PUC Minas

Arquitetura de Computadores II

Relatório III

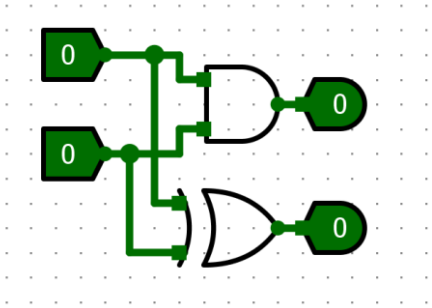
Aluno: Hyalen Neves Caldeira (Turno Manhã - 9292.1.01) e Vinícius Francisco da Silva (Turno Tarde - 9781.1.01)*

Matrícula 576920

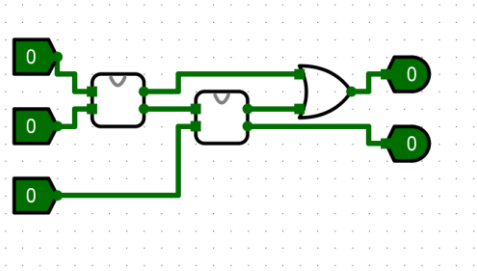
Disciplina: Arquitetura de Computadores II

Professor: Romanelli

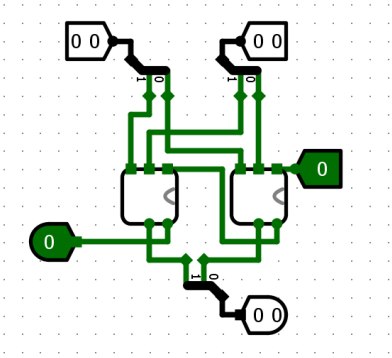
- Construção dos circuitos através do Logisim.



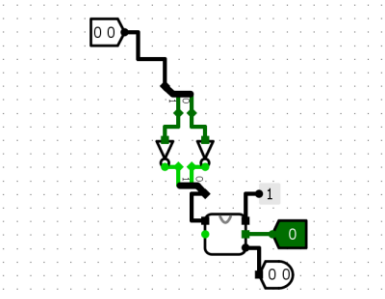
A imagem mostra o primeiro circuito construído, que é o da Meia Soma.



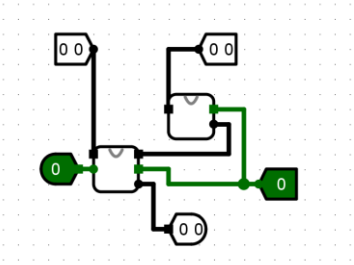
Utilizando dois circuitos de meia soma construímos o circuito de soma completa.



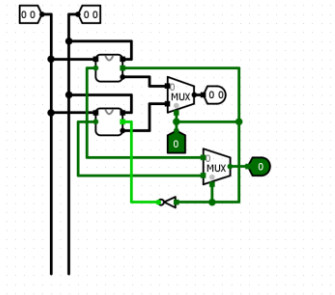
Utilizando dois circuitos completos fazemos a soma para 4 bits, recebendo um carry-in e trazendo de saída um carry-out.



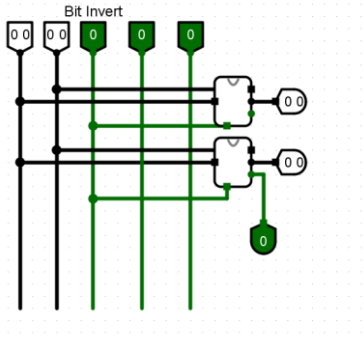
Nesse circuitos fazemos um complemento de 2 para utilizarmos no subtrador.



Este é o circuito do subtrador, que é realizado através da soma de uma entrada A com o complemento da entrada B.



Aqui temos uma ULA que através das entradas op0 e op1 realizam ou uma soma de A com B ou uma subtração.



E aqui temos uma Unidade Lógica e Aritmética completa.

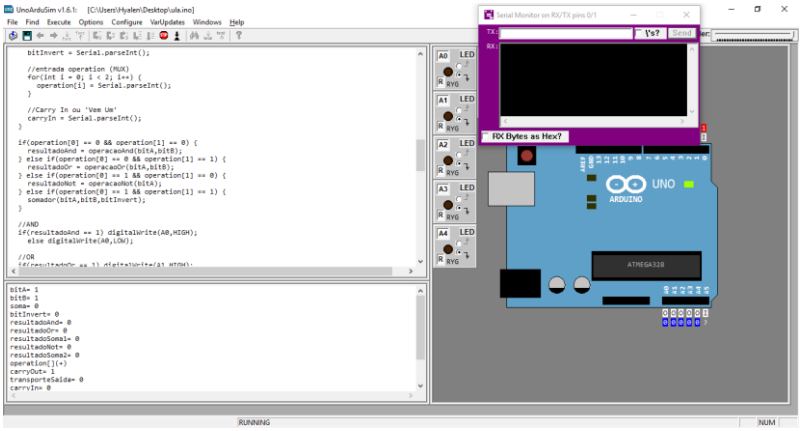
```
int bitA;
int bitB;
int soma;
int bitInvert = 0;
int resultadoAnd = 0;
int resultadoOr = 0;
int resultadoSoma1 = 0;
int resultadoNot = 0;
int resultadoSoma2 = 0;
int operation[2] = {0,0};
int carryOut = 0;
int transporteSaida = 0;
int carryIn;
int subtracao;

void setup(){
    Serial.begin(9600);
    //AND(A,B)
    pinMode(A0,OUTPUT);
    //OR(A,B)
    pinMode(A1,OUTPUT);
    //SOMA(A,B)
    pinMode(A2,OUTPUT);
    //NOT(A)
    pinMode(A3, OUTPUT);
    //SOMA(A,-B);
    pinMode(A4, OUTPUT);
}

int operacaoAnd(int bitA, int bitB){
    return bitA & bitB;
}

int operacaoOr(int bitA, int bitB){
    return bitA | bitB;
}

int orTransporteSaida(int bitA, int bitB, int bitC) {
    return bitA | bitB | bitC;
}
```

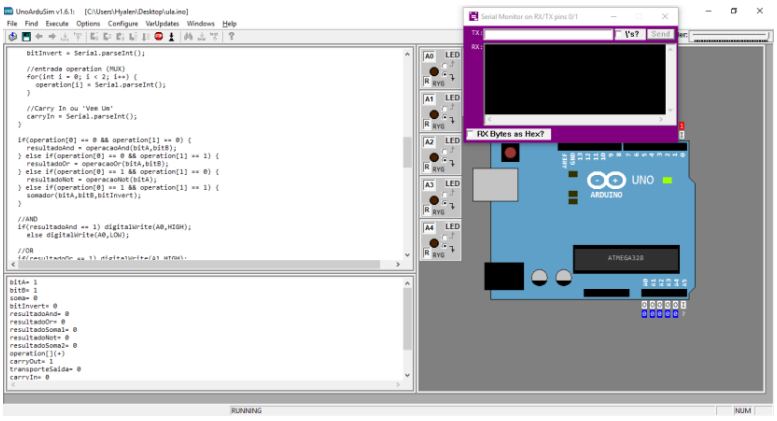
```
//Negacao do bit A ou B
int operacaoNot(int bit){
    return !bit;
}

int operacaoXor(int bitA, int bitB) {
    return bitA ^ bitB;
}

void loop(){
    if(Serial.available() > 0){
        //Leitura das 5 entradas, conforme descrito no relatorio
        //entrada bit A
        bitA = Serial.parseInt();
        //entrada bit B
        bitB = Serial.parseInt();
        //entrada bitInvert (0 para somador ou 1 para subtrador)
        bitInvert = Serial.parseInt();
        //entrada operation (MUX)
        for(int i = 0; i < 2; i++) {
            operation[i] = Serial.parseInt();
        }
        //Carry In ou 'Vem Um'
        carryIn = Serial.parseInt();
    }

    if(operation[0] == 0 && operation[1] == 0) {
        resultadoAnd = operacaoAnd(bitA,bitB);
    } else if(operation[0] == 0 && operation[1] == 1) {
        resultadoOr = operacaoOr(bitA,bitB);
    } else if(operation[0] == 1 && operation[1] == 0) {
        resultadoNot = operacaoNot(bitA);
    } else if(operation[0] == 1 && operation[1] == 1) {
        somador(bitA,bitB,bitInvert);
    }
```

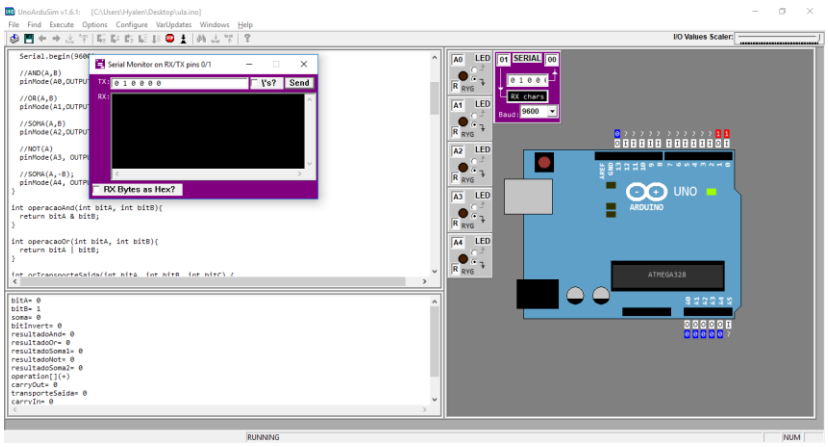
```
    //AND
    if(resultadoAnd == 1) digitalWrite(A0,HIGH);
        else digitalWrite(A0,LOW);
    //OR
    if(resultadoOr == 1) digitalWrite(A1,HIGH);
        else digitalWrite(A1,LOW);
    //SOMA
    if(resultadoSoma1 == 1) digitalWrite(A2,HIGH);
        else digitalWrite(A2,LOW);
    //NOT(A)
    if(resultadoNot == 1) digitalWrite(A3,HIGH);
        else digitalWrite(A3,LOW);
    //SOMA(A,-B)
    if(resultadoSoma2 == 1) digitalWrite(A4,HIGH);
        else digitalWrite(A4,LOW);
    }

    int operacaoSubtracao(int bitA, int bitB) {
        int notB = operacaoNot(bitB);

        //soma-se normalmente, como no somador completo
        int xorAux = operacaoXor(bitA, bitB);
        int andAux = operacaoAnd(bitA, bitB);

        //logica do transporte de saida
        int andSub1 = operacaoAnd(bitA,notB);
        int andSub2 = operacaoAnd(carryIn,notB);
        int andSub3 = operacaoAnd(bitA,carryIn);
        subtracao = operacaoXor(carryIn, xorAux);
        transporteSaida = orTransporteSaida(andSub1, andSub2, andSub3);

        return subtracao;

    }
```

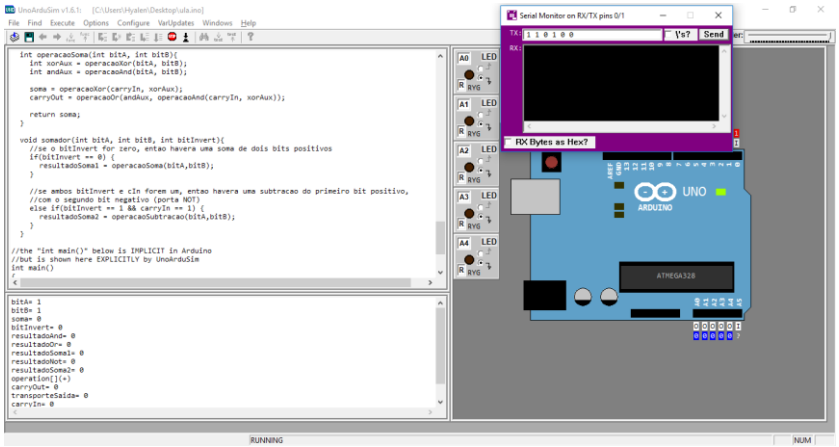
```
    int operacaoSoma(int bitA, int bitB){
        int xorAux = operacaoXor(bitA, bitB);
        int andAux = operacaoAnd(bitA, bitB);
        soma = operacaoXor(carryIn, xorAux);
        carryOut = operacaoOr(andAux, operacaoAnd(carryIn, xorAux));

        return soma;
    }

    void somador(int bitA, int bitB, int bitInvert){
        //se o bitInvert for zero, entao havera uma soma de dois bits positivos
        if(bitInvert == 0) {
            resultadoSoma1 = operacaoSoma(bitA,bitB);
        }

        //se ambos bitInvert e cIn forem um, entao havera uma subtracao do primeiro bit positivo,
        //com o segundo bit negativo (porta NOT)
        else if(bitInvert == 1 && carryIn == 1) {
            resultadoSoma2 = operacaoSubtracao(bitA,bitB);
        }
    }
```

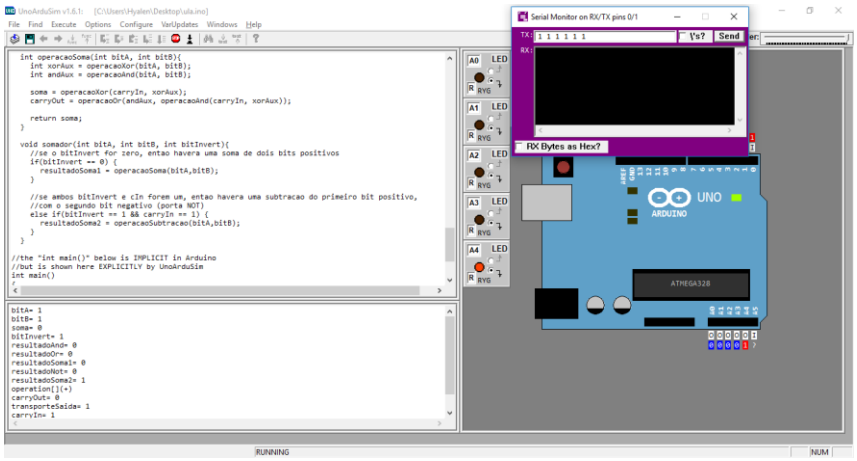
Imagens do código feito no Arduino e abaixo consta as imagens dos testes realizados











Imagens dos testes realizados a partir do código para teste que está no relatório.

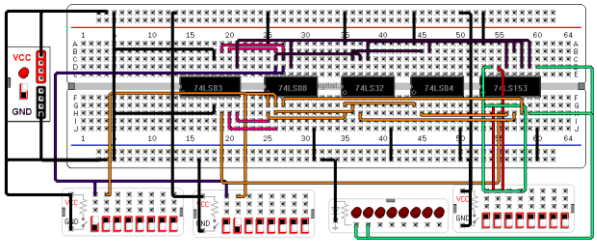
a	b	c	d	op1	op0	s
0	0	0	0	0	0	00
0	0	0	0	0	1	00
0	0	0	0	1	0	1
0	0	0	0	1	1	10

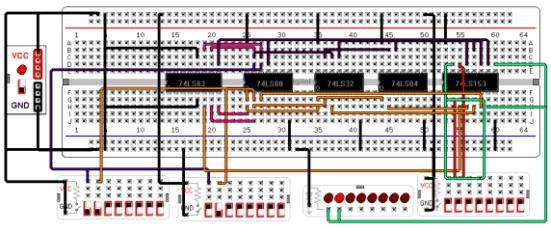
0	0	0	1	0	0	00
0	0	0	1	0	1	01
0	0	0	1	1	0	11
0	0	0	1	1	1	11
0	0	1	0	0	0	00
0	0	1	0	0	1	10
0	0	1	0	1	0	11
0	0	1	0	1	1	01
0	0	1	1	0	0	00
0	0	1	1	0	1	11
0	0	1	1	1	0	11
0	0	1	1	1	1	00
0	1	0	0	0	0	00
0	1	0	0	0	1	01
0	1	0	0	1	0	10
0	1	0	0	1	1	11
0	1	0	1	0	0	01
0	1	0	1	0	1	01
0	1	0	1	1	0	10
0	1	0	1	1	1	10
0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

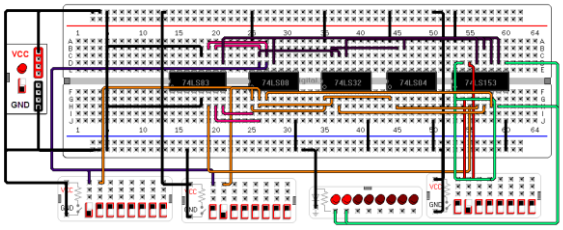
1	1	0	0	0	1	11
1	1	0	0	1	0	00
1	1	0	0	1	1	00
1	1	0	1	0	0	01
1	1	0	1	0	1	11
1	1	0	1	1	0	00
1	1	0	1	1	1	01
1	1	1	0	0	0	10
1	1	1	0	0	1	11
1	1	1	0	1	0	00
1	1	1	0	1	1	10
1	1	1	1	0	0	11
1	1	1	1	0	1	11
1	1	1	1	1	0	00
1	1	1	1	1	1	11

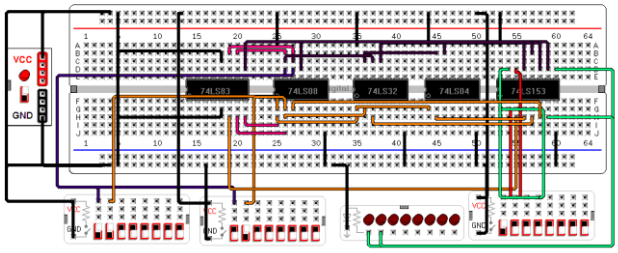
A	B	Op1	Op0	S
A	B	0	0	And(a,b)
A	B	0	1	Or(a,b)
A	B	1	0	Not(a)
A	B	1	1	Soma(a,b)

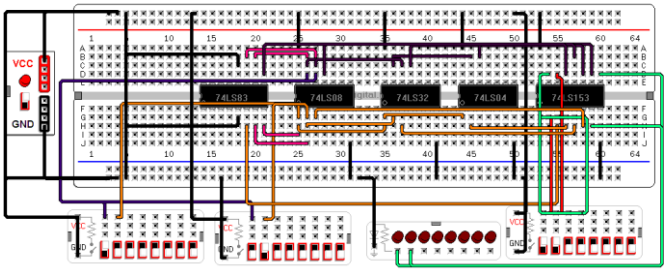
- Tabela verdade adotada no projeto da Unidade Lógica e Aritmética com todas as saídas possíveis











- Imagens com a Unidade Lógica e Aritmética implementada no Simulador 97

e com todos os testes.

Instrução realizada	Binário (A,B,Op.code)	Valor em Hexa	Resultado em binário
And (a,b)	10 01 00	24	00
Or(a,b)	10 01 01	25	11
Soma(a,b)	10 01 11	27	00
Not(a)	11 01 10	36	00
And (a,b)	11 01 00	34	01

- Tabela de testes devidamente preenchida de acordo com as instruções e os valores