CDU 002.6:681.3

INTRODUÇÃO À COMPUTAÇÃO ELECTRÓNICA

UNIVERSIDADE DE SÃO PAULO
Campus de São Carlos
Centro de processamento de dados e estudos de sistemas

RESUMO: Trata-se do material difundido pelos frequentadores dum curso de iniciação, como se depreende do título, cuja reprodução foi amavelmente autorizada e tem como objectivo fornecer, por intermédio de «Cadernos», a quem se encontra ligado a serviços biblioteconómicos, arquivísticos, documentais e informativos mais um elemento que permita a sua sensibilização para os problemas da automatização e transmita aquele mínimo de conhecimentos necessários, à partida, para qualquer realização neste campo.

São sucessivamente tratados temas como conjuntos, funções, algoritmos e fluxogramas, variáveis e constantes, sistemas numéricos e erros, precedidos duma introdução histórica e descritiva de equipamentos, linguagens e funções especializadas em processamento de dados.

SUMMARY: As the title implies, it consistes of material which these who had attended an introductory course were kindly permitted to publish. Its object is, through «Cadernos», to give those who are connected with the organization of libraries, and documentary and information centres not only further elements which will help them to understand the problems of automation but also that minimum of knowledge which is necessary as a starting point for any kind of enterprise in this field.

After a historical and descriptive introduction about equipment, language and functions, the following subjects are dealt with in turn: — conjuncts, functions, algorisms, fluxions, variable and constant, numerical systems and errors.

1. INTRODUÇÃO HISTÓRICA

Os computadores têm-se desenvolvido ao longo de dois caminhos perfeitamente distintos:

1.1. COMPUTADOR ANALÓGICO

A régua de cálculo é um exemplo de computador analógico. Números são representados por distâncias numa escala, e as distâncias são determinadas pelos logaritmos dos números. Como um exemplo: a multiplicação de dois números é feita por adição de suas distâncias na escala.

Podemos notar que o computador tem o seu funcionamento baseado na medida contínua de um determinado comprimento ou a distância entre dois pontos.

Alguns exemplos de computadores analógicos: velocímetros, voltímetros, termómetros, etc.

Uma instalação do Controle de Tráfego da Força Aérea Norte Americana utiliza um computador analógico para acompanhar o curso das aeronaves e programar suas aterrissagens, à razão de duas por minuto, de modo que não interfiram entre si. O computador calcula o curso de vôo apropriado para cada aeronave, enquanto os operadores introduzem valores indicando a direcção e a velocidade do vento, a rota própria a ser usada, e a designação da aeronave. Depois disto feito, o computador compara o curso real da aeronave com o curso teórico que ele locou, e então dá as correcções necessárias.

São complexos os computadores analógicos? Realmente não o são. Além das fontes de alimentação, seus principais componentes são amplificadores e redes passivas de resistores, capacitadores e indutores. Conectando os amplificadores e redes passivas apropriadas, é possível representar qualquer problema.

Cadernos Bibl. Arq. Doc., Coimbra, 11 (3-4) Jul.-Out. 1974, p. 88-130

1.2. COMPUTADORES DIGITAIS

Cuidadosamente, a natureza forneceu aos nossos mais antigos ancestrais o modo mais simples de cálculo — *os dedos* — isto é, um computador digital no sentido estrito da palavra.

Num computador digital as quantidades são contadas em vez de medidas como num computador analógico. Desde a invenção dos próprios números, o homem vem criando dispositivos para ajudá-lo a lidar com um grande volume de dados.

Vamos, então, tentar mostrar por ordem cronológica de tempo o aparecimento desses inventos.

ÁBACO (antigo)

O ábaco estava entre os primeiros inventos usados pela computação e foi tomado sob muitas formas no decorrer dos anos. As atuais formas são de armação de madeira ou metal com colunas de contas em finas varas. Os mercadores da China e de outros países asiáticos usavam-no para cálculos apurados.

As colunas de contas são usadas para representar as unidades, dezenas, centenas e assim sucessivamente, dependendo do número de colunas.

Apesar dos métodos não serem explicados aqui, o ábaco pode ser usado para somas, subtrações, multiplicações (repetidas somas) e divisões (repetidas subtrações).

CALCULADORA MECÂNICA DE RODAS (1642)

Blaisy Pascal inventou a primeira máquina de adição mecânica, do mundo. Sua calculadora era baseada em um conjunto de rodas em forma de engrenagens; uma roda representava a unidade, outra a dezena e assim por diante.

O número 15, por exemplo, entraria pela movimentação da roda de unidades o número 5, e pela movimentação da roda de dezenas o número 1. Para somarmos 23 com esse número, movemos a roda de unidade 3 a mais e a roda das dezenas 2 a mais. O resultado é que na roda das unidades teremos movido 8 e na roda das dezenas 3. Temos, portanto, um total de 38.

As engrenagens são feitas de tal maneira que se houver necessidade de um carregamento, ele será feito de maneira correta. Vamos supor que exista um 8 em uma roda e desejamos somar 5 a essa roda: o resultado aparecerá como 3 na roda, e será carregado 1 para a roda seguinte.

OS MOTORES DE BABBAGE (1812-1834)

Foi em 1812 que Charles Babbage, um matemático, ficou particularmente aborrecido pelas inexatidões que descobriu nas tabelas de cálculo que usava.

Nesse tempo as tabelas eram preparadas manualmente. Isso, logicamente, era um processo que consumia muito tempo e deixava margem a erros.

Babbage logo começou a trabalhar em uma máquina de calcular automática, que teria capacidade para gerar muitas tabelas, fazendo o uso de diferenças de alta ordem.

A máquina foi construída e recebeu o nome de «máquina de diferenças» e fornecia resultados impressos, sem o auxílio do homem.

Com a idade de 30 anos Babbage foi persuadido pelo governo britânico a projetar uma máquina de maiores dimensões, mas em vista do alto custo o projeto foi abandonado.

ÁLGEBRA DE BOOLE (1854)

George Boole, um matemático, desenvolveu um sistema para representação lógica de declarações em termos de símbolos matemáticos, usando símbolos e algumas regras que poderiam determinar se as declarações seriam logicamente verdadeiras ou falsas.

Seus métodos não eram amplamente aceitos e não os foram até o século seguinte, onde sua idéia foi aplicada.

CARTÕES PERFURADOS (1880-1890)

Herman Hollerith desenvolveu uma moderna máquina leitora de cartões e a associou a um equipamento mecânico processador de cartões. Hired, do Departamento de Recenseamento dos Estados Unidos, o desenvolveu e usou sua inversão em uma grande quantidade de dados coletados no censo de 1890.

Ainda hoje utilizamos o código de cartão perfurado de H. Hollerith.

LÓGICA DE CIRCUITOS (1938)

Claude Shannon aplicou a álgebra de Boole no sistema de representação de complexos circuitos de redes. O resultado dos ensinamentos e pesquisas simplificados por Shannon, no campo de circuitos, pode ser encontrado nos futuros computadores eletrônicos.

MARK I (1937-1944)

Em 1937, Howard Aiken da Universidade de Harward, projetou um grande calculador mecânico capaz de realizar uma grande sequência de operações aritméticas e lógicas. Propos que o calculador reconhecesse instruções provenientes de fitas perfuradas, instruções que passavam pelas unidades de controle, aritméticas e, se necessário, de armazenamento, produzindo então os resultados. Mas, o MARK I, foi relativamente diminuído porque a sua velocidade de cálculo depende da velocidade de vários componentes eletromecânicos.

A IBM harmonizando-se com a construção do calculador de Aiken, completou a máquina e doou-a à Universidade de Harward, em 1944.

ENIAC (1934-1945)

O Eletronic Numerical Integrator and Calculator (ENIAC), foi projetado por J. P. Eckert e J. W. Mauchly, da Universidade da Pennsylvania. Foi o primeiro computador eletrônico (usando 18.000 válvulas) e foi muito mais rápido do que o MARK I. Entretanto, a máquina não tinha armazenamento interno. Ele recebia suas instruções externas através de interruptores e plugs, ocasionando uma séria limitação.

Pode-se notar que o ENIAC era um computador com o especial propósito de resolver problemas matemáticos voltados para a balística e a aeronáutica.

EDVAC (1945-1952)

O Eletronic Discrete Variable Automatic Computer (EDVAC) foi também desenvolvido pela equipe Eckert e Mauchly. Era maior que o ENIAC, usava números binários (base 2) para operações aritméticas e armazenava instruções internamente.

UNIVAC I (1951)

Foi o primeiro computador a ser utilizado comercialmente; foi utilizado para processamento de dados não científicos.

O UNIVAC I (Universal Automatic Computer) foi construido pela Remington Rand (atual Sperry Rand) companhia originária da Eckert and Machly. É usada fita magnética para entrada e saída; anteriormente os computadores usavam fita ou cartões perfurados, consideravelmente mais lentos. O UNIVAC foi também o primeiro computador a aceitar o processamento de dados alfabéticos, tão bom como os numéricos.

COMPUTADOR DE SEGUNDA GERAÇÃO (1959)

Em pouco tempo a UNIVAC, a IBM e outras companhias se juntaram para desenvolver uma nova série de computadores.

Como vimos, os computadores de primeira geração usavam válvulas como componentes básicos de seus circuitos internos. As máquinas eram volumosas e necessitavam de muita força para funcionar; produziam muito calor e necessitavam de um cuidadoso sistema de ar condicionado para proteger e mante-los funcionando.

A primeira geração de computadores não era tão digna de confiança como se esperava e sua capacidade de armazenamento interno era muito pequena.

Pesquisas levaram ao desenvolvimento de computadores que usam transistores em lugar de válvulas; máquinas bem menores com necessidade de bem menos força que as anteriores e produzindo pouco calor. Sua capacidade de armazenamento era maior, assim como sua velocidade de cálculo.

COMPUTADORES DE TERCEIRA GERAÇÃO (1964)

Em 1964, apareceram no mercado, os sistemas de computadores de terceira geração. Eles têm grande vantagem sobre os seus antecessores, incluindo características que faltam aos computadores de segunda geração.

Um novo projeto usa microcircuitos lógicos no estado sólido para cada condutor, resistor, diodo e transistores que têm sido miniaturizados e combinados em placas de cerâmica de meia polegada.

Outro projeto usa pequenas placas em que os circuitos e seus componentes são impressos. Essas chapas são chamadas circuitos integrantes monolíticos.

Computadores utilizando esses pequenos circuitos, tornam trabalhos, que antes eram considerados impraticáveis devido ao tempo, em trabalhos perfeitamente praticáveis.

Grande capacidade de armazenamento e baixo custo por «bit» de armazenamento. Muitas indústrias de computadores de terceira geração produzem séries similares e compatíveis. Isto significa que o programa que é processado em um computador de uma determinada série, poderá ser processado em um modelo maior da mesma série. Em outras palavras, eles aceitam e executam as mesmas instruções e trabalham os dados da mesma maneira.

O usuário (bancos, linhas aéreas, centros de pesquisa, etc.) pode alugar um computador e trocá-lo por um maior, quando necessário, sem precisar modificar seus programas nem a forma de seus dados.

É agora muito mais fácil, para pessoas especializadas em operação ou programação de um computador, ajustar-se a diferentes computadores.

Os sistemas de computadores de 3.ª geração podem ser aplicados tanto no campo científico como no comercial, com igual facilidade, sendo que antes necessitaríamos instalar dois computadores: um científico e um comercial — ou um só computador que prejudicaria uma das áreas.

Juntamente com os computadores de 3.ª geração, novos e mais rápidos equipamentos têm sido introduzidos para o trabalho de entrada e saída.

A evolução das gerações, tipos de circuitos e caracterização de tempo podem ser melhor observados na fig. 1.

Geração	*Data*	*Circuito*	*Tempo*
1ª	até 1959	Válvulas	Milisegundos (10^{-3})
2ª	até 1964	Transistores	Microsegundos (10^{-6})
3ª	até 1964	Transistores. Circuitos micro-miniaturizados	Nanosegundos (10^{-9})

Fig. 1

Fábricas existentes

Atualmente existem muitos fabricantes de computadores de alta velocidade. Relacionamos alguns deles:

— Burroughs Corporation (B)
— Control Data Corporation (CDC)
— Digital Equipment Corporation (PDP)
— General Eletric Company (GE)
— Honeywell (H)
— International Business Machines Corporation (IBM)
— National Cash Register Company (NCR)

— RCA Corporation (RCA)
— Univac Division of Sperry Rand (UNIVAC)
— Xerox Data Systems (XDS)

Aplicações

Os computadores foram desenvolvidos, originalmente, para resolverem cálculos matemáticos. Hoje, o computador vai muito além de resultados matemáticos; vai desde aplicações jurídicas, governamentais e esportes até música e vôos espaciais.

A descoberta do DNA, RNA, o código genético e a produção sintética de matéria viva, são baseados nos resultados de computadores.

Autoridades de todos os níveis usam o computador para a administração, estudos de poluição do ar, controle do tráfego e prevenção do crime. Através de dados armazenados no computador, a polícia pode fazer uma pesquisa instantânea de propriedades roubadas e pessoas procuradas.

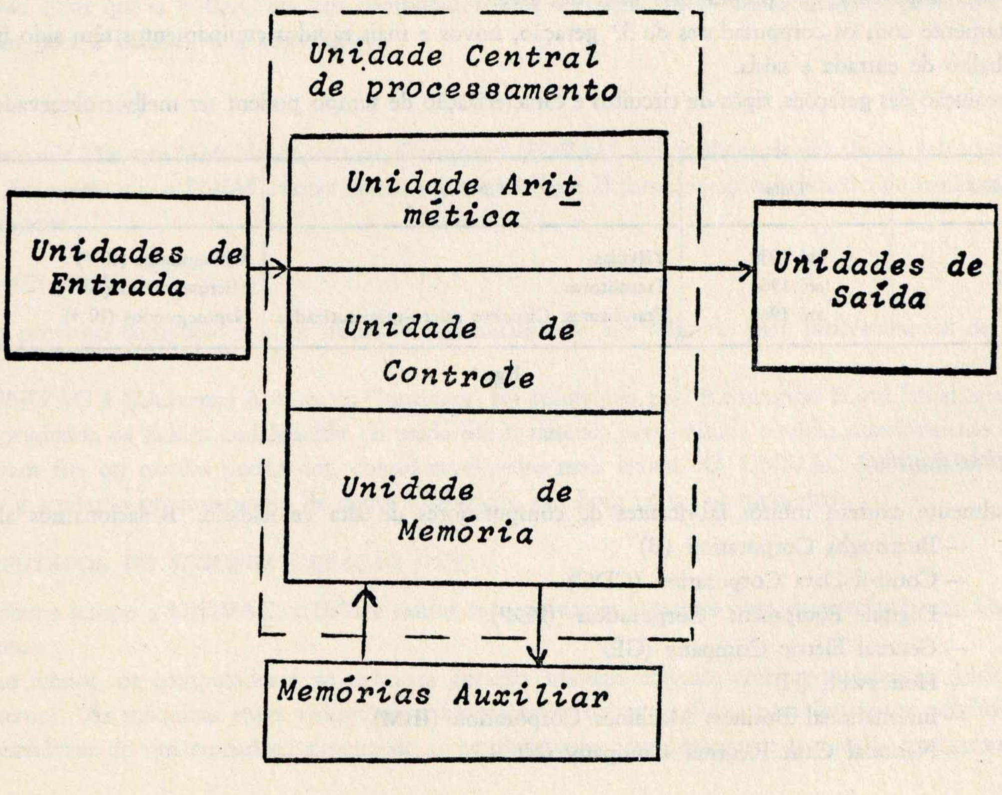
No comércio e na indústria os computadores são usados para fazer reservas de passagens aéreas, registrar transações bancárias, controlar usinas convencionais e nucleares geradoras de vapor. Ajudam companhias a operar usinas de aço, estradas de ferro e refinarias de petróleo e auxiliam os agricultores a melhorar sua produtividade.

Os computadores são usados para instrução e pesquisa educacional. Auxiliam médicos nos sinais, ou melhor, na análise dos sinais de instrumento de diagnóstico. Aceleram os trabalhos de salvamento no mar, ajudam a encontrar doadores de sangues raros e previnem o perigo de incêndios nas florestas distantes.

Esses exemplos são somente uns poucos dos múltiplos usos que os homens fazem do computador. E eles estão trabalhando constantemente em novas aplicações dos computadores na educação, medicina, literatura e em muitas outras áreas.

1.3. DESCRIÇÃO DE COMPUTADOR IBM-1130

1.3.1. *Esquema geral de um sistema de computação electrónico*



Um computador eletrónico é uma máquina que apresenta as seguintes unidades básicas:

— unidades de entrada
— unidade central de processamento, composta de:
  — unidade aritmética
  — unidade de controle
  — unidade de memória
— unidades de saída

1.3.2. ***Unidades de entrada***

As unidades de entrada são dispositivos que lêem os veículos que servem de suporte para informação codificada

No IBM 1130 existem as seguintes unidades:

— leitora de cartões — IBM 1442
— leitora de fita de papel — IBM
— teclado da console
— chaves da console
— «drive» de disco magnético (leitura)

Para outros computadores existem unidades tais como:

— leitora de fita magnética
— leitora de caracteres óticos
— terminais — teclado
— «drive» de tambor magnético (leitura)

1.3.3. ***Unidades de saída***

As unidades de saída são dispositivos que fornecem os resultados de um determinado processamento, resultados esses contidos na memória.

No IBM 1130 existem as seguintes unidades:

— perfuradora de cartões — IBM 1442
— perfuradora de fita de papel — IBM
— impressora rápida — IBM 1403
— impressora lenta — IBM 1132
— painel da console
— «drive» de disco magnético (gravação)
— traçador de gráficos — IBM 1627 (Plotter)
— máquina de impressão da console

Para outros computadores existem unidades tais como:

— terminais — máquina de impressão
— «Displays» (telas ou vídeos)
— «drive» de tambor magnético (gravação)

1.3.4. ***Memória***

A unidade de memória armazena o programa, dados recebidos da unidade de entrada e os resultados parciais e finais da unidade aritmética.

As memórias são — em sua maioria — magnéticas e, em geral, dos seguintes tipos:

a) memórias internas: de núcleos magnéticos, de tambores magnéticos e outros de menor uso;

b) memórias auxiliares: disco magnético, fita magnética, tambor magnético.

Memória interna é aquela intrinsecamente incorporada à unidade central de processamento. Tem, em geral, uma capacidade de armazenamento limitado, mas é de processamento rapidíssimo e de pequeno tempo de acesso (da ordem de nanosegundos).

As memórias auxiliares são, em geral, de grande capacidade de armazenamento. São usadas para arquivo, armazenamento dos sistemas de programação e para grandes áreas intermediárias de trabalho. O processamento, porém, é mais lento e é maior o tempo de acesso (milisegundos).

1.3.5. ***Unidade aritmética***

A unidade aritmética é aquela que recebe os operandos da memória e executa a parte aritmética propriamente dita. Ela também realiza operações lógicas de comparação, teste de sinal, etc..

1.3.6. ***Unidade de controle***

A unidade de controle supervisiona as unidades já mencionadas. Comanda a entrada de dados e programas, interpreta cada instrução, que já deve estar registrada na memória e, de acordo com o seu código, pode comandar a unidade aritmética para apanhar os operandos na memória e processá-los, ou comandar a transferência de resultados para a saída, ou transferir, condicionalmente, ou não, para outra instrução, a execução do programa, ou interromper o programa.

1.4. O CONCEITO DE PALAVRA EM COMPUTADOR

Internamente, um computador digital trabalha representando toda a informação através de números.

A lógica de funcionamento de um computador é bivalente. Supõe a existência de apenas duas situações: falso ou verdadeiro, sim ou não.

Pelo facto de um computador digital utilizar-se da lógica binária, e trabalhar internamente apenas com números o sistema numérico por ele utilizado é o binário, composto dos dígitos 0 e 1.

Logicamente, o *zero* corresponderia à ausência de informação e o *um* corresponderia à presença de informação.

Fisicamente, os núcleos que compõem a memória do computador são capazes de serem magnetizados, tanto no sentido dos ponteiros de um relógio, quanto em sentido inverso. Uma magnetização no sentido dos ponteiros do relógio significa «sim» (um) e a magnetização no sentido contrário «não» (zero).

Magnetização dos núcleos

Em qualquer circunstância, a informação contida na direcção de magnetização de um núcleo é a menor unidade de informação, chamada BIT. Um núcleo pode, então, armazenar um dígito binário, 0 ou 1, mas uma coleção de núcleos pode armazenar uma quantidade enorme de bits.

Embora o BIT seja a menor unidade de informação, é a palavra — um determinado conjunto de BITS, que representa a menor unidade de informação à qual se tem acesso, a qual se pode manipular.

No Computador IBM-1130 uma palavra é composta de 16 bits, numerados de 0 a 15, cada um podendo conter 0 ou 1.

Cadernos Bibl. Arq. Doc., Coimbra, 11 (3-4) Jul.-Out. 1974, p. 88-130

Na palavra deve-se distinguir o seu conteúdo e o seu endereço. O conteúdo será sempre um conjunto de zeros e / ou uns, os quais representam uma informação; o endereço é único para cada palavra e é através dele que se consegue localizar, na memória do computador, uma determinada palavra.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

1.4.1. ***Representação dos números nas palavras***

Os números inteiros são representados ocupando uma palavra de 16 bits.

O bit zero é a posição reservada para indicar o sinal.

O maior inteiro positivo que pode ser representado é o número 32767^{+}, isto é, 2^{15}-1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

O menor inteiro negativo, que pode ser representado é o número — 32768, isto é, 2^{15}

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Existe um código que faz a conversão de números do sistema decimal para o binário. É o código EBCDIC (Extended Binary Coded Decimal Interchange Code) que quer dizer: Código Extendido de Intercâmbio dos Decimais Codificados em Binário.

Os números reais são representados ocupando duas palavras de 26 bits cada uma.

Os números reais são escritos em duas partes: mantissa e expoente. A mantissa é a parte decimal (com ou sem sinal) e é escrita com um ponto decimal que pode aparecer no começo, no meio ou no fim da mesma.

O expoente consiste da letra E seguida por um número inteiro, com ou sem sinal, que indica a potência de 10 pela qual deve ser multiplicada a mantissa para manter a indicação correcta do número. Caso o expoente no número seja zero, não será necessário escrever a parte exponencial.

Exemplos:

$$4{,}5 = 4.5E0 = 4.5 \times 10^{0} = 4{,}5$$

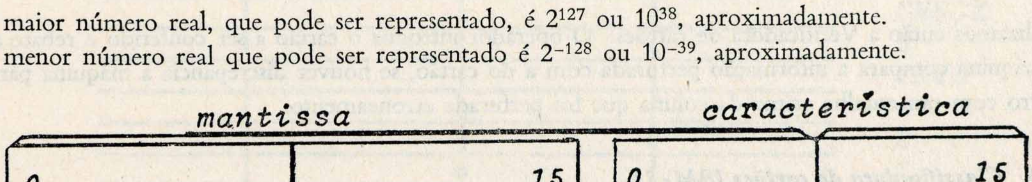
$$4{,}5 = 45.E^{-}1 = 45. \times 10^{-1} = 45 \times \frac{1}{10} = 4{,}5$$

$$4{,}5 = 4500.E^{-}3 = 4500. \times 10^{-3} = 4500. \times \frac{1}{10^{-3}} = 4{,}5$$

$$4{,}5 = 0.00045.E4 = 0.00045 \times 10^{4} = 4{,}5$$

O maior número real, que pode ser representado, é 2^{127} ou 10^{38}, aproximadamente.

O menor número real que pode ser representado é 2^{-128} ou 10^{-39}, aproximadamente.



1.4.2. ***Representação das letras nas palavras***

Todas as letras, internamente, quando o código de intercâmbio é o EBCDIC, são representadas por oito dígitos binários. Assim, em cada palavra, podem ser armazenadas no máximo duas letras.

Por exemplo:

A = 1100 0001

B = 1100 0010

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	0
			A									B			

1.5. EQUIPAMENTOS PARA PREPARAÇÃO DE DADOS DE ENTRADA

Existem inúmeros equipamentos cujas funções executadas desempenham a preparação de dados de entrada, como por exemplo perfuração de cartões, perfuração de fita de papel, verificação, multiperfuração, reprodução, perfuração por percepção de marcas, interpretação, interpretação marginal, classificação, seleção, comparação, impressão em detalhe, impressão por grupo, perfuração resumo, prova de cartão, conversão, transferência de lançamentos, coleta de dados.

As unidades existentes no C. P. D. E. S. são as seguintes:

1.5.1. ***Perfuradora de cartões IBM-029***

As perfuradoras executam basicamente duas funções:

1.5.1.1. *Perfuração*

É basicamente a mesma espécie de operação que dactilografar em uma máquina de escrever comum. A máquina é alimentada comprimindo-se uma tecla especial, e o operador simplesmente transcreve as informações para o cartão pressionando as teclas correspondentes aos caracteres desejados; terminada a perfuração a máquina ajusta o cartão com a pressão de uma nova tecla.

1.5.1.2. *Duplicação*

Se tomarmos uma série de dados a serem perfurados, sendo que existe uma informação que deva se repetir em todos os cartões, em vez de transmitirmos a informação à máquina batendo tecla por tecla, simplesmente perfuramos a informação no primeiro cartão e a duplicamos nos demais.

1.5.2. ***Verificadora de cartões IBM-59***

Para conferência de um grande volume de dados, seria muito trabalhoso e muito pequena a margem de segurança, se conferissemos os dados manualmente.

Utilizamos então a Verificadora de cartões. O operador introduz o cartão a ser conferido e rebate as instruções. A máquina compara a informação perfurada com a do cartão, se houver discrepância a máquina pára, assinalando o erro com um entalhe acima da coluna que foi perfurada erroneamente.

1.5.3. ***Classificadora de cartões IBM-82***

Trabalhando por meio de escovas eletromagnéticas e lâminas de seleção, a IBM-82 tem a função de classificar cartões de acordo com nossas necessidades (alfabeticamente, numericamente).

1.5.4. ***Flexowriter 2301***

Aparentemente tem o aspecto de uma máquina de escrever elétrica, através do teclado são inseridas as informações que são dactilografadas e ao mesmo tempo perfuradas em fita de papel. Assim podemos perfurar programas e dados em fita de papel.

A operação inversa também pode ser feita. Colocamos a fita já perfurada, contendo um programa ou dados, pressiona-se uma determinada tecla, e automaticamente a máquina transporta as informações da fita para o carro da máquina.

1.6. VEÍCULOS PARA SUPORTE DE INFORMAÇÕES

1.6.1. ***Cartão perfurado***

Todas as 80 colunas do cartão podem receber perfurações. Cada coluna tem 12 posições de perfuração, chamadas alturas, sendo uma para cada um dos dígitos de 1 a 9, uma para a zona 0, outra para a zona 11 e uma última para a zona 12. A perfuração na zona 11 é também chamada de perfuração X. Os números são registrados no cartão pela perfuração da altura correspondente, sendo o 0 perfurado na zona 0.

As letras são obtidas com uma combinação de perfuração em zona e altura.

Os caracteres especiais são conseguidos com a combinação de uma, duas ou três perfurações de uma mesma coluna.

1.6.2. ***Fita de papel***

Cada coluna tem 9 posições de perfuração.

Números combinação da perfuração das posições.
0, 1, 2, 4, 8

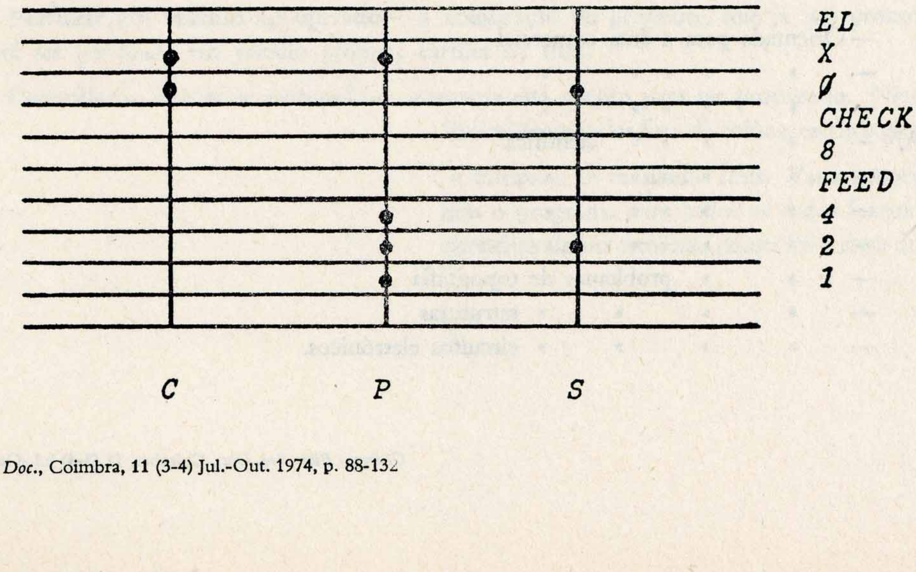
Letras 1) A-I — posição X e 0 mais a combinação de perfuração nas colunas 1, 2, 4, 8, tal que a soma dessas colunas forneça a posição da letra no grupo.
2) J-R — posição X, mais combinação.
3) S-Z — posição 0, mais combinação começando a partir da coluna 2.

Caracteres combinação das posições restantes.

Feed — colunas pré-perfurada, servindo como apoio para o movimento da fita
Check — coluna de verificação de perfuração.
EL — indicar fim de registro

Exemplo:

C = perfuração X, 0, 2, 1 (3.ª letra do grupo 1)
P = perfuração X, 1, 4, 2 (7.ª letra do grupo 2)
S = perfuração 0, 2 (1.ª letra do grupo 3)



1.6.3. *Formulários para preparação de programas e dados*

1.6.3.1. *Formulários para preparação de programas*

As declarações de uma linguagem devem ser escritas pelo programador conveniente, numa folha de codificação apropriada.

Examinando a folha, vemos que cada linha é constituída de 80 espaços, correspondentes às 80 colunas existentes no cartão. Cada uma destas linhas é a imagem do que será o cartão depois de perfurado.

1.6.3.2. *Formulário para preparação de dados*

Serão do mesmo tipo dos formulários para preparação de programas. O programador, após decidir qual vai ser o formato de seus dados, transcreve-os para o formulário de dados, o qual posteriormente será entregue ao operador de uma perfuradora de cartões.

Os dois tipos de formulários devem ser preenchidos com letra de forma, ocupando cada caracter o espaço correspondente a uma só coluna.

1.6.4. *Formulários para impressão de relatórios*

Devemos ter uma idéia a respeito da disposição das saídas de nossos programas. Para isso usamos um formulário que representa a folha de saída de um computador.

Esse formulário contém um número qualquer de linhas, sendo que cada linha é constituída de 120 espaços, correspondentes às 120 colunas de impressão do computador.

Assim o usuário representa a saída de seu programa em um formulário, depois translada-os para os comandos de impressão de seu programa.

1.7. TIPOS DE LINGUAGEM

As linguagens são o meio de comunicações entre o homem e o computador. Existem muitos tipos de linguagens, as quais são classificadas em três categorias.

1.7.1. **Absoluta ou de máquina** — é a linguagem que o computador entende. As instruções são em forma numérica. Considerada como a linguagem de mais baixo nível.

1.7.2. **Simbólica ou mnemónica** — tem a mesma estrutura da linguagem absoluta, mas em vez de utilizarmos códigos numéricos, substituimos por códigos mnemónicos, isto é, nomes que estejam associados com as funções que a ordem executa. Considerada como linguagem de baixo nível.

1.7.3. **Automáticas** — são aquelas que têm suas instruções próximas à linguagem comum. Foi com o aparecimento das linguagens automáticas que um grande número de pessoas teve acesso aos computadores. Várias são as linguagens automáticas:

COBOL	— Orientada para a área comercial
RPG	— » » » » »
CSP	— » » » » »
FORTRAN	— » » » » científica
ALGOL	— » » » » »
PL1	— » » » » »
BASIC	— » » » » »
COGO	— » » problemas de topografia
STRESS	— » » » » estruturas
ECAP	— » » » » circuitos eletrónicos.
Etc.	

1.8. FUNÇÕES ESPECIALIZADAS EM PROCESSAMENTO DE DADOS

Análise de Sistemas: tem por funções liderar projetos de sistemas e tratamento de informações, definir documentos de entrada e relatório, assessorar o usuário quanto a conveniência ou não da execução do serviço em equipamento eletrónico ou eletromecânico.

Análise de Suportes: elaborar diagramas dos programas definidos pelo analista de sistema, definir a melhor situação do equipamento disponível, definir linguagem.

Análise de Métodos: elaboração de documentos de entrada, definir rotinas de trabalho, elaborar manuais de serviço.

Análise de Custos: definir sistema que permita o controle, a avaliação e a análise do desempenho da mão de obra na utilização do equipamento e consumo de material de processamento.

Programação: elaborar diagramas de bloco detalhados dos programas, codificar programas.

Operação: planejar a utilização do computador coordenando o processamento de operação de acordo com instruções específicas, analisando sua prioridade, tempo de processamento, volume de memória ocupada e partição.

Documentalista de material especializado: manter o arquivo de fitas e discos magnéticos.

Perfuração: definir e elaborar cartões, programas, operar máquina perfuradora e verificadora.

1.9. FASES DE RESOLUÇÃO DE PROBLEMAS

Supondo-se que tivéssemos que resolver o seguinte problema:

1.9.1. **Definir o trabalho** — resolução de uma equação do segundo grau.

1.9.2. **Analisar o problema** — Análise de problema é pesquisa de métodos e recursos existentes para resolução do mesmo, a fim de compilar elementos que funcionam como ingrediente na escolha do melhor método de solução. Ao lado dessa análise sobre técnicas específicas de resolução de problemas devemos também nos preocupar com a transposição dos métodos e técnicas para o computador, cuidando para que seja encontrada uma solução óptima.

1.9.3. **Escolher os métodos de resolução** — de acordo com a análise do problema devemos seleccionar o melhor método e aplicá-lo.

1.9.4. **Descrever graficamente o método utilizado para a resolução** — transpor o método escolhido para a forma de diagramas de bloco. Além de termos uma melhor visão do problema, poderemos mudar as possíveis falhas que o problema apresentar.

1.9.5. **Codificar o problema em linguagem apropriada** — pela análise do problema temos condições de saber qual é a área em que o problema está incluso. No nosso caso, escolhemos uma linguagem voltada para a área científica. A codificação deverá ser feita em formulários especiais.

1.9.6. Simular os passos que vão ser seguidos pelo computador. Nessa fase o nosso problema já está montado no diagrama de blocos e na linguagem escolhida. Seguindo o diagrama como se fôssemos o computador tentaremos descobrir os possíveis erros de lógica, ou a falta de alguma instrução.

1.9.7. **Perfurar em veículo apropriado** — a codificação do programa que já está pronto em formulário especial, deverá ser perfurada em veículo próprio, cartões ou fitas.

1.9.8. **Compilar** — depois de perfurado o programa está pronto para ser processado. Nessa fase o compilador da linguagem escolhida vai-nos apontar os erros de sintaxe (comandos fora de coluna, escritos erroneamente, etc.).

1.9.9. **Testar** a execução com dados amostrais e comparar os resultados reais. Vamos supor que tivéssemos que resolver mais de 500 equações. Antes de processarmos o programa com todos os dados fazemos um teste com quatro ou cinco dados. Dessa maneira, caso o programa apresente algum erro, não corremos o risco de perder o tempo de execução para as 500 equações.

2. CONJUNTOS

Em matemática, bem como na vida cotidiana, é muitas vezes desejável pensar em vários objectos como uma simples unidade.

Os matemáticos têm adoptado o uso da palavra CONJUNTO para denotar a idéia de um objecto constituído de vários objectos individuais distintos. Os objectos individuais são chamados ELEMENTOS do conjunto e são ditos pertencerem ao conjunto.

Um conjunto S, composto pelos objectos *a*, *b*, e *c*, é descrito:

$$S = \{a, b, c\}$$

Exemplos:

1) $N = \{0;\ 1;\ 2;\ 3;\ldots\}$ = conjunto dos números Naturais.
2) $Z = \{\ldots;\ -2;\ -1;\ 0;\ 1;\ 2;\ldots\}$ = conjunto dos números Inteiros.
3) $Q = \{\ldots;\ -1/2;\ -0{,}2;\ 0;\ 0{,}333\ldots;\ 5;\ldots\}$ = conjunto dos números Racionais.
4) $R = \{\ldots;\ -0{,}8;\ -\sqrt{3};\ 3;\ 3/4;\ldots\}$ = conjunto dos números Reais.

O conjunto dos números Reais inclue os números irracionais (têm infinitas casas decimais e não periódicas) tal como:

$$\sqrt{2} = 1{,}4142\ldots$$

Para indicar que um objeto *a* pertence a um conjunto S escreve-se:

$$a \in S$$

Se *a* não é um elemento de S (*a* não pertence a S) escreve-se:

$$a \notin S$$

Exemplos:

1) $7 \in N$
2) $7{,}5 \notin N$

Se $a \in S$, $b \in S$ e $c \in S$ pode-se abreviar este comando para: $a, b, c \in S$.

Um conjunto é denominado CONJUNTO INFINITO se possuir infinitos elementos e é chamado CONJUNTO FINITO se possuir um número finito de elementos.

Exemplo: O conjunto N é infinito e o alfabeto é finito.

Um conjunto é denominado CONJUNTO UNITÁRIO se possuir um único elemento e é denominado CONJUNTO VAZIO se não possuir nenhum elemento. Um conjunto vazio é representado por:

$$\{\ \} \quad \text{ou} \quad 0$$

Exemplo: Se S é o conjunto dos meses com 32 dias, então:

$$S = 0$$

Dois conjuntos S e T são IGUAIS se eles têm precisamente os mesmos elementos e escreve-se:

$$S = T$$

Se S e T são DIFERENTES (não iguais) escreve-se:

$$S \neq T$$

Exemplo: Se $S = \{1, 2\}$, $T = \{2, 4\}$ e $J = \{1, 2\}$ então:

$$S = J,\ S \neq T,\ \text{e}\ T \neq J$$

Um conjunto T é um SUBCONJUNTO de um conjunto S se todo elemento de T é também elemento de S e escreve-se:

$$T \subset S \quad \text{ou} \quad S \supset T$$

Se T não é subconjunto de S escreve-se:

$$T \not\subset S \quad \text{ou} \quad S \not\supset T$$

Os símbolos $\subset$, $\not\subset$, $\supset$, $\not\supset$, significam respectivamente, «contido», «não contido», «contém», «não contém». Se $T \subset S$ e S tem um elemento não pertencente a T, então T é um SUBCONJUNTO PRÓPRIO do conjunto S. Se todo elemento de S é também elemento de T então T é SUBCONJUNTO TRIVIAL do conjunto S. O conjunto $\varnothing$ é um subconjunto de qualquer conjunto.

Exemplo: Se $S = \{1, 2, 3,\}$, $T = \{2, 3\}$ e $X = \{1, 2, 3\}$, então:

T é subconjunto próprio do conjunto S e
X é subconjunto trivial do conjunto S.

Um conjunto é muitas vezes especificado como consistindo de todos os elementos de outro conjunto, que tem uma propriedade. Um conjunto S de todos $t \in T$ tendo a propriedade P denotado:

$$S = \{t\text{: } t \in T \text{ e } t \text{ tem } P\}$$

onde o símbolo : significa «tal que».

Exemplos:

1) $E = \{x : x \in Z \text{ e } x \text{ é par}\}$ então:
$E = \{\ldots, -2, 0, 3, 4, \ldots\}$

2) $\varnothing = \{a : a \in Z \text{ e } a^2 \text{ é negativo}\}$

Chama-se CONJUNTO UNIÃO de dois conjuntos S e T ao conjunto cujos elementos pertencem a S *ou* T. A união dos dois conjuntos é denotada por $S \cup T$ e o conjunto UNIÃO pode ser descrito:

$$S \cup T = \{x\text{: } x \in S \text{ ou } x \in T\}$$

Exemplo: Se $S = \{1, 2, 3\}$ e $T = \{2, 3, 4\}$ então:

$$S \cup T = \{1, 2, 3, 4\}$$

Chama-se CONJUNTO INTERSECÇÃO de dois conjuntos S e T ao conjunto cujos elementos pertencem a S e a T. A intersecção dos dois conjuntos é denotada por $S \cap T$ e o conjunto INTERSEÇCÃO pode ser descrito:

$$S \cap T = \{x : x \in S \text{ e } x \in T\}$$

Exemplo: Se $S = \{1, 2, 3\}$ e $T = \{2, 3, 4\}$, então:

$$S \cap T = \{2, 3\}$$

Chama-se CONJUNTO DIFERENÇA de dois conjuntos S e T ao conjunto cujos elementos pertencem a S e não pertencem a T. A diferença dos dois conjuntos é denotada por S — T e o conjunto DIFERENÇA pode ser descrito:

$$S\text{-}T = \{x : x \in S \text{ e } x \notin T\}$$

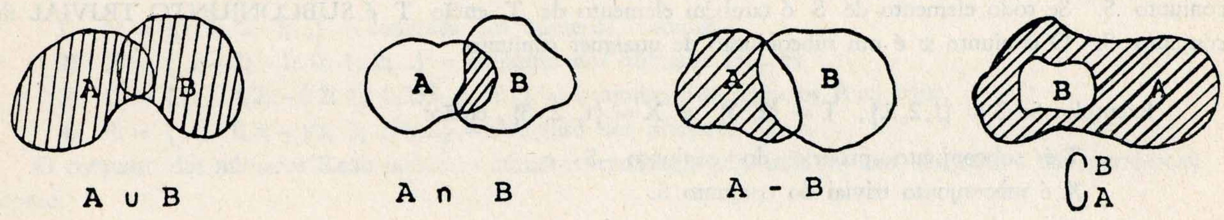
Exemplo: Se $S = \{1, 2, 5, 7, 8\}$ e $T = \{1, 5, 6, 9\}$, então:

$$S\text{-}T = \{2, 7, 8\}$$

Quando T é subconjunto de S , o conjunto S-T é chamado CONJUNTO COMPLEMENTAR de T em relação a S e ele é denotado: C_S^T.

Exemplo: Se $S = \{1, 2, 3\}$ e $T = \{1, 2\}$ então: $C_S^T = \{3\}$

A união, intersecção e a diferença de conjuntos podem ser interpretadas visualmente pelos diagramas de Venn:



Chama-se PRODUTO CARTESIANO de dois conjuntos A e B ao conjunto $A \times B$ cujos elementos são pares ordenados (a, b) onde $a \in A$ e $b \in B$. O conjunto $A \times B$ é, portanto descrito:

$$A \times B = \{(a,b) : a \in A \ \text{ e } \ b \in B\}$$

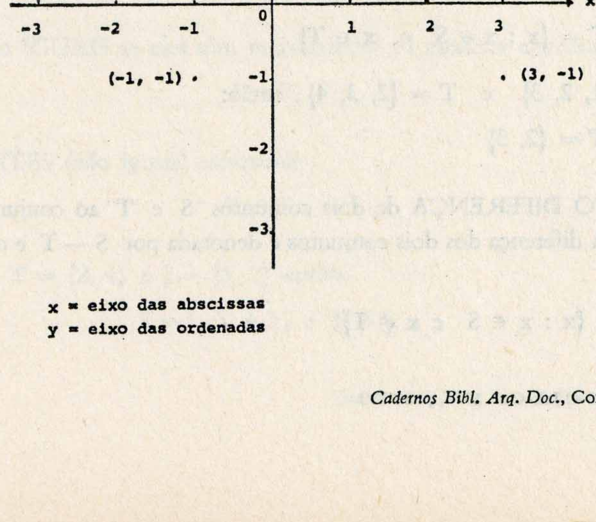
Em (a,b), *a* é denominado 1.ª coordenada (abcissa) e *b* é denominado 2.ª coordenada (ordenada).

Exemplo: Se $A = \{-1, 3\}$ e $B = \{-1, 2\}$, então:

$$A \times B = \{(-1,-1), (-1, 2), (3,-1), (3, 2)\}$$

Se A e B são conjuntos numéricos pode-se usar a representação gráfica cartesiana para $A \times B$, onde cada elemento de $A \times B$ é representado por um ponto.

Exemplo: O conjunto $A \times B$ do exemplo anterior tem a seguinte representação gráfica cartesiana:



x = eixo das abscissas
y = eixo das ordenadas

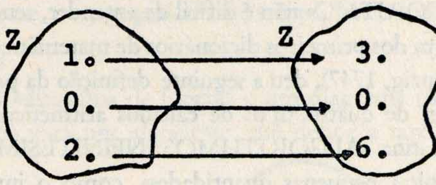
Cadernos Bibl. Arq. Doc., Coimbra, 11 (3-4) Jul.-Out. 1974, p. 88-130

3. FUNÇÕES

Uma função f de um conjunto x em um conjunto y é uma correspondência tal que a cada elemento $x \in X$ corresponde a um único elemento $y \in Y$. Notação: $y = f(x)$ ou $f : x \rightarrow y$.

Os conjuntos x e y são denominados, respectivamente, o DOMÍNIO e o CONTRA-DOMÍNIO de f. Se a é um valor de x então o único elemento de y, associado por f, com a é denotado por f (a) e é chamado o valor de f para $x = a$. A função f é dita ser uma função da variável x.

Exemplo: $x, y \in Z, \; y = 3x$



O domínio e o contra-domínio é o conjunto Z. O valor da função para $x = -2$ e $y = f(-2) = 3(-2) - 6$.

Se $b = f(a)$, então b é chamado IMAGEM de a por f.

Seja $f: X \rightarrow Y$ e $A \subset X$, chama-se IMAGEM de A por f o conjunto $f(A)$ tal que:

$$\boxed{f(A) = \{y : x \in A \;\; e \;\; y = f(x)\}}$$

Exemplo: $X = \{-1, 0, 1\}$, $Y = \{-2, -1, 0, 1, 2\}$. Se $f : X \rightarrow Y$ é $y = x^2$ então o domínio é o conjunto X, o contra-domínio é o conjunto Y e a imagem de X é $f(X) = \{1, 0\}$.

Uma função $f : X \rightarrow Y$ é dita INJECTORA se para todo $y \in Y$ existe no máximo um $x \in X$ tal que $y = f(x)$.

Exemplos:

1) $y = 3x$, $x, y \in R$ é injectora porque cada valor de y é obtido por apenas um valor de x.
2) $y = x^2$, $x, y \in R$ não é injectora porque, por exemplo, tanto para $x = -2$, como para $x = 2$, o valor de y é 4.

Uma função $f : X \rightarrow Y$ é dita SOBREJECTORA se para todo $y \in Y$ existe $x \in X$ tal que $y = f(x)$.

Exemplos:

1) $X = \{1, 2, 3\}$, $Y = \{2, 3, 4\}$,
   $y = x + 1$ é uma função sobrejectora de X em Y.
2) $X = \{1, 2, 3\}$, $Y = \{1, 2, 3, 4\}$,
   $y = x + 1$ não é uma função sobrejectora de X em Y.

Se $f : X \rightarrow Y$ é uma função injectora e sobrejectora então ela é chamada função BIJECTORA.

Exemplo: $f : R \rightarrow R$ tal que $y = 3x$ é sobrejectora.

4. ALGORITMOS E FLUXOGRAMAS

4.1. ALGORITMOS

A noção de algoritmo é básica para toda programação em computador; assim, devemos iniciar com uma cuidadosa análise deste conceito.

A palavra «ALGORITMO» é com efeito interessante; numa primeira olhada podemos enxergá-la como alguém que, intencionado em escrever «LOGARITMO», confunde as quatro primeiras letras. A palavra aparecia no «WEBSTER'S NEW WORLD DICTIONARY» antes de 1957 com a velha forma «ALGORISMO» e seu antigo significado, o processo de aplicar aritmética usando numerais arábicos. Na idade média, abacistas computavam em ÁBACOS e algoristas computavam em «ALGORISMOS». Seguindo a idade média, a origem desta palavra foi duvidosa, e experiências anteriores de linguística supõem esta derivação fazendo combinações como «ALGIROS» (trabalhoso) + «ARITHMOS» (números); outras citações dizem que a palavra veio do «King Algor of Castile». Finalmente, historiadores de matemática acharam a verdadeira origem da palavra ALGORISMO: ela veio do nome de um famoso autor de um texto arábico, ABU JA'FAR MOHAMMED ibn Mûsâ al-Khowârizmî (c. - 825) — literalmente, Pai de JA'FAR, Mohammed, filho de Moisés, natural de Khowârizmî. Al-Khowârizmí escreveu o célebre livro KIBAT AL JABR W'AL-MUQABALA («Regras de restauração e redução»); uma outra palavra «ALGEBRA», originou-se do título de seu livro, embora o livro não fosse realmente muito algébrico.

Gradualmente, a forma e a idéia de «ALGORISMO» deturpou-se; como exposto pelo «OXFORD ENGLISH DICTIONARY», a palavra foi «reformulada erroneamente» por «CONFUSÃO LÓGICA» com a palavra ARITMÉTICA. A mudança de «ALGORISMO» para «ALGORITMO» não é difícil de entender, tendo em vista o facto de que o povo esqueceu a original derivação da palavra. Um dos primeiros dicionários de matemática, alemão, «VOLLSTANDIGES MATEHEMATISCHES LEXICON» (Leipzig, 1747), deu a seguinte definição da palavra «ALGORITMOS»: «Sobre esta designação estão combinadas as noções de quatro tipos de cálculos aritméticos, nominalmente, adição, multiplicação, subtracção e divisão». A expressão latina «ALGORITHMOS INFINITESIMALIS» era o termo usado para denominar «Caminhos de cálculos com infinitas pequenas quantidades», como o inventado por LEIBNITZ.

Após 1950, a palavra algoritmo foi frequentemente associada com «ALGORITMO EUCLIDIANO», um processo para encontrar o maior divisor comum de dois números, o qual aparece no livro ELEMENTOS EUCLIDIANOS.

Mas, o que é realmente um algoritmo? Um algoritmo é uma lista de instruções para se executar algum processo, passo a passo. Uma receita de bolo é um excelente exemplo de algoritmo. A preparação do bolo é quebrada em passos simples de tal modo que qualquer cozinheiro possa entendê-la. Algoritmos para grandes processos são, sem dúvida, muito complexos, mas são sempre constituídos de peças, como no exemplo abaixo.

O facto de conseguirmos imaginar um algoritmo para um processo basta para que possamos modificá-lo de diferentes maneiras. Aqui está um algoritmo para o processo trivial de trocar um pneu furado.

E.1. — Levante o carro
E.2. — Desaparafuse a roda
E.3. — Retire-a
E.4. — Coloque o estepe
E.5. — Reaperte os parafusos
E.6. — Abaixe o carro.

Podemos adicionar muitos detalhes a este algoritmo. Podemos incluir os passos de pegar o material no porta--malas, retirar a calota, etc.. Para algoritmos descrevendo processos é melhor, antes de começar, decidir o quanto de detalhes se vai incluir. Quando trabalhamos com algoritmos matemáticos, temos que ser muito mais precisos.

Todo algoritmo deverá receber um nome, ou uma letra de identificação, e os seus passos serão identificados por este nome seguido de um número (exemplo: E.1, E.2, etc.).

Cada passo de um algoritmo será representado por uma frase, tão breve quanto possível, a qual denota o conteúdo principal do passo.

Um algoritmo começa no passo com numeração mais baixa, geralmente passo 1, e os passos são executados em ordem sequencial, excepto se for especificado de outra maneira.

O significado moderno para algoritmo é quase similar ao de receita, processo, método, técnica, procedimento, rotina, com a excepção de que a palavra «algoritmo» significa alguma coisa diferente. Ao lado de ser meramente um conjunto finito de regras, as quais dão uma sequência de operações para solucionar um tipo específico de problema, um algoritmo tem cinco características importantes:

a — finitude: — um algoritmo deve sempre terminar após um número finito de passos. Um procedimento que tem todas as características de um algoritmo, excepto sua possivelmente carência de finitude pode ser chamado um «método computacional». Euclides desenvolveu, ao lado de seu algoritmo de maior divisor comum de dois inteiros, um outro para obter a «maior dimensão comum» dos comprimentos de duas linhas; este é um método computacional que não termina se os comprimentos dados são incomensuráveis.

b — definitude: — cada passo de um algoritmo deve ser precisamente definido; as acções a serem executadas devem ser rigorosa e inambiguamente especificadas para cada caso.

c — entrada: — um algoritmo tem uma ou mais entradas, isto é, quantidades as quais são dadas a ele inicialmente antes do algoritmo iniciar. Estas entradas são tomadas dos conjuntos específicos de objetos.

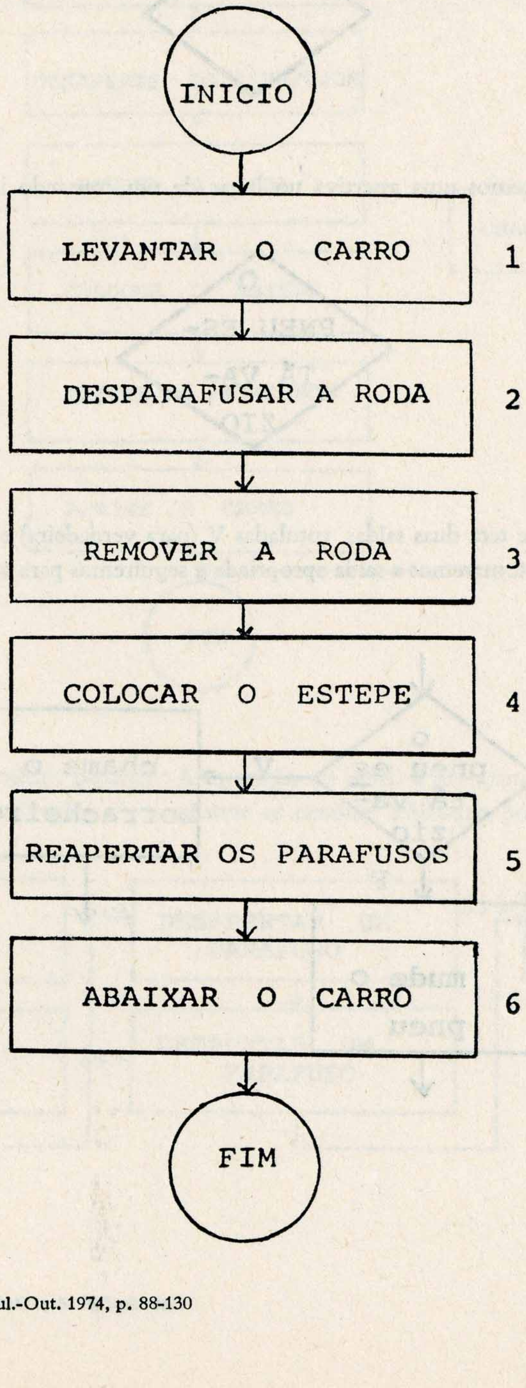
d — saída: — um algoritmo tem uma ou mais saídas, isto é, quantidades que tem uma relação específica com as entradas.

e — efectividade: — um algoritmo é também geralmente pensado para ser efectivo. Isto significa que todas as operações a serem executadas no algoritmo devem ser suficientemente básicas de tal modo que elas possam em princípio ser feitas exactamente, e num período de tempo finito, por um homem usando lápis e papel.

4.2. FLUXOGRAMAS

Um «fluxograma» ou «diagrama de blocos» é um diagrama para representar um algoritmo. Na figura abaixo vemos o diagrama de blocos para um algoritmo que define a tarefa de trocar um pneu furado.

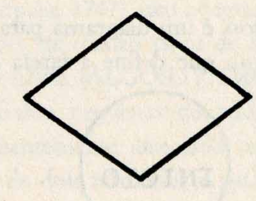
FIGURA A — Fluxograma do pneu furado



Os símbolos INÍCIO e FIM no diagrama de blocos nos lembram os botões para ligar e desligar uma máquina. Cada instrução no «diagrama de blocos está contida numa caixa» ou bloco. Como logo veremos, o conteúdo da caixa indica um comando para executar alguma acção.

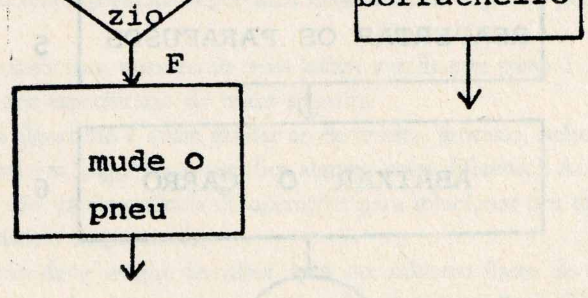
Para se atingir o objetivo proposto pelo diagrama de blocos, começamos no bloco de INÍCIO e seguimos as setas de caixa para caixa, executando as instruções assim que chegamos a elas.

Depois de elaborar o primeiro diagrama de blocos, devemos analisá-lo para verificar se conseguimos melhorá-lo. Por exemplo, no diagrama do pneu furado deixamos de verificar se o estepe estava vazio. Se o estepe estivesse vazio, deveríamos chamar o borracheiro antes de trocá-lo. Isto traz a necessidade de uma decisão entre dois cursos de acção. Para este fim nós introduzimos uma nova forma de bloco no nosso diagrama.

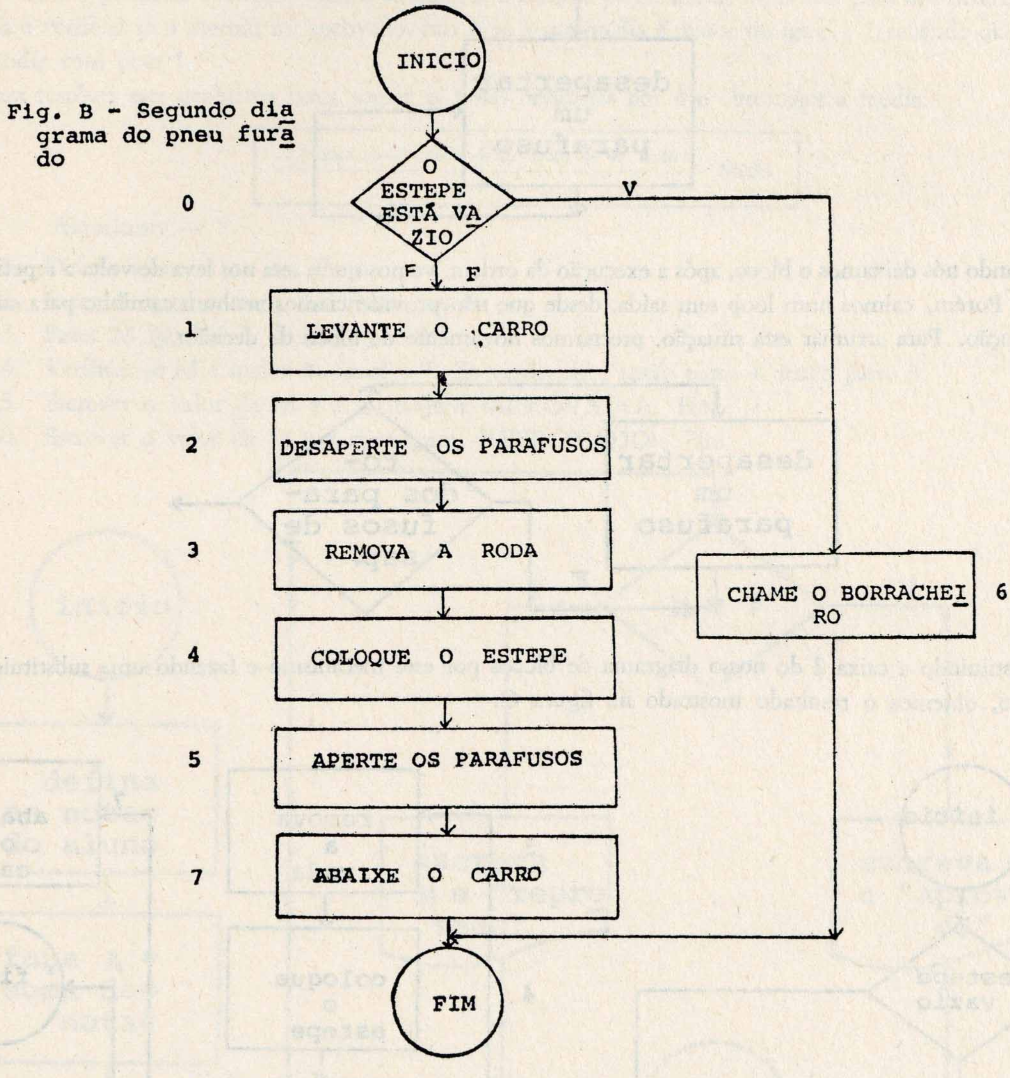


Dentro do bloco, escreveremos uma assertiva no lugar de um comando imperativo.

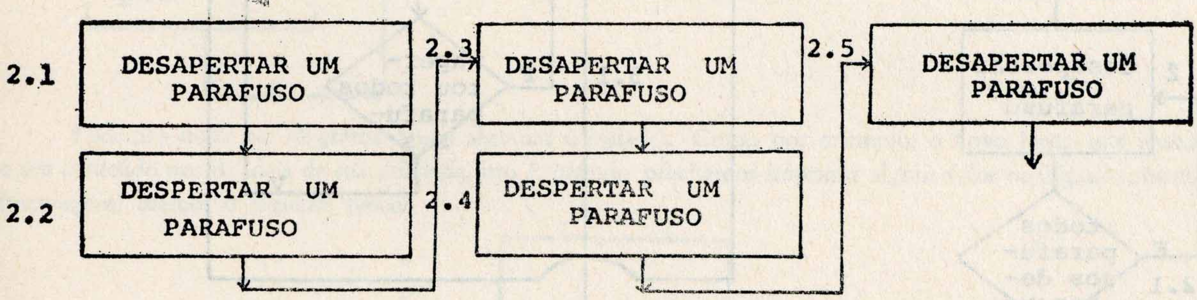
Este é o bloco de decisão e terá duas saídas, rotuladas V (para verdadeira) e F (para falsa). Depois de testar a verdade ou falsidade da assertiva, tomaremos a saída apropriada e seguiremos para a actividade indicada pela seta.



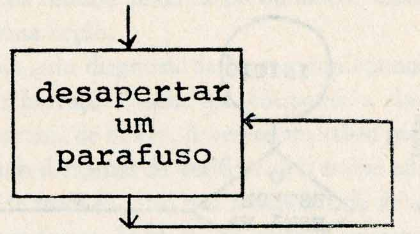
Encaixando este fragmento de bloco na fig. A obtemos o diagrama de blocos da figura B.



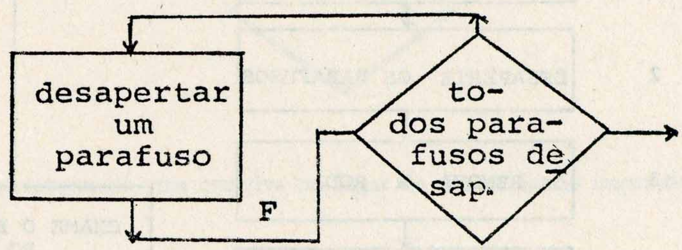
Existe ainda outro melhoramento possível. A instrução no bloco 2 do nosso diagrama serve para indicar um número de repetições da mesma operação. Para mostrar os detalhes adicionais podemos substituir o bloco 2 por:



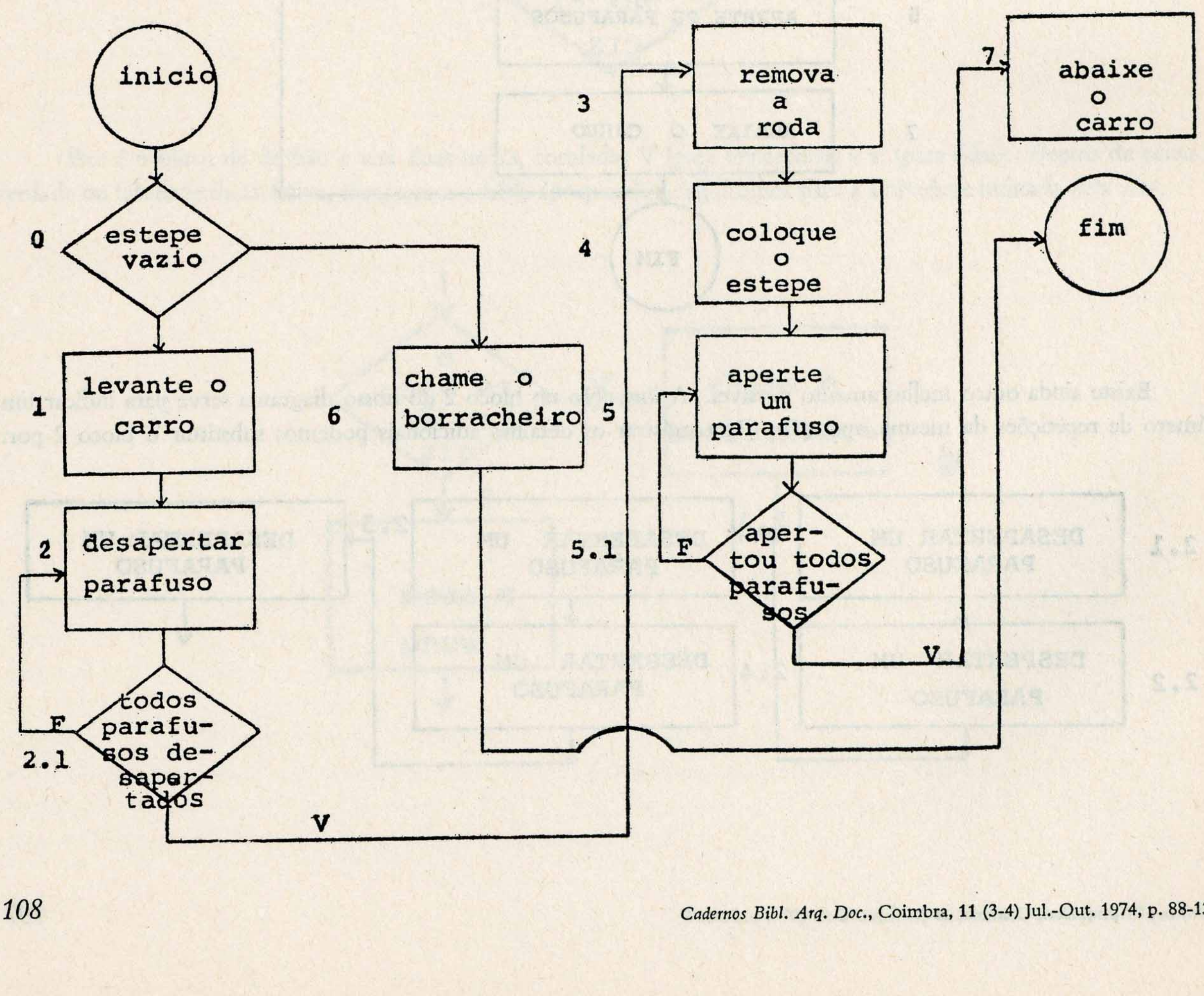
A inconveniência desta instrução repetida pode ser eliminada pela instrução de um ciclo (loop).



Quando nós deixamos o bloco, após a execução da ordem, vemos que a seta nos leva de volta a repetir a mesma instrução. Porém, caimos num loop sem saída, desde que não providenciamos nenhum caminho para sair e ir para outra instrução. Para arrumar esta situação, precisamos novamente de bloco de decisão:



Substituindo a caixa 2 do nosso diagrama de blocos por este mecanismo e fazendo uma substituição similar no bloco 5, obtemos o resultado mostrado na figura C.



 Cadernos Bibl. Arq. Doc., Coimbra, 11 (3-4) Jul.-Out. 1974, p. 88-130

4.3. ALGORITMO NUMÉRICO

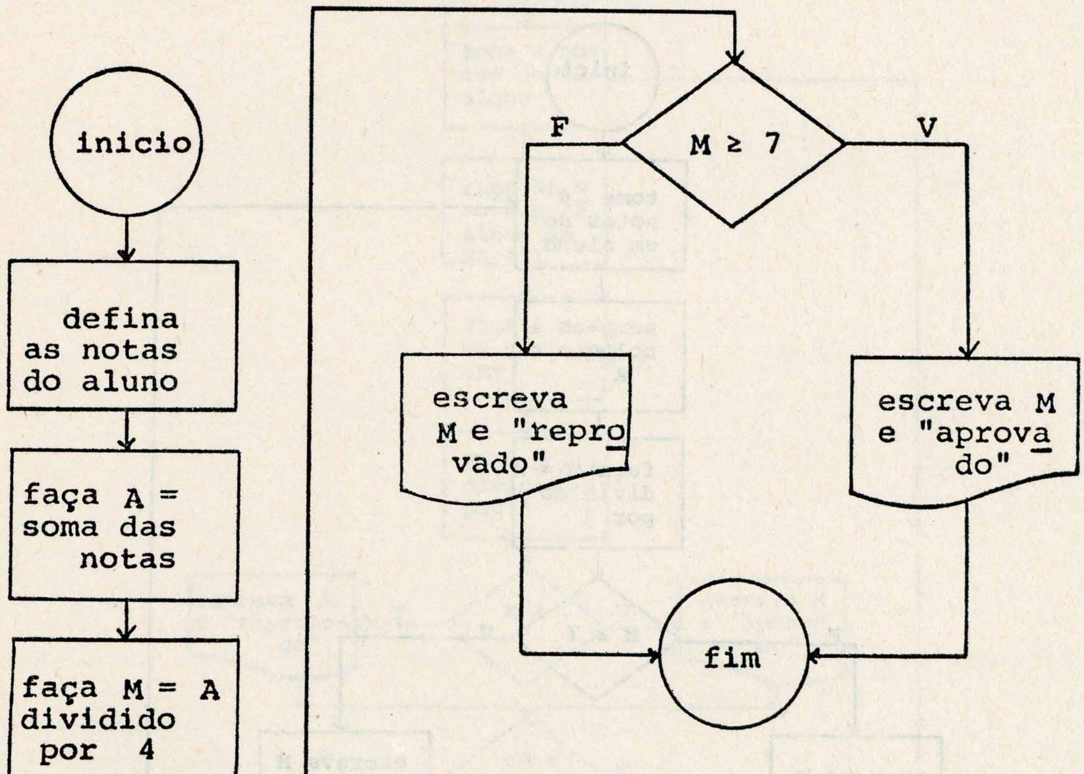
Como um primeiro exemplo, vamos considerar o simples problema de encontrar para um determinado aluno sua média e verificar se o mesmo foi aprovado, isto é, se a sua média é maior ou igual a 7, sabendo que o aluno tem 4 notas todas com peso 1.

Para resolver este problema basta somar as notas dividi-las por 4 e obteremos a média.

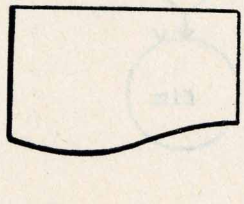
$$\boxed{\frac{1.^a \text{ nota} + 2.^a \text{ nota} + 3.^a \text{ nota} + 4.^a \text{ nota}}{4} = \text{Média}}$$

Algoritmo:

E.1. Definir as notas de um aluno
E.2. Fazer A igual à soma das quatro notas
E.3. Fazer M igual a A dividido por 4
E.4. Verificar se M é maior ou igual a 7. Se verdadeiro, então passo 4, senão passo 5.
E.5. Escrever o valor de M e a mensagem «APROVADO». Fim
E.6. Escrever o valor de M e a mensagem «REPROVADO». Fim.



Podemos notar no diagrama acima algumas novidades. Como por exemplo, o novo bloco que apareceu; o seu conteúdo nos dá ideia de sua utilidade, isto é, quando precisamos imprimir algum valor ou algum comentário (mensagem) usamos o seguinte bloco:



5. VARIÁVEIS E CONSTANTES

Aparecem também letras assumindo valores; no exemplo, A assumiu o valor da soma das 4 notas e M o valor da média. São chamadas VARIÁVEIS, e podem assumir o valor de um número, como também de um conjunto de números, e ainda um conjunto de valores alfabéticos ou alfanuméricos.

Considerando o cálculo da média de quatro notas para vários alunos, e não para um único, teremos:

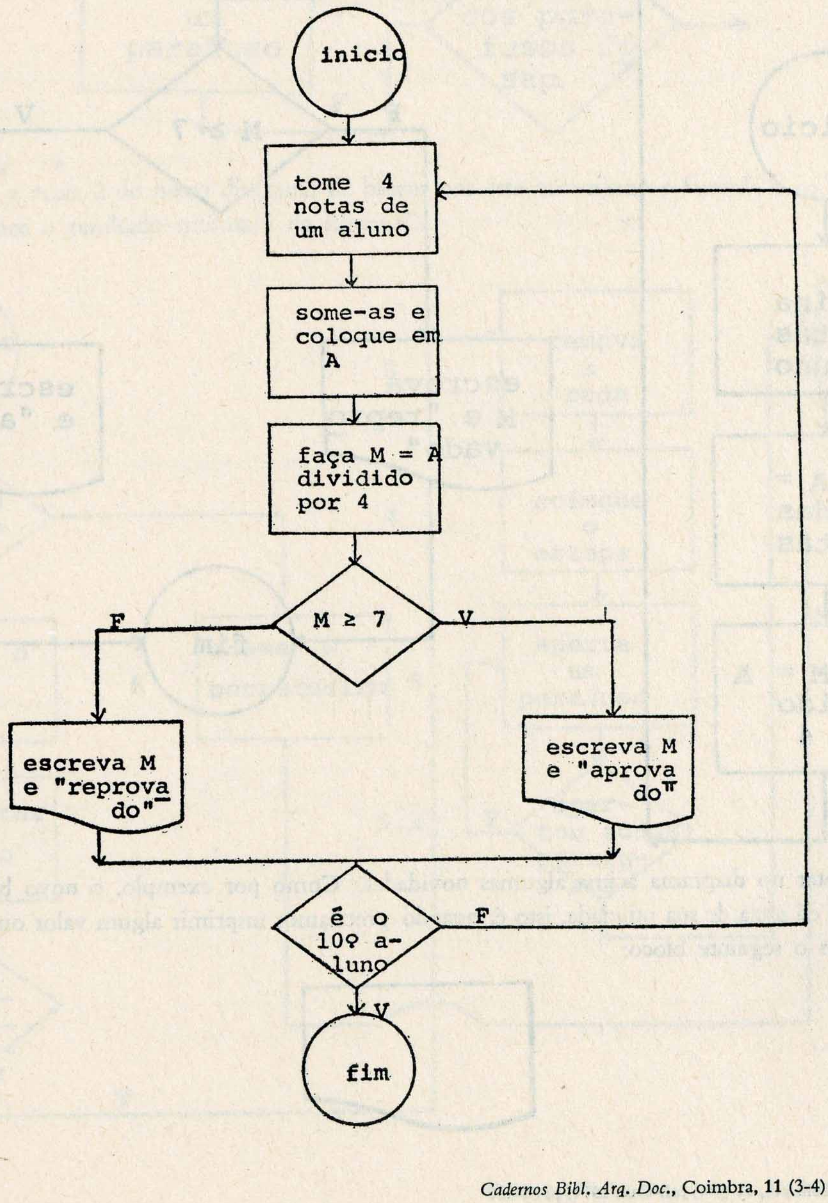
A — como a soma das notas, as quais variam de um aluno para outro.

M — como média, a qual varia de um aluno para outro.

O divisor 4, que aparece para todos alunos, sem se alterar, chamamos de CONSTANTE. Esta será sempre um valor numérico ou seja um número; esse tipo de informação não apresenta a necessidade de associação com uma variável.

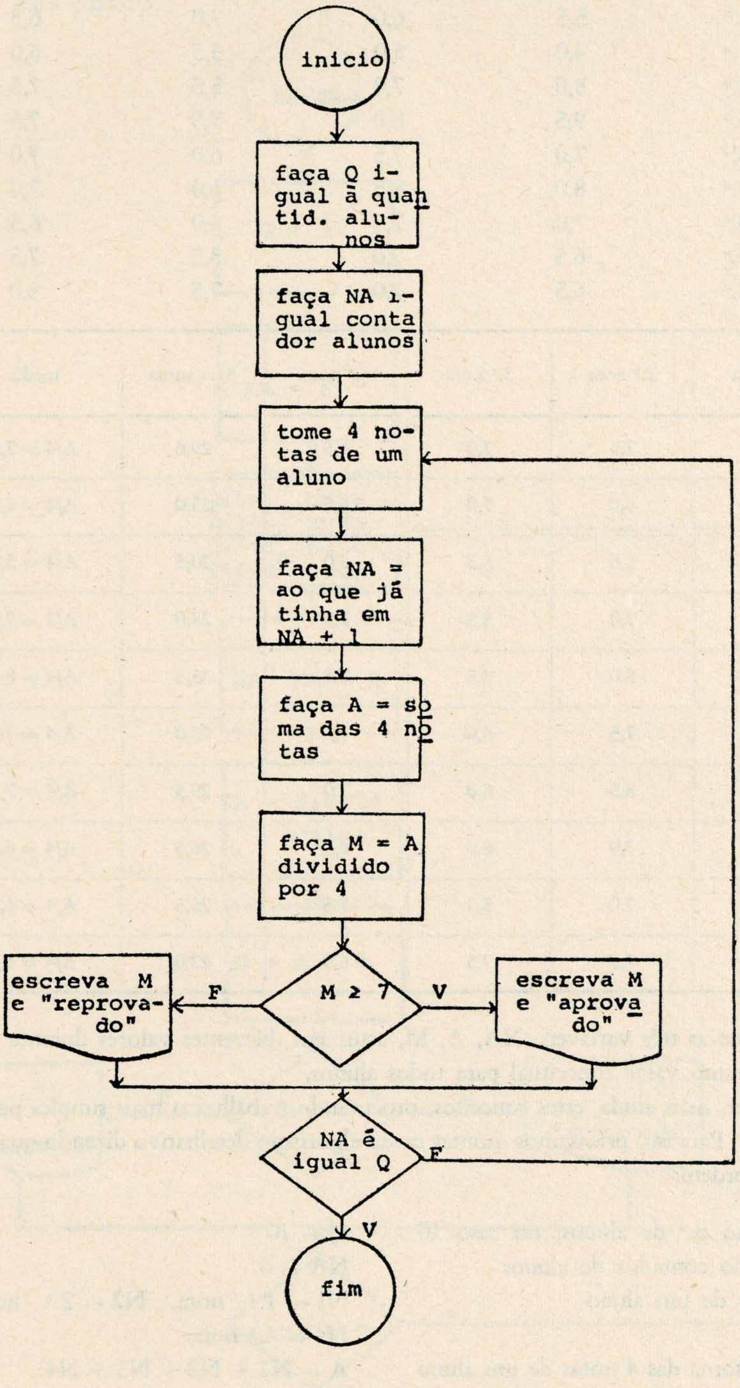
Para fazer uso de uma variável ou de uma constante devemos verificar a natureza da informação; se houver mudança de seu valor durante o processamento ela deve ser representada por uma variável; caso contrário, e desde que seja um valor numérico, deverá ser introduzida sob forma de constante.

Para entendermos melhor estes conceitos vamos exemplificá-los usando o problema da média para 10 alunos.



Cadernos Bibl. Arq. Doc., Coimbra, 11 (3-4) Jul.-Out. 1974, p. 88-130

Podemos notar que o número de alunos varia de 1 até 10; poderíamos, então, melhorar nosso diagrama fazendo o número de alunos ser representado por uma variável; com esse procedimento o diagrama ficaria generalizado para um número qualquer de alunos, e não apenas para 10.



Podemos ver a utilidade e o recurso que as variáveis nos dão. Para completar a explicação sobre variável, vamos mostrar o que acontece durante o processamento do nosso diagrama, supondo que os alunos tenham as seguintes notas:

Aluno	1.ª nota	2.ª nota	3.ª nota	4.ª nota
1.º	8,5	7,5	7,0	6,5
2.º	5,5	6,0	7,0	6,5
3.º	4,0	5,0	5,5	6,0
4.º	8,0	7,0	5,5	7,5
5.º	9,5	8,0	8,5	7,5
6.º	7,0	7,5	6,0	7,0
7.º	8,0	8,5	6,0	7,0
8.º	7,0	7,0	6,0	6,5
9.º	6,5	7,0	5,5	7,5
10.º	6,5	7,0	7,5	6,0

NA = n.º 10 aluno	1.ª nota	2.ª nota	3.ª nota	4.ª nota	A = soma	média	situaç.
1	8,5	7,5	7,0	6,5	29,6	A/4 = 7,4	aprov.
2	5,5	6,0	7,0	6,5	25,0	A/4 = 6,3	reprov.
3	4,0	5,0	5,5	6,0	20,5	A/4 = 5,1	reprov.
4	8,0	7,0	5,5	7,5	28,0	A/4 = 7,0	aprov.
5	9,5	8,0	8,5	7,5	33,5	A/4 = 8,4	aprov.
6	7,0	7,5	6,0	7,5	28,0	A/4 = 7,0	aprov.
7	8,0	8,5	6,0	7,0	29,5	A/4 = 7,5	aprov.
8	7,0	7,0	6,0	6,5	26,5	A/4 = 6,6	reprov.
9	6,5	7,0	5,5	7,5	26,5	A/4 = 6,6	reprov.
10	6,5	7,0	7,5	6,0	27,0	A/4 = 6,7	reprov.

Podemos notar que as três variáveis, NA, A, M, assumem diferentes valores durante o processamento, mas são utilizados com o mesmo valor conceitual para todos alunos.

Podemos esclarecer, mais ainda, estes conceitos, procurando trabalhar o mais simples possível com os mesmos e os diagramas de blocos. Para isto precisamos montar nosso algoritmo detalhativo dizendo quais os passos que devem ser seguidos e em qual ordem:

R. 1 — fazer Q igual ao n.º de alunos, no caso 10	Q ← 10
R. 2 — faça NA igual ao contador de alunos	NA ← 0
R. 3 — tome as 4 notas de um aluno	N1 ← 1.ª nota, N2 ← 2.ª nota, N3 ← 3.ª nota, N4 ← 4.ª nota.
R. 4 — faça A igual à soma das 4 notas de um aluno	A ← N1 + N2 + N3 + N4
R. 5 — faça NA igual ao que já tinha em NA + 1	NA ← NA + 1
R. 6 — faça M igual a A dividido por 4	M ← A/4
R. 7 — verifique se M é maior ou igual 7, se for, vá para a ordem 7.ª, senão 8.ª.	Se M $\geqslant$ 7 então passo 7, senão passo 8.

R. 8 — escreva o valor de M e «APROVADO». Vá para passo 10.

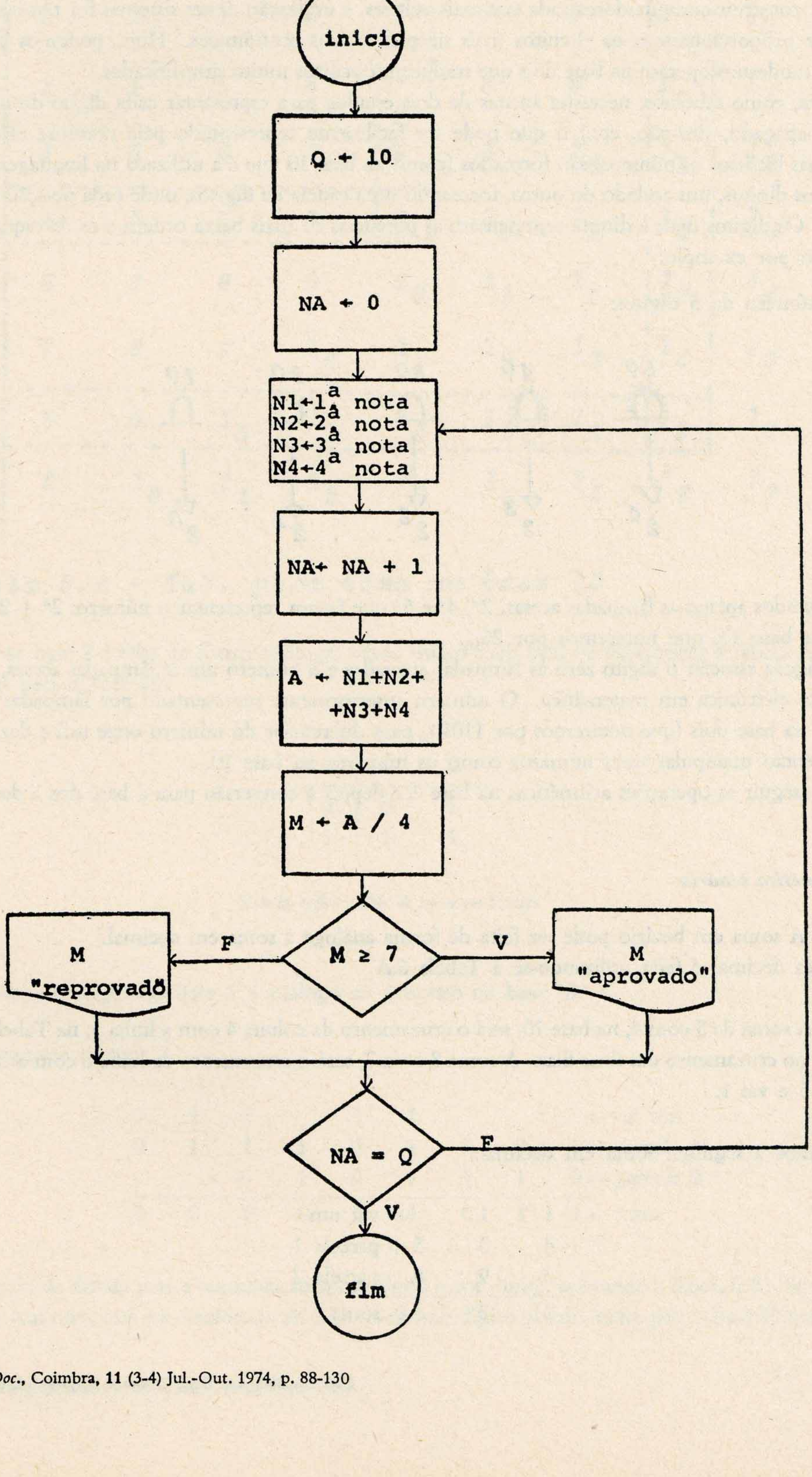
Imprima M e APROVADO. Vá para passo 10.

R. 9 — escreva o valor de M e «REPROVADO»

Imprima M e REPROVADO.

R.10 — verifique se NA é igual a Q; se for vá para FIM, senão vá para passo 3.

Se NA = Q, então FIM, senão passo 3.



6. SISTEMAS NUMÉRICOS

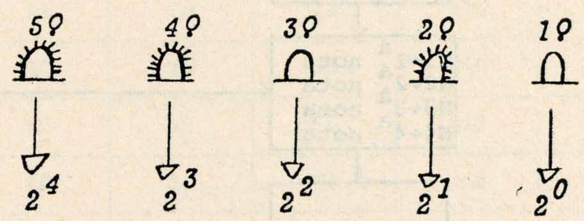
6:1. *Números binários*

Com o aparecimento dos computadores, desenvolveu-se uma grande quantidade de sistemas numéricos. Uns computadores operavam na base cinco, outros na base oito, dez, dezasseis, etc.. Com o avanço da tecnologia e a necessidade de se construir computadores cada vez mais velozes, a utilização desses sistemas foi também caminhando para sistemas que proporcionassem os circuitos mais simples e mais económicos. Hoje, podemos dizer que todos os computadores modernos operam na base dois que traduzem circuitos muito simplificados.

A base dois, como sabemos, necessita apenas de dois estados para representar cada dígito do número (ligado--desligado, aceso-apagado, sim-não, etc.) o que pode ser facilmente representado pela corrente eléctrica.

Nos sistemas binários os números são formados (como na base 10 que é a utilizada na linguagem comum) pela colocação de vários dígitos, um ao lado do outro, formando uma cadeia de dígitos, onde cada posição representa uma potência de dois. Os dígitos mais à direita representam as potências de mais baixa ordem e os da esquerda os de mais alta ordem, como por exemplo:

Seja um número de 5 dígitos:



São computados apenas as lâmpadas acesas, 2ª, 4ª e 5ª que fazem representar o número: $2^4 + 2^3 + 2^1 = 16 + 8 + 2 = 26$ na base 10, que notaremos por 26_{10}.

Podemos agora associar o dígito zero às lâmpadas apagadas e o número um às lâmpadas acesas, transformando uma representação eletrónica em matemática. O número anteriormente representado por lâmpadas é representado agora por 11010, na base dois (que notaremos por 11010_2 para diferenciar do número onze mil e dez).

Podemos então manipular esses números como os números na base 10.

Veremos a seguir as operações aritméticas na base 2 e depois a conversão para a base dez e dezasseis.

6.2. *Aritmética binária*

SOMA: A soma em binário pode ser feita de forma análoga à soma em decimal.

A soma em decimal é feita utilizando-se a Tabela 6.A.

Exemplo: A soma de 5 com 4, na base 10, será o cruzamento da coluna 4 com a linha 5, na Tabela 6.A. Obteremos o número 9, no cruzamento das duas filas. A soma 8 com 7, será o cruzamento da linha 8 com a coluna 7, obtendo-se o número 5 e vai 1.

Consideramos a seguinte soma em decimal:

1	1		← vai um
4	3	5 +	parcela 1
3	9	8	parcela 2
8	3	3 ←	soma

Cadernos Bibl. Arq. Doc., Coimbra, 11 (3-4) Jul.-Out. 1974, p. 88-130

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	1_0
2	2	3	4	5	6	7	8	9	1_0	1_1
3	3	4	5	6	7	8	9	1_0	1_1	1_2
4	4	5	6	7	8	9	1_0	1_1	1_2	1_3
5	5	6	7	8	$\boxed{9}$	1_0	1_1	1_2	1_3	1_4
6	6	7	8	9	1_0	1_1	1_2	1_3	1_4	1_5
7	7	8	9	1_0	1_1	1_2	1_3	1_4	1_5	1_6
8	8	9	1_0	1_1	1_2	1_3	1_4	$\boxed{1_5}$	1_6	1_7
9	9	1_0	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_8

Tabela 6.A - Tab. para soma na base 10

A soma na base 2 é feita de forma análoga, sendo muito mais fácil de memorizar a tabela da soma nesta base do que a tabela para a base 10.

+	0	1
0	0	1
1	1	1_0

Tabela 6.B — *Tab. de soma em binário*

O processo de adição na base 2 é análogo ao processo na base 10.

Seja a seguinte soma:

1	1	1		1	1				← vai um
0	1	1	1	0	1	1	0	1	← parcela 1
		+	1	0	1	1	1	0	← parcela 2
1	0	0	0	1	1	0	1	1	← soma

Começando da direita para a esquerda, soma-se dígito e por dígito, aplicando a tabela 6.B. Se a soma do último dígito tiver um «vai um», esse *um* é colocado ao lado do último dígito obtido, tanto para a base 10 como para a base 2.

Nota-se que na adição em binário, o dígito «vai um» frequentemente se propaga por quase todo o número, como pode ser visto no seguinte exemplo:

	1	1	1	1	1	1	1	1	1		← «vai um»
	1	1	0	1	1	1	0	0	1	1	parcela 1
		+	1	1	0	1	1	1	0	1	parcela 2
1	0	0	0	1	0	1	0	0	0	0	← soma

SUBTRACÇÃO: O processo de subtracção em binário também não traz nenhuma dificuldade, se bem que para um principiante traga mais dificuldade que a adição.

Temos para a adição, o seguinte conjunto de regras:

$$0 + 0 = 0$$
$$1 + 0 = 1$$
$$0 + 1 = 1$$
$$1 + 1 = 10$$

Podemos obter daí as regras para a subtracção, uma vez que a subtracção é a operação inversa da adição:

$$0 - 0 = 0$$
$$1 - 0 = 1$$
$$1 - 1 = 0$$
$$10 - 1 = 1$$

A diferença (0 — 1) pode ser obtida pelo processo usual de «emprestar 1» do dígito vizinho mais próximo à esquerda e depois utilizar a regra (10 — 1) = 1.

Exemplo:

2 15			0	1	1	10		
35	← minuendo	→	1	0	0	0	1	1
− 7	← subtraendo	→				1	1	1
28	← diferença	→	0	1	1	1	0	0

			0	10	10		
− 27	← minuendo	→	1	1	0	1	1
15	← subtraendo	→	−	1	1	1	1
12	← diferença	→		1	1	0	0

MULTIPLICAÇÃO: A tabela utilizada para a multiplicação na base dois é muito mais simples do que aquela utilizada na base 10.

X	0	1
0	0	0
1	0	1

Cadernos Bibl. Arq. Doc., Coimbra, 11 (3-4) Jul.-Out. 1974, p. 88-130

A memorização é muito simples porque a resposta apresenta somente dois dígitos ou é 1 ou 0.

Exemplo:

$$
\begin{array}{ccccccccccc}
 & & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
 & & & & & & & & 1 & 0 & 1 \\
\hline
 & & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
 & + & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & & \\
\hline
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}
\qquad\qquad
\begin{array}{r}
221 \\
\times\, 5 \\
\hline
1105
\end{array}
$$

A dificuldade deste método está no acúmulo de somas que por facilidade poderão ser efetuadas por etapas como por exemplo:

$$
\begin{array}{cccccccccccccc l}
 & & & & & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & \\
 & & & & & & & \times & 1 & 1 & 1 & 0 & 1 & 1 & \\
\hline
 & & & & & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & \\
 & & & & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & & \\
\hline
 & & & & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & \to 1.^{a} \text{ soma parcial} \\
 & & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & & & & \\
\hline
 & & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \to 2.^{a} \text{ soma parcial} \\
 & & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & & & & & \\
\hline
 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & \to 3.^{a} \text{ soma parcial} \\
 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & & & & & & \\
\hline
1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & \to \text{produto} = 13.039_{10}
\end{array}
$$

DIVISÃO: A divisão é efectuada de maneira análoga à multiplicação. Aplicando o processo inverso (idêntico à divisão na base dez).

Teremos uma sequência sucessiva de subtrações.

Exemplo:

= decimal =

$$
\begin{array}{r|l}
39 & 3 \\
\cline{2-2}
3\downarrow & 13 \\
\cline{1-1}
0\downarrow & \\
9 & \\
9 & \\
\cline{1-1}
0 &
\end{array}
$$

= binário =

$$
\begin{array}{rr|ll}
\text{dividendo} \to & 100111 & 11 & \text{divisor} \\
\cline{3-3}
 & 11\downarrow & 1101 & \text{quociente} \\
\cline{2-2}
 & 0011 & & \\
 & 11\downarrow & & \\
\cline{2-2}
 & 001 & & \\
 & 000\downarrow & & \\
\cline{2-2}
 & 0011 & & \\
 & 11 & & \\
\cline{2-2}
 & 0000 & \text{resto} &
\end{array}
$$

6.3. ***Conversão de Bases***

6.3.1. ***Binário para decimal***

Método 1: Esse método é muito simples e basta expandir o número binário em potências de dois.

Exemplo:

$$
\begin{aligned}
110111_2 &= 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = \\
&= 32 + 16 + 4 + 2 + 1 = 55_{10}
\end{aligned}
$$

$$
\begin{aligned}
1101101_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^3 = \\
&= 8 + 4 + 0 + 1 + \tfrac{1}{2} + 0 + \tfrac{1}{8} = \\
&= 13 + 0{,}5 + 0{,}125 \quad = \quad 13{,}625
\end{aligned}
$$

Método 2: Esse método pode trazer um pouco de confusão para o principiante, que com um pouco de prática pode fazer a conversão mais rapidamente que o método anterior. A vantagem desse método sobre o anterior é que dispensa a memorização das potências de dois.

Exemplo:

$$\begin{array}{l}
1\quad 1\quad 0\quad 1\quad 1\quad 1 \\
\times 2 + 1 \\
\overline{\quad 3 \quad} \\
\quad \times 2 + 0 \\
\quad \overline{\quad 6 \quad} \\
\qquad \times 2 + 1 \\
\qquad \overline{\quad 13 \quad} \\
\qquad\quad \times 2 + 1 \\
\qquad\qquad 27 \\
\qquad\qquad \times 2 + 1 \\
\qquad\qquad\quad 55_{10}
\end{array}
\qquad 110111_2 = 55_{10}$$

O algoritmo que define esse método pode ser dado da seguinte forma:

BD2.1: Multiplica-se o dígito binário de mais alta ordem por dois e a soma com o próximo dígito à direita.

BD2.2: Multiplica-se a soma por dois e soma com o próximo dígito à direita.

BD2.3: Repete-se o passo BD2.2 até o último dígito (de mais baixa ordem) ser somado.

BD2.4: O valor da soma obtido é o correspondente em decimal do número binário.

O diagrama de blocos da página seguinte, ilustra melhor o processo.

Exemplo:

N.º binário	1	1	0	1	1	1	0	1
Produtos		2	6	12	26	54	110	220
Somas		3	6	13	27	55	110	$\boxed{221}$

Resultado: $11011101_2 = 221_{10}$

6.3.2. *Decimal para binário*

Método 1: Esse método é o inverso do método 1 para conversão de binário para decimal. Para executar a operação inversa é conveniente que se tenha uma tabela com as potências de dois:

$2^0 = 1$	$2^5 = 32$
$2^1 = 2$	$2^6 = 64$
$2^2 = 4$	$2^7 = 128$
$2^3 = 8$	$2^8 = 256$
$2^4 = 16$	$2^9 = 512$

Esse processo consiste em transformar o número em somas sucessivas das potências de dois; para isso escolhe-se a maior potência de dois, menor que o número e efectua-se a subtração e assim sucessivamente obtendo o número em binário:

Exemplo: transformar o número 55_{10}

$$\left.\begin{array}{r} 55 \\ -32 \rightarrow 2^5 \\ \hline 23 \\ -16 \rightarrow 2^4 \\ \hline 7 \\ -\ 4 \rightarrow 2^2 \\ \hline 3 \\ -\ 2 \rightarrow 2^1 \\ \hline 1 \\ -\ 1 \rightarrow 2^0 \\ \hline 0 \end{array}\right\} = 2^5 + 2^4 + 2^2 + 2^1 + 2^0 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 110111^{a}$$

Portanto $55_{10} = 110111_2$

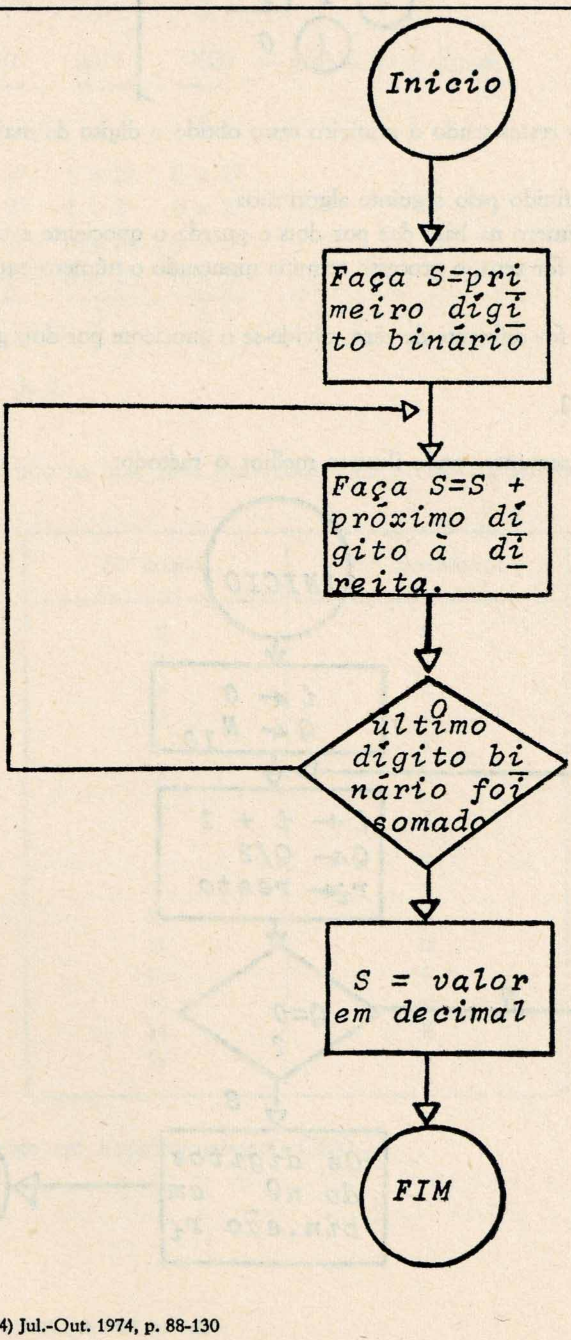
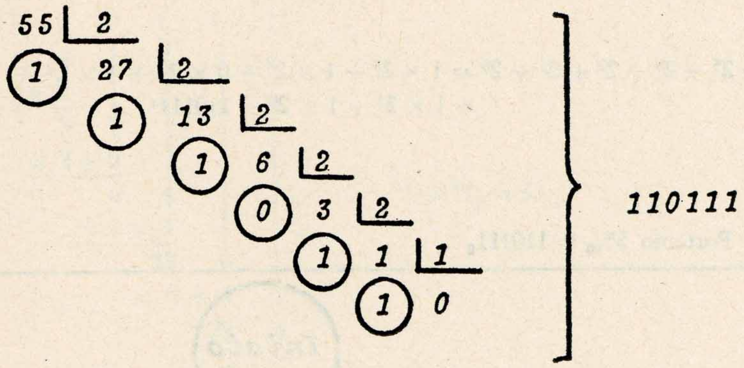


Fig. C — Binário para decimal método 2, 6.3.1.

Método 2: Esse método envolve sucessivas divisões por dois e é o inverso do método 2 para o conversão de binário para decimal. A vantagem desse método sobre o anterior é ainda a não necessidade da memorização das potências de dois.

Exemplo:



Toma-se o conjunto dos restos sendo o primeiro resto obtido o dígito de mais baixa ordem e o último o de mais alta ordem.

O processo pode ser definido pelo seguinte algoritmo:

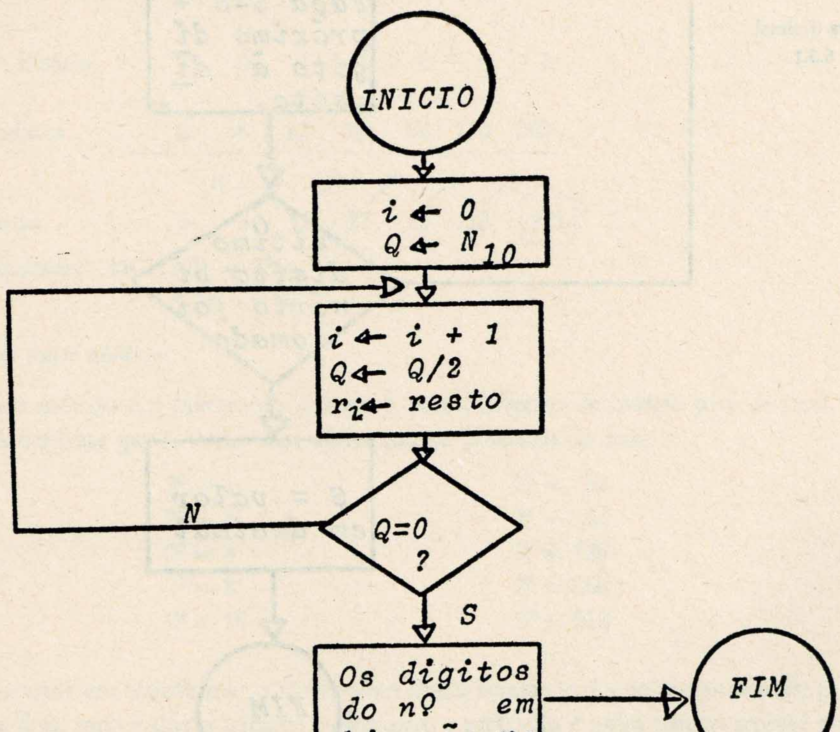
DB2.1: Divide-se o número na base dez por dois e guarda o quociente e o resto.

DB2.2: Se o quociente for zero, o processo termina montando o número equivalente em binário a partir dos restos obtidos.

DB2.3: Se o quociente for diferente de zero, divide-se o quociente por dois guardando o quociente e o resto da divisão.

DB2.4: Vá para DB2.2.

O diagrama de blocos seguinte, pode ilustrar melhor o método:



Cadernos Bibl. Arq. Doc., Coimbra, 11 (3-4) Jul.-Out. 1974, p. 88-130

6.3.3. *Binário para hexadecimal*

Apesar dos computadores utilizarem a base dois para o processamento aritmético, normalmente é feita a conversão das mensagens que seriam em binário, para uma outra base. Para isso a base 16 é muito usada porque a conversão é feita facilmente.

Para converter um número em binário para a base hexadecimal basta agrupar o número em conjuntos de quatro e aplicar para cada um o processo que é idêntico à conversão de binário para a base 10.

Exemplo:

$$\begin{array}{cccccl}
 & \underline{1100} & \underline{0110} & \underline{0011} & \underline{0000} & \text{— n.º binário} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \\
 & \underbrace{1100} & \underbrace{0110} & \underbrace{0011} & \underbrace{0000} & \text{— grupos de 4 dígitos} \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \\
 & 0 \times 2^0 & 0 \times 2^0 & 1 \times 2^0 & 0 \times 2^0 & \\
 & 0 \times 2^1 & 1 \times 2^1 & 1 \times 2^1 & 0 \times 2^1 & \\
+ & 1 \times 2^2 & 1 \times 2^2 & 0 \times 2^2 & 0 \times 2^2 & \\
 & 1 \times 2^3 & 0 \times 2^3 & 0 \times 2^3 & 0 \times 2^3 & \\
\hline
 & 12 & 6 & 3 & 0 &
\end{array}$$

Obtém-se os números: 12, 6, 3, 0.

Cada número desses, obtido na base 10 é convertido para a base 16, através da seguinte tabela:

N.º decimal	N.º hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

O número correspondente em hexadecimal será: $C630_{16}$.

6.3.4. *Hexadecimal para binário*

É usado o processo inverso. Os dígitos são expandidos em grupos de 4 dígitos binários e depois agrupados obedecendo-se a posição correspondente.

Exemplo:

F	D	9	1	3	(Hexadecimal)
15	13	9	1	3	
1111	1101	1001	001	0011	

1111110110010010011 (Binário)

7. CONSIDERAÇÕES SOBRE ERROS

Nesta secção chamamos a atenção sobre alguns erros comuns em processamento de dados e devem ser contornados ou pelo menos minimizados em função às vezes dos dados que serão manipulados.

a) Erros de impressão

Esse tipo de erro é comum em processamentos comerciais, quando se tem por exemplo uma soma calculada pelo computador.

Exemplo: Seja o seguinte relatório impresso:

EN —	35.98%
PQ —	50.01%
TA —	14.00%
Total —	100.00%

A soma dessas três percentagens, na realidade é 99.99%. O computador no entanto calculou 100.00%. A explicação que pode ser dada é a seguinte:

— Cada número ao ser impresso, foi truncado para obedecer ao formato de impressão, e por isso perdeu-se informação sobre os números. Se esse mesmo exemplo fosse impresso com cinco casas decimais poderíamos ter por exemplo o seguinte resultado:

EN —	35.98	786
PQ —	50.01	212
TA —	14.00	002
Total —	100.00	000

— Ficou explicado porque ele calculou 100.00% e não 99.99%.

Erros desse tipo podem ser corrigidos geralmente, fazendo-se o arredondamento dos números em função do número de casas decimais desejadas.

Cadernos Bibl. Arq. Doc., Coimbra, 11 (3-4) Jul.-Out. 1974, p. 88-130

b) Erros de arredondamento

Os números em um computador, por maior que seja, são representados por um número finito de dígitos. Isso significa que os números são de tamanhos fixos, pré-determinados.

É fácil ver que, se multiplicamos dois números, o produto precisaria de um número maior de dígitos, por exemplo:

$$1543 \times 25 = 38575$$

um número de 4 dígitos multiplicado por um de 2 dígitos resultou um número de 5 dígitos.

Geralmente os números são normalizados, chamados de ponto flutuante, onde se guarda o número multiplicado pela potência de 10. No caso anterior teríamos:

$$0.1543 \times 10^4 \times 0,25 \times 10^2 = 0,38575 \times 10^5$$

Se como exemplo, fixarmos 4 dígitos para cada número, o resultado desse produto seria alterado, cometendo-se o que chamamos *erro de arredondamento.*

$$0.1543 \times 10^4 \times 0,2500 \times 10^2 = 0,3857 \times 10^5$$
$$\text{ou} \quad = 0,3858 \times 10^5$$

Dependendo do computador, os dígitos que serão desprezados poderão ser arredondados, ou simplesmente eliminados do número, podendo dar um dos dois resultados obtidos acima.

Esse tipo de erro pode ser minimizado fazendo-se um estudo pormenorizado de cada expressão aritmética a ser calculada pelo programa.

c) Erros de «estouro»

Como número de dígitos é fixo, os números possíveis de serem armazenados é finito, tendo então limites superior e inferior.

Esses limites nunca devem ser ultrapassados, mesmo dentro de uma expressão aritmética.

Exemplo: N = IX + IY — IZ

— Se tivéssemos disponíveis 4 dígitos decimais e os valores fossem:

IX = 6000
IY = 7800
IZ = 5000

O valor N calculado seria absurdo, às vezes bem longe da realidade.

N = 6000 + 7800 — 5000 = ?

— essa soma ultrapassa o limite de 4 dígitos.

— Se tivéssemos escrito a fórmula:

N = IX — IZ + IY o valor calculado seria:
N = 6000 — 5000 + 7800 = 8800

obtendo-se o resultado correcto.

CÓDIGOS DE CARACTERES

Um mesmo caracter pode ser representado em localização (cartão, memória, etc.) diferente ou idêntica por diferentes códigos. A letra X, por exemplo, quando representada em cartão segundo o código EBCDIC, toma a forma:

12	11	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	0	0	1	0	0

onde os uns representam perfurações e os zeros ausências de perfurações. Quando este caracter é lido, é colocado em uma palavra da memória, ficando com a representação:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	0	0	0	0	0	0	1	0	0				

onde os bits hachurados não são usados. Para que este caracter possa ser manuseado adequadamente pelo computador, ele é então convertido para o código EBCDIC interno, e então toma a forma:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1

Finalmente, para que ele possa ser impresso na impressora IBM — 1403 ele é então convertido para o código desta impressora, ficando com a representação:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	0	1	0	0	1	0		1	1	0	0	1	0	0

Os códigos de armazenamento mais utilizados são o EBCDIC (Extended Binary Coded Decimal Interchange Code), o ASCII (American Standard Code for Information Interchange) e o BCD (Binary Coded Decimal).

FORMATOS DE DADOS

O valor alfanumérico AXY1 pode ser representado na memória do computador, em EBCDIC, da seguinte forma:

00C1 00E7 00E8 00F1

em quatro palavras de computador.

O valor numérico sem sinal, 345 pode ser representado na memória, em EBCDIC, da seguinte forma:

00F3 00F4 00F5

em três palavras de computador.

O valor numérico com sinal + 345 pode ser representado na memória, em EBCDIC, da seguinte forma:

00F3 00F4 00C5

em três palavras de computador. O sinal do número é guardado junto com o último dígito. A última palavra representa a letra E, mas isto não causa confusão porque por programação pode-se distinguir que é o número + 345 ao invés de 34E.

O valor numérico com sinal − 345 pode ser representado na memória, em EBCDIC, da seguinte forma:

00F3 00F4 00D4

em três palavras de computador. Onde o D é utilizado para indicar que o número é negativo.

Um valor numérico pode ser armazenado na forma binária, onde o primeiro bit indica se o número é negativo ou positivo. 1 indica negativo e 0 indica positivo.

Seja, por exemplo, uma palavra composta de 4 bits. O número 2_{10} ou $+ 2_{10}$ é armazenado:

0	0	1	0

Os números negativos são armazenados na forma complementar.

Seja, por exemplo, armazenar o número -2_{10}. O número -2_{10} corresponde ao número positivo 2_{10}, que é representado em binário por 0010.

Para se obter a forma complementar, inverte-se o número (trocam-se os zeros por 1 e os uns por zero) e adiciona-se 1.

0010
inverte-se
1101
adiciona-se 1 ao número invertido.

$$\begin{array}{r} 1101 \\ +\quad 1 \\ \hline 1110 \end{array}$$

Logo, o número -2_{10} é armazenado:

1	1	1	0

O maior número que pode ser armazenado nesta palavra é:

0	1	1	1

cujo valor na base 10 é: $2^3 - 1 = 7$

O menor número que pode ser armazenado nesta palavra é:

1	0	0	0

$$1000 \rightarrow \begin{array}{r} 0111 \\ +\quad 1 \\ \hline 1000 \end{array}$$

cujo valor na base 10 é: $-(2)^3 = -8$

OPERAÇÕES NA FORMA BINÁRIA

1) Seja, por exemplo, adicionar os valores:

a) $\overset{3}{\boxed{0\,|\,0\,|\,1\,|\,1}}$ e $\overset{4}{\boxed{0\,|\,1\,|\,0\,|\,0}}$

$$\begin{array}{r} 0011 \\ +\ 0100 \\ \hline 0111 \end{array} \rightarrow \overset{7}{\boxed{0\,|\,1\,|\,1\,|\,1}}$$

b) $\overset{3}{\boxed{0\,|\,0\,|\,1\,|\,1}}$ e $\overset{-8}{\boxed{1\,|\,0\,|\,0\,|\,0}}$

$$\begin{array}{r} 0011 \\ +\ 1000 \\ \hline 1011 \end{array} \rightarrow \overset{-5}{\boxed{1\,|\,0\,|\,1\,|\,1}}$$

c) $\overset{-1}{\boxed{1\,|\,1\,|\,1\,|\,1}}$ e $\overset{-2}{\boxed{1\,|\,1\,|\,1\,|\,0}}$

$$\begin{array}{r} 1111 \\ +\ 1110 \\ \hline 11101 \end{array} \rightarrow \overset{-3}{\boxed{1\,|\,1\,|\,0\,|\,1}}$$

2) Seja, por exemplo, subtrair os valores:

a) $\overset{3}{\boxed{0\,|\,0\,|\,1\,|\,1}}$ e $\overset{4}{\boxed{0\,|\,1\,|\,0\,|\,0}}$

Determina-se o complemento do 2.º e soma-se ao 1.º.

$$0100 \rightarrow \begin{array}{r} 1011 \\ +\ \ \ 1 \\ \hline 1100 \end{array}$$

$$\begin{array}{r} 0011 \\ +\ 1100 \\ \hline 1111 \end{array} \rightarrow \overset{-1}{\boxed{1\,|\,1\,|\,1\,|\,1}}$$

b) $\overset{-1}{\boxed{1\,|\,1\,|\,1\,|\,1}}$ e $\overset{-2}{\boxed{1\,|\,1\,|\,1\,|\,0}}$

$$1110 \rightarrow \begin{array}{r} 0001 \\ +\ \ \ 1 \\ \hline 0010 \end{array}$$

$$\begin{array}{r} 1111 \\ +\ 0010 \\ \hline 10001 \end{array} \rightarrow \overset{1}{\boxed{0\,|\,0\,|\,0\,|\,1}}$$

c) $\overset{6}{\boxed{0|1|1|0}}$ e $\overset{2}{\boxed{0|0|1|0}}$

```
0010 ──> 1101
       +    1
       ──────
         1110

  0110
+ 1110          4
──────
1 0100 ──> |0|1|0|0|
```

3) Multiplicações podem ser feitas por somas sucessivas e divisões por subtrações sucessivas.

PALAVRA, BYTE, DIGITO e BIT

Uma PALAVRA é um conjunto de bits (16, 32, etc.), possuindo um endereço para que seu conteúdo possa ser acessado.

Exemplo:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
13425	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

o conteúdo do endereço 13425 é 0000000000000100.

Um BYTE é um conjunto de bits (6 ou 8 geralmente) com um endereço, contido em uma palavra. Exemplo:

	PALAVRA	
	BYTE	BYTE
	0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0
Endereços ——→	13436	13437

Um DIGITO é um conjunto de 4 bits com um endereço.

Exemplo:

PALAVRA			
BYTE		BYTE	
-DIGITO-	-DIGITO-	-DIGITO-	-DIGITO-
0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0
13436	13437	13438	13439

A razão do nome DIGITO é porque seus 4 bits podem representar qualquer digito na base 16.

TÁBUA DE EQUIVALÊNCIAS

Decimal	Hexadecimal	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
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

TÁBUA DE ADIÇÃO — SUBTRAÇÃO HEXADECIMAL

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

TÁBUA DE MULTIPLICAÇÃO — DIVISÃO HEXADECIMA

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

TÁBUA DECIMAL

ADIÇÃO E SUBTRAÇÃO — MULTIPLICAÇÃO E DIVISÃO

±	0	1	2	3	4	5	6	7	8	9	× ÷	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9	0	0	0	0	0	0	0	0	0	0	0
1	1	2	3	4	5	6	7	8	9	10	1	0	1	2	3	4	5	6	7	8	9
2	2	3	4	5	6	7	8	9	10	11	2	0	2	4	6	8	10	12	14	16	18
3	3	4	5	6	7	8	9	10	11	12	3	0	3	6	9	12	15	18	21	24	27
4	4	5	6	7	8	9	10	11	12	13	4	0	4	8	12	16	20	24	28	32	36
5	5	6	7	8	9	10	11	12	13	14	5	0	5	10	15	20	25	30	35	40	45
6	6	7	8	9	10	11	12	13	14	15	6	0	6	12	18	24	30	36	42	48	54
7	7	8	9	10	11	12	13	14	15	16	7	0	7	14	21	28	35	42	49	56	63
8	8	9	10	11	12	13	14	15	16	17	8	0	8	16	24	32	40	48	56	64	72
9	9	10	11	12	13	14	15	16	17	18	9	0	9	18	27	36	45	54	63	72	81