UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA DE ELETRICIDADE
LABORATÕRIO DE SISTEMAS DIGITAIS

MONTADOR DO ‘‘PATINHO FEIO’’

Antonio Marcos de Aguirra Massola
João José Neto
Moshe Bain

Julho
1977

Em memória de
Laís Costa Ortenzi

INDICE

Assunto | Página

1 - INTRODUÇÃO

O ante-projeto do minicomputador Patinho Feio nasceu de um curso de pós-graduação dado pelo Professor Glen George Langdon Jr., em 1972. A seguir, os engenheiros e estagiários do Laboratório de Sistemas Digitais (LSD) da EPUSP terminaram o projeto e montaram o Patinho Feio que, dessa forma, se tornou o primeiro computador projetado e construído no Brasil.

Os circuitos do Patinho Feio são totalmente constituídos por circuitos integrados da família TTL (“transistor - transistor logic”), apresentando uma memória de núcleos de fer-rite, e tendo um ciclo de máquina de dois microsegundos.

O Patinho Feio foi destinado a pesquisas no LSD, tan-to na área de programação (“software”) como dos circuitos eletrônicos (“hardware”).

Cuidou-se do desenvolvimento de um “software” que per-mitisse um uso mais eficiente do minicomputador, já que, de início só se podia programá-lo em linguagem de máquina, manual-mente, através do seu painel. Em particular, foi definida uma linguagem de montador (“assembly language”), que associa a cada instrução de máquina um mnemônico, e um programa montador (“assembler”), cuja função é traduzir programas escritos em linguagem de montador para linguagem de máquina, os quais são os assuntos tratados neste manual.

Este manual foi escrito de forma a tratar cada tópico de forma mais ou menos extensa, na suposição de que o leitor tenha tido previamente apenas um pequeno contato com a área de computação, e pouco ou nenhum conhecimento de lingua-gens de baixo nível, como um montador. Por causa disso,tentou-se fazer com que o manual fosse o mais auto-explicativo e inde-pendente possível de outros textos. Naturalmente é impossível que um texto seja completamente independente de outros; por isso, recomenda-se consultar outros textos, tais como manuais de operação do Patinho Feio e de seus equipamentos periféricos(de entrada/saída), textos sobre números binários, etc.

Foi feito um bom esforço para apresentar os conceitos com clareza e para padronizar as notações, com o objetivo de tornar o manual realmente útil. Contudo, certamente muitas falhas subsistem, de forma que são bem recebidas quaisquer sugestões e críticas de modo a melhorar o manual em futuras edições.

Observações:

a) As informações contidas neste manual são as melhores que se pôde obter na época em que o manual foi escrito (setembro de 1975). Contudo, devido ao constante desenvolvimento de novos projetos de "hardware" e "software" para o Patinho Feio, alguns detalhes podem ter sofrido alterações até a presente data.

b) Os programas e trechos de programa existentes no manual foram aí colocados por estarem sintaticamente corretos, mas não representam necessariamente exemplos de boa técnica de programação.

2 - ARITMÉTICA BINÃRÍA E HEXADECIMAL

(Com números inteiros)

1. Bases de Numeração

Utiliza-se, na vida diária, a base decimal de numeração para representar os números. Isto significa duas coisas:

a) existem dez algarismos com os quais todos os números são representados (pois a base de numeração é dez), a saber: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

b) emprega-se uma notação posicional onde está subentendido que, quando um algarismo é deslocado de uma posição para a esquerda, seu valor é multiplicado por dez. Por exemplo:
$295 = 2 \times 10^2 + 9 \times 10^1 + 5 \times 10^0$

Generalizando, quando se escreve o número $N = d_n\ d_{n-1} \ldots d_2\ d_1\ d_0$ (sem sinal), onde os d_i ($i = 0, 1, 2, \ldots, n$) são os seus algarismos (ou dígitos), está-se querendo dizer que: $N = d_n \times 10^n + d_{n-1} \times 10^{n-1} + \ldots + d_2 \times 10^2 + d_1 \times 10^1 + d_0 \times 10^0$.

Nada obriga a que se use apenas a base dez. Na verdade, qualquer base b(inteira) pode ser escolhida para representar um número. Para tanto, escolhem-se b símbolos distintos (os algarismos da base) que representam os números de zero a (b - 1). Escrevendo-se agora n + 1 algarismos adjacentes $d_n d_{n-1} \ldots d_1 d_0$ e subentendida a notação posicional descrita acima, tem-se o número N representado por essa notação:

$$N = d_n \cdot b^n + d_{n-1}\ b^{n-1} + \ldots + d_1\ b^1 + d_0\ b^0$$

Inversamente, pode-se provar que cada número N tem uma única representação, numa dada base b, que satisfaz as condições mencionadas acima.

Exemplo: Escolhendo b = 3, têm-se três algarismos;

convencionalmente usa-se 0, 1, 2. Então tem-se:

$$(1202)_3 = 1 \times 3^3 + 2 \times 3^2 + 0 \times 3^1 + 2 \times 3^0 = (47)_{10}$$

Pode-se começar a perceber a importância do que foi dito acima quando se considera que os computadores modermos trabalham sempre, em última análise, com a base dois.

2. Bases mais empregadas em computação

Além da base dez, que é de uso geral, empregam-se comumente as seguintes bases:

a) base dois (binária) - necessita dois algarismos distintos para representar os números zero e um. Por convenção utilizam-se os símbolos 0 e 1, Um algarismo binário é também chamado "bit" (do inglês "binary digit").

   A base dois é extremamente importante pois, como já foi citado, os computadores só entendem sequências de zeros e uns , que são usadas tanto para representar as instruções dadas à máquina quanto números prop iamente ditos.

b) base oito (octal) - utiliza os algarismos de 0 a 7. Não será aqui tratada com mais detalhes porque não é utilizada no Patinho Feio, embora o seja em vários outros computadores.

c) base dezesseis (hexadecimal) - os dígitos hexadecimais são: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F; usados para representar os números de zero a quinze.

   Exemplo: $(AB)_{16} = 10 \times 16^1 + 11 \times 16^0 = (171)_{10}$

A correspondência entre os valores binários, decimais e hexadecimais é apresentada na tabela seguinte (note-se que são necessários quatro bits para representar todos os dígitos hexadecimais na base dois).

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

3. Conversão de números entre as bases dois, dez e dezesseis

Conforme já se deve ter percebido, surge frequentemente a necessidade de converter números escritos em uma base psta outra. Para isso existem algoritmos gerais, dos quais são apresentados abaixo alguns casos particulares:

a) Conversão para a base dez de números escritos em outra base. Basta escrever o número na forma $d_n \cdot b^n + \ldots + d_0$ e efetuar as operações indicadas.

Exemplos: 1º) $(101111100001)_2$ para a base 10

$$= 1.2^{11} + 0.2^{10} + \ldots + 0.2^1 + 1 = (3041)_{10}$$

2º) $(BE1)_{16}$ para a base 10

$$= 11 \times 16^2 + 14 \times 16 + 1 = (3041)_{10}$$

Uma forma conveniente de fazer isso é dada nos diagramas abaixo:

$(BE1)_{16}$ para a base 10

$$
\begin{array}{l|lll}
 & 11 & 14 & 1 \\
\hline
\text{base original}=16 \xleftarrow{x} & 11 & 190 & (3041)_{10}
\end{array}
$$

Método usado: $11 \times 16 + 14 = 190$

$190 \times 16 + 1 = 3041$

$(10110)_2$ para a base 10

$$
\begin{array}{l|lllll}
 & 1 & 0 & 1 & 1 & 0 \\
\hline
\text{base original}= 2 & 1 & 2 & 5 & 11 & (22)_{10}
\end{array}
$$

Método usado: $1 \times 2 + 0 = 2$

$2 \times 2 + 1 = 5$

$5 \times 2 + 1 = 11$

$11 \times 2 + 0 = 22$

b) Conversão de números escritos na base dez para uma outra base. Divide-se repetidamente o número dado pela base de destino até que o quociente seja zero. Os restos obtidos são a representação desejada, em ordem invertida. Ver os esquemas abaixo:

$(3041)_{10}$ base 16

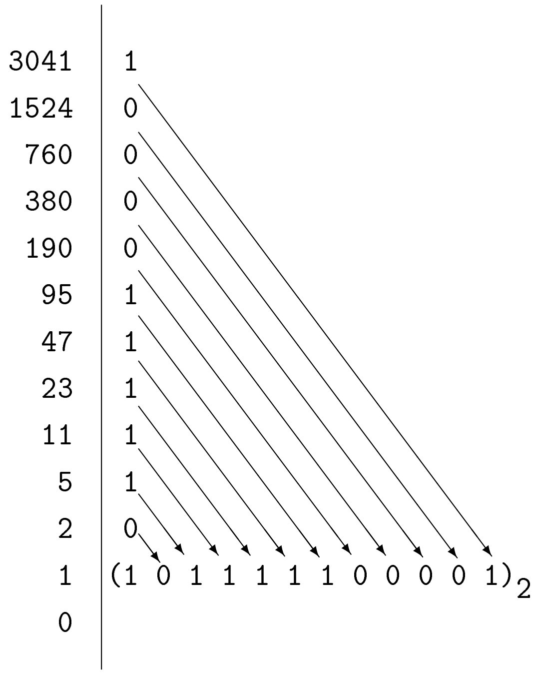
resto de 3041 : 16

resto de 190 : 16

resto de 11 : 16

$$
\begin{array}{rcr|l}
 & & 3041 & 1 \\
3041 : 16 & \longrightarrow & 190 & 14 \\
190 : 16 & \longrightarrow & 11 & 11 \quad 14 \quad 1 \\
11 : 16 & \longrightarrow & 0 & (BE1)_{16}
\end{array}
$$

$(3041)_{10}$ base 2

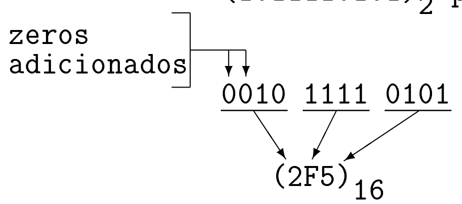


c) Conversão entre as bases dois e dezesseis.

c-1) Da base dois para a base dezesseis. Basta agrupar os dígitos binários de quatro em quatro (a partir da direita) e substituí-los pelo respectivo dígito hexadecimal, conforme a tabela apresentada mais atrás (item 2.c).

Exemplo:

$(1011110101)_2$ para base 16



c-2) Da base dezesseis para a base dois. Basta substituir cada dígito hexadecimal pelo seu código de quatro bits.

Exemplo:

$(2F5)_{16}$ para base 2

$(0010\ 1111\ 0101)_2$

Obs.: 1º) Para a base oito, como é fácil perceber, o método é inteiramente análogo, dividindo-se o número binário em grupos de três bits.

2º) Pode-se agora notar porque são tão usadas as bases oito e dezesseis em computação: elas permitem dividir por três (pois $8 = 2^3$) e por quatro (pois $16 = 2^4$), respectivamente, o comprimento em algarismos do número escrito na base dois, que costuma ser inconvenientemente longo.

3º) Está-se dando mais ênfase à base hexadecimal porque no Patinho Feio os números têm ou oito ou doze bits de comprimento, podendo então, ser representados com dois ou três dígitos hexadecimais, enquanto que, por exemplo, para transformar um número de oito bits (também chamado "byte") em um número octal, tem-se que adicionar um zero à frente do número, para dividí-lo em três grupos de três bits cada.

4. Soma de números binários positivos

Realiza-se de forma inteiramente análoga à soma somum, de números decimais. Para ver isso, examine-se detalhadamente uma soma decimal, por exemplo, de 1672 com 729.

```
   0   1   1   1

     1   6   7   2
   +   +   +   +        +
         7   2   9
 -----------------
 0   2   4   0   1
```

Começando a partir da direita, foram realizadas as seguintes operações:

```
    2 + 9 = 1  e  vai-um  (= 11)
1 + 7 + 2 = 0  e  vai-um  (= 10)
1 + 6 + 7 = 4  e  vai-um  (= 14)
    1 + 1 = 2  e  vai-zero(= 02)
        0 = 0
```

Com os números binários procede-se da mesma forma, segundo as seguintes regras:

```
0 + 0 + 0  = 00  ———→ 0  e  vai-um
0 + 0 + 1  = 01  ———→ 1  e  vai-zero
0 + 1 + 1  = 10  ———→ 0  e  vai-um
1 + 1 + 1  = 11  ———→ 1  e  vai-um
```

Exemplo:

1º) Seja somar 10111010 com 10011. Tem-se:

```
0   0   1   1   0   0   1   0
  1   0   1   1   1   0   1   0
                1   0   0   1   1   +
-------------------------------
0 1   1   0   0   1   1   0   1
```

Começa-se a partir da direita, realizando as seguintes operações:

```
        0 + 1   =  1  e  vai-zero
    0 + 1 + 1   =  0  e  vai-um
    1 + 0 + 0   =  1  e  vai-zero
    0 + 1 + 0   =  1  e  vai-zero
    0 + 1 + 1   =  0  e  vai-um
    1 + 1       =  0  e  vai-um
    1 + 0       =  1  e  vai-zero
    0 + 1       =  1  e  vai-zero
    0           =  0

coluna dos                 bits do segundo número
vai-um ou
vai-zero                   bits do primeiro número
```

2º) Somar 11101 com 110.

```
1    1   1   0   0
   1   1   1   0   1
           1   1   0  +
--------------------
1  0   0   0   1   1
```

5. <u>Representação de números negativos</u>

Obs.: Nos itens seguintes assume-se sempre que um número tem oito bits de comprimento, quando for binário.

Até agora, só foram tratados os números positivos. Contudo, é óbvia a necessidade de se manipular números negativos, de modo que é preciso uma representação adequada para os mesmos. Especialmente, é necessária essa representação para números binários, de modo que o computador possa reconhecer os números que sejam negativos como tais.

Existem três modos de representar números negativos em notação binária, chamados de: sinal e amplitude, complemento de um e complemento de dois.

a) Representação de sinal e amplitude.

Usualmente, quando se quer denotar um número como negativo (em qualquer base), coloca-se à sua frente um sinal de menos (-), e quando positivo,às vezes, o sinal de mais (+). Mas,como um computador não reconhece os sinais + e -, mas apenas zeros e uns, vê-se que é necessário reservar um bit do número (geralmente o primeiro)para indicar o seu sinal (usa-se zero para indicar um número positivo e um para indicar um negativo). Supondo um número de oito bits, tem-se, por exemplo:

$(+6)_{10}$ 0000 0110

$(-6)_{10}$ 1000 0100

sinal | amplitude

Desta forma pode-se representar os números inteiros de -127 a +127. Note-se que existem duas representações do número zero, a saber: 0000 0000 e 1000 0000.

b) Representação em complemento de um.

Nesta representação, para indicar um número negativo troca-se os seus zeros por uns e vice-versa. Como sempre, o primeiro bit indicará o sinal do número. Exemplo:

$(+6)_{10}$ 0000 0110

$(-6)_{10}$ 1111 1001

sinal

Deste modo, analogamente ao anterior, pode-se representar os números de -127 a +127 e o zero continua com duas representações, a saber: 0000 0000 e 1111 1111.

c) Representação em complemento de dois.
Para se obter a representação em complemento de dois, soma-se um (em binário) à representação em complemento de um, retendo-se apenas os oito bits mais à direita. Exemplo:

```
(+6)₁₀= 0000 0110 ──── ida ────→ 1111 1001
          ↑                              1 +
          │  ─────────  complemento  ──────────
        sinal              de um
                    1 +                1111 1010  = (-6)₁₀ em com-
          0000 0101 ←── volta ──────   ↑              plemento de
                                      sinal           dois.
```

A representação em complemento de dois tem as seguintes propriedades:

1a.) O primeiro bit do número indica o seu sinal: positivo se zero e negativo se um.

2a.) São representáveis os números de -128, cuja representação é 1000 0000; a +127, cuja representação é 0111 1111. Desta forma, o número -128 não tem complemento de dois. De fato, tem-se:

```
-128 → 1000 0000 → 0111 1111
                           1 +
                   ─────────
                   1000 0000 → +128?
```

que está errado, pois é a representação de -128, não de +128.

3a.) O número zero tem apenas uma representação: 0000 0000. De fato, tem-se:

```
            0000 0000 → 1111 1111
                                1 +
                        ──────────
[desprezado → 1]0000 0000 ──→ 0000 0000
```

6. Aritmética no Patinho Feio

Já foi visto como somar números binários positivos e como representar números negativos em oito bits. Deste modo, pode-se passar à soma (e subtração) de números de oito bits, que é o que o Patinho Feio consegue fazer diretamente através de seus circuitos eletrônicos. Ver-se-á também, como operar com números de mais de oito bits e como reconhecer quando o resultado de uma soma não pode ser representado em oito bits (isto é, o número é menor que -128 ou maior que +127).

a) Vai-um:

Denomina-se vai-um de uma soma entre dois numeros de oito bits ao vai-um na última soma realizada (bit mais significativo), ou seja, ao que seria o nono bit da soma (se fossem considerados números de nove bits). Exemplo: