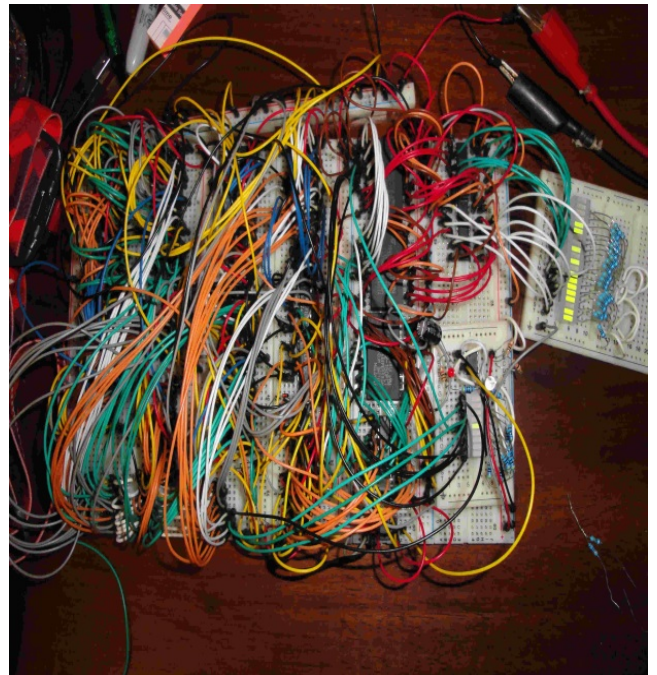
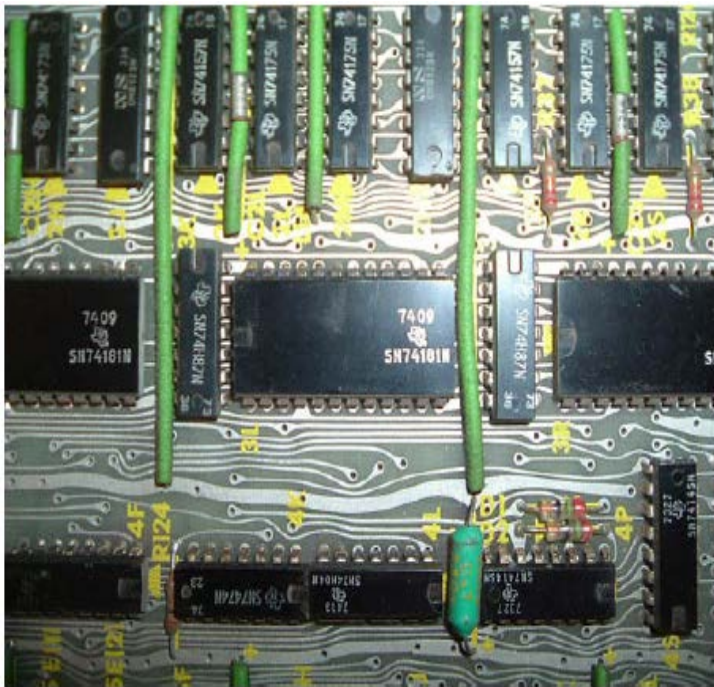
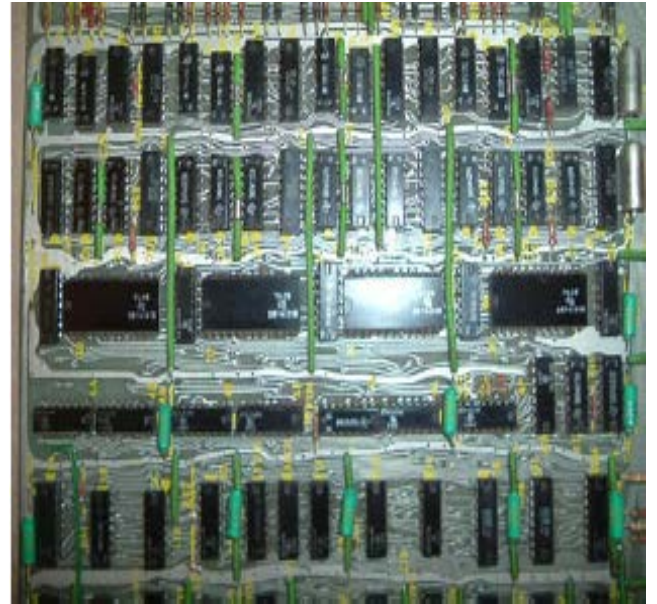
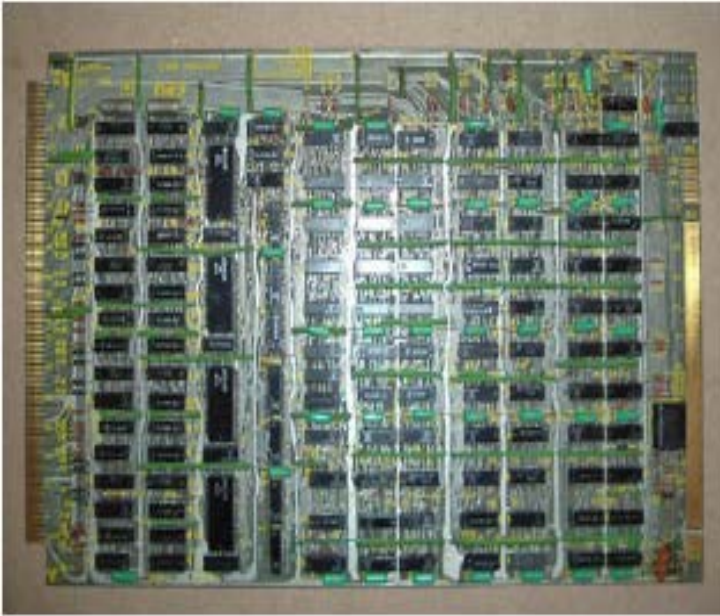


Lab. AC II – Exercício Prático 03

Parte 1

Conhecendo uma ULA de 4 bits (dados e instruções)

Nesta experiência 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 esta mesma ULA dentro do Arduino, por isso é importante conhecê-la.

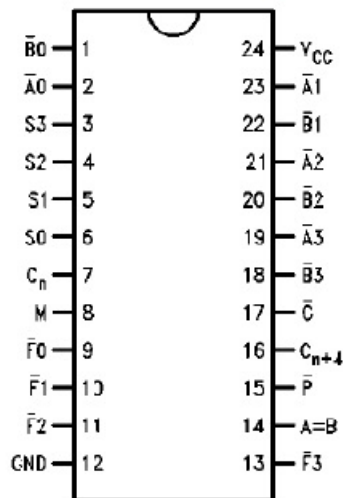


Como a ULA funciona.

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:

SELECTION				M = H LOGIC FUNCTIONS	ACTIVE-HIGH DATA	
S3	S2	S1	S0		M = L; ARITHMETIC OPERATIONS	
					$\overline{C}_n = H$ (no carry)	$\overline{C}_n = L$ (with carry)
L	L	L	L	$F = \overline{A}$	$F = A$	$F = A \text{ PLUS } 1$
L	L	L	H	$F = \overline{A} + B$	$F = A + B$	$F = (A + B) \text{ PLUS } 1$
L	L	H	L	$F = \overline{A}B$	$F = A + \overline{B}$	$F = (A + \overline{B}) \text{ PLUS } 1$
L	L	H	H	$F = 0$	$F = \text{MINUS } 1 \text{ (2's COMPL)}$	$F = \text{ZERO}$
L	H	L	L	$F = \overline{A}B$	$F = A \text{ PLUS } \overline{A}B$	$F = A \text{ PLUS } \overline{A}B \text{ PLUS } 1$
L	H	L	H	$F = \overline{B}$	$F = (A + B) \text{ PLUS } \overline{A}B$	$F = (A + B) \text{ PLUS } \overline{A}B \text{ PLUS } 1$
L	H	H	L	$F = A \oplus B$	$F = A \text{ MINUS } B \text{ MINUS } 1$	$F = A \text{ MINUS } B$
L	H	H	H	$F = \overline{A}B$	$F = \overline{A}B \text{ MINUS } 1$	$F = \overline{A}B$
H	L	L	L	$F = \overline{A} + B$	$F = A \text{ PLUS } AB$	$F = A \text{ PLUS } AB \text{ PLUS } 1$
H	L	L	H	$F = A \oplus B$	$F = A \text{ PLUS } B$	$F = A \text{ PLUS } B \text{ PLUS } 1$
H	L	H	L	$F = B$	$F = (A + \overline{B}) \text{ PLUS } AB$	$F = (A + \overline{B}) \text{ PLUS } AB \text{ PLUS } 1$
H	L	H	H	$F = AB$	$F = AB \text{ MINUS } 1$	$F = AB$
H	H	L	L	$F = 1$	$F = A \text{ PLUS } A$	$F = A \text{ PLUS } A \text{ PLUS } 1$
H	H	L	H	$F = A + \overline{B}$	$F = (A + B) \text{ PLUS } A$	$F = (A + B) \text{ PLUS } A \text{ PLUS } 1$
H	H	H	L	$F = A + B$	$F = (A + \overline{B}) \text{ PLUS } A$	$F = (A + \overline{B}) \text{ PLUS } A \text{ PLUS } 1$
H	H	H	H	$F = A$	$F = A \text{ MINUS } 1$	$F = A$

Connection Diagram



Pin Descriptions

Pin Names	Description
$\overline{A}0-\overline{A}3$	Operand Inputs (Active LOW)
$\overline{B}0-\overline{B}3$	Operand Inputs (Active LOW)
S0-S3	Function Select Inputs
M	Mode Control Input
C_n	Carry Input
$\overline{F}0-\overline{F}3$	Function Outputs (Active LOW)
$A = B$	Comparator Output
G	Carry Generate Output (Active LOW)
\overline{P}	Carry Propagate Output (Active LOW)
C_{n+4}	Carry Output

Nessa primeira parte do experimento você deverá conhecer 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**.

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 ($0100 = 4$).

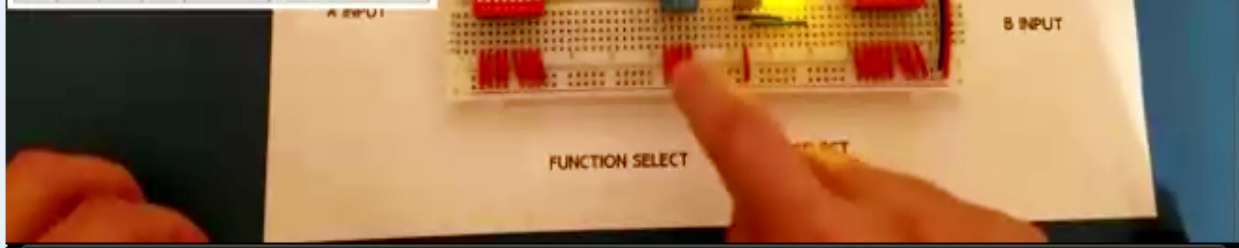
Complete agora a tabela a seguir onde todas as instruções que a ULA pode fazer serão testadas.

Instruções	Binário	Resultado da operação
450		
CB1		
A32		
C43		
124		
785		
9B6		
CD7		
FE8		
649		
D9A		
FCB		
63C		
98D		
76E		
23F		

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.

FUNCTION TABLES

MODE SELECT INPUTS				ACTIVE HIGH INPUTS AND OUTPUTS	
S_3	S_2	S_1	S_0	LOGIC ($M=H$)	ARITHMETIC ⁽²⁾ ($M=L; C_n=H$)
L	L	L	L	\bar{A}	A
L	L	L	H	$A + \bar{B}$	$A + B$
L	L	H	L	$\bar{A}\bar{B}$	$A + \bar{B}$
L	L	H	H	logical 0	minus 1
L	H	L	L	$\bar{A}\bar{B}$	A plus $\bar{A}\bar{B}$
L	H	L	H	\bar{B}	(A + B) plus $\bar{A}\bar{B}$
L	H	H	L	$A \oplus B$	A minus B minus 1
L	H	H	H	$\bar{A}\bar{B}$	$\bar{A}\bar{B}$ minus 1
H	L	L	L	$\bar{A} + B$	A plus $\bar{A}B$
H	L	L	H	$\bar{A} \oplus \bar{B}$	A plus B
H	L	H	L	\bar{B}	(A + \bar{B}) plus $\bar{A}\bar{B}$
H	L	H	H	$\bar{A}\bar{B}$	$\bar{A}\bar{B}$ minus 1
H	H	L	L	logical 1	A plus $A^{(1)}$
H	H	L	H	$A + \bar{B}$	(A + B) plus A
H	H	H	L	$A + B$	(A + \bar{B}) plus A
H	H	H	H	A	A minus 1



O que apresentar nesta parte do relatório:

- 1) A tabela de teste com as funções da ULA completamente preenchida.

Parte 2

ULA 4 bits (programa montador) + Arduino

Nesta parte do exercício você deverá criar 2 programas. Um no hardware externo (Arduino) e outro no PC, que será a interface com o usuário. A ideia é ler um programa escrito pelo usuário, transformá-lo em mnemônicos gerando outro programa e finalmente passá-lo ao Hardware externo através da porta serial e realizar algum processamento nesse Hardware. O resultado será observado nos 4 Leds conectados no Hardware externo.

Arquitetura proposta

Você deverá projetar uma ULA com 4 bits para um dado A, 4 bits para um dado B e 4 bits para a instrução desejada. O funcionamento é similar à ULA anteriormente estudada.

Uma arquitetura do sistema proposto pode ser vista na Figura 1.

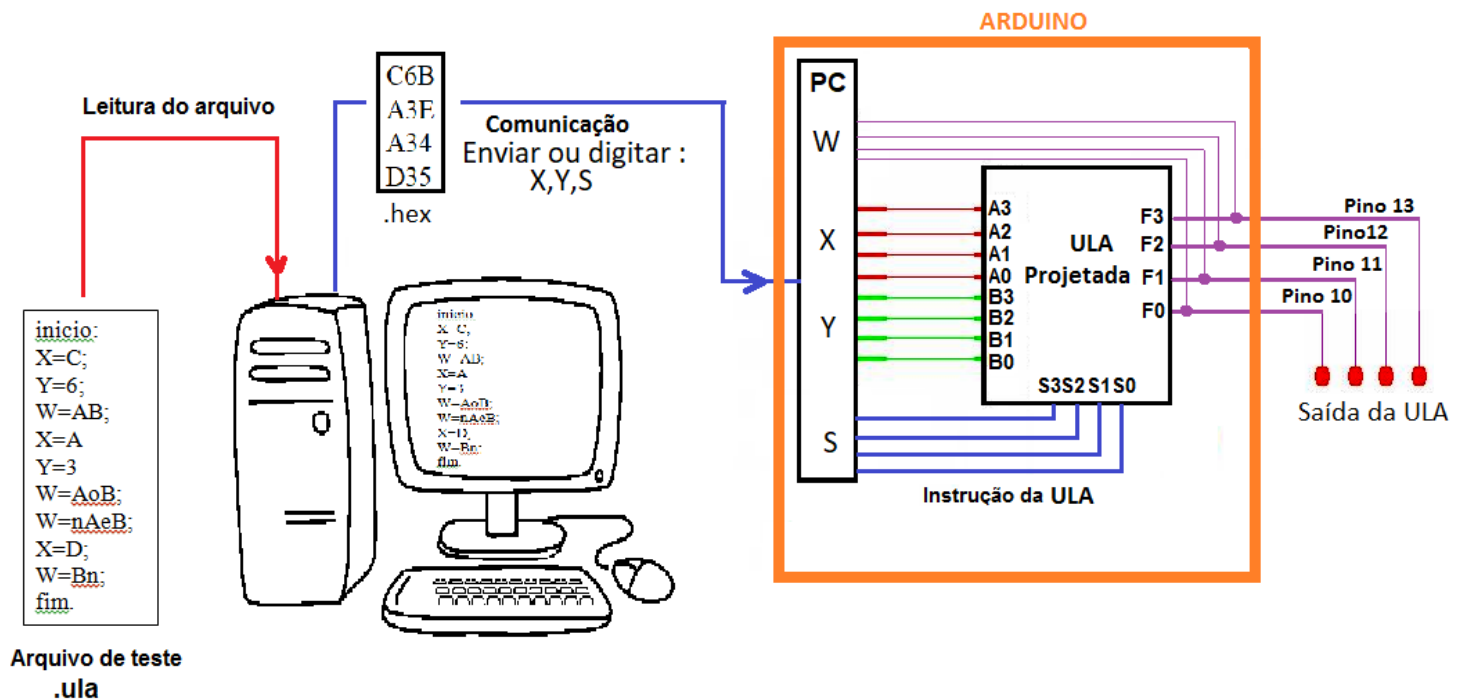


Figura 1: Arquitetura do sistema proposto

Software no PC

O software no PC poderá ser escrito em C, C++, C# , Java ou Python.

Você deverá criar um programa que transforme um texto lido de um arquivo nas instruções a serem executadas e permita a sua execução linha a linha através do console. Para isso, o programa deverá inicialmente ler um arquivo contendo um texto original com os mnemônicos (instruções a serem executadas) e gerar um segundo texto, onde cada linha seja transformada nos valores que serão disponibilizados para a porta USB/serial (ou digitados) no Arduino. Esse segundo texto deverá ser um arquivo gravado com os respectivos valores a serem enviados para a porta USB/serial (ou digitados) porém no formato hexadecimal.

Você deverá utilizar o conjunto de instruções que a ULA possui ilustrado na Figura 2 a seguir:

Função	Mnemônico	Código Hexa
A'	An	0
(A+B)'	nAoB	1
A'B	AnB	2
0 Lógico	zeroL	3
(AB)'	nAeB	4
B'	Bn	5
$A \oplus B$	AxB	6
AB'	ABn	7
A' + B	AnoB	8
$(A \oplus B)'$	nAxB	9
B	copiaB	A
AB	AB	B
1 lógico	umL	C
A+B'	AoBn	D
A+B	AoB	E
A	copiaA	F

Figura 2: Instruções e Mnemônicos
(ativos em 1- high)

A Figura 3 ilustra um pequeno exemplo de código a ser transformado ou o programa fonte. A Figura 4 ilustra o programa a ser gerado ou o programa hex. Os nomes dos arquivos indicados nas Figuras 3 e 4 correspondem aos nomes que você deverá utilizar no programa para leitura e escrita.

```

inicio:
X=C;
Y=6;
W=AB;
X=A;
Y=3;
W=AoB;
W=nAeB;
X=D;
W=Bn;
fim.

```

Figura 3: Exemplo do programa de teste
"testeula.ula"

```

C6B
A3E
A34
D35

```

Figura 4: Programa gerado
"testeula.hex"

Como se vê, o programa fonte (testeula.ula) é convertido no programa executável (testeula.hex).

O ciclo de execução da máquina pode ser entendido através da Figura 5 a seguir e que é executado sobre o programa hex. Imagine que cada linha do programa hex esteja em uma posição da memória e que aconteça a busca de uma instrução de cada vez.

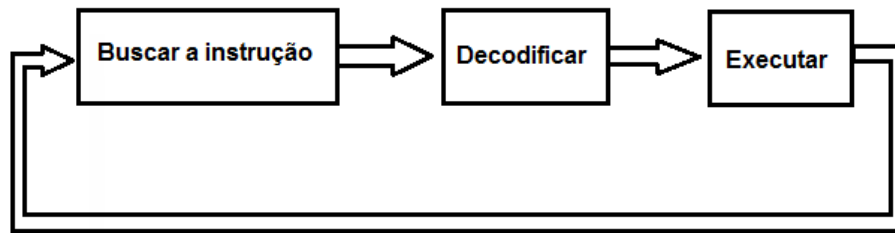


Figura 5: Ciclo de execução de uma instrução

O Programa no Arduino

A ideia é executar exatamente o ciclo de instruções proposto na figura 5 dentro do Arduino.

Você deverá elaborar um programa no Arduino que utilize a entrada serial para receber as entradas necessárias ao funcionamento da ULA (dados e instruções) e as saídas deverão ser 4 Leds ligados aos pinos 13, 12, 11 e 10 (o bit mais significativo no pino 13 e o menos significativo no pino 10).

Seu programa no arduino deverá ser capaz de receber 3 dados da seguinte forma:

Um primeiro valor representando a entrada X (X0, X1, X2 e X3).

Um segundo valor representando a entrada Y (Y0, Y1, Y2 e Y3).

Um terceiro valor representando a instrução desejada S (S0, S1, S2 e S3).

Assim, se fornecermos pela comunicação serial na IDE do Arduino os seguintes 3 valores:

124, estaremos passando para a ULA as seguintes informações:

Valor de X=1, valor de Y=2 e a instrução desejada=4 ou S=4. A ULA projetada no arduino deverá então realizar, conforme o conjunto de instruções da ULA (de acordo com a Fig. 2), a instrução (nAeB), ou seja (AB)' que sobre as variáveis X e Y ficaria (XY)'.

Observe que as operações sempre serão realizadas sobre as variáveis X e Y e o resultado sempre será em W.

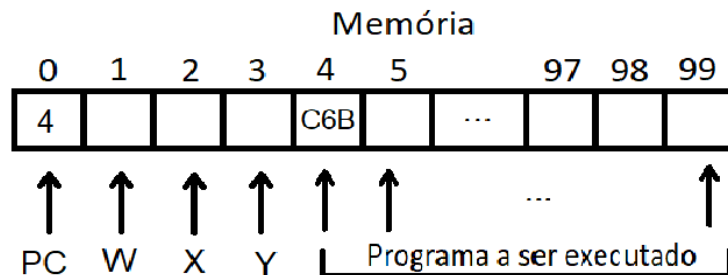
Para não haver confusão nos valores, deveremos usar os números em Hexadecimal, assim, se passarmos ao Arduino os seguintes dados AAA, o significado será:

Valor de X = 10, valor de Y=10 e a instrução desejada ou S=10. A ULA projetada no arduino deverá então realizar, conforme o conjunto de instruções da ULA (e de acordo com a Fig. 2) a instrução B, atenção que a instrução B apenas coloca o valor da entrada Y na saída (não confunda a instrução B com a entrada tendo o valor de B).

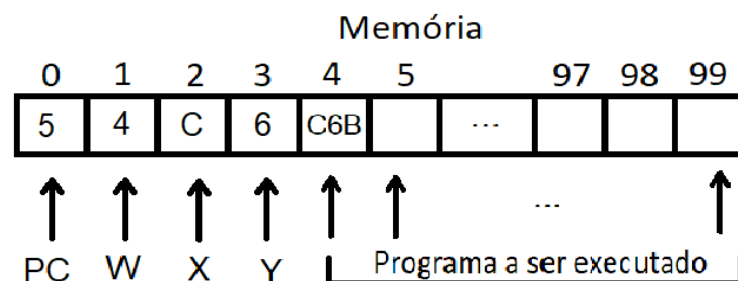
Em um segundo momento deveremos executar a instrução ou seja, verificar no PC qual a posição do vetor que contem a instrução, ler esta instrução, atribuir os valores de X, Y e após decodificar e executar a instrução escrever o valor de W.

- 1) X = 1100 ou C;
- 2) Y = 0110 ou 6;
- 3) Como deveremos executar a instrução B, de acordo com a tabela de instruções, a instrução B (11) é AB, ou seja como S=1011 indica que queremos a instrução AB (and das entradas), a saída F seria 0100, o and bit a bit entre X e Y e cuja resposta é 0100 (4 em hexadecimal) ou o led do pino 12 ligado.
- 4) Além disso a memória deverá ser atualizada ou seja, deveremos escrever os valores de W, X, Y e atualizar o PC, isto é apontar para a próxima instrução a ser realizada (no caso 5).

Antes da execução da instrução (primeiro momento):



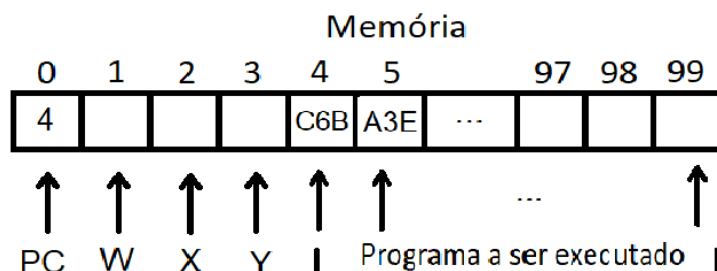
Após a execução da instrução(após o segundo momento):



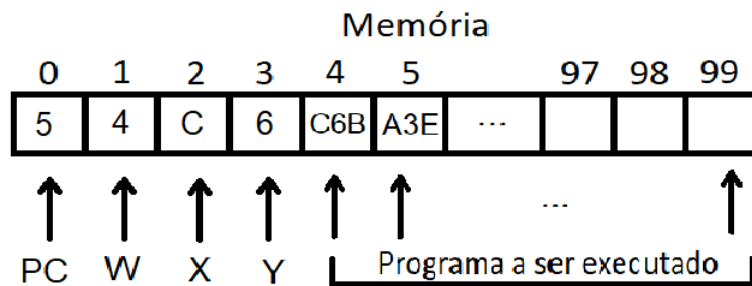
Observe que agora o PC está indicando que a próxima instrução está na posição 5, que no nosso caso não possui nenhuma instrução.

Exemplo 2: Vamos supor que iremos executar agora duas instruções, C6B e logo a seguir A3E. Como iremos executar 2 instruções, os valores de PC, W, X e Y variarão duas vezes. A sequencia será a seguinte:

- 1) Carga do programa no vetor:

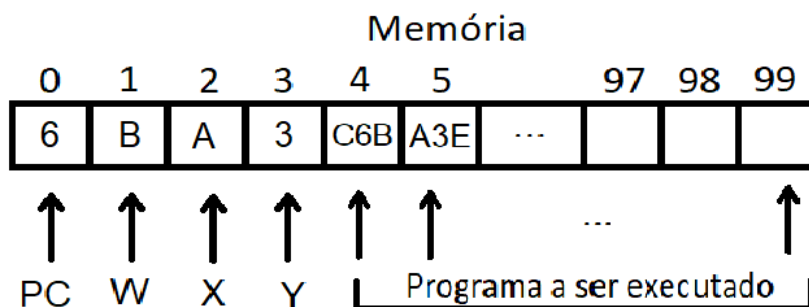


2) Após a execução da primeira instrução:



E o Led do pino 12 ligado, já que o resultado em W é 4 ou 0100 e o PC apontando para a próxima instrução (5).

3) Após a execução da segunda instrução



Para a execução desta segunda instrução, o valor de X=A (1010), o valor de Y=3(0011) e a operação desejada S=E(1110). A operação E, pela tabela é o “ou” de X com Y, ou seja (1011) e que corresponde ao valor B. O PC também deverá ser incrementado de 1 indicando a próxima instrução a ser realizada.

Além disto, os Leds dos pinos 13, 11 e 10 ligados.

Atenção (detalhe de projeto 1): Fica a critério do grupo a utilização do vetor para armazenar os valores, quer seja números na base decimal, binária ou mesmo String. Da mesma forma, fica a critério do grupo determinar uma forma de indicar que o programa acabou (não existem instrução a ser realizada).

Funcionamento:

1)

Ao iniciarmos o Arduino, inicialmente deveremos fazer a carga do programa no vetor que representa a memória.

Atenção (detalhe de projeto 2): O grupo deverá propor uma forma de entrar com os dados que estarão no formato descrito pelo arquivo testeula.hex e que será descrito posteriormente nesta especificação. Esta carga poderá ser realizada digitando-se cada instrução no Arduino e completando o vetor.

Neste momento não é para executar as instruções, apenas fazer a carga do vetor.

2)

Ao iniciarmos a execução do programa, o Arduino deverá ler as instruções da memória a partir do local indicado pelo valor armazenado na primeira posição (ou o PC, como é a primeira instrução ele deverá conter o valor 4), decodificar a instrução, executar a instrução, escrever os valores das variáveis X e Y na memória e o resultado em W, incrementar o PC de 1 (posição 0 do vetor) assim como mostrar o valor do resultado nos leds.

Posteriormente passar para a próxima instrução e assim sucessivamente. Vamos considerar inicialmente um intervalo de 2 segundos entre cada instrução, para podermos acompanhar as respostas nos LEDs.

Ao final da execução de cada instrução deveremos ter um DUMP da memória, ou seja, o Arduino mostrará todos os valores contidos na memória para acompanharmos a execução. Esta opção poderá estar continuamente ativa ou ser ativada caso o usuário deseje. Procure mostrar neste DUMP apenas as posições onde existem valores na memória e não toda ela. A cada linha executada uma nova linha deverá ser mostrada exatamente como a memória está. Para o exemplo 2, poderíamos ter as seguintes respostas:

Carga do vetor:

- >| 4 | 0 | 0 | 0 | C6B | A3E |

Após a execução da primeira instrução:

- >| 5 | 4 | C | 6 | C6B | A3E |

Após a execução da segunda instrução:

- >| 6 | B | A | 3 | C6B | A3E |

Um possível formato da tela que irá aparecer no arduino, com um intervalo de 2 segundos entre cada linha:

- >| 4 | 0 | 0 | 0 | C6B | A3E | | |

- >| 5 | 4 | C | 6 | C6B | A3E | | |

- >| 6 | B | A | 3 | C6B | A3E | | |

Observe que **não** mostramos todo o vetor (ou a nossa memória), **apenas** até onde temos alguma coisa na memória.

Como o programa no PC e o programa no Arduino se completam:

O que será executado no arduino deverá ser o programa testeula.hex, Figura 4, (Não é o programa com os mnemônicos).

Inicialmente cada linha do programa deverá ser digitada no Monitor Serial e enviada para o Arduino, isso irá preencher o vetor Memória com todo o programa que deverá ser executado.

ATENÇÃO : Não é para digitar uma instrução e esta instrução ser imediatamente executada, neste momento apenas deveremos preencher o vetor com todas as instruções que estão no programa testeula.hex.

Após esta carga da memória o programa do Arduino terá início, executando uma instrução de cada vez e mostrando os resultados no Dump de memória e nos Leds de saída.

O que apresentar ao final do projeto

- 1) Um programa no Arduino que simule uma ULA e receba os valores dos dados e instruções através da porta serial (no tinkercad através do “Monitor Serial”).
- 2) Um programa de acesso em C/C++/Java/Python (com muitos comentários!) que:
 - a) **leia um programa fonte (com os dados e os mnemônicos), você deverá criar um programa fonte de teste. Durante a aula um outro programa será utilizado para verificação do trabalho.**
 - b) **gere um arquivo hexa correspondente aos dados e instruções**

O programa de teste possuirá o nome "testeula.ula" e você só terá acesso a ele no momento do teste. O formato será o mesmo descrito na Figura 3 e os mnemônicos da Figura 2. O programa gerado deverá possuir o nome "testeula.hex".

Para o seu teste crie seu próprio programa fonte, lembre-se de procurar testar todas as instruções possíveis.

Para o teste de todo o sistema deveremos copiar o texto do programa gerado “testeula.hex” no Monitor Serial do Arduino e executar todas as linhas acompanhando a execução através do Dump de memória.

Cada grupo fará a apresentação dos programas, **(será a nota desse relatório, não haverá entrega pela internet, apenas a apresentação.)**

Cada grupo deverá estar com os programas disponíveis e fazer a apresentação do trabalho.

Eu irei avaliar/ testar individualmente os programas de cada grupo durante a apresentação, **alunos que participaram do trabalho mas ausentes na apresentação, não terão nota.**

Grupos que não estiverem com os programas não terão nota no relatório, o mesmo acontecendo com trabalhos copiados.

Comece já, você nunca terá tanto tempo!