

m4_SI2025.txt

este arquivo: m4_SI2025.txt

=====

TEMAS e MATERIAIS PARA ESTUDO - SI2025

Consulte também: <https://pater.web.cip.com.br/MaquInfo/>

É ilustrativo imaginar uma sequência de números inteiros a partir do zero como o resultado do movimento de avançar uma posição em uma catraca decimal.

Assim podemos observar posições (casas) que se movimentam mais ou menos rapidamente, e como depende seu movimento do movimento das demais, como se propagam alterações e em que direção.

E o que acontece ao alcançarem, todas as posições, o dígito de maior valor representável no sistema - no caso decimal, o algarismo 9 (nove).

Um "número", nesta acepção limitada, seria então uma disposição de dígitos alcançada por movimentos sucessivos da catraca, a partir do zero.

Em uma catraca decimal de 3 casas, teríamos as seguintes regras de deslocamento a partir do zero e avançando uma unidade a cada movimento, da direita para a esquerda:

[A capacidade de representação da máquina abaixo vai de 000 a 999; ao todo mil estados ou representações diferentes entre si]

0	0	0	<-----zero inicial-----início do primeiro ciclo
0	0	1	
0	0	2	
0	0	3	
0	0	4	
0	0	5	
0	0	6	
0	0	7	
0	0	8	
0	0	9	<-----dígito de maior valor
0	1	0	<-----retorno ao zero--propagação do movimento para a posição imediatamente à esquerda-----início do segundo ciclo
0	1	1	(o 1 na posição das dezenas indica na máquina que o repertório elementar de 0 a 9 se esgotou 1 vez)
0	1	2	
0	1	3	
0	1	4	
0	1	5	

```

0  1  6
0  1  7
0  1  8
0  1  9
0  2  0 <-----novo ciclo-----nova propagação do movimento para a esquerda
0  2  1 (o 2 na posição das dezenas indica na máquina que o repertório elementar de 0 a 9 se esgotou pela segunda vez)
0  2  2

...

0  8  8
0  8  9
0  9  0 <-----novo ciclo-----nova propagação
0  9  1 (o 9 na posição das dezenas indica na máquina que o repertório elementar de 0 a 9 se esgotou pela nona vez)
0  9  2
0  9  3
0  9  4
0  9  5
0  9  6
0  9  7
0  9  8
0  9  9 <-----dígito de maior valor na primeira e na segunda posição-----fim de ciclo na primeira e na segunda posições
1  0  0 <-----novo ciclo de 00 a 99-----propagação do movimento das duas primeiras posições para a terceira
1  0  1 (o 1 na posição das centenas indica na máquina que o repertório composto de 00 a 99 se esgotou pela primeira vez)
1  0  2
1  0  3
1  0  4
1  0  5
1  0  6
1  0  7
1  0  8
1  0  9
1  1  0
1  1  1
1  1  2
1  1  3
1  1  4

...

9 8 9
9 9 0

```

```

9 9 1
9 9 2
9 9 3
9 9 4
9 9 5
9 9 6
9 9 7
9 9 8
9 9 9
0 0 0 <-----retorno ao ZERO NAS TRÊS POSIÇÕES-----o repertório completo de 000 a 999 se esgotou
0 0 1 (não foram esgotados - ou foram esgotados mas não registrados pela máquina o repertório de 0 a 9 nem o   de 00 a 99
0 0 2
0 0 3
0 0 4
0 0 5
0 0 6
0 0 7
0 0 8
0 0 9
0 1 0
0 1 1
0 1 2
0 1 3
0 1 4
...

```

Sem necessidade de fazer contas, pois costumamos falar e escrever em decimal, sabemos que a catraca decimal acima de três posições pode representar mil estados diferentes, de 000 a 999.
(Porque começamos a contar do 000, temos $999 + 1 = 1000$ estados diferentes)

E se movimentássemos uma catraca de 3 casas mas com apenas 8 algarismos, de 000 a 777, sem os 8 e sem os 9?
A volta ao algarismo inicial 0 e a consequente propagação do movimento para a esquerda viriam após o avanço do algarismo de maior valor, o 7

```

0   0   0
0   0   1
0   0   2
0   0   3
0   0   4
0   0   5

```

0	0	6
0	0	7
0	1	0
0	1	1
0	1	2
0	1	3
0	1	4
0	1	5
0	1	6
0	1	7
0	2	0
0	2	1
0	2	2
0	2	3

...

0	7	6
0	7	7
1	0	0
1	0	1
1	0	2

...

7	6	7
7	7	0
7	7	1
7	7	2
7	7	3
7	7	4
7	7	5
7	7	6
7	7	7

0 0 0
0 0 1
etc.

Para saber quantos desenhos diferentes esta catraca "octal" pode fazer precisamos calcular. Pois chegar a 777 não significa somar sete centenas a sete dezenas e a sete unidades.

Assim como, em octal, 10 não significa dez, pois 10 octal é o sucessor de 7 (não existem os algarismos 8 ou 9). A leitura 077 em octal significa que esgotamos o repertório de oito dígitos (0 1 2 3 4 5 6 7) sete vezes e adicionamos 7 movimentos ao resultado.

Interpretando o significado (o correspondente decimal) de 77 octal, temos $7 \times 8 + 7 = 63$; sete vezes o repertório de oito dígitos com o acréscimo de 7 unidades.

Obtivemos no movimento entre 00 e 77 octal 64 (63+1) representações diferentes, contando o 00.

Assim como, em decimal, no movimento entre 00 e 99 obtivemos cem representações diferentes contando o zero.

A leitura 777 em octal significa que esgotamos o conjunto 00 a 77 (64 símbolos, ou pares de algarismos diferentes).

Vimos na segunda aula (m2_SI2025) as sequências hexadecimais e binárias, que seguem regras de formação similares: esgotamento dos ciclos e propagação dos movimentos.

Recapitulando, 10 em hexadecimal não significa dez, pois 10 hexadecimal é o sucessor de F (F = 15 decimal, pois temos os algarismos A B C D E F após o 9).

O sucessor de quinze é dezesseis, representado em hexadecimal como 10 ($0F + 1 = 10$).

F0 em hexadecimal representa F vezes o esgotamento do ciclo de 16 algarismos de 0 a F ($0Fh \times 10h$, ou $15d \times 16d$). Com o acréscimo de F, temos $F0h + F = FFh$. Em decimal, $240d + 15d = 255d$.

Portanto, de 00 a FF, em hexadecimal, contando o zero, temos FFh representações de 01 a FF + 1 representação para o zero. $FFh + 1 = 100h$, que corresponde a $255d + 1 = 256d$.

Do mesmo modo, 90d representa 9 vezes o esgotamento do ciclo de 0 a 9 ($9 \times 10d = 90d$). Com o acréscimo de 9, temos $90d + 9 = 99d$.

O que está descrito aqui se aplica a qualquer base de numeração (codificação): à base 64, muito usada para transmitir por email anexos binários na forma de texto, assim como se aplica à base 58 nas moedas digitais, como o bitcoin.

Algumas operações com binários são triviais, outras nem tanto.

Complementar a um um padrão binário é substituir os uns por zeros e os zeros por uns, de modo que a "soma" dos dois padrões resulte no dígito 1 em todas as posições.

Por exemplo,

0101010111 é o complemento a 1 de
1010101000 (e vice-versa)

Complementar a dois (também na base 2) é um pouco mais trabalhoso, mas necessário na representação de negativos.

Para trabalhar com números negativos, podemos pensar em uma máquina-catraca que reserva parte de seus estados, em particular os números mais altos para representar negativos, através de movimentos retrógrados, do tipo

000
999
998
997
...
502
501
500

para indicar

000	zero
999	-1
998	-2
997	-3
...	
502	-498
501	-499
500	-500

Com binários, teríamos algo assim, representando para cima os negativos,

1000	0000	-128
1000	0001	-127
...		
1111	1101	-3
1111	1110	-2
1111	1111	-1
0000	0000	zero
0000	0001	+1
0000	0010	+2
...		
0111	1110	+126
0111	1111	+127

É conveniente o 1 à esquerda dos negativos e o 0 à esquerda dos positivos para identificar imediatamente o significado do

padrão binário que representa positivos e negativos em complemento a 2.

O nome "complemento a dois" é, a rigor, um abuso de linguagem. Significa "[complemento a um] + 1", e não "complemento a 1+1".

Na lista de bytes (binários com 8 bits) em complemento a dois acima, se quiséssemos saber a que distância está o padrão 1111 1101 do zero, isto é, se quiséssemos saber quão negativo é o padrão (seu valor absoluto, podemos operar do modo seguinte.

Sabendo a distância de um negativo qualquer ao valor -1 (menos um), para saber a distância ao zero basta acrescentar uma unidade.

Se calculamos que -5 (menos 5) está a uma distância de 4 unidades (movimentos de catraca) do -1 (menos 1), sabemos que a distância do -5 ao zero é $4 + 1$, isto é, sua distância ao $-1 + 1$.

Calcular a distância ao -1 é imediato. Como a representação de -1 em complemento a dois é sempre um agrupamento de uns (no exemplo com 1 byte, o -1 corresponde a oito uns: 1111 1111), a distância ao -1 de qualquer padrão binário negativo (representado em complemento a dois) é o complemento a 1

Assim, no exemplo dado, 1111 1101, padrão binário negativo de 1 byte representado em complemento a dois, difere (dista) de 1111 1111 (menos 1) do seu complemento a 1, a saber 0000 0010.

Pois o complemento a 1 de

1111 1101 é

0000 0010

este último valor equivalendo a quanto o 1111 1101 deve "andar" para chegar ao menos 1 (quanto falta para 1111 1101 chegar a 1111 1111).

Somando uma unidade a 0000 0010, temos a distância do -5 ao zero (quanto falta para 1111 1111 chegar ao 0000 0000), isto é, $0000\ 0010 + 1 = 0000\ 0011$ (no caso, três, em decimal, mas a conversão decimal é irrelevante para nosso cálculo).

Se faltam 3 movimentos diretos para o 1111 1101 chegar ao zero, sua distância ao zero é três, o valor absoluto do negativo 1111 1101.

Há outros métodos para representar números negativos, com outras finalidades e propriedades, como a notação chamada "por excesso", ou como a notação de "inteiros com sinal", parecida à que usamos normalmente.

Uma balança que utiliza uma "tara", um peso inicial para calibrar o zero, ilustra a representação de um negativo por excesso. O zero real da balança está na faixa "negativa" da medição. É necessário "subtrair" o valor "excessivo" medido pelo peso "inicial" do prato da balança.

As observações acima sobre são introdutórias e devem ser complementadas.

Não deixe de consultar

<https://pater.web.cip.com.br/MaquInfo/ch04.html>

e

<https://pater.web.cip.com.br/MaquInfo/ch07.html>

para uma primeira leitura, exemplos e exercícios adicionais, em particular representações de números negativos (excesso, complemento a um e complemento a dois) assim como conversões binário-decimal-binário e hexa-decimal-hexa.

Estudaremos, no decorrer do curso, diversos exercícios como esses, tanto sobre bases de numeração e codificação como operações com diversos códigos.

As leituras preliminares sugeridas podem esclarecer alguma dúvidas e, certamente, tornar outras mais aparentes.

=====

Usamos marcações binárias singulares (de 1 bit) para indicar a presença ou ausência de certo item em uma lista

x	tem	
o	não	tem
o	não	tem
o	não	tem
x	tem	
x	tem	
o	não	tem
o	não	tem
x	tem	
x	tem	
o	não	tem
x	tem	

ou, com zeros e uns
100011001101 binário

ou, escrito da esquerda para a direita,

101100110001 binário

(dependendo da convenção que adotarmos)

É tão usual como "marcar" ('check/uncheck') certo item para selecioná-lo, "comentar" ('comment/uncomment') certa linha de código, de modo a que, comentada, a tal linha não seja executada em um script, por exemplo.

Um comentário, em um código executável é uma sequência de bytes (caracteres) que será excluída do texto/programa antes da execução. Afinal, trata-se de um "comentário".

Marcar como "comentário" alguma linha de código, tem, então, como efeito, "colateral" ou principal, a sua eliminação no processamento anterior à execução.

Também usamos diferentes agrupamentos de bits para indicar mais alternativas de representação.

Note que (segunda aula, m2_SI2025) agrupamos binários de 4 em 4 bits para obter uma correspondência entre desenhos binários e hexadecimais,

...

1001bin	9hexa
1010b	Ah
1011b	Bh
1100b	Ch
1101b	Dh
1110b	Eh
1111b	Fh

pois de 0000b a 1111b, assim como de 0h a Fh, temos dezesseis desenhos diferentes.

Pode ser conveniente abreviar o padrão tem-tem-tem-nãotem, ou 1110 como E hexa.

Também é muitas vezes conveniente agrupar binários de 3 em 3 bits para obter uma correspondência entre desenhos binários e octais.

000	0
001	1
010	2
011	3
100	4
101	5

```
110    7
111    8
```

Por exemplo, nas permissões de acesso aos recursos de um sistema, é importante saber que usuários podem ler, ou modificar, ou executar conteúdos de certos arquivos.

Há, grosso modo, três categorias de usuários de um recurso. O proprietário (user/owner), o grupo (group) associado ao recurso, e (other) os que não pertencem às categorias anteriores.

Ordenando essas categorias, da esquerda para a direita, como u,g,o, podemos representar permissões de acesso - r(ead),w(rite),(e)x(exute) - (a um arquivo, por exemplo) no formato

```
(u      g      o)
rwx    rwx    rwx
```

Por exemplo, supondo que

- .o usuário proprietário tem permissão para ler, escrever e executar o arquivo
- .o grupo do proprietário tem permissão para ler e executar, mas não para escrever no arquivo
- .os demais podem apenas executar o arquivo

representaríamos esta situação assim:

```
rwx rwx rwx
111 101 001
```

ou, simplesmente,

```
111101001
```

ou, ainda, como os bits estão agrupados de 3 em 3, usando octais,

```
751
```

Em outras palavras, uma permissão '751' de acesso a algum arquivo significa que o 'user' pode ler, escrever e executar; o grupo pode ler e executar; e os demais apenas ler o recurso.

=====

fim_de_m4_SI2025.txt