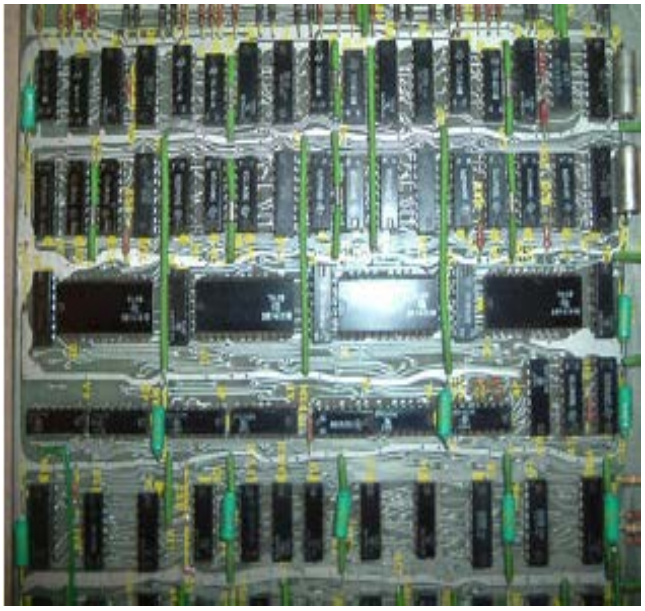
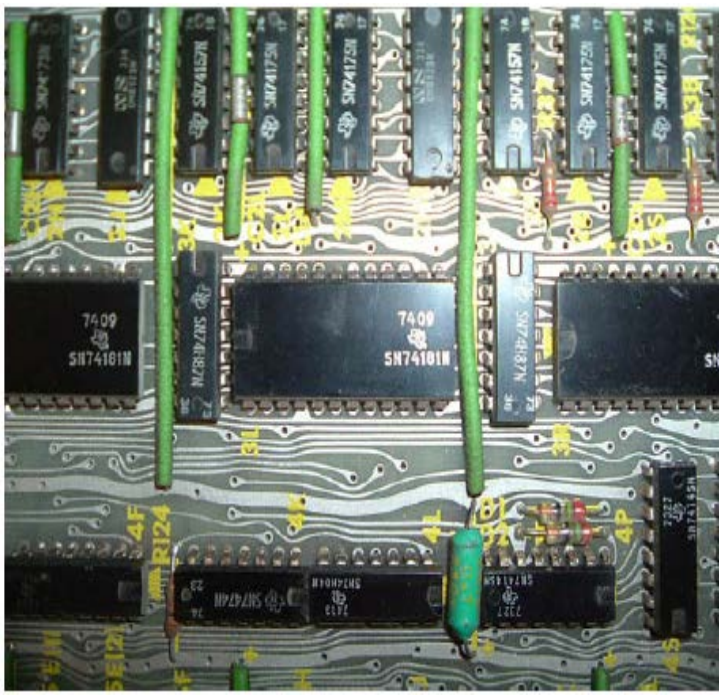
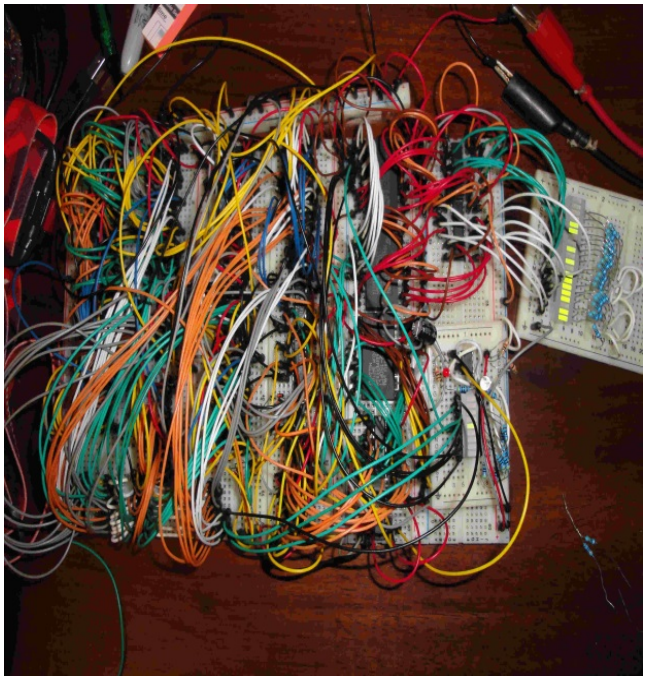
Exercício Prático 4 - AC II – ULA 4 bits + Arduino

Neste exercício utilizaremos 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.





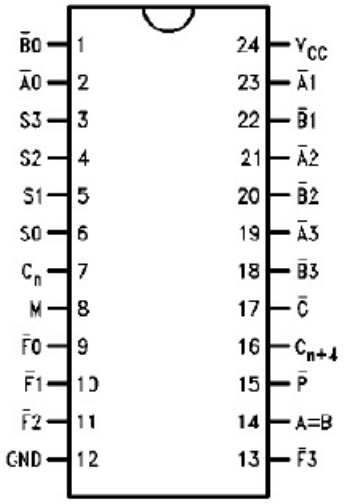


Como a ULA funciona.

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

Neste exercício, remos utilizar apenas as instruções lógicas.

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.

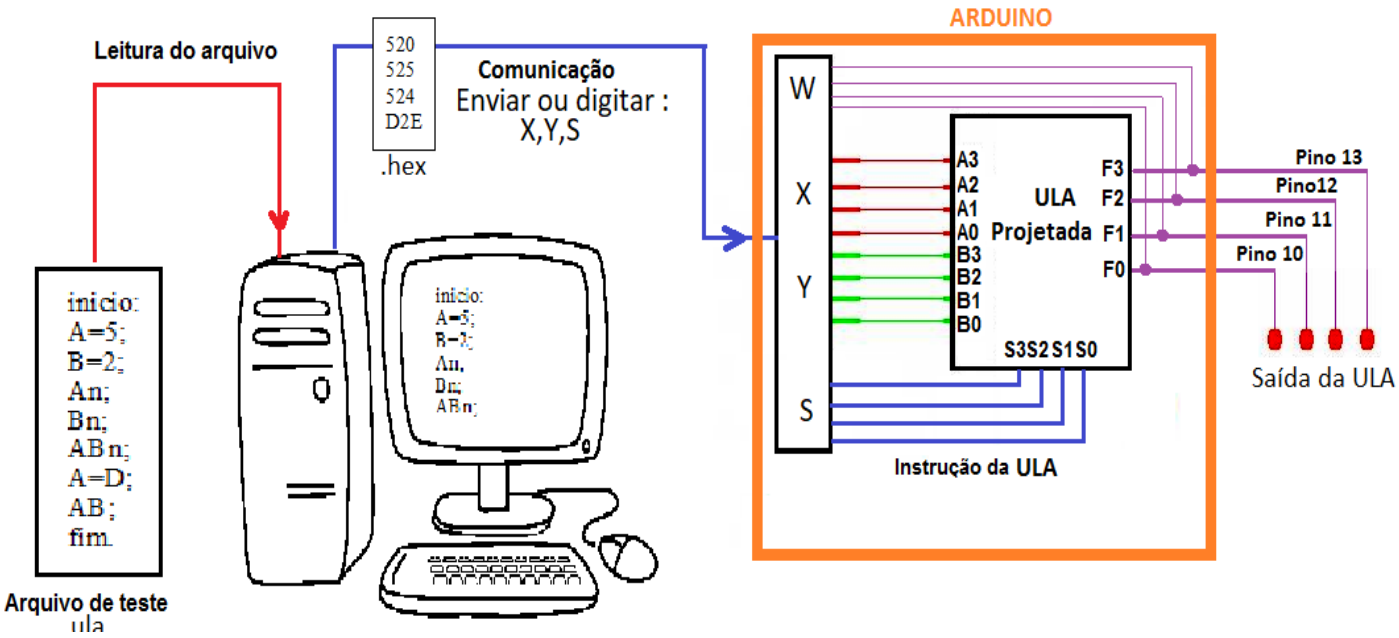


Figura 1: Arquitetura do sistema proposto

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
A’	An	0
(A+B)’	nAoB	1
A’B	AnB	2
0 Lógico	zeroL	3
(AB)’	nAeB	4
B’	Bn	5
A.B	AxB	6
AB’	ABn	7
A’+ B	AnoB	8
(A.B)’	nAxB	9
B	B	A
AB	AB	B
1 lógico	umL	C
A+B’	AoBn	D
A+B	AoB	E
A	A	F

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

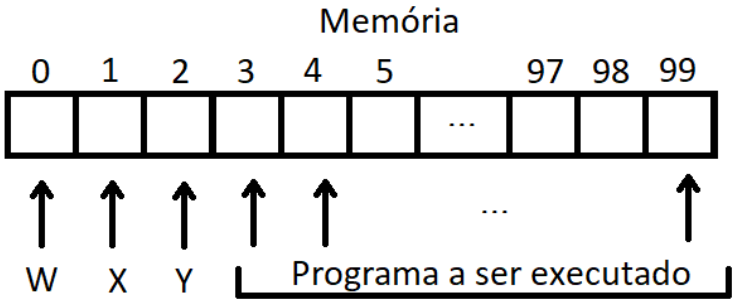
O programa no Arduino

Seu programa no arduino deverá ser capaz de receber 3 dados da seguinte forma:
Um primeiro operando representando a entrada X (X0, X1, X2 e X3).
Um segundo operando 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:
1 2 4, 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.

Observe que, para não haver confusão nos valores, deveremos usar os números em Hexadecimal, assim, se passarmos ao Arduino os seguintes dados A A A, 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 da Unidade ou seja, deverá conter nos três primeiros campos os valores do resultado após a execução de uma instrução (W) e os dois operandos (X e Y). Vamos considerar que iremos utilizar um espaço relativo a 100 posições (esta será a nossa área de memória) onde as três primeiras posições serão as variáveis e as 97 restantes o programa a ser executado.



Se, durante o programa a ser executado, houver uma alteração no valor dos operandos (as variáveis X ou Y), 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:

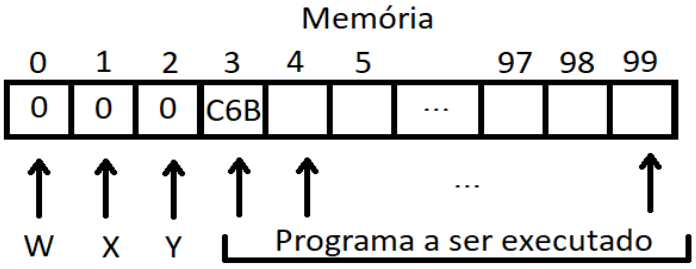
Pino 13 = F3
Pino 12 = F2
Pino 11 = F1
Pino 10 = F0

Pergunta: Qual seria o significado de passarmos para o Arduino os seguintes valores “C 6 B”, como ficariam os LEDs ligados na saída e a memória antes e após a execução da instrução?

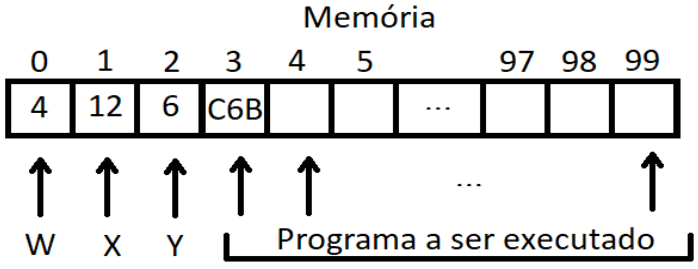
Resp:

Entrada dos valores para o primeiro operando ou X = 1100 ou C;
Entrada para os valores para o segundo operando ou Y = 0110 ou 6;
Entrada para os valores da instrução a ser executada ou S = 1011 ou B;
Como S=1011 indica que queremos a instrução AB (and das entradas), a saída F seria 0100 (4 em decimal) ou o led do pino 12 ligado.
Além disso a memória deverá conter os seguintes valores antes e após a execução do programa

Antes da execução da instrução:



Depois da execução da instrução:



Atenção (detalhe de projeto 1): Fica a critério do grupo a utilização no vetor de números na base decimal, binária ou mesmo String.

O que Fazer:

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. Eventualmente esta carga poderá ser realizada através de comunicação USB/serial (caso um Arduino real esteja presente) ou digitada, caso tenhamos uma versão virtual do Arduino.

2)

Ao iniciarmos a execução do programa, o Arduino deverá ler as instruções da memória a partir do início do programa a ser executado (posição 3 do vetor), decodificar a instrução, executar a instrução e escrever os valores do primeiro operando, do segundo operando e o resultado (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.**

Um exemplo do que deverá ocorrer:

1) Suponha que o programa a ser executado seja o seguinte:

   520
   525
   524
   D2E

2) O primeiro passo será carregar o programa acima para o Arduino (podemos digitar uma instrução de cada vez ou todas de uma vez, esta decisão é do grupo). Este programa deverá estar armazenado em um vetor, que sera a nossa memória, a partir da quarta posição.

0	1	2	3	4	5	6	7	8	9
0	0	0	520	525	524	D2E	…	…	…

Vetor Memória

3) As três primeiras posições serão os registradores que deverão, a cada instrução executada, conter:
   Posição 0 : o resultado da operação
   Posição 1 : o valor do primeiro operando
   Posição 2 : o valor do segundo operando
4) Quando o programa já estiver carregado iremos acionar a execução (o grupo deverá decidir como este processo terá início). Como anteriormente mencionado, cada instrução deverá gastar 2 segundos (é o tempo para observarmos a memória).
5) A cada instrução:
   a. seu programa deverá ler da memória a instrução;

b. decodifica-la ou seja, separar os dois primeiros valores da instrução (os operandos X e Y e a instrução a ser realizada);
c. executá-la ou seja, escrever na primeira posição do vetor o resultado da operação;
d. mostrar como estão os valores na memória.

Se considerarmos como exemplo o programa lido, a primeira instrução é 520, ou seja o primeiro operando vale 5, o segundo operando vale 2 e deveremos executar a operação 0, que é nA (primeiro operando negado). Como o primeiro operando vale 5 (0101) ele negado será 1010 ou o valor A em hexadecimal assim, após a execução desta instrução, o vetor memória deverá estar da seguinte forma:

0	1	2	**(3)**	4	5	6	7	8	9
A	5	2	520	525	524	D2E	…	…	…

Após 2 segundos iremos executar a segunda instrução 525, novamente o primeiro operando vale 5, o segundo 2 porém deveremos realizar a operação 5 (segundo operando negado). Como o segundo operando vale 2 (0010), ele negado será (1101) ou o valor D em hexadecimal assim, após a execução desta instrução, o vetor memória deverá estar da seguinte forma:

0	1	2	3	**(4)**	5	6	7	8	9
D	5	2	520	525	524	D2E	…	…	…

6) Lembre-se de, a cada instrução efetuar um DUMP da memória ou seja, toda a memória deverá ser mostrada (assim como os dois passos acima mostrados). É importante destacar qual a instrução foi realizada no momento, o que está sendo mostrado pelo índice do vetor em negrito (cada grupo deverá decidir como isto será realizado).

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=5;
Y=2;
An;
Bn;
nAeB;
X=D;
AoB;
fim.
```

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

```
520
525
524
D2E
```

Figura 4: Programa gerado "testeula.hex"

Como se vê, o programa a ser enviado(ou digitado) deverá ser a partir do arquivo contendo o programa gerado (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.

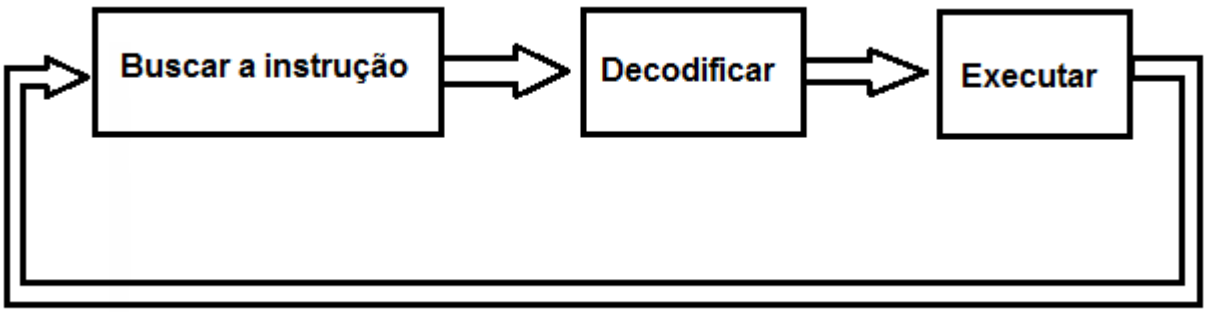


Figura 5: Ciclo de execução de uma instrução

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** através da porta Serial para a ULA (Caso um Arduino real esteja presente)

   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 virtualmente.

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 também não terão nota, o mesmo acontecendo com trabalhos copiados.

Comece já, você nunca terá tanto tempo!