Unidade Lógica Aritmética (Arithmetic Logic Units)

A maioria dos computadores contem um único circuito para a realização das operações AND, OR e soma de duas palavras de máquina. Tipicamente, tal circuito de palavras de *n*-bit é construído por n circuitos idênticos para as posições de bits individuais. A figura 3-19 é um exemplo simples de tal circuito, chamado de Unidade Lógica Aritmética ou ULA (ALU - Arithmetic Logic Unit). Ela pode calcular qualquer uma das quatro funções listadas a seguir: **A** AND **B**, **A** OR **B**, $\overline{\mathbf{B}}$, ou **A** + **B**, dependendo se as linhas de entrada de função **F0** e **F1** contiverem os valores 00, 01, 10, ou 11 (binário). Note-se que aqui **A** + **B** significa a soma aritmética de **A** e **B**, não o booleano OU.

O canto inferior esquerdo da nossa ALU contém um decodificador de 2-bits para gerar sinais de ativação para as quatro operações, com base nos sinais de controle **F0** e **F1**. Dependendo dos valores de **F0** e **F1** exatamente uma das quatro linhas de ativação é selecionada. A definição dessa linha permite que a saída para a função selecionada seja passada para a porta OR final.

O canto superior do lado esquerdo tem a lógica para calcular **A** AND **B**, **A** OR **B**, e $\overline{\mathbf{B}}$, mas apenas um destes resultados é passado para a porta final OU, dependendo das linhas habilitadas que saem do decodificador. Em razão de apenas uma das saídas do decodificador ser 1, apenas uma das quatro portas AND habilitará a porta OR final; as outras três terão saída 0, independente de A e B.

Além de ser capaz de usar **A** e **B** como entradas para operações lógicas ou aritméticas, é também possível forçar cada uma delas a 0, negando ENA ou ENB, respectivamente. Também é possível obter $\overline{\mathbf{A}}$, definindo **INVA**. Veremos usos para **INVA**, **ENA** e **ENB** no Cap. 4. Sob condições normais, **ENA** e **ENB** são ambos 1 para habilitar ambas as entradas (**A**, **B**) e **INVA** é 0. Neste caso, os valores de **A** e **B** são apenas alimentados para dentro da unidade lógica não sendo modificados.

O canto inferior do lado direito da ALU contém um somador completo para calcular a soma de **A** e **B**, incluindo a manipulação dos vai-1 (Carry-Out), pois é provável que vários destes circuitos, eventualmente, serão ligados em conjunto para executar operações de palavra inteira. Circuitos como a da FIG. 3-19 estão realmente disponíveis e são conhecidos como *bits slices*. Eles permitem que o projetista de computadores construa uma ALU de qualquer largura desejada. A figura 3-20 mostra um ALU de 8-bit construída por oito 1-bit-slices de ALU. O sinal INC é útil apenas para operações de adição. Quando presente, ele incrementa (isto é, adiciona 1 a)o resultado, tornando-se possível calcular somas como **A** + **1** e **A** + **B** + **1**.

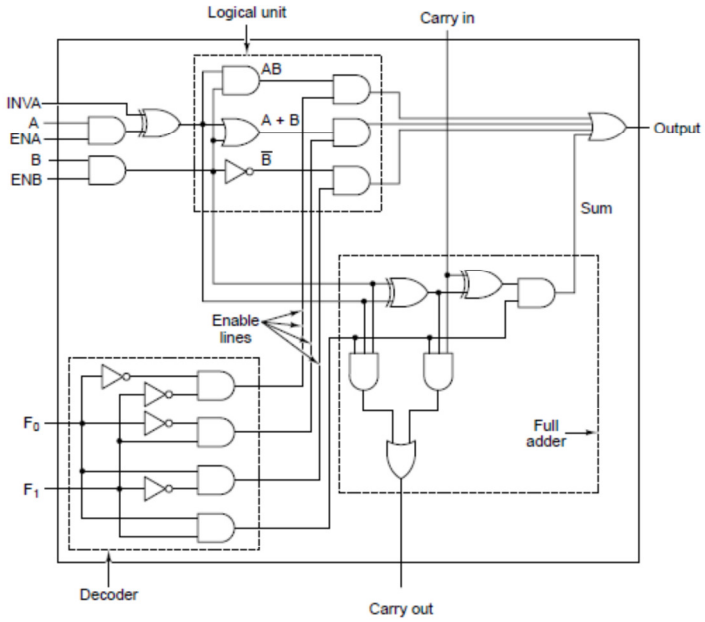

Figure 3-19. A 1-bit ALU.

F0	F1	ENA	ENB	INVA	Carry in	Função
0	1	1	0	0	0	*A*
0	1	0	1	0	0	*B*
0	1	1	0	1	0	*A (inverso)*
1	0	1	1	0	0	*B (inverso)*
1	1	1	1	0	0	*A+B*
1	1	1	1	0	1	*A+B+1*
1	1	1	0	0	1	*A+1*
1	1	0	1	0	1	*B+1*
0	0	1	1	0	0	*A AND B*
0	1	1	1	0	0	*A OR B*
0	1	0	0	0	0	*0*
0	1	0	0	1	0	*1*

Combinações de sinais da ALU e função executada

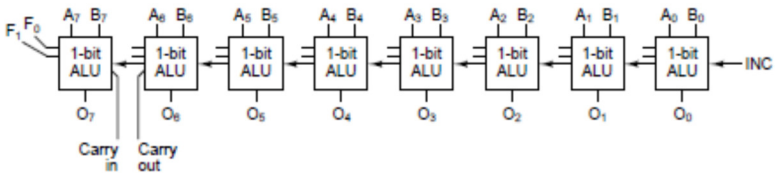

Figure 3-20. Eight 1-bit ALU slices connected to make an 8-bit ALU. The enables and invert signals are not shown for simplicity.

Structured Computer Organization, Fifth Edition, Andrew S. Tanenbaum