ILP500 – Laboratório de Arquitetura e Organização de Computadores
Prof. Sérgio Luiz Banin

Registro da Aula

Aritmética com Números Binários

Abordaremos este tópico com um foco computacional, o que significa que vamos trabalhar sempre com no máximo 2 operandos.

Adição

	128	64	32	16	8	4	2	1
Vai (pode ser 0 ou 1)	0	1	1	1	1	1	0	
Operando 1 (valor 54)	0	0	1	1	0	1	1	0
Operando 2 (valor 31)	0	0	0	1	1	1	1	1
Resultado (valor 85)	0	1	0	1	0	1	0	1

Coluna mais à direita (dígito menos significativo):

- operação de adição com 2 bits de entrada
- serão produzidas duas saídas: o bit resultante na coluna e o bit que vai para a coluna à esquerda
- são 4 possibilidades: 0 + 0 ou 0 + 1 ou 1 + 0 ou 1 + 1

0 0 --- 00	0 1 --- 01	1 0 --- 01	1 1 --- 10

Em vermelho temos o resultado (bit 0 ou 1) que ficará na própria coluna
Em pink temos o resultado (bit 0 ou 1) que será transferido para a coluna à esquerda

Todas as demais Colunas:

- operação de adição com 3 bits de entrada
- serão produzidas duas saídas: o bit resultante na coluna e o bit que vai para a coluna à esquerda
- são 8 possibilidades:
  0 + 0 + 0 ou 0 + 0 + 1 ou 0 + 1 + 0 ou 0 + 1 + 1 ou 1 + 0 + 0 ou 1 + 0 + 1 ou 1 + 1 + 0 ou 1 + 1 + 1

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

Em verde temos o bit originário da operação feita na coluna imediatamente à direita
Em vermelho temos o resultado (bit 0 ou 1) que ficará na própria coluna
Em pink temos o resultado (bit 0 ou 1) que será transferido para a coluna à esquerda

Exercícios

	128	64	32	16	8	4	2	1
Vai (pode ser 0 ou 1)	0	0	0	1	1	1	1	
Operando 1 (valor 43)	0	0	1	0	1	0	1	1
Operando 2 (valor 15)	0	0	0	0	1	1	1	1
Resultado (valor 58)	0	0	1	1	1	0	1	0

	Flag Vai1	128	64	32	16	8	4	2	1
Vai (pode ser 0 ou 1)		1	1	1	1	1	0	0	
Operando 1 (valor 189)		1	0	1	1	1	1	0	1
Operando 2 (valor 78)		0	1	0	0	1	1	1	0
Resultado (valor 267)	X	0	0	0	0	1	0	1	1

	Flag Vai1	128	64	32	16	8	4	2	1
Vai (pode ser 0 ou 1)		1	1	1	1	0	0	0	
Operando 1 (valor 200)		1	1	0	0	1	0	0	0
Operando 2 (valor 56)		0	0	1	1	1	0	0	0
Resultado (valor 256)	X	0	0	0	0	0	0	0	0

Subtração
Desejamos ter condição de executar no processador a operação: R = 17 – 3
Na prática os processadores executam: R = 17 + (-3)

A diferença entre as duas situações apontadas acima é que na primeira indica-se uma subtração direta, enquanto que na segunda indica-se a "soma do valor negativo".

O que é um número inteiro negativo em um sistema computacional?
Por exemplo: sabemos que 3 + (-3) resulta em zero. Então temos que encontrar uma forma binária para o -3 de modo que quando adicionado à forma do binária do 3 o resultado seja zero.

Neste nosso tópico vamos adotar a seguinte nomenclatura
Diremos -3 é o complemento de 3. E o mesmo vale para suas representações binárias.

Vamos agora para as formas binárias (vou usar 4 bits nos próximos exemplos)
$(3)_{10} = (0011)_2$

Agora vamos produzir o complemento de $(0011)_2$ e faremos aplicando a regra conhecida como "**Complemento de 2**". É um processo de duas etapas.

Valor original = 3	**0011**
1. Inversão dos bits do valor original	**1100**
2. Somar 1 aos bits invertidos	**1100** **0001+** **-----** **1101**
Este é o que vamos convencionar como sendo o -3	**1101**

Por consequência da aplicação do método "Complemento de 2" temos que

$$\text{se } (3)_{10} = (0011)_2 \text{ então } (-3)_{10} = (1101)_2$$

Prova que o método funciona: se fizermos a adição das duas formas binária deve resultar o valor zero.

$$(0011)_2 + (1101)_2 = (0000)_2$$

Vamos executar essa soma

	Flag Vai1	8	4	2	1
Vai (pode ser 0 ou 1)		1	1	1	
Operando 1 (valor 3)		0	0	1	1
Operando 2 (valor -3)		1	1	0	1
Resultado (valor 0)	X	0	0	0	0

Agora que já vimos como produzir um negativo utilizando o método "Complemento de 2", vamos passar a trabalhar com 8 bits

Valor original = 3	00000011
1. Inversão dos bits do valor original	11111100
2. Somar 1 aos bits invertidos	11111100 00000001+ -------- 11111101
Este é o que vamos convencionar como sendo o -3	11111101

Vamos fazer a adição 3 + (-3) agora com 8 bits

	Flag Vai1	128	64	32	16	8	4	2	1
Vai (pode ser 0 ou 1)		1	1	1	1	1	1	1	
Operando 1 (valor 3)		0	0	0	0	0	0	1	1
Operando 2 (valor -3)		1	1	1	1	1	1	0	1
Resultado (valor 0)	X	0	0	0	0	0	0	0	0

Vamos fazer a adição 7 + (-3) com 8 bits

	Flag Vai1	128	64	32	16	8	4	2	1
Vai (pode ser 0 ou 1)		1	1	1	1	1	1	1	
Operando 1 (valor 7)		0	0	0	0	0	1	1	1
Operando 2 (valor -3)		1	1	1	1	1	1	0	1
Resultado (valor 4)	X	0	0	0	0	0	1	0	0

Valores para usar no programa em C que, para o mesmo conjunto de 8 bits (tipo char), exibe o valor positivo, o negativo e o caractere correspondente – programa "positivoOUnegativo.cpp"

	Flag Vai1	128	64	32	16	8	4	2	1
valor 245 ou -11		1	1	1	1	0	1	0	1
Complemento de 2									
Inverte		0	0	0	0	1	0	1	0
Soma 1		0	0	0	0	0	0	0	1
		0	0	0	0	1	0	1	1

Link para a Tabela ASCII usada na aula https://web.fe.up.pt/~ee96100/projecto/Tabela%20ascii.htm

Deslocamentos de bits

Um deslocamento de dígitos numéricos representa uma multiplicação ou uma divisão pelo valor da base adotada no sistema de numeração. Essa operação se aplica a qualquer base.

Exemplo na base 10: valor escolhido = 12450

Valor original	0	0	0	1	2	4	5	0
Deslocamento à esquerda 1 coluna (multiplicação por 10)	0	0	1	2	4	5	0	0

Valor original	0	0	0	1	2	4	5	0

Deslocamento à esquerda 2 colunas (multiplicação por 100)	0	1	2	4	5	0	0	0

Valor original	0	0	0	1	2	4	5	0
Deslocamento à esquerda 5 colunas (multiplicação por 100000)	4	5	0	0	0	0	0	0

Valor original	0	0	0	1	2	4	5	0
Deslocamento à direita 1 coluna (divisão por 10)	0	0	0	0	1	2	4	5

Valor original	0	0	0	1	2	4	5	0
Deslocamento à direita 2 colunas (divisão por 100)	0	0	0	0	0	1	2	4

Acima exemplificamos o deslocamento de dígitos decimais para a esquerda e para a direita de modo a executar operações de multiplicação e divisão, respectivamente, pela base numérica adotada.

Vamos agora ver o mesmo para base binária

		128	64	32	16	8	4	2	1
Valor original	38	0	0	1	0	0	1	1	0
Deslocamento à esquerda 1 coluna (multiplicação por 2)	76	0	1	0	0	1	1	0	0

		128	64	32	16	8	4	2	1
Valor original	38	0	0	1	0	0	1	1	0
Deslocamento à esquerda 2 colunas (multiplicação por 4)	152	1	0	0	1	1	0	0	0

		128	64	32	16	8	4	2	1
Valor original	38	0	0	1	0	0	1	1	0
Deslocamento à direita 1 coluna (divisão por 2)	19	0	0	0	1	0	0	1	1

		128	64	32	16	8	4	2	1
Valor original	38	0	0	1	0	0	1	1	0
Deslocamento à direita 2 colunas (divisão por 4)	9	0	0	0	0	1	0	0	1

Resumindo: deslocar à esquerda implica em multiplicar por 2 e à direta implica em dividir por 2

	128	64	32	16	8	4	2	1
3	0	0	0	0	0	0	1	1
6	0	0	0	0	0	1	1	0
12	0	0	0	0	1	1	0	0
24	0	0	0	1	1	0	0	0
48	0	0	1	1	0	0	0	0
96	0	1	1	0	0	0	0	0
192	1	1	0	0	0	0	0	0

Multiplicação

1ª Etapa: Para cada bit 1 do Operando 2 vamos copiar e deslocar (como é multiplicação o deslocamento será sempre para a esquerda) os bits do Operando 1
2ª Etapa: somamos as linhas copiadas e deslocadas (as linhas coloridas)

17 x 3 = 51

		128	64	32	16	8	4	2	1
Operando 1	17	0	0	0	1	0	0	0	1
Operando 2	3	0	0	0	0	0	0	1	1
Cópia de Op.1 sem deslocar		0	0	0	1	0	0	0	1
Cópia de Op.1 deslocando 1 bit		0	0	1	0	0	0	1	0
	51	0	0	1	1	0	0	1	1

21 x 11 = 231

		128	64	32	16	8	4	2	1
Operando 1	21	0	0	0	1	0	1	0	1
Operando 2	11	0	0	0	0	1	0	1	1
Cópia de Op.1 sem deslocar	21	0	0	0	1	0	1	0	1
Cópia de Op.1 deslocando 1 bit	42	0	0	1	0	1	0	1	0
Vai 1			1	1	1				
Resultado parcial		0	0	1	1	1	1	1	1
Cópia de Op.1 deslocando 3 bits	168	1	0	1	0	1	0	0	0
	231	1	1	1	0	0	1	1	1

Na prática o que foi feito é: 21 x 8 + 21 x 2 + 21 x 1 = 21 x (8 + 2 + 1) = 21 x 11 = 231

23 x 12 = 276

		128	64	32	16	8	4	2	1
Operando 1	23	0	0	0	1	0	1	1	1
Operando 2	12	0	0	0	0	1	1	0	0
	1	1	1	1	1				
Cópia de Op.1 deslocando 2 bits		0	1	0	1	1	1	0	0
Cópia de Op.1 deslocando 3 bits		1	0	1	1	1	0	0	0
Resultado final		0	0	0	1	0	1	0	0

Neste exemplo ocorre Vai1 no dígito mais significativo. O flag Carry (Vai1) do processador será ligado, mas em termos práticos o valor 256 correspondente à 9ª coluna é perdido pois estamos trabalhando apenas com 8 bits.

12 x 15 = 180

		128	64	32	16	8	4	2	1
Operando 1	12	0	0	0	0	1	1	0	0
Operando 2	15	0	0	0	0	1	1	1	1
Vai1		1	1	1	1				
Cópia de Op.1 sem deslocar	12	0	0	0	0	1	1	0	0
Cópia de Op.1 deslocando 1 bit	24	0	0	0	1	1	0	0	0
Cópia de Op.1 deslocando 2 bits	48	0	0	1	1	0	0	0	0
Cópia de Op.1 deslocando 3 bits	96	0	1	1	0	0	0	0	0
	180	1	0	1	1	0	1	0	0

Divisão

Queremos calcular: 17 / 3
E estamos interessados em executar divisão de inteiros produzindo como resultado o Quociente e o Resto.

É um processo de subtrações sucessivas
17 – 3 = 14 → Q = 1

14 – 3 = 11 → Q = Q + 1 = 2
11 – 3 = 8 → Q = Q + 1 = 3
8 – 3 = 5 → Q = Q + 1 = 4
5 – 3 = 2 → Q = Q + 1 = 5

Ordem de grandeza de tempo de execução de operações aritméticas
Soma 2 ciclos
Subtração 5 ciclos
Multiplicação 12 ciclos
Divisão 110 ciclos
Essas proporções são antigas (por volta das décadas 1970/1980). Hoje em dia com o padrão IEEE-754 e os coprocessadores matemáticos, as operações de divisão se tornaram muito mais rápidas.