

Exercício Prático 04 - ULA 4 bits + Arduino

Neste exercício você deverá criar 2 programas. Um no hardware externo (Arduino) e outro no PC, que será a interface com o usuário. A idéia 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.

O Hardware externo

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.

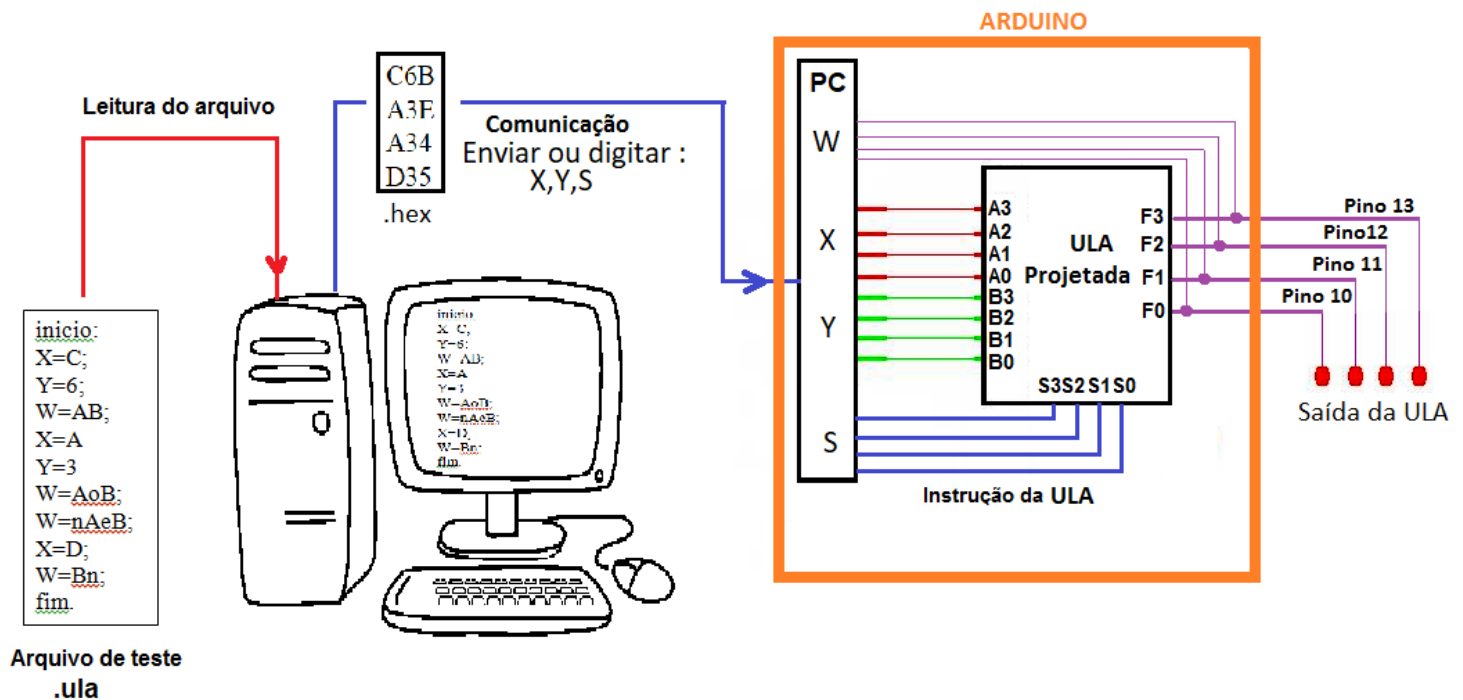


Figura 1: Arquitetura do sistema proposto

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).

A figura 2 a seguir mostra o conjunto de instruções da ULA e que você deverá inserir no Arduino.

Função	Mnemônico	Código Hexa
B'	nB	0
$(A'.B)'$	nAeBn	1
$A'.B$	nAeB	2
0 Lógico	Lzero	3
$(A.B)'$	AeBn	4
A'	nA	5
$A \oplus B$	AxB	6
$A.B'$	AenB	7
$A' + B'$	nAonB	8
$A' \oplus B'$	nAxnB	9
B	Bcopia	A
A.B	AeB	B
1 lógico	Lum	C
$A+B'$	AonB	D
$A+B$	AoB	E
A	Acopia	F

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

O programa no Arduino

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 (AeBn), 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**).

Deverá existir internamente no Arduino um vetor que será a memória (e também os registradores) da Unidade. Este vetor deverá conter nos quatro primeiros campos (que serão os registradores da máquina) os seguintes valores:

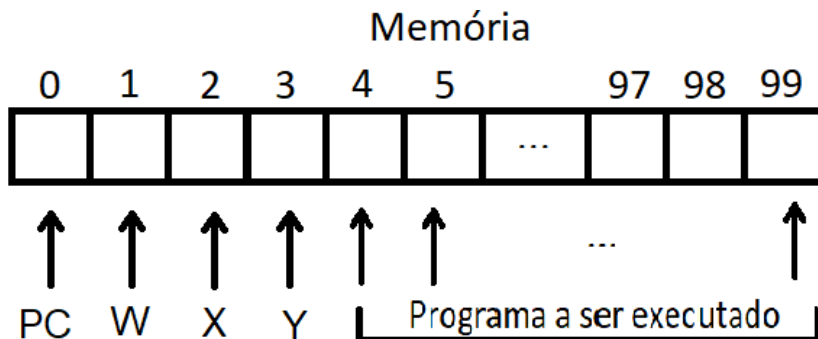
Primeira posição = o índice do vetor onde a instrução está armazenada, que chamaremos de PC

Segunda posição = o conteúdo da variável W (que contém os resultados das operações)

Terceira posição = o conteúdo da variável X

Quarta posição = o conteúdo da variável Y

Vamos considerar que iremos utilizar um espaço relativo a 100 posições (esta será a nossa área de memória) onde as quatro primeiras posições serão as variáveis e as 96 restantes o programa a ser executado.



Se, durante o programa a ser executado, houver uma alteração no valor das variáveis X, Y ou W este valor deverá ser alterado na memória (nas respectivas posições do vetor). A alteração se dá através da atribuição de valores no programa fonte original.

Deveremos ainda acompanhar a evolução do programa observando 4 leds indicadores dos resultados ou seja, a saída da ULA deverá estar presente nos seguintes pinos:

Pino 13 = F3

Pino 12 = F2

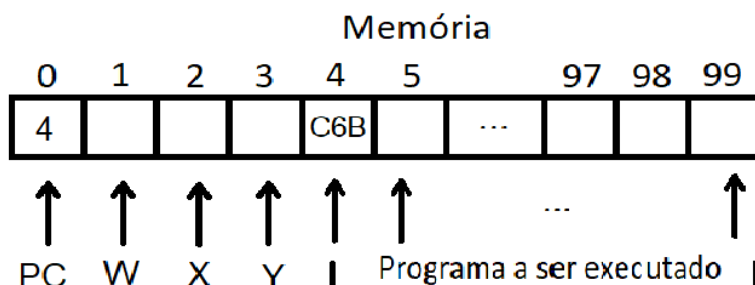
Pino 11 = F1

Pino 10 = F0

Exemplo 1: Qual seria o significado de passarmos para o Arduino os seguintes valores “C6B”, como ficariam os LEDs ligados na saída e a memória antes e após a execução da instrução?

Resp:

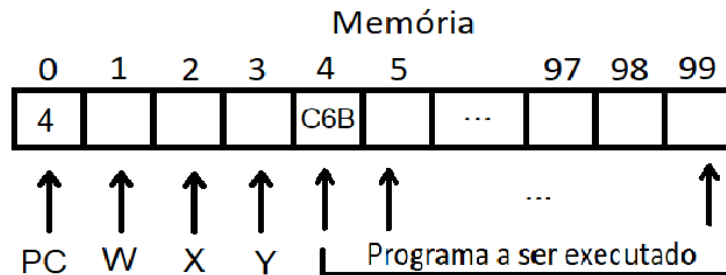
Em um primeiro momento faremos a carga do programa, que no nosso caso será executar apenas a instrução C6B. Assim, a instrução será colocada na primeira posição do vetor onde o programa tem início, ou seja o índice 4 do vetor. A primeira posição (PC) deverá indicar qual o índice do vetor onde esta instrução está (como é a primeira deverá indicar o valor 4). Assim o vetor deverá ficar da seguinte forma:



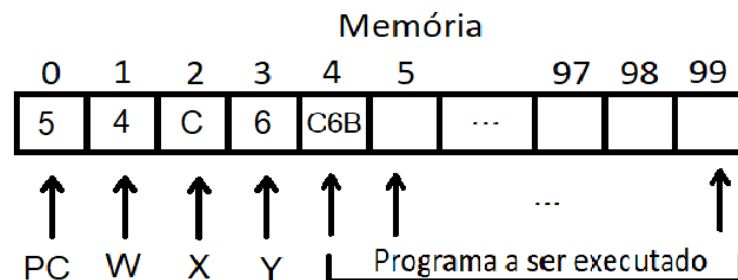
Em um segundo momento deveremos executar a instrução ou seja, verificar no PC qual a posição do vetor que contem a instrução, ler esta instrução, atribuir os valores de X, Y e após decodificar e executar a instrução escrever o valor de W.

- 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 AeB, ou seja como S=1011 indica que queremos a instrução AeB (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):



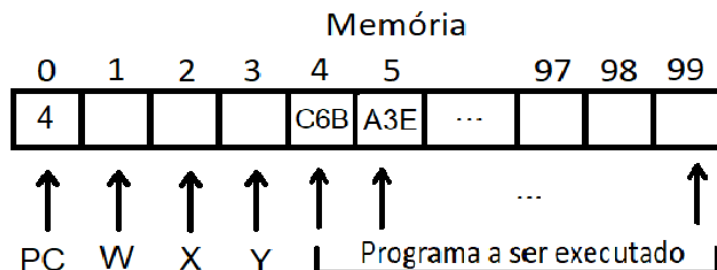
Após a execução da instrução (após o segundo momento):



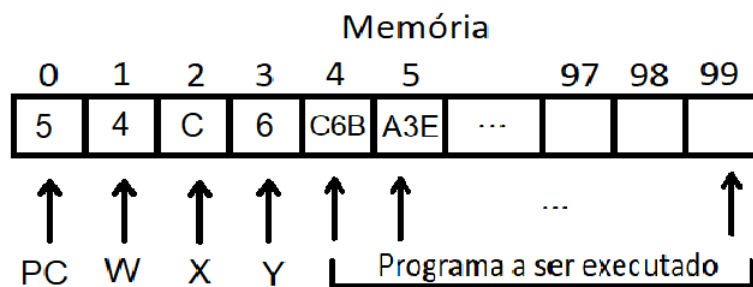
Observe que agora o PC está indicando que a próxima instrução está na posição 5, que no nosso caso não possui nenhuma instrução.

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 sequência será a seguinte:

- 1) Carga do programa no vetor:

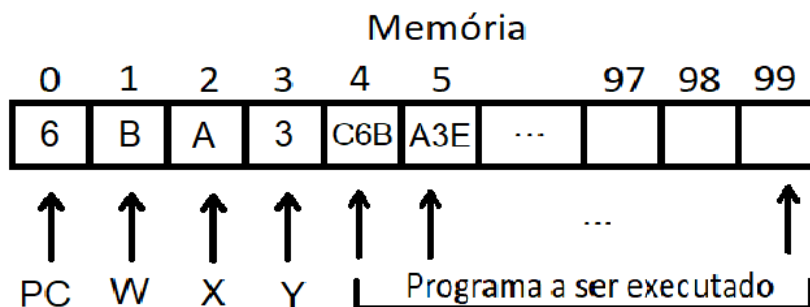


2) Após a execução da primeira instrução:



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 ou selecionando-se todo o código e copiando no campo “Enviar” do Arduino.

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 a 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 4 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.

O 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 Figura 3 ilustra um pequeno exemplo de código a ser transformado. A Figura 4 ilustra o programa a ser gerado. 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=AeB;  
X=A;  
Y=3;  
W=AoB;  
W=AeBn;  
X=D;  
W=nB;  
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) e que deverá ser carregado no arduino para posterior execução.

O que será executado no arduino deverá ser o programa no formato .hex, Figura 4 e não o programa fonte original (programa no formato .ula, Figura 3).

O ciclo de execução da máquina pode ser entendido através da Figura 5 a seguir.

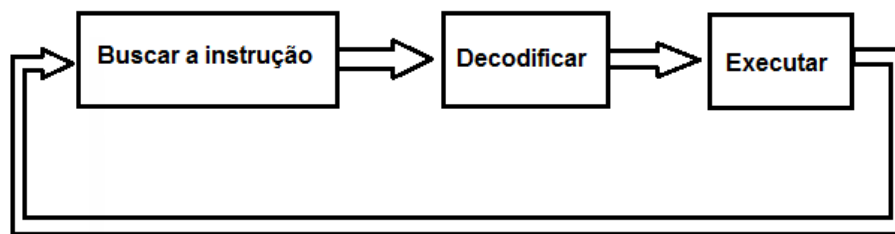


Figura 5: Ciclo de execução de uma instrução

2. 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 ou sejam digitados.
- 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 e**
 - c) **envie dados e instruções** para a ULA (Arduino)

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.

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, o mesmo acontecendo com trabalhos copiados.

Principais situações onde se pode perder pontos:

- 1) O programa .hex ser executado conforme as instruções serem inseridas e não após copiar todas as instruções para a memória (ou o vetor memória).
- 2) Não usar o valor da posição PC para indexar no vetor memória a instrução a ser executada.
- 3) O formato da execução mostrado no Arduino não mostrar o vetor memória de forma clara (PC, W, X, Y e as instruções) e não for mostrado conforme a execução instrução por instrução.
- 4) Algum mnemônico não for interpretado corretamente.
- 5) Alguma instrução não for executada corretamente.
- 6) Os códigos tanto do Arduino quanto do PC não estiverem comentados.

Comece já, você nunca terá tanto tempo!