  - A representação em complemento para 2 com 4 bits de -3?

$$3_{10} = 0011_2$$

  - O complemento para 2 de 0011_2 é:

$$1100_2 + 1$$

  - Logo -3 em complemento para 2 com 4 bits escreve-se:

$$1101_2$$

Representação de Números com Sinal

- Representação em complemento para 2:
- Qual o valor de 1001_2 assumindo que está representado em complemento para 2 com 4 bits?
  - O bit mais significativo é 1, logo o número é negativo.
  - Então precisamos de calcular o complemento para 2, negando todos os bits e somando 1:

$$0110_2 + 1 = 0111_2$$

$$=7$$

  - Concluímos que 1001_2 representa o -7.

Representação de Números com Sinal

- Representação em complemento para 2:
- A vantagem esmagadora do complemento para 2 é que as somas funcionam de modo transparente quer se trate de números positivos quer sejam negativos.
- Exemplo $+2_{10} + (-1_{10})$ (com 4 bits)
  - $2_{10} = 0010_2$
  - 1_{10}= 0001 em complemento para 2: 1110_2+1 = 1111_2

Importante:
Desaparece porque estamos em 4 bits

```
  1110
  0010
 +1111
 10001
```

Representação de Números com Sinal

- Representação em complemento para 2:
- Exemplo $+5_{10} + (-3_{10})$ (com 4 bits)
  - $5_{10} = 0101_2$
  - $-3_{10} = 1101_2$
    - 3_{10}=0011 em complemento para 2: $1100_2+1 = 1101_2$

```
 1101
 0101
+1101
-----
~~1~~0010
```


Representação de Números com Sinal

- Representação em complemento para 2:
- Exemplo $+3_{10} + (-5_{10})$ (com 4 bits)
  - $3_{10} = 0011_2$
  - $-5_{10} = 1011_2$
    - 5_{10}=0101 em complemento para 2: $1010_2+1 = 1011_2$

```
  011
 0011
+1011
-----
 1110
```

- 1110_2 em complemento para 2 é 0001+1 = 0010 = 2
- $1110_2 = -2$

Representação de Números com Sinal

- Representação em complemento para 2:
- Deteção de Overflow
- Exemplo $+3_{10} + (5_{10})$ (com 4 bits)
  - $3_{10} = 0011_2$
  - $5_{10} = 0101_2$

$$\begin{array}{r} 111 \\ 0011 \\ +0101 \\ \hline 1000 \end{array}$$

- A soma de dois números positivos deu um número negativo!
  - Ocorreu overflow!

Representação de Números com Sinal

- Representação em complemento para 2:
- Deteção de Overflow:
  - Um **overflow ocorre se a soma de dois números positivos** for maior que o máximo positivo representável (2^{N-1}-1), ou seja **originar um número negativo**.
  - Ou se a **soma de dois números negativos** for menor que o menor número representável (-2^{N-1}), ou seja **originar um número positivo**.
  - Nunca ocorre overflow quando somamos números de sinais diferentes. (Porquê?)
- A existência de um carry out no bit mais significativo não é indicadora de overflow.

Representação de Números com Sinal

- Extensão do número de bits.
- Podemos necessitar de mudar o número de bits usado na representação de um número.
- Em números sem sinal:
  - Basta acrescentar zeros à esquerda.
- Em sinal e módulo:
  - Mudamos o sinal para a esquerda e acrescentar zeros no módulo.
- Em complemento para 2:
  - Acrescenta-se o sinal à esquerda:
    - 0's se o número é positivo
    - 1's se o número é negativo

Comparação de Representação de Números

Sistema Numeração	Gama
Sem sinal	$[0, 2^N-1]$
Sinal e Módulo	$[-(2^{N-1}-1), 2^{N-1}-1]$
Complemento para 2	$[-2^{N-1}, 2^{N-1}-1]$

	Unsigned	Two's Complement	Sign/Magnitude
-8		1000	
-7		1001	1111
-6		1010	1110
-5		1011	1101
-4		1100	1100
-3		1101	1011
-2		1110	1010
-1		1111	1001
0	0000	0000	0000 1000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111
8	1000		
9	1001		
10	1010		
11	1011		
12	1100		
13	1101		
14	1110		
15	1111		

Indice

Representação de Números Reais

- Até aqui vimos como representar quantidades inteiras (positivas e/ou negativas).
- Podemos generalizar a representação de inteiros na base 2, para representar números reais usando a virgula, à semelhança do que fazemos em base 10:

Parte_Inteira,Parte_Fracionária

$$x = b_3 b_2 b_1 b_0 , b_{-1} b_{-2} b_{-3}$$

$$x = b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 2^0 + b_{-1} 2^{-1} + b_{-2} 2^{-2}$$

Representação de Números Reais

- Virgula Fixa: Gama e Erro de Representação
- O Maior (e menor) valor representável é função do número de algarismos da parte inteira
- O erro de representação (absoluto) é função do número de algarismos da parte fracionária:
  - O erro é metade do peso do último algarismo. (Porquê?)
- Exemplo:
  - Representação decimal, com sinal mais 3 dígitos na parte inteira e 3 dígitos na parte decimal.
  - Gama de representação =
  - Erro de representação =

Representação de Números Reais

- Virgula Fixa Binária
- A virgula no sistema binário funciona do mesmo modo que no sistema decimal.
  - A função da virgula é identificar as potências de 2 positivas e as potências de 2 negativas.
- Desde que a soma (subtração) seja feita com as virgulas alinhadas **somar números inteiros ou fracionários em complemento para 2 é idêntico**.

Representação de Números Reais

- Virgula Fixa Binária
- Exemplo:
  - Representar com 8 bits, cinco inteiros e três fracionários os números:
    - 10.25_{10}
    - 4.625_{10}
    - -2.5_{10}
  - Efetuar em complemento para 2 as operações:
    - $10.25_{10}+4.625_{10}$
    - $4.625_{10}-2.5_{10}$

Representação de Números Reais

- Virgula Fixa Binária: Problemas.
- Representar valores muito grandes (ou muito pequenos) em virgula fixa requer uma enorme quantidade de algarismos.
- Em decimal a solução encontrada foi a notação científica:
  - $6.02*10^{23}$
  - $1.6*10^{-19}$
- Isto é **Virgula Flutuante**

Representação de Números Reais

- Virgula Flutuante Binária:

$$X = \pm m * b^{exp}$$

  - m: mantissa
  - exp: expoente
  - b: base
- A mantissa determina precisão.
- O expoente determina a gama de representação.
- A representação em Virgula Flutuante expressa um compromisso entre gama de representação e precisão.

O Formato IEEE 754

- Especifica o modo de representar números binários em virgula flutuante com:
  - Precisão simples: 32 bits
  - Precisão dupla: 64 bits
- As regras da soma, subtracção, multiplicação e divisão, raiz quadrada, resto e comparação.
- As conversões entre formatos.
- Os modos de arredondamento.
- As excepções.
  - A representação de 0, $\pm\infty$, NaN, e valores desnormalizados.

O Formato IEEE 754

	S	EXP	MANTISSA
Simples		8 bits	23 bits
Dupla		11 bits	52 bits

$$X = (-1)^S * 1,MANTISSA * 2^{(EXP-BIAS)}$$

- S: sinal (0: positivo 1: negativo)
- EXP: expoente representado em excesso do *Bias.*
  - *Bias* = 127 em *Single* 1023 em *Double*
- MANTISSA tem um bit escondido pois o algarismo à esquerda da virgula é sempre 1.

O Formato IEEE 754

- Exemplo 1:
- Que número está representado em formato IEEE 754 precisão simples como:

  1011 1111 0100 0000 0000 0000 0000 0000

  ou

  0xBF400000
- Solução:
  - Determinar, sinal, expoente e mantissa e escrever o número como no slide anterior.

O Formato IEEE 754

- Exemplo 2:
- Determinar a representação em formato IEEE 754 precisão simples do número: 5.5

- Solução:
  - Escrever o número na forma canónica.
  - Agrupar cada um dos campos nos 32 bits.

$$X = (-1)^S * 1, MANTISSA * 2^{(EXP-BIAS)}$$

O Formato IEEE 754

- Casos Especiais:

Número	Sinal	Expoente	Mantissa
0	X	00000000	00000000000000000000000
∞	0	11111111	00000000000000000000000
$-\infty$	1	11111111	00000000000000000000000
NaN	X	11111111	Non-zero

O Formato IEEE 754

- Adição:
- 1. Extrair os bits do expoente e da mantissa.
- 2. Acrescentar 1 à esquerda (o bit escondido) para obter a mantissa completa.
- 3. Comparar os expoentes
- 4. Deslocar à direita a mantissa do número menor para alinhar os expoentes.
- 5. Somar as mantissas e ajustar o expoente se necessário.
- 6. Arredondar o resultado
- 7. Juntar expoente e fração no formato do standard

O Formato IEEE 754

- Adição exemplo:
- Somar os números seguintes representados na norma IEEE 754.

0x3FC00000 + 0x40500000

Indice

Representação de Caracteres

- O código ASCII (American Standard Code for Information Interchange)
- É um código binário que codifica 128 sinais:
  - 95 Gráficos: letras do alfabeto, caracteres de pontuação e símbolos matemáticos
  - 33 Controlo: Esc, Del, mudanças de linhas, tabulações, etc.
- Podemos explorar propriedades da tabela ASCII:
  - Para determinar a ordem alfabética.
  - Para determinar a representação ASCII de um algarismo.
  - Para converter maiúsculas em minúsculas.

Representação de Caracteres: ASCII

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Representação de Caracteres

- O código ASCII apenas permite representar o alfabeto latino.
- Para representar outros alfabetos (chinês, cirílico, árabe, etc.) são necessários mais símbolos.
  - Logo mais bits para cada caractere.
- O standard UNICODE representa os caracteres em 8, 16 ou 32 bits. Sendo que a representação de 8 bits (UTF-8) é compatível com o código ASCII.

Outros tipos de dados

- Imagens
  - São tratadas como arrays de pixels
    - Monocromáticas: 1 bit define se é preto ou branco
    - Cor:
      - sistema RGB com 8 bits para a intensidade de cada cor
      - Sistema HSV, etc.
- Som
  - O som é um sequência de números armazenados que representam a amplitude sonora ao longo do tempo.

Sumário

- Nos computadores os números são representados em **base 2**.
- Para números inteiros positivos é possível com N bits armazenar os valores entre 0 e 2^N-1
- Inteiros com sinal representados em complemento para 2, com N bits podemos representar entre -2^{N-1} e $2^{N-1}-1$
- Os números reais são representados em vírgula flutuante (sinal, expoente e mantissa) em 32 ou 64 bits: norma IEEE 754.
- Caracteres são representados em ASCII (1 byte) ou UNICODE.

Próxima Aula…

- Lógica binária. As operações lógicas básicas.
- Álgebra de Boole.