m5_040924.txt
este arquivo: m5_040924.txt
==========
TEMAS e MATERIAIS PARA ESTUDO - 040924
Consulte também: https://pater.web.cip.com.br/MaquInfo/
Em particular, para esta aula: https://pater.web.cip.com.br/MaquInfo/ch09.html ("PROCESSANDO PADRÕES SIMBÓLICOS")
Esquema: https://pater.web.cip.com.br/SI2024/cdpam2024.pdf

Acompanhe a leitura "Processando Padrões Simbólicos"
https://pater.web.cip.com.br/MaquInfo/ch09.html

com o esquema cdpam (computador digital de programa armazenado em memória)
https://pater.web.cip.com.br/SI2024/cdpam2024.pdf

=====
Vamos recapitular e complementar a matéria dada até agora e fazer uma primeira

apresentação muito geral e introdutória do funcionamento de um computador (ordenador simbólico) digital de programa armazenado em memória (chamado de "arquitetura Von-Neumann")

=====

Até agora, falamos de padrões simbólicos, binários, hexadecimais, decimais, etc. que representam informações estáticas, ou "estados de coisas", por assim dizer.

Para circularem como mensagens e produzirem trabalho útil os padrões simbólicos devem ser guardados, recuperados, transferidos, comparados, manipulados.

Um computador, ou "ordenador simbólico" ("ordinateur" em francês, ou "ordenador" em espanhol ou português de Portugal), é uma máquina capaz de organizar diferentes conjuntos de símbolos seguindo instruções.

O cômputo de valores é um caso particular importante entre as muitas faculdades de organizar símbolos que os computadores possuem.

Assim, usamos os computadores para movimentar aqueles padrões estáticos e obter, assim, novos estados de coisas. atuais ou potenciais, concretos ou abstratos.

Pois um computador, para poder funcionar, organiza símbolos através de variações mecânicas, ópticas, elétricas ou eletromagnéticas dos seus estados.

Aqueles símbolos manipulados têm necessáriamente, além do valor simbólico atribuído a eles, um efeito sobre entorno da máquina, capaz de produzir trabalho.

Os símbolos, em um computador, não são meramente símbolos, portanto, mas símbolos-interruptores, por assim dizer, capazes de controlar, ligar e desligar dispositivos direta ou remotamente acoplados a eles.

Basta que se encontrem em locais "ativos" da bitsfera (barramentos ou portas de Entrada e Saída) para exercerem seus efeitos sobre tais dispositivos acoplados.

Que efeito terão externamente aqueles sinais, isso já depende do funcionamento particular do dispositivo externo ou remoto, do "periférico", não diz mais respeito ao núcleo do computador.

Certamente, se houver retroalimentação, como geralmente ocorre, entre os dispositivos acoplados, é possível gerenciar o dispositivo externo através do tratamento coordenado dos dados de entrada e de saída disponíveis para o computador.

Os gerenciadores de dispositivos ("drivers"), por exemplo, são programas - sequências de instruções processadas pela máquina - que servem para isso.

=====

Entre os códigos que usamos até agora, há os que servem para indicar ("numerar") as posições de outros símbolos em uma lista, evitando a tarefa de contar linhas para encontrar a posição de um símbolo.

E há os outros símbolos propriamente ditos, cujas posições na lista dependem das convenções usadas na formação da lista ordenada, como a tabela ascii que vimos anteriormente.

Podemos falar das posições relativas entre dois símbolos quaisquer em uma sequência calculando a distância entre eles, contando na ordem direta - do "anterior" para o "posterior" - ou vice-versa.

As expressões "15 para as 9", ou "9 e 15", representam certa distância, retrógrada ou direta (negativa ou positiva, faltante ou excedente), do ponteiro dos minutos em relação à hora 9 exata do relógio.

O "zero" da medição parte das horas exatas, na direção negativa para as "faltas", e na direção positiva para os "excessos".

Podemos, então, representar números negativos, as "faltas", as "dívidas", como movimentos retrógrados a partir de um zero convencional qualquer.

O ponteiro dos minutos sobre o 9 significa, como sabemos, tanto um "+ 45" como um "- 15".

Como evitar ambiguidades em um sistema que usa os movimentos retrógrados ou diretos para expressar faltas ou excessos, isto é, valores negativos ou positivos?

Se reservarmos parte do nosso repertório simbólico para os excessos e outra parte para as faltas, eliminamos a ambiguidade simbólica à custa do repertório agora reduzido para cada parte.

Em um registrador decimal de três posições que se movimenta livremente nos dois sentidos
(...,997,998,999,000,001,002,003,...,998,999,000,001,...,etc.) poderíamos reservar praticamente o mesmo número de posições
negativas e positivas, assim

```
500   (menos quinhentos)
501   (menos quatrocentos e noventa e nove)
502   (menos quatrocentos e noventa e oito)
503   (menos quatrocentos e noventa e sete)
...
996   (menos quatro)
997   (menos três)
998   (menos dois)
999   (menos um)
000   ZERO   <-----o sucessor mecânico de 999 não é 1000 porque nossa catraca só tem três posiçoes
001   (um)
002   (dois)
003   (três)
004   (quatro)
...
497   (quatrocentos e noventa e sete)
498   (quatrocentos e noventa e oito
499   (quatrocentos e noventa e nove)
```

Como essa "máquina" acima contém, além de uma posição para o ZERO, 499 posições "positivas" e 500 posições "negativas", pode representar inteiros entre -500 e +499 incluindo o ZERO.

Note que quatrocentos e noventa e sete somado a menos quatrocentos e noventa e sete deve, aritmeticamente, dar zero.

Na nossa máquina de representação "complementar" acima, isso é facilmente verificável.

Pois

```
503      (menos quatrocentos e noventa e sete)
497  +   (somado a mais quatrocentos e noventa e sete)
000      (dá ZERO)
```

Note que para retroceder do ZERO até o desenho '503' precisamos andar para trás 497 posições.

Andar para trás 497 posições e, depois, andar para frente outras 497 posições nos leva ao lugar inicial, no caso ao ZERO.

Há portanto a adição "mecânica" 503 + 497 que produz o resultado "mecânico" 000; e uma interpretação aritmética, 503 significando o símbolo da posição obtida pelo retrocesso de 497 passos a partir do ZERO.

Assim, para interpretar o significado dos desenhos "negativos" (isto é, para calcular o valor absoluto do negativo representado por certo símbolo), precisamo "complementar" o desenho, ou dito de outro modo, calcular o retrocesso.

Como 497 é o complemento que falta a 503 para chegar ao ZERO do sistema, chamamos a essa representação de negativos de "notação por complemento" (no caso, complemento a mil, aritmeticamente, ou complemento a ZERO, mecanicamente).

Note que não precisamos "complementar" representações positivas: a distância de um desenho "positivo" ao ZERO da catraca é o próprio valor positivo. A distância do valor positivo 497 do ZERO é 497.

Para facilitar o cálculo da distância de um desenho negativo (entre 500 e 999) ao ZERO, podemos buscar a distância ao 'menos um' (no caso, ao desenho '999') e acrescentar 1 para obtermos a distância ao ZERO.

Isso porque é imediato subtrair qualquer valor de 999. No caso, a diferença (999 - 503) é representada pelos algarismos (9-5)(9-0)(9-3) = 496.

496 é, portanto, a distância de 503 ao 999 (do desenho negativo 503 ao desenho negativo 999 que sabemos significar 'menos um').

Para saber a distância do desenho 503 ao ZERO, basta acrescentar 1 ao 496. Portanto, 496+1=497 é a distância do desenho 503 ao ZERO. Assim, o desenho 503 representa o número negativo -497.

Como discutimos acima, o minuto 45 de um reÍógio também representa o valor negativo -15.

Uma breve recapitulação sobre sequências binárias e suas conversões antes de falarmos das catracas (registradores) binárias que representam negativos de modo equivalente ao descrito acima para os decimais.

Com dois símbolos elementares, os bits 0 e 1, podemos representar apenas duas coisas diferentes. Ou alguma coisa e sua ausência, se quisermos.

Mas, pela justaposição bits em certo regime de formação de sucessores e antecessores, podemos, como sabemos, representar qualquer número de coisas diferentes.

O número de representações possíveis depende do número de casas binárias de que dispomos. Se certo registrador de um computador possui uma casa binária, é possível representar com aquele registrador apenas dois estados, zero ou um.

Mas se tivermos um registrador com duas posições, poderemos representar quatro estados ou desenhos diferentes, a saber

00
01
10
11

Se quisermos representar mais estados diferentes, precisamos de mais posições binárias para os desenhos. Por exemplo, podemos representar oito coisas diferentes com registadores de três bits:

000
001
010
011
100
101
110
111

Obtivemos os sucessores de cada desenho à luz do que fazemos com os decimais: primeiro movimentamos, até o último, os algarismos das unidades; depois de esgotado o repertório, movimentamos os algarismos mais à esquerda, e assim por diante.

Como em binário só existem dois algarismos elementares, 0 e 1, o sucessor do 01 é o 10, o sucessor do 011 é o 100. Assim como, em decimal, o sucessor de 09 (o valor mais alto com uma posição) é 10, ou o sucessor de 099 (o valor mais alto com duas posições) é 100.

Veja que o fato de termos

quatro representações possíveis para duas posições binárias, ou

oito representações possíveis para três posições binárias

não é apenas evidente, mas necessário.

0
1

esgotam o repertório de desenhos (símbolos compostos, ou simplesmente símbolos) com uma posição binária.

Para ter mais desenhos, precisamos acrescentar novas posições. Porque se simplesmente repetíssemos a sequência anterior

0
1
0
1

para representar, digamos, o dobro de coisas diferentes, não haveria diferença perceptível entre o primeiro e o terceiro, ou entre o segundo e o quarto símbolos.

Uma primeira solução para diferenciar os quatro símbolos seria, talvez, marcá-los:

0
1
.0
.1

ou

0x
1x
0y
1y

Usamos aqui símbolos adicionais para elimianar as ambiguidades de representação. Uma complicação desnecessária, pois poderíamos combinar os próprios simbolos elementares entre si.

Assim, imitando a lei de formação dos sucessores em decimal, temos a sequência (I):

00
01
10
11

Ou, invertendo a ordem do marcador à esquerda, primeiro o 1, depois o 0, temos a sequência (II):

10
11
00
01

Ou ainda, mudando apenas 1 bit de cada vez, a sequência (III)

00
01
11
10

Todas as sequências acima atendem a exigência de serem diferentes e, desse modo, podem reprepresentar quatro coisas diferentes.

Por várias razões, representamos sequências binárias correntes com a ordem da primeira sequência (I).

A primeira sequência mantém da direita para a esquerda, como nos decimais, a ordem de modificação sucessiva de algarismos; e considera 0 antecessor de 1 em todas as posições dos desenhos.

Ao repetir a sequência

0
1

obtemos, claro, o dobro de estados representados:

0 0
0 1
1 0
1 1

marcando com 0 à esquerda o primeiro par, e com 1 à esquerda o segundo par repetido.

Marcações similares podem ser feitas à esquerda dos quatro desenhos obtidos, de modo a obtermos mais símbolos diferentes, assim:

0 00
0 01
0 10
0 11
1 00
1 01
1 10
1 11

Não é difícil ver que "repetir o conjunto anterior" dobra o número de representações diferentes; e que "marcar à esquerda" o conjunto anterior, a primeira vez com 0, e a repetição com 1, aumenta de 1 unidade o número de bits (casas) da representação.

Assim, é equivalente "dobrar o número de representações" e "acrescentar 1 bit às posições (casas) existentes".

Quando passamos de

01

para

10

este 10 indica, como vimos para os decimais, que o repertório à direita foi esgotado (1 vez). Pois passamos por 00 e 01 para chegar ao 10.

Do mesmo modo, quando passamos de

011

para

100

este 100 indica que todo o repertório à direita do 1 foi esgotado, pois atravessamos todas as representações de 000 a 011 para chegar ao 100:

0 00
0 01
0 10
0 11
1 00

Cada vez que um 0 à esquerda se torna 1, começa a repetição do padrão à direita até que todos os bits do padrão sejam 1. Numerando os bits de 0 a 7, da direita para a esquerda:

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 <------esgotado o repertório 0 a 1 do bit 0 (segundo desenho - número 1)
0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 1 <------esgotado o repertório 00 a 11 dos bits 0 e 1 (quarto desenho - número 3)
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 1

0 0 0 0 0 1 1 0
0 0 0 0 0 1 1 1   <------esgotado o repertório 000 a 111 dos bits 0, 1 e 2 (oitavo desenho - número 7)
0 0 0 0 1 0 0 0   <------marcador indicando que o repertório de 000 a 111 já foi esgotado - nono desenho - número 8)

EXERCÍCIO: completar a sequência de oito bits acima até alcançar o padrão completo de uns nos oito bits, isto é até
...
1 1 1 1 1 1 1 1   <------esgotado o repertório completo do registrador de 8 bits, que representa padrões de 0000 0000 a 1111 1111 (desenho 256 - número 255)

Temos, então:

1   valendo 1
10   valendo 2
100   valendo 4
1000   valendo 8
10000   valendo 16

etc.

E, também,

1   valendo 1 e antecessor de 10
11   valendo 3 e antecessor de 100
111   valendo 7 e antecessor de 1000

etc.

Assim, 5 "palitos" é o antecessor de 1 seguido de 5 zeros; cem "palitos" é o antecessor de 1 seguido de cem zeros.

Cada deslocamento de um padrão binário de uma posição para a esquerda multiplica o padrão por 2.

Assim como, em decimal, cada deslocamento do padrão decimal para a esquerda multiplica o padrão por dez.

E assim como cada deslocamento de um padrão hexadecimal para a esquerda multiplica o padrão por 16.

Podemos falar, de modo mais apropriado, em vez de "deslocamento do padrão para a esquerda", de "deslocamento da vírgula para a direita".

Em decimal, um deslocamento da vírgula para a direita multiplica o padrão decimal por dez. Para a esquerda, o deslocamento da vírgula divide o padrão por dez.

De modo semelhante, deslocar a vírgula de um padrão binário para a direita, multiplica o padrão por 2; e deslocar para a esquerda a vírgula em um padrão binário divide o padrão por 2.

O dobro do binário 101b é 1010b. Se soubermos que o correspondente decimal de 101b é 5, é fácil deduzir que o correspondente de 1010b é dez, que vinte corresponde ao binário 10100b, quarenta a 101000b, e assim por diante.

Sabemos acrescentar ou subtrair uma unidade a um binário (obter seu sucessor ou antecessor); e sabemos multiplicar ou dividir por 2 um padrão binário (bastando mudar a vírgula de lugar).

Podemos, então, calcular - e, mais importante, entender - facilmente as conversões decimal-binário e suas derivadas decimal-hexadecimal, decimal-octal, e vice-versa.

Finalizando esta recapitulação, abaixo se procurou deduzir os passos da conversão de um padrão decimal em um padrão binário representando o mesmo número, isto é, a mesma distância do zero a partir de uma lista decimal ou binária, respectivamente.

Isto significa codificar, ou "converter": encontrar a posição correspondente de um símbolo de uma lista de origem na lista de destino. Encontrando a posição, localizamos - ou podemos deduzir - os símbolos correspondentes em cada lista (ou catraca).

Recordando sempre que não se trata aqui de "saber calcular", de um virtuosismo no cálculo, mas de "entender o que se está fazendo", de poder, em circunstâncias diversas aplicar conhecimentos gerais, aqui está um

exercício resolvido:

Encontre o padrão binário correspondente ao inteiro decimal 1.234.567d explicando os passos sucessivos da conversão.

Vamos listar o que já sabemos e tentar deduzir disso o que ainda não sabemos.

SABEMOS que 1234567d corresponde a algum binário com posições de 0 a n, da direita para a esquerda.

SABEMOS que, seja qual for esse padrão binário, terá a seguinte forma, numerando os bits da direita para a esquerda a partir do bit 0, com um número de bits ainda não calculado:

```
...    .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
...    26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9  8  7  6  5  4  3  2  1  0
```

SABEMOS que o número que 1234567d representa É ÍMPAR, portanto o padrão binário correspondente a 1234567d não pode representar um múltiplo de 2. Em outras palavras, como não pode ter ZERO como bit 0, o bit 0 do padrão binário será 1:

```
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1   <-----esta sequência binária ÍMPAR desconhecida corresponde ao decimal 1234567d
```

É imediato encontrar o binário correspondente a 1234566d (que representa um número PAR) se já houvéssemos calculado o binário correspondente à sua METADE (???????b ----> 617283d):

2 x 616283d = 1234566d, que corrresponde a

10b x ???????b = ????????0b

(Por quê?)

SABEMOS que o ANTECESSOR do padrão binário tem ZERO como bit 0, que podemos representar provisoriamente assim:

. . . . 0 <-----esta sequência binária PAR, antecessora da primeira, corresponde ao decimal 1234566d

SABEMOS também que quando conhecermos os demais bits do padrão binário já teremos a resposta, pois basta acrescentar 1 ao antecessor acima para obter o padrão binário correspondente a 1234567d, assim:

. . . . 0 + 1
26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Repetindo o que dissemos acima,

SUPONDO que conhecêssemos o padrão binário correspondente à METADE de 1234566d, isto é, se soubéssemos quais são os bits correspondentes a 1234566d / 2 = 617283d
(Vamos representar o tal binário hipotético por um rabisco, por uma sequência de traços):

"--------------------" = padrão binário correspondente ao decimal 617283d, metade de 1234566d

SABEMOS que, neste caso, seria fácil obter o padrão binário original, correspondente ao DOBRO de 617283d, o decimal 1234566d e, em seguida, seu sucessor 1234567d:

Para o DOBRO, colocamos um ZERO à direita do padrão binário; para o sucessor de um binário PAR, substituímos o 0 do bit mais à direita por 1. Isto é,

"--------------------"0 (= DOBRO de "--------------------" obtido pelo ZERO à direita) + 1 (="--------------------"1, SUCESSOR de "--------------------"0)

Pelas mesmas razões,

SABEMOS que o ÍMPAR 617283d deve ser representado por um binário (o rabisco de pontos "..............." abaixo) com um bit 1 à direita:

"..............."1

e seu antecessor, correspondente ao decimal 617282d, com um bit 0 à direita:

"..............."0

CONHECENDO os bits correspondentes à METADE de 617282d (= 308641d), seria fácil encontrar o DOBRO, seu SUCESSOR, novamente o DOBRO e seu SUCESSOR, e assim por diante.

A ideia é reconstituir os dobros a partir das representações binárias das metades, atentos para as operações subtração de uma unidade usadas para obter valores divisíseis por 2 (pares).

Reconstruímos DE CIMA PARA BAIXO o que calculamos DE BAIXO PARA CIMA:

```
      1d x 2 =
      2d + 1 =
      3d x 2 =
      6d x 2 =
     12d x 2 =
     24d x 2 =
     48d + 1 =
     49d x 2 =
     98d x 2 =
    196d + 1 =
    197d x 2 =
    294d + 1 =
    295d x 2 =
    590d + 1 =
    591d x 2 =
   1182d + 1 =
   1183d x 2 =
   2286d x 2 =
   4572d x 2 =
   9144d + 1 =
   9145d x 2 =
  19290d x 2 =
```

```
  38580d x 2 =
  77160d x 2 =
 154320d x 2 =
 308640d + 1 =
 308641d x 2 =
 617282d + 1 =
 617283d x 2 =
1234566d + 1 =
1234567d

1d corresponde a 1b, logo, com a coluna à direita representando os binários
      1d x 2 = 10
      2d + 1 = 11
      3d x 2 = 110
      6d x 2 = 1100
     12d x 2 = 11000
     24d x 2 = 110000
     48d + 1 = 110001
     49d x 2 = 1100010
     98d x 2 = 110000100
    196d + 1 = 110000101
    197d x 2 = 1100001010
    294d + 1 = 1100001011
    295d x 2 = 11000010110
    590d + 1 = 11000010111
    591d x 2 = 110000101110
   1182d + 1 = 110000101111
   1183d x 2 = 1100001011110
   2286d x 2 = 11000010111100
   4572d x 2 = 110000101111000
   9144d + 1 = 110000101111001
   9145d x 2 = 1100001011110010
  19290d x 2 = 11000010111100100
  38580d x 2 = 110000101111001000
  77160d x 2 = 1100001011110010000
 154320d x 2 = 11000010111100100000
 308640d + 1 = 11000010111100100001
 308641d x 2 = 110000101111001000010
 617282d + 1 = 110000101111001000011
 617283d x 2 = 1100001011110010000110
1234566d + 1 = 1100001011110010000111
1234567d
```

Consulte também:

https://pater.web.cip.com.br/MaquInfo/ch03.html

"Posições, conversões e aritmética binária"

EXERCÍCIO PARA RESOLVER:

Encontre o padrão binário correspondente ao inteiro decimal de nove dígitos que representa o seu DRE.

Note que não se trata de obter o código ascii de cada algarismo do seu DRE, mas de obter a representação binária do múmero que o DRE representa.

Evite decorar os cálculos, ou fazer cálculos por fazer. Procure entender a razão de ser de cada passo na obtenção das correspondências.

De modo similar, como vimos na aula anterior (020924), podemos representar em binário o "menos 1" pelo desenho obtido quando movimentamos para trás uma catraca binária com o valor inicial zero.

Com oito bits, temos

1000 0000 (menos cento e vinte e oito)
1000 0001 (menos cento e vinte e sete)
1000 0010 (menos cento e vinte e seis)
1000 0011 (menos cento e vinte e cinco)

...

1111 1101 (menos três)
1111 1110 (menos dois)
1111 1111 (menos um)
0000 0000 ZERO <-----o sucessor mecânico binário de 1111 1111 não é 1 0000 0000 porque nossa catraca só tem oito posiçoes binárias
0000 0001 (um)
0000 0010 (dois)
0000 0011 (três)

```
0000 0100   (quatro)

...

0111 1100   (cento e vinte e quatro)
0111 1101   (cento e vinte e cinco)
0111 1110   (cento e vinte e seis)
0111 1111   (cento e vinte e sete)
```

Abaixo, reproduzo o mesmo argumento (e a mesma construção, a bem da verdade) usado na explicação sobre a representação de negativos por "complemento" na catraca decimal de três posições acima. Veja que são usadas as mesmas expressões, mudando apenas os desenhos e progressões dos símbolos.

Como essa "máquina" binária acima contém, além de uma posição para o ZERO, 127 posições "positivas" e 128 posições "negativas", pode representar inteiros entre -128 e +127 incluindo o ZERO.

Note que cento e vinte e sete somado a menos cento e vinte e sete deve, aritmeticamente, dar zero.

Na nossa máquina de representação "complementar" acima, isso é facilmente verificável.

Pois, sempre em binário,

```
1000 0001b   (menos cento e vinte e sete)
0111 1111b + (somado a cento e vinte e sete)
0000 0000b   (dá ZERO)
```

Use, se quiser, o mesmo procedimento da soma de decimais (vai-um, etc.) recordando que há somente dois valores possíveis por casa binária.

Ou use um artifício, para não ter que operar com tantos "vai-um":

```
1000 0001b = 1000 0000b + 1
[menos cento e vinte e sete] = [menos cento e vinte e oito] + 1

0111 1111b + 1000 0000b + 1 = 1111 1111b + 1 = 0000 0000b
[cento e vinte e sete] mais [menos cento e vinte e oito] mais um = menos um mais um = ZERO
```

Para o cálculo binário não precisamos nos expressar (nem falar, nem escrever) em decimal.

Note que para retroceder do ZERO até o desenho '1000 0001' precisamos andar para trás 0111 1111b posições.

Andar para trás 0111 1111b posições e, depois, andar para frente outras 0111 1111b posições nos leva ao lugar inicial, no caso ao ZERO.

Há portanto a adição "mecânica" 1000 0001b + 0111 1111b que produz o resultado "mecânico" 0000 0000b; e uma interpretação aritmética, 1000 0001b significando o símbolo da posição obtida pelo retrocesso de 0111 1111b passos a partir do ZERO.

Assim, para interpretar o significado dos desenhos "negativos" (isto é, para calcular o valor absoluto do negativo representado por certo símbolo), precisamo "complementar" o desenho, ou dito de outro modo, calcular o retrocesso.

Como 0111 1111b é o complemento que falta a 1000 0001b para chegar ao ZERO do sistema, chamamos a essa representação de negativos de "notação por complemento a dois" (pois se trata de um "complemento a um" + 1, como veremos abaixo).

Note que não precisamos "complementar" representações positivas: a distância de um desenho "positivo" ao ZERO da catraca é o próprio valor positivo. A distância do valor positivo 01111 1111b do ZERO é 0111 1111b.

Para facilitar o cálculo da distância de um desenho negativo (entre 1000 0000b e 1111 1111b) ao ZERO, podemos buscar a distância ao 'menos um', o "complemento a um" (no caso, ao desenho '1111 1111b') e acrescentar 1 ap resultado para obtermos a distância ao ZERO.

Isso porque é imediato subtrair qualquer byte de 1111 1111b. No caso, a diferença (1111 1111b - 1000 0001) é representada por 0111 1110b, chamado "complemento a um" - ou o inverso - de 1000 0001b.

0111 1110b é, portanto, a distância de 1000 0001b ao 1111 1111b (do desenho negativo '1000 0001' ao desenho negativo '1111 1111' que sabemos significar 'menos um').

=====

Imagine um jogo com quatro caixas, chamadas A, B e C e Z, inicialmente vazias, ou com um número desconhecido de objetos (bolas, pedras, moedas, palitos).

Variando a quantidade de objetos em cada caixa (observadas simultaneamente) podemos modificar e registrar o "estado" (situação) geral do jogo em determinado momento.

O jogo progride através da repetição de uma sequência de movimentos idênticos, cada um deles compreendendo a realização de um ato predeterminado capaz de alterar o estado do jogo a cada momento.

Os passos modificam o estado do jogo, mas não a sequência de movimentos.

A sequência de atos (com efeito sobre o estado do jogo) dinamizados através aqueles movimentos repetidos não é sempre a mesma.

Depende de uma "sequência predeterminada de instruções", de um "programa".

Nos jogos de tabuleiro, por exemplo, estamos acostumados com a alternância entre jogadores: eu jogo, você joga, eu jogo, você joga, etc., um jogo diferente do outro, todos seguindo esta mesma regra.

Escrevemos textos muito diferentes com as mesmas regras gerais: da esquerda para a direita, de cima para baixo, o verso da página do lado esquerdo do caderno.

Há "jogos de escrita" com outras regras, mas o argumento é o mesmo.

As caixas A, B, C e Z, como dissemos, estão "vazias" ao início do jogo, ou, o que é indiferente para o nosso jogo, com uma quantidade desconhecida de objetos dentro delas.

Note que a caixa vazia aqui representa uma caixa com o valor zero dentro dela. Não é assim que representamos hoje o zero na escrita de números em geral.

O zero é normalmente representado por alguma marca ou símbolo, e não por uma ausência.

Para ser mais preciso, temos sistemas diferentes de representação de ordem e de quantidade que parecem conviver muito bem entre si. Mas precisamos estar atentos.

Nas planilhas, por exemplo, "numeramos" as colunas com um sistema muito diferente do que usamos para "numerar" as linhas.

Isso apesar de começarmos, geralmente, a contar a partir da unidade, isto é, da linha '1' e da coluna 'A'.

Ao chegarmos à linha '9', passamos para a linha '10'; ao chegarmos à '99', passamos para a linha '100', etc.

Mas a coluna sucessora de 'Z' é a coluna 'AA', a de 'ZZ' é a 'AAA', o que equivale a dizer que o sucessor de '9' é o '11', ou que o sucessor de '99' é o '111', usando algarismos em vez de letras para numerar as colunas.

A numeração das colunas é posicional sem o zero. A das linhas é posicional com o zero, mas sem utilizar o zero como primeiro elemento.

Há muitas formas de se representar o zero, como sabemos. Em geral, 000 significa normalmente a mesma coisa que 00, ou que 0.

Mas, nas colunas das planilhas, também sabemos que 'A' é muito diferente de 'AA' e de 'AAA'. E de nada serve anteceder de zeros os primeiros 'A's. Pois '0Z' não tem como sucessor 'A0' nas planilhas, e sim 'AA'.

A complicação e a perplexidade que a representação do zero (do vazio, do nada, da ausência) - sempre produziram através dos tempos vão aparecer aqui também, mas estaremos atentos para não errar (muito).

Nas próximas aulas, veremos o funcionamento de uma máquina de programa armazenado em memória, a partir de um jogo de quatro caixas

de conteúdo variável como o mencionado aqui.

====
fim_de_m5_040924.txt