NOÇÕES GERAIS SOBRE O COMPUTADOR
NATIONAL-ELLIOTT 803-B E SOBRE PROGRAMAÇÃO

1) Um computador electrónico digital (de que o ELLIOTT-803 é um exemplar) é um conjunto de dispositivos electrónicos que permitem efectuar operações aritméticas elementares (adição, subtracção, multiplicação e divisão) e realizar decisões quantitativas elementares "automàticamente" e a grande velocidade.

Esclareçamos o sentido de alguns termos:

a) Realizar decisões quantitativas elementares significa a possibilidade que o computador tem de, conforme a natureza do número (negativo, nulo ou positivo) contido num registo especial (chamado acumulador, conforme adiante se verá), seguir sequências de operações aritméticas diferentes; mais tarde se verá como nesta simples possibilidade se fundamenta grande parte da potencialidade dos computadores.

b) O termo automàticamente precisa de ser bem interpretado. Realmente a realização efectiva das operações e decisões envolve um trabalho preliminar fundamental, que consiste na formulação do problema a ser resolvido como uma sequência daquelas operações e decisões e além disso na expressão dessa sequência numa linguagem aceitável pelo computador. Só depois desse trabalho realizado pela inteligência humana é que o computador é capaz de executar os cálculos sem intervenção directa do homem.

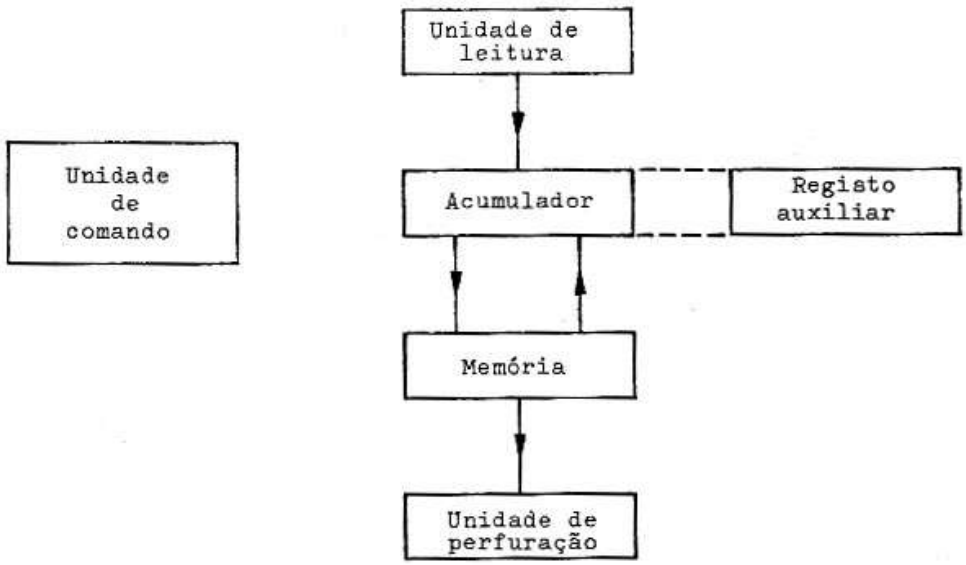
c) Para fazer ideia da velocidade com que um computador opera basta dizer que o ELLIOTT-803 é capaz de realizar cerca de 2000 adições algébricas por segundo.

Falta acrescentar que as operações aritméticas são executadas com

elevada precisão; para o citado computador essa precisão é de cerca de 12 ou 9 algarismos decimais significativos, conforme se trabalhe em vírgula fixa ou vírgula flutuante (o significado destes termos será compreendido mais tarde).

2) Atendendo ao que ficou dito atrás compreende-se que o emprego dum computador estará indicado naqueles problemas que, por técnicas adequadas, se possam transformar numa sequência daquelas operações e decisões, sendo as primeiras em grande número ou em pequeno número mas tendo que ser executadas muitas vezes com diferentes valores numéricos. Não faz sentido empregar um computador num cálculo que envolva um pequeno número de operações e que tenha que ser executado uma única vez, visto que, embora o tempo de execução fosse muitíssimo curto, o trabalho preliminar a que atrás nos referimos, levaria um tempo de preparação que não compensaria o tempo de execução manual ou mecânica do problema.

3) Seguidamente apresentamos o esquema teórico do computador Elliott-803



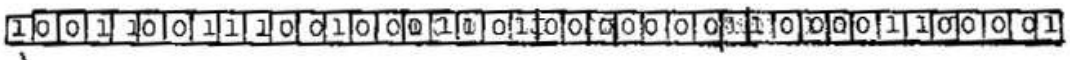
Existe ainda uma unidade de controle que superintende ao funcionamento das várias unidades do computador. No entanto o conhecimento desta unidade não é essencial para um programador.

4) Analisaremos agora as várias partes do Elliott-803.

a) Memória

É um conjunto de memórias. Cada memória é um registo onde se formam fieiras de dígitos, fieiras essas que se chamam palavras; cada memória tem um número identificador, que se chama o endereço da memória.

No 803 existente no LNEC há 8192 memóriaa, numeradas de 0 a 8191.Em cada uma delas forma-se uma palavra de 39 dígitos binários, binits ou bits (do inglês binary digits) – 0 e 1.



dígito de paridade: é um quadragésimo dígito que faz com que o número total de 1 na palavra seja ímpar.

Obs: A palavra contida na memória de endereço N será designada por C (N).

Das 8192 memórias do 803 há 5 (0 a 4) que são usadas para fins especiais; as restantes são usadas para armazenar palavras.

Cada palavra pode ser interpretada como um número (em binário) ou duas instruções. Com efeito, para execução dum cálculo é necessário não só armazenar um certo número de valores numéricos mas também "instruir" o computador na maneira de actuar sobre eles. Por exemplo, se tivermos dois números guardados em duas memórias e os quisermos adicionar, é necessário que haja uma instrução ou instruções que permitam ao computador executar aquela operação. Essas instruções são guardadas nas memórias (duas por cada memória) por meio de convenções que serão estudadas adiante.

Um problema que se põe imediatamente é o do computador interpretar correctamente uma palavra como um número ou duas instruções. O computador fará essa distinção consoante as medidas tomadas pelo operador. Uma confusão nessas medidas poderá conduzir à interpretação dum número como duas instruções e vice-versa, anulando assim a boa sequência de todo o processo.

b) Acumulador

As operações aritméticas elementares fazem essencialmente corresponder a dois números (a e b) um terceiro c; assim temos:

$$a+b = c \text{ (soma)}$$

$$a-b = c \text{ (diferença)}$$

$$a \times b = c \text{ (produto)}$$

$$a:b = c \text{ (quociente)}$$

A maneira como o 803 executa cada uma destas operações é combinando o conteúdo duma memória (que é um dos termos da operação) com o conteúdo dum dispositivo chamado acumulador, que é um registo especial capaz de conter uma palavra com 39 bits (mesmo número que o de uma memória) e onde está contido o segundo termo de operação.

Obs: O acumulador designar-se-á por A e o seu conteúdo por C(A).

Quando multiplicamos dois números cada um com n algarismos, o produto tem sensivelmente 2n algarismos; inversamente, para que o quociente de dois números tenha n algarismos, tendo o divisor n algarismos também, é necessário que o dividendo tenha sensivelmente 2n algarismos. Estes factos justificam o aparecimento do chamado registo auxiliar, que é um registo onde se podem inscrever 38 bits e que permite, juntamente com o acumulador, armazenar números em duplo comprimento.

A cada uma das operações aritméticas elementares correspondem várias

instruções, que estão agrupadas nos grupos 0,1,2,3,5[1] e cujo conhecimento, assim como as dos outros grupos, é essencial para a chamada programação em código-máquina. Como estas notas se destinam a dar umas noções gerais sobre programação, não se entrará no estudo detalhado dessas instruções.

c) Unidade de leitura

No computador 803 existente no LNEC esta unidade é constituída por dois leitores de fitas perfuradas. Nestas fitas vão perfuradas, segundo um determinado código, as instruções e constantes cujo conjunto se chama programa de resolução de um determinado problema e que, combinadas com os dados numéricos desse problema, o permitem resolver.

d) Unidade de perfuração e impressão

No computador 803 existente no LNEC esta unidade é constituída por dois perfuradores de fita e um tele-impressor. Estes são os meios através dos quais o computador nos dá os resultados numéricos de um problema. No caso de saída dos resultados em fita perfurada é necessário descodificá-los.

Há instruções para ordenar ao computador a leitura e perfuração (impressão) de caracteres, instruções essas que fazem parte do grupo 7.

e) Unidade de comando

Tem os botões, lâmpadas e outros dispositivos que permitem ao operador controlar o computador e verificar se ele está a operar correctamente. Esta unidade será analisada detalhadamente mais tarde.

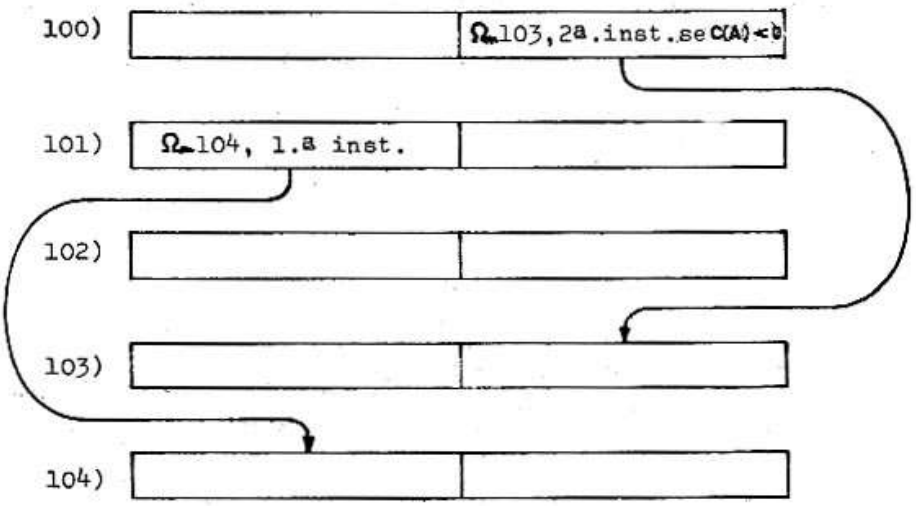
5) Já afirmámos a existência de vários tipos de instruções, ligadas

---

(1) - Consultar as publicações "A guide to programming the National-Elliott 803 electronic digital computer" ou "803 facts"

respectivamente às operações aritméticas e à leitura e perfuração de caracteres. Faremos agora referência às instruções de salto (grupo 4), que fazem quebrar a sequência normal de execução das instruções.

Expliquemos o que se entende por esta expressão e para isso imaginemos que nas 5 memórias 100 a 104 estão memorizadas 10 instruções, que fazem parte de um programa. Se não houvesse instruções de salto nesta secção do programa, o computador executaria as instruções da maneira que nos aparece à primeira vista como mais lógica, isto é,



executaria primeiro a 1ª instrução de 100, depois a 2.ª, depois a 1.ª instrução de 101, depois a 2.ª e assim sucessivamente. No entanto, se houver instruções de salto, a sequência já não será esta, podendo assumir vários aspectos. Com efeito há dois tipos de instruções de salto:

a) de salto incondicional: quando o computador encontra uma destas instruções, a instrução seguinte a ser executada é aquela indicada na instrução de salto; assim, se no exemplo apresentado a 1.ª instrução da memória 101 é uma instrução de salto para a 1.ª instrução da memória 104, o computador ignora as instruções seguintes àquela e executa imediatamente a 1.ª instrução da memória 104;

b) de salto condicional: quando o computador encontra uma destas ins

truções, a instrução seguinte a ser executada é aquela indicada na instrução de salto, se se verificar uma determinada condição; no exemplo apresentado, quando o computador chega à 2.ª instrução da memória 100, testa o conteúdo do acumulador: se ele for negativo, a instrução seguinte a ser executada é a 2.ª instrução da memória 103; se for nulo ou positivo segue a sequência normal.

São estas instruções de salto condicional que estão relacionadas, conforme é fácil concluir, com o que atrás designámos por decisões quantitativas elementares.

6) Vejamos agora como as várias unidades cujo estudo resumido foi feito atrás concorrem para a resolução dum problema.

Suponhamos que o programa de resolução do problema já está elaborado e perfurado. O primeiro passo é a leitura da fita ou fitas perfuradas pelos leitores; ao mesmo tempo que vão sendo lidas, as instruções passam pelo acumulador e vão sendo memorizadas na memória do computador(1); a leitura é ordenada através da unidade de comando. Seguidamente, igualmente através da unidade de comando, dá-se ordem ao computador para começar a executar o programa; as instruções são executadas em sequência normal, a não ser que haja instruções de salto, conforme já foi referenciado; normalmente as primeiras instruções são instruções de leitura, que permitem a leitura e memorização dos dados numéricos do problema; seguidamente temos as instruções aritméticas que vão actuar sobre os números memorizados, fazendo intervir frequentemente o acumulador (com ou sem registo auxiliar); finalmente há as instruções de perfuração, que permitem fazer sair os resultados do problema.

---

(1) - Frequentemente as instruções estão numa forma que não permite a sua memorização imediata; é por exemplo o caso dos códigos simbólicos (ex: auto-código, algol) em que as respectivas instruções precisam primeiro de ser traduzidas para se poderem inscrever nas memórias do computador da maneira que adiante será estudada.

7) Sistemas de numeração: aconselhamos o leitor a rever os seus conhecimentos sobre o assunto, que se encontra em qualquer compêndio de Aritmética Racional.

Faremos aqui apenas uma referência à conversão de números de binário para decimal e vice-versa, servindo-nos de exemplos.

a) Passagem de binário para decimal:

$A = 10101.11_{(2)}$ (1)

$A = 1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+1\times2^{-2}$

$= 21.75_{(10)}$

b) Passagem de decimal para binário

$A = 100$

100	
64	2^6
36	
32	2^5
4	
4	2^2
0	

$100_{(10)}=2^6+2^5+2^2=1100100_{(2)}$

$B = 3.7$

2	2^1
1.7	
1	2^0
0.7	
0.5	2^{-1}
0.2	
0.125	2^{-3}
0.075	
0.0625	2^{-4}
0.0125	
0.0078125	2^{-7}
0.0046875	
⋮	⋮

$3.7_{(10)}=2^1+2^0+2^{-1}+2^{-3}+2^{-4}+2^{-7}+\ldots$

$= 11.1011001\ldots.$

---

(1) Usar-se-á sistemàticamente o ponto em vez da vírgula, porque nas publicações em língua inglesa e equipamentos de origem inglesa usa-se normalmente esta notação.

Dos exemplos apresentados é fácil concluir que a maneira de proceder utilizada é subtrair ao número dado a maior potência de 2 nele contida e proceder anàlogamente com os sucessivos restos que se vão obtendo.

Também se conclui que um número representável em decimal por um número finito de dígitos pode não admitir (e em geral não admitirá) uma representação em binário com um número finito de bits.

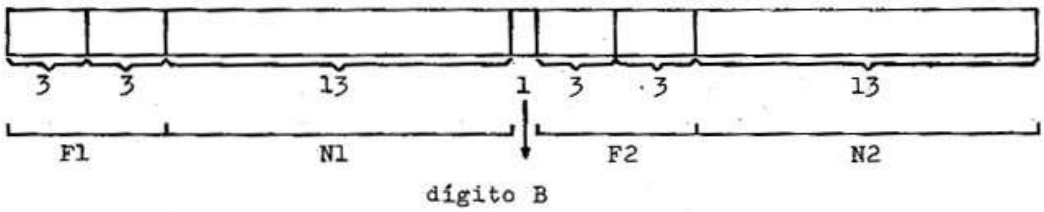
8) Representação das instruções

Cada instrução é composta de duas partes, designadas respectivamente por F e N:

F – especifica a operação a ser realizada

N – é um número inteiro que representa o endereço duma memória ou uma especificação complementar à operação a ser realizada.

No caso de uma palavra constituir duas instruções, os 39 bits distribuem-se por vários grupos, cada um deles com o seu significado próprio; esses grupos estão representados no seguinte esquema de uma memória:



Cada instrução ocupa portanto 19 bits; as duas ocuparão 38 bits que, com o bit do meio[1], perfazem os 39 bits da memória.

F1, F2 — 6 bits { 3 bits — grupo
                { 3 bits — função

N1, N2 — 13 bits

(1) - Este bit tem uma função especial que não nos interessa nestas notas referir.

Com 3 bits podem-se representar os números inteiros compreendidos entre 0 e $2^3 - 1 = 7$. Há portanto a possibilidade de representar 8 grupos e 8 funções, o que dá um máximo possível de $8 \times 8 = 64$ tipos de operações diferentes possíveis.

Com 13 bits podem-se representar os números inteiros compreendidos entre 0 e $2^{13} - 1 = 8191$, que são tantos quantas as memórias do 803. Por aqui se vê que não pode haver um computador com 39 bits por palavra e com este mesmo sistema de representação das instruções com mais de 8192 memórias.

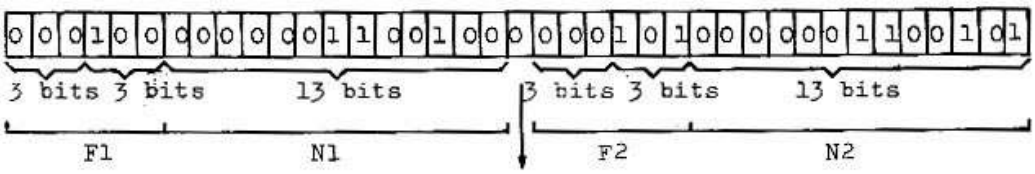
Exemplo: Suponhamos que C(A)=a, C(100)=b e C(101)=c e queremos formar a soma algébrica a+b=c no acumulador, isto é, queremos que fique

$$C(A) = a+b-c$$

Podemos fazer isso por intermédio do seguinte par de instruções

04 100 05 101

conforme se pode concluir da análise das instruções 04 e 05. Essas duas instruções são memorizadas da seguinte maneira



dígito B=0

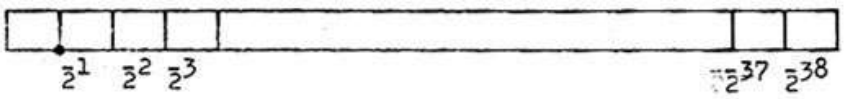
9) Representação dos números em vírgula fixa

Os números representáveis no 803 estão no intervalo

$$\left[-1,\ 1-2^{-38}\right]$$

"realizados" da seguinte maneira:

a) Supõe-se que a vírgula está entre o dígito da esquerda e o seguin



b) Os números positivos entre zero (que é considerado como positivo) e $1-2^{-38}$ são representados por palavras começando por zero, sendo os restantes 38 bits a representação binária do número. Assim

Zero	——	0 $\underbrace{0}_{38}$ (1)
2^{-38}	——	0 $\underbrace{0}_{37}$ 1
menor número positivo		
0.875	——	0 111 $\underbrace{0}_{35}$
$1-2^{-38}$	——	0 $\underbrace{1}_{38}$
maior número positivo		

Resumindo, e supondo que uma palavra tem apenas 4 bits, podemos estabelecer que

$$\underbrace{0XXX}_{\text{representação no computador}} = \underbrace{+\ (0.XXX)}_{\text{significado aritmético}}$$

c) Os números negativos entre -1 e -2^{-38} são representados por palavras começando por um, sendo os restantes 38 bits a representação binária de 1-x (-x é o número negativo). Assim

---

(1) Com a notação $\underbrace{a}_{n}$ designar-se-á uma sucessão de n caracteres a

-1 —— 1 $\underbrace{0}_{38}$

menor número negativo

-0.375 —— 1 11 $\underbrace{0}_{36}$

$$\begin{array}{r} 1 \\ \underline{0.375} \\ 0.625 \end{array} = 0.11_{(2)}$$

-2^{-38} —— 1 $\underbrace{1}_{38}$

maior número negativo

Resumindo, e com uma convenção análoga à adoptada para os números positivos, podemos estabelecer que

$$\underbrace{1XXX}_{\text{representação no computador}} = \underbrace{-(1-0.XXX) = -1+0.XXX}_{\text{significado aritmético}}$$

Esta expressão indica-nos uma maneira imediata de passar da representação dos números negativos no 803 para o seu verdadeiro valor: basta adicionar a -1 o número positivo representado pelos últimos 38 bits.

O primeiro dígito duma palavra chama-se dígito do sinal, porque é zero para os números positivos e um para os negativos.

A representação dos números em vírgula fixa dá-nos uma precisão de 38 bits, o que corresponde aproximadamente a 12 algarismos decimais significativos.

Sempre que, em consequência de uma operação aritmética, um número exceda a capacidade de uma memória, entra em acção um dispositivo chamado indicador do excesso de capacidade, acendendo-se a lâmpada "overflow" da mesa de comando.

10) Representação dos números em vírgula flutuante

Por meio desta designação entende-se uma maneira especial de inscrever os números na memória do computador. A sua finalidade é a de permitir

que números muito pequenos ou muito grandes sejam representados por uma palavra do computador.

Vimos na representação em vírgula fixa que os números representáveis são os do intervalo $\left[-1, 1-2^{-38}\right]$. Ora na maior parte dos casos os números que nos aparecem não estão neste intervalo. Uma das possibilidades que nos aparecem é a de multiplicarmos esses números por potências de 10 ou 2 (conforme o sistema que se estiver a utilizar) de forma a obter números naquele intervalo. Por exemplo, suponhamos que queremos multiplicar os números 721.3 e 4203.47 : podemos proceder da seguinte maneira

$$721.3 \times 4203.47 = 0.7213 \times 10^{3} \times 0.420347 \times 10^{4}$$

$$= 0.7213 \times 0.420347 \times 10^{7}$$

Os dois primeiros factores deste produto já estão no intervalo considerado, assim como o respectivo produto; isso significa que esta operação já pode ser efectuada pelo computador, não nos podendo esquecer que o resultado obtido vem dividido por 10^{7}.

Vê-se imediatamente que em sequências longas de cálculo se torna muito trabalhoso e demorado o controle dos números de forma a evitar o que se designa por excessos de capacidade. É essa desvantagem que é suprimida pela representação dos números em vírgula flutuante, que permite a inscrição de números muito pequenos ou muito grandes e efectuar as operações sobre eles sem que o operador tenha que controlar os resultados intermédios. No entanto este processo tem a desvantagem de diminuir a precisão dos cálculos (número de algarismos significativos)

Passemos então ao estudo da representação dos números em vírgula flutuante.

Qualquer número A pode ser representado de muitas maneiras por um par

de números (a,b) que satisfaçam a igualdade

$$A = a \times 2^b$$

em que a é a mantissa ou argumento e b o expoente (binário)[1]. Por exemplo, se A = 6, A pode ser representado pelos pares (6,0), (0,75,3), etc. No entanto, pode-se estabelecer que todo o número A diferente de zero pode ser representado de uma única maneira por um par de números tais que o valor absoluto da mantissa fica compreendido entre 1/2 e 1 (podendo atingir um destes limites) e o expoente é inteiro. Com efeito, tem-se, com x e y inteiros:

$A > 0$	$A < 0$
$2^x \leq A < 2^{x+1}$	$-2^{y+1} \leq A < -2^y$
$\frac{1}{2} \leq \frac{A}{2^{x+1}} < 1$	$-1 \leq \frac{A}{2^{y+1}} < -\frac{1}{2}$
$A = \frac{A}{2^{x+1}} \times 2^{x+1}$	$A = \frac{A}{2^{y+1}} \times 2^{y+1}$

Na prática há limites superior e inferior para os expoentes binários, de forma que há limites no valor absoluto dos números que podem ser representados no computador.

Nos computadores 803-B (computadores dotados de uma unidade automática de vírgula flutuante) a representação dos números obedece às seguintes convenções:

a) Se $A > 0$ —— $\frac{1}{2} \leq a < 1$

b) Se $A = 0$ —— $a=0$ $b=-256$ (sempre)

---

(1) A razão da designação "vírgula flutuante" compreende-se melhor se considerarmos a representação

$$A = a \times 10^b$$

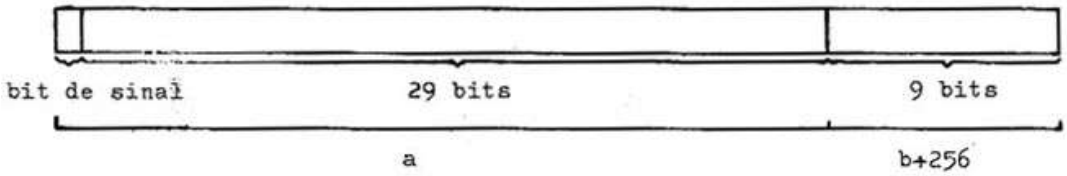
Com efeito, para A = 293.2, tem-se

$$A = 293.2 \times 10^0 = 29.32 \times 10^1 = 2932 \times 10^{-1} = \ldots ;$$

há como que uma "flutuação" na posição da vírgula.

c) Se $A < 0$ —— $-1 \leq a < -\frac{1}{2}$

d) Os 39 bits duma palavra dividem-se em dois grupos:



o primeiro grupo, de 30 bits, serve para a representação da mantissa a da mesma maneira que uma fracção em vírgula fixa; o segundo serve para a representação do expoente b adicionado de 256 unidades.

Vejamos a razão da adição de 256 unidades ao expoente b. Com 9 bits podemos representar os números inteiros compreendidos entre 0 e $2^9-1=511$ (inclusivé). É para podermos representar expoentes negativos que nos servimos daquele artifício. Com efeito

$$0 \leq b+256 \leq 511$$

$$-256 \leq b \leq +255$$

Vejamos agora a representação de alguns números em vírgula flutuante:

Zero —— 0 $\underbrace{0}_{29}$ 000000000

$a=0$; $b=-256$

mesma representação que em vírgula fixa

$-1 = -1 \times 2^0$ —— 1 $\underbrace{0}_{29}$ 100000000

$a=-1$; $b=0$

$+1 = 0.5 \times 2^1$ —— 0 1 $\underbrace{0}_{28}$ 100000001

$a=0.5$; $b=1$

$+30=\frac{30}{32} \times 32=\frac{15}{16}\times 2^5$ —— 0 1111 $\underbrace{0}_{25}$ 100000101

$a=\frac{15}{16}$; $b=5$

$-0.0875=-\frac{0.0875}{0.125}\times 0.125=-0.7\times 2^{-3}$ —— 1 010011... 011111101

$(1-2^{-29})\times 2^{255} \approx 5.8\times 10^{76}$ maior número positivo	—	0	1 (29) 111111111
$\frac{1}{2} \times 2^{-256}$ menor número positivo	—	0	1 0 (28) 000000000
$-(\frac{1}{2}+2^{-29})\times 2^{-256} \approx 4.3\times 10^{-78}$ maior número negativo	—	1	0 1 (28) 000000000
-1×2^{255} menor número negativo	—	1	0 (29) 111111111

Em resumo: em vírgula flutuante o número zero é representado exactamente e qualquer número A satisfazendo a

$$4.3\times 10^{-78} \leq |A| \leq 5.8\times 10^{76}$$

pode ser representado com uma precisão de 29 bits, o que corresponde aproximadamente a 9 algarismos decimais significativos.

12) Se o computador é chamado a executar qualquer operação aritmética em vírgula flutuante de que o resultado é demasiado pequeno para ser representado, o resultado gerado será zero. Este efeito é designado em inglês por "floating-point underflow" e não afecta a execução de um programa ou qualquer lâmpada indicadora.

Se o computador é chamado a executar qualquer operação aritmética em vírgula flutuante de que o resultado é demasiado grande para ser representado, o computador pára e a lâmpada "floating-point overflow" ficará acesa.

Há um grupo de instruções (grupo 6) que permite executar operações em vírgula flutuante.

13) Passamos agora a descrever a mesa de comando.

a) No cimo à direita encontram-se os botões que permitem ligar e desligar o computador. Para ligar carrega-se primeiro em "battery-on" e

depois em "computer-on" e para desligar carrega-se primeiro em "computer--off" e depois em "battery-off".

b) O botão "clear store" tem por efeito fazer todos os dígitos de paridade nas memórias 4 a 8191 iguais a 1 e os restantes iguais a zero. Para realizar esta operação, carrega-se sucessivamente nos botões "clear store" e "normal" e depois na barra de operação (operate bar).

c)c) Gerador de palavras: é constituído por 4 filas de botões, contendo 6, 14, 6 e 13 botões, correspondendo respectivamente aos dígitos F1, N1 e B, F2 e N2 de uma palavra. Um botão em baixo representa um 1 e em cima um 0. Quando um botão é cpremidodo fica preso; para ser liberto carrega-se no botão vermelho à esquerda da fila respectiva.

d) Botões principais de comando: "read", "normal", "obey". São botões exclusivos, isto é, só pode estar premido um de cada vez.

Para perceber o emprego destes botões é preciso ter um conhecimento, se bem que rudimentar, da maneira como o computador trabalha.

Já vimos atrás que o computador trabalha fundamentalmente em dois tempos: memorização, precedida ou não duma tradução, e execução das instruções dum programa. Vejamos agora como é executada cada instrução do programa.

Em primeiro lugar o computador determina qual a instrução a obedecer por meio dum registo especial que controla a sequência do programa; seguidamente o computador, ao mesmo tempo que faz o teste de paridade da palavra contida na memória onde está a referida instrução (isto é, determina se o número de 1 na palavra é ímpar), tira uma cópia dessa instrução para a unidade de controle; finalmente interpreta-a e executa-a.

Quando o computador pára, quer por meio dos botões de comando quer por falhar o teste de paridade, fica sempre na posição de ter completado a segunda fase. Há portanto uma instrução na unidade de controle pronta

a ser obedecida; é o que se chama a instrução presente.

Vamos agora estudar várias acções possíveis que se podem executar por intermédio destes botões.

d_1) Pararo computador — comprime-se o botão "read" ou o "obey".

d_2) Fazer com que o computador, tendo parado, leia uma instrução do gerador de palavras e substitua a instrução presente por ela — carrega-se no botão "read", põe-se a instrução nos botões F1 e N1 do gerador de palavras e comprime-se a barra de operação uma vez.

d_3) Fazer com que o computador, tendo parado, obedeça à instrução presente e pare de novo — carrega-se no botão "obey" e comprime-se a barra de operação uma vez.

d_4) Fazer com que o computador, tendo parado, obedeça à instrução presente e continui a obedecer às instruções da sua memória a toda a velocidade — carrega-se no botão "normal" e comprime-se a barra de operação uma vez.

Explicaremos agora, à luz do que foi exposto, o procedimento habitual de começo de execução dum programa: põe-se a instrução 40 M (onde M é o endereço da memória onde estão as primeiras instruções do programa) nos botões F1 e N1 do gerador de palavras; seguidamente comprime-se o botão "read" e a barra de operação, o que faz passar esta instrução para a unidade de controle; finalmente comprime-se o botão "normal" e a barra de operação, o que faz com que aquela instrução seja descodificada e executada; ora o seu efeito é colocar no registo de controle de sequência, duma maneira apropriada, o endereço M e a indicação de que se trata da primeira instrução dessa memória; o computador executa essa instrução da maneira que atrás foi esquematizada, assim como as seguintes.

Quando M=0, a chamada, digamos assim, é ao programa que ocupa as

memórias 0 a 3 do computador (utilizando a memória 4 como memória de trabalho); esse programa, conhecido por T1, é um programa que não pode ser destruído por qualquer operação exterior sobre a máquina e que permite a leitura de fitas perfuradas num código especial.

e) Há ainda dois botões,"manual data" e "selected stop", que estão relacionados com determinadas operações em código máquina e cujo emprego específico não nos interessa referir nestas breves notas; e finalmente um terceiro, "reset", de que falaremos a propósito da lâmpada "busy".

f) Existe um alto-falante que é activado por todas as instruções dos grupos 4,5,6 e 7;a altura do som varia com a frequência da ocorrência de tais instruções no programa e pode dar indicações sobre a execução dum programa a um operador experiente.

Em ligação com o alto-falante existe uma roda de controle do volume do som.

g) Lâmpadas de sinalização: dão indicações ao operador sobre o comportamento do computador.

g_1) "Parity" - acende quando o teste de paridade indica que há um número par de 1 numa memória; ao mesmo tempo o computador pára.

g_2) "Block transfer" - acende durante as operações longas.

g_3) "Busy" - acende quando uma das unidades periféricas não está em condições de trabalho; quando isto se verifica deve carregar-se no botão "reset".

g_4) "Floating-point overflow" - acende quando se verifica um excesso de capacidade em vírgula flutuante.

g_5) "Step by step" - acende durante uma operação passo a passo (botões "read" ou "obey" premidos).

g_6) "Overflow" - acende quando se verifica um excesso de capacidade

em vírgula fixa

14) Neste parágrafo, e baseando-nos nas considerações anteriores, vamos descrever os vários passos que se devem percorrer na resolução dum problema com um computador.

a) Identificação do problema:-temos que enquadrar a questão a resolver no conjunto dos vários ramos da ciência e (ou) técnica e descrever exactamente os seus dados e objectivos.

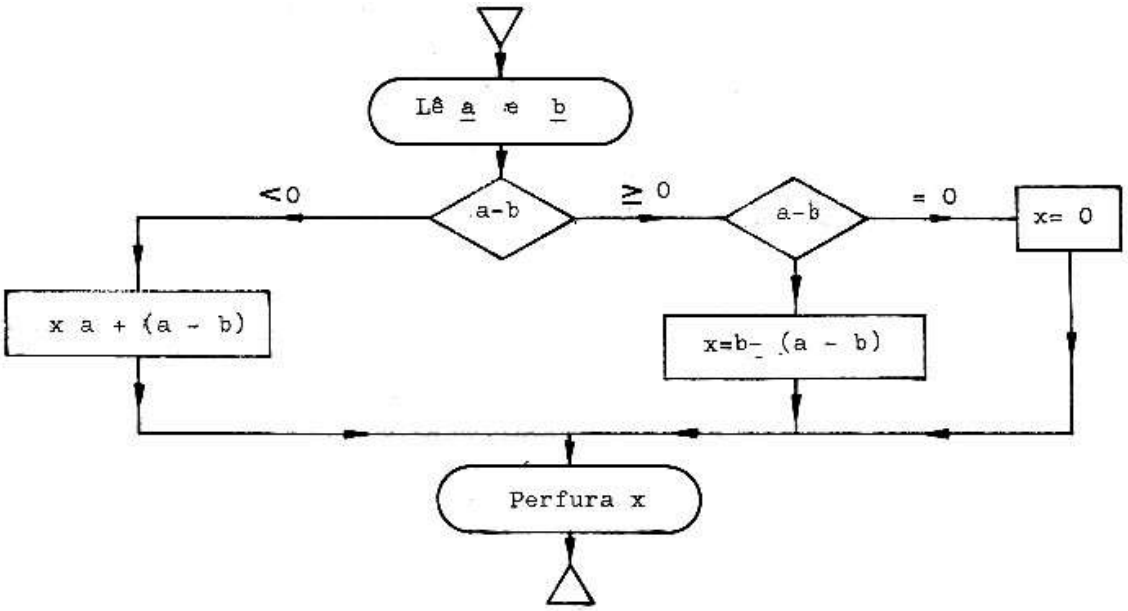
b) Descrição matemática: há geralmente várias maneiras de descrever um processo matemàticamente; tem portanto que se escolher uma destas ou então desenvolver uma nova, se nenhuma das existentes se mostrar satisfatória; está-se no campo da física teórica, física matemática, investigação operacional, etc.

c) Recurso à Análise Numérica: a formulação matemática do problema pode não ser (e em geral não é) traduzível directamente na linguagem do computador, atendendo às capacidades deste. Funções trigonométricas, exponenciais, logarítmicas, raizes quadradas, etc. e processos que se estudam em Matemática Superior (integrais, equações diferenciais, equações às derivadas parciais, etc.) têm que ser expressos em termos de operações elementares. Aí é que intervém as técnicas da Análise Numérica que, duma maneira aproximada (podendo determinar-se frequentemente o grau de aproximação) permitem efectuar essa passagem.

d) Ordenação: é o estabelecimento da sequência de operações e sua concatenação; normalmente faz-se sob a forma de organigrama, de que se apresenta um exemplo a seguir.

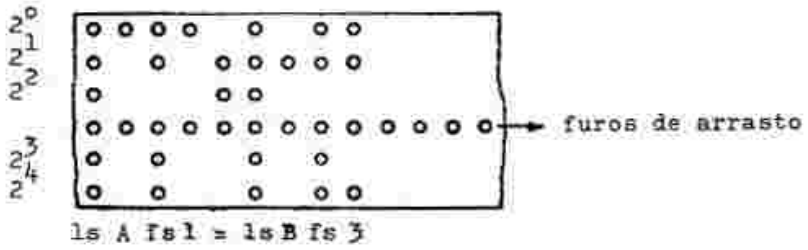
Problema: Dados os números a e b, calcular o número x tal que

$$\begin{array}{lll} x = 2a-b & \text{se} & a<b \\ x = 0 & \text{se} & a=b \\ x = 2b-a & \text{se} & a>b \end{array}$$



e) Programação: é a passagem do processo sob a forma de organigrama para uma linguagem aceitável pelo computador. Essa linguagem pode ser a própria do computador, isto é, aquela constituída por instruções e números obedecendo às convenções atrás indicadas (no caso do computador 803) e que normalmente se designa por linguagem máquina; ou então pode ser uma linguagem simbólica, que faça o compromisso entre a linguagem algébrica e a linguagem máquina: é formada por um conjunto de instruções que tomam um aspecto semelhante ao algébrico, instruções essas que por outro lado são susceptíveis de ser traduzidas(por intermédio de programas especiais chamados tradutores ou compiladores) para linguagem máquina. No caso do 803 há duas linguagens simbólicas aceitáveis pelo computador: o auto-código e o algol, sendo esta uma linguagem universal, isto é, susceptível de ser interpretada em computadores de outras marcas.

f) Perfuração: depois de estabelecido o programa é preciso perfurá-lo, isto é, fazer corresponder a cada símbolo do programa um caracter, ou seja, uma fiada de furos em determinadas posições perfurados numa fita especial, de que a seguir se apresenta um esquema



O número máximo de furos que se podem perfurar no sentido da largura da fita, ou, o que é o mesmo, o número máximo de posições da fita, dá-nos o chamado <u>número de canais</u> da fita; no 803 podem-se utilizar fitas de 5 e 8 canais, usando-se normalmente as de 5.

A cada posição na fita faz-se corresponder uma potência de 2 da maneira indicada na figura; à presença dum furo nessa posição associa-se a potência de 2 respectiva e à sua ausência o valor zero: assim dir-se-á que o 5º caracter da fita tem o valor decimal $2^1+2^2=6$. Conclui-se também fàcilmente que os diferentes caracteres possíveis têm valores decimais compreendido entre 0 (ausência de furos-"blank") e $31=2^0+2^1+2^2+2^3+2^4$ (letter shift).

Cada caracter pode ser interpretado como representando dois símbolos, consoante a natureza do chamado "shift" situado antes (não necessàriamente <u>imediatamente antes</u>) e que por sua vez é um caracter de valor 27 (figure shift) ou 31 (letter shift). Exemplificando, na figura apresentada estão perfurados os caracteres correspondentes aos símbolos

A1 = B3

antecedidos dos respectivos"shifts".

g) <u>Testes:</u> normalmente cometem-se erros em qualquer das fases anteriores, por maior que sejam os cuidados que se tenham. Para os eliminar é necessário realizar testes, que, para comodidade de exposição, vamos dividir em dois grupos:

g_1) Na memorização ou tradução e memorização (caso habitual) dum programa: consistem na observação da leitura da fita ou fitas onde estão as instruções do programa; se houver quaisquer instruções inadmissíveis pela sua forma ou se houver qualquer erro de perfuração, o computador acusa o erro quer por paragem da leitura da fita quer pela emissão de qualquer sinal através da unidade de saída.

g_2) Na execução: consistem na observação do comportamento do computador durante a execução das instruções do programa; pode este parar, e neste caso as lâmpadas de sinalização normalmente dão-nos indicação do tipo de erro ocorrido ou pode dar resultados que se revelem incorrectos, logo à primeira vista ou por comparação com os resultados obtidos por resolução manual ou com máquinas de calcular[1]. Algumas vezes é preciso percorrer novamente todos os passos desde o princípio para detectar a causa dos erros que surgem nos resultados.

h) Elaboração do relatório: o programador deve, após ter dado o programa como apto a funcionar, estabelecer o relatório respectivo, que conterá uma parte destinada a esclarecer o problema que o programa permite resolver e outra destinada pròpriamente ao sector de operação, na qual se especificarão as acções a tomar para execução dum problema concreto. Existem normas no LNEC para o estabelecimento desses relatórios.

O programador deve igualmente preencher uma ficha com indicações relativas ao programa em questão.

i) Entrada do programa na biblioteca de programas do LNEC: finalmente o programador dá por concluído o seu trabalho com a entrega no centro de cálculo do dossier com o relatório, que irá para um arquivo especial e da fita ou fitas que constituem o programa, que entrarão por sua vez num

(1) Deve-se resolver sempre o problema em questão com determinados valores numéricos para poder estabelecer o confronto e verificar se o programa está "afinado".

arquivo de fitas.

Lisboa e Laboratório Nacional de Engenharia Civil, em Abril de 1964

VISTO

JÚLIO FERRY BORGES
Engenheiro Chefe do Serviço de
Edifícios e Pontes

JOSÉ F. V. PALMA FERNANDES
Estagiário para Especialista

PF/JFFF