**Exercício Prático 2 Parte 1**

# Laboratório de ac2

**Objetivo:**

**Cosntruir uma Unidade Lógica e Aritmética (ULA) de 1 bit, 4 bits e implementar no Logisim e Arduino.**

**Parte 1 (estudo da ALU usando Logisim):**

1. Considere a Unidade Lógica e Aritmética de 1 bit ilustrada na Figura 1 a seguir:



*Figura 1.: Unidade Lógica e Aritmética*

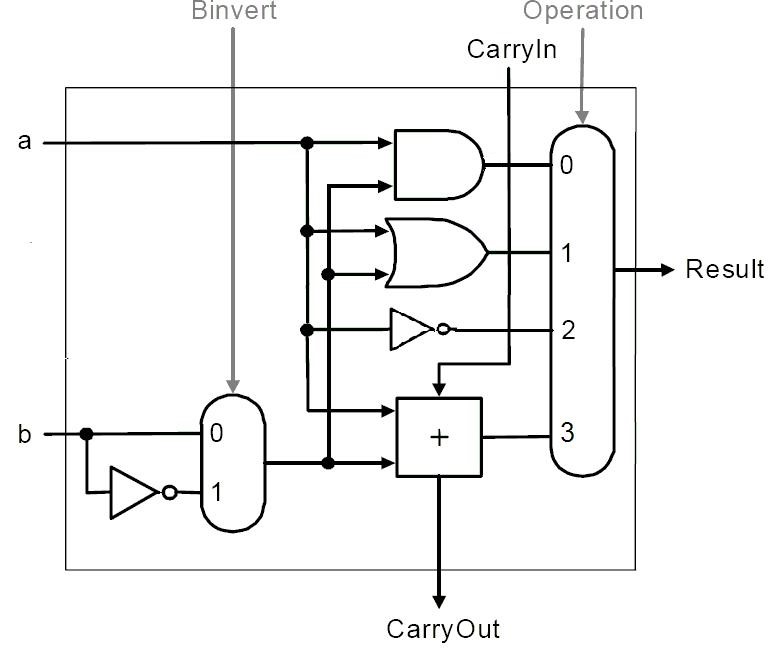


Figura 1

1. Procure entender o esquema, principalmente a subtração.
2. Sua ULA possui a seguinte tabela de opcodes:

|  |  |
| --- | --- |
| **Op. Code (Operation)** | **Instrução (Result)** |
| 0 | AND (a,b) |
| 1 | OR (a,b) |
| 2 | NOT (a) |
| 3 | SOMA(a,b) |

1. Teste a sua ULA de acordo com o seguinte roteiro:

***Inicio:***

A=0;

B=1;

AND(A,B);

A=1;

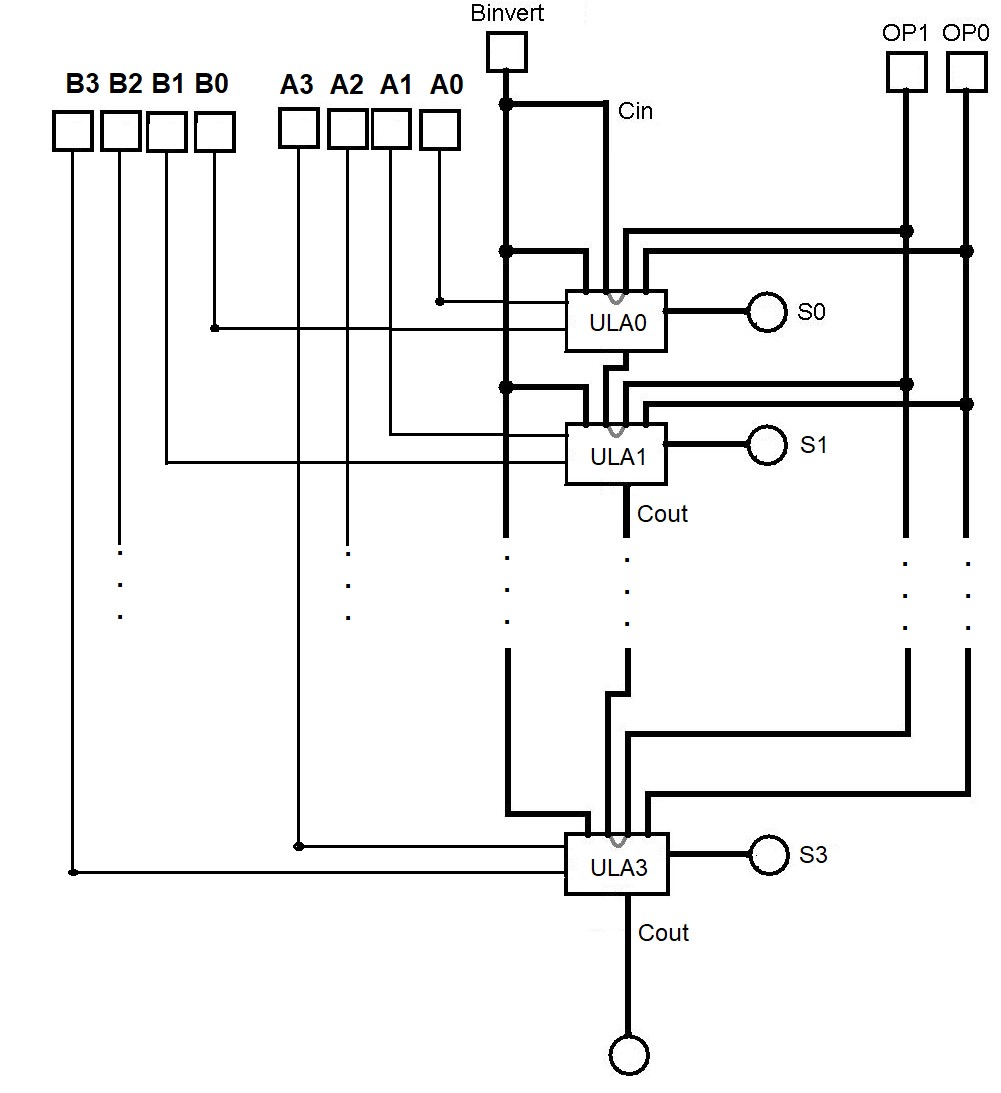
B=1;

OR(A,B);

SOMA(A,B);

NOT(A); SOMA (A,-B); ***Fim.***

1. Usando essa ula de 1 bit, construa essa ULA para **4 bits** no Logisim e verifique o seu funcionamento. Veja como funciona o barramento de instruções (operation) e o barramento de dados (a e b). Observe a ligação do Binvert ao Carry\_in da primeira ULA. Procure usar subcircuitos, seu circuito deverá estar como a figura a seguir:



1. Teste a sua ULA de acordo com o seguinte roteiro (considerando os números de 4 bits):

***Inicio:***

A=2; ( ou A=0010)

B=1; ( ou B=0001)

AND(A,B);

B=3; ( ou B=0011)

OR(A,B);

SOMA(A,B);

A=12; ( ou A=1100)

NOT(A);

B=13; ( ou B=1101) AND(B,A); ***Fim.***

Para o programa de teste acima, preencher a tabela a seguir considerando que cada linha corresponderá à execução de uma instrução (a primeira linha já foi realizada, observe que a palavra deverá conter 10 bits, para escrevermos em hexa completamos os dois bits à esquerda com zero):

|  |  |  |  |
| --- | --- | --- | --- |
| Instrução realizada | Binário (A,B,Op.code) | Valor em Hexa (0x …) | Resultado em binário |
| **AND(A,B)** | 0010 0001 00 | (0000 1000 0100) = 0x084 | 0000 |
| **OR(A,B)** | 0010 0011 01 | (0000 1000 1101) = 0x08d | 0011 |
| **SOMA(A,B)** | 0010 0011 11 | (0000 1000 1111) = 0x08f | 0101 |
| **NOT(A)** | 1100 0011 10 | (0011 0000 1110) = 0x30e | 0011 |
| **AND(B,A)** | 1100 1101 00 | (0011 0011 0100) = 0x334 | 1100 |

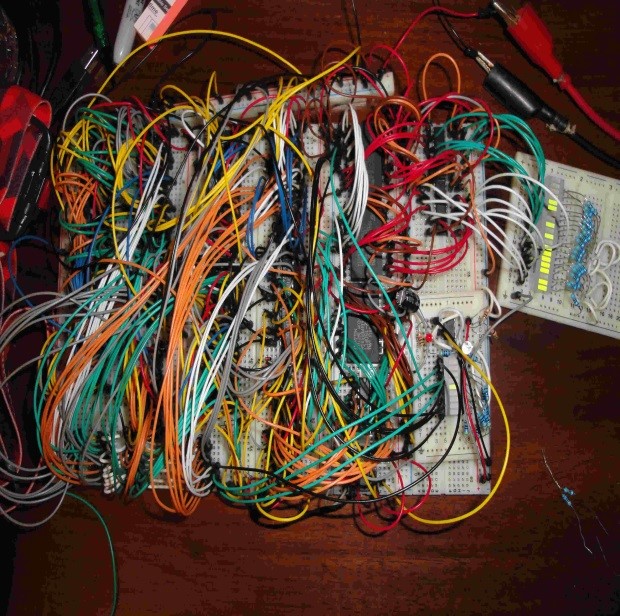
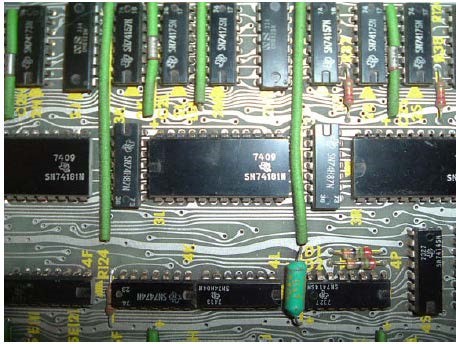
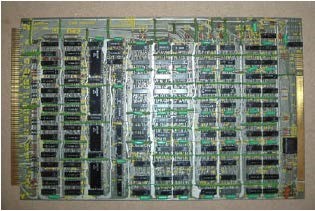
**O que apresentar nesse exercício:**

**Parte 1 (Logisim)**

* **Os prints de todos os circuitos elaborados (ULA de 1 bit e ULA de 4 bits).**
* **Preencher a tabela de resultados para a unidade de 4 bits.**
* **Um print de cada um dos testes realizados (use o roteiro indicado no item 6). Não é necessário dar prints quando no programa de teste apenas uma atribuição de valores às variáveis for realizada.**

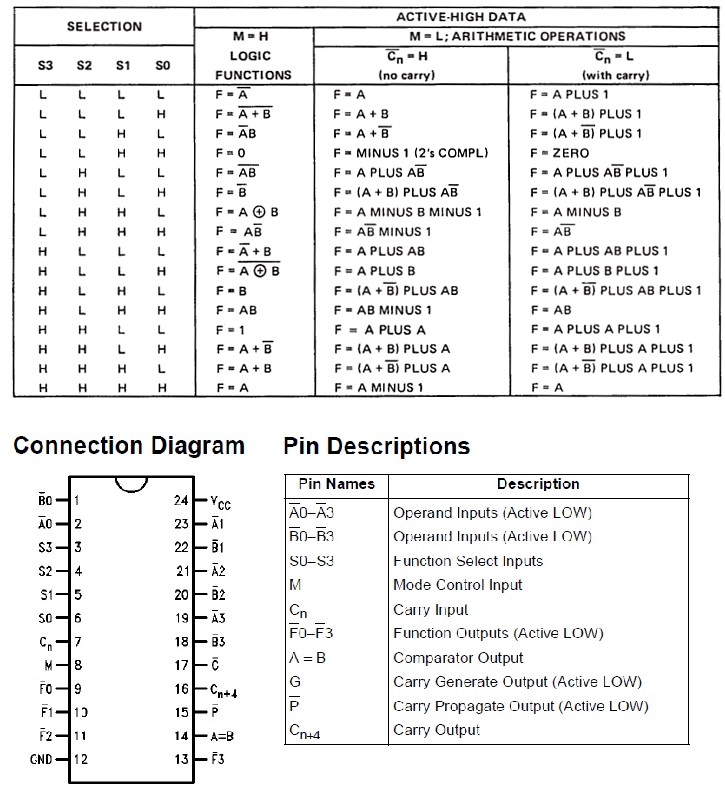
# Parte 2

Nesta experiência você irá projetar no logisim o circuito 74181, que foi inicialmente utilizado para a construção de computadores de 8 e 16 bits (conforme as figuras abaixo). Posteriormente iremos implementar uma ULA semelhante dentro do Arduino, por isso é importante conhecê-la.



**Como a ULA funciona.**

A ULA a ser utilizada é a 74LS181, que possui 4 bits de controle e é uma ULA de 4 bits (saída). Portanto, opera sobre duas entradas de 4 bits. A distribuição dos pinos pode ser vista a seguir:



Nessa primeira parte do experimento você deverá implementar toda a ULA no Logisim. Esta ULA permite a execução de instruções lógicas e aritméticas e permite que usemos entradas ativas em nível alto e em nível baixo. **Atenção:** Usaremos entradas em **nível alto**, conforme a tabela de funções ilustrada e iremos utilizar **apenas as instruções lógicas.**

O próximo passo será um projeto no LOGISIM dessa mesma ULA.

Agora você poderá utilizar os componentes presentes no Logisim (MUX, somadores, portas de múltiplas entradas, etc). Você também deverá utilizar o conceito de barramento para cada entrada e/ou saída, isso evitará um número muito grande de conexões.

Para um teste, você deverá dar um valor para A, um valor para B e executar todas as funções que a ALU permite através de S0, S1, S2 e S3. A saída da ALU deverá ser verificada nos pinos F0, F1, F2 e F3.

Iremos testar todas as funções lógicas da ULA da seguinte forma:

- criaremos uma palavra de 12 bits (os primeiros 4 bits para A0, A1, A2 e A3), os próximos 4 bits para B ( B0, B1, B2 e B3) e os 4 bits finais para a operação desejada (S0, S2 e S3). O valor a ser preenchido da tabela será o resultado da operação. Exemplo:

|  |  |  |
| --- | --- | --- |
| Instrução | Binário | Resultado da operação |
| 4CB | 010011001011 | 4 |

O significado da instrução é o seguinte (observe que escrevemos os valores em Hexadecimal para simplificar):

* O valor de A = 4 ( ou **0100** em binário)
* O valor de B = C (ou **1100** em binário)
* A operação será B (ou **1011** em binário)

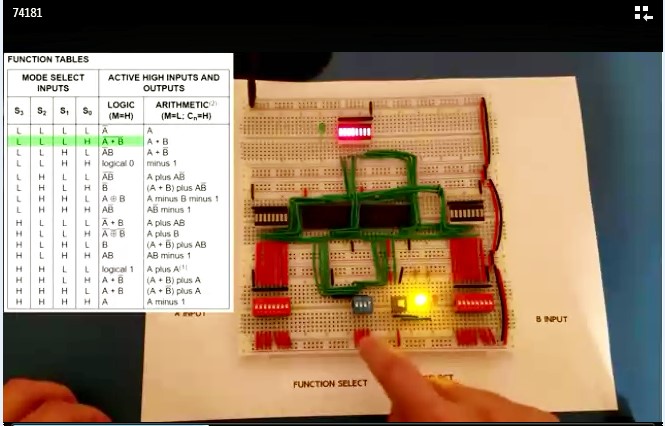
O que deveremos então fazer será a operação **1011** sobre os dados **0100** (que é o valor de A ou o primeiro operando) sobre **1100** (que é o valor de B ou o segundo operando).

Quando olhamos na tabela da ULA, a operação 1011 (ou H L H H) corresponde a F = AB, ou seja, a saída da ULA será o AND de A com B. Como A=0100 e B=1100, o AND de A e B será 0100, que é o resultado da operação e que deverá ser colocado na tabela (0100 = 4).

Complete agora a tabela a seguir onde todas as instruções que a ULA pode fazer serão testadas.

|  |  |  |
| --- | --- | --- |
| Instruções | Binário | Resultado da operação |
| 450 | 0100 0101 0000 | 1010 |
| CB1 | 1100 1011 0001 | 0111 |
| A32 | 1010 0011 0010 | 0001 |
| C43 | 1100 0100 0011 | 0000 |
| 124 | 0001 0010 0100 | 1110 |
| 785 | 0111 1000 0101 | 1000 |
| 9B6 | 1001 1011 0110 | 0010 |
| CD7 | 1100 1101 0111 | 0000 |
| FE8 | 1111 1110 1000 | 0001 |
| 649 | 0110 0100 1001 | 0010 |
| D9A | 1101 1001 1010 | 1001 |
| FCB | 1111 1100 1011 | 1100 |
| 63C | 0110 0011 1100 | 1111 |
| 98D | 1001 1000 1101 | 1111 |
| 76E | 0111 0110 1110 | 0111 |
| 23F | 0010 0011 1111 | 0010 |

A figura a seguir ilustra, apenas como exemplo, uma montagem real de uma ULA de 8 bits utilizando 2 circuitos 74181 e a implementação de uma função lógica (A+B)’ , cujo opcode é LLLH ou 0001.



**O que apresentar nesse relatório:**

1. A tabela de teste com as funções da ULA completamente preenchida.
2. O projeto da ULA no Logisim com um printscreen de alguma instrução da tabela sendo executada. Responder:

- Se o objetivo fosse realmente testar esta ULA, quantas linhas a nossa tabela verdade deveria ter, ou seja na verdade a tabela que você preencheu deveria ter quantas linhas?

2 elevado a 12 (4096)