

## Experimento 7

### TUTORIAL DA LINGUAGEM ASSEMBLY

	Turmas A e C	Turmas B e D
Pré Relatório	20/10/2015	22/10/2015
Visto	20/10/2015	22/10/2015
Relatório	03/11/2015	05/11/2015

#### I. OBJETIVO

O objetivo deste experimento é apresentar a linguagem Assembly, sua estrutura e sua sintaxe. Para alcançar esse objetivo serão implementados um somador e um contador de pulsos e, para cada um destes, será mostrada cada etapa desde o desenvolvimento à simulação. Este experimento também servirá de tutorial para que os alunos aprendam a utilizar as ferramentas necessárias para o desenvolvimento de outros experimentos com a linguagem Assembly, em especial para o microcontrolador 8051. Para a avaliação do experimento, será solicitado ao aluno implementar um contador de pulsos com reset e uma calculadora.

#### II. INTRODUÇÃO AO ASSEMBLY

Uma linguagem de montagem ou assembly é uma notação legível por humanos para o código de máquina que uma arquitetura de computador específica usa. A linguagem de máquina, que é um mero padrão de bits, torna-se legível pela substituição dos valores em bruto por símbolos chamados **mnemônicos**.

Por exemplo, enquanto um computador sabe o que a instrução de máquina E5-80 (11100101 10000000) faz, para os programadores é mais fácil interpretar a representação equivalente em instruções mnemônicas **MOV A, P0**. Tal instrução ordena que o valor que se encontra na porta P0 seja movido para o acumulador 'A'.

Ao contrário do que acontece nas linguagens de alto nível, existe (até certo ponto) uma correspondência de 1 para 1 entre a linguagem de montagem simples e a linguagem de máquina. Por isso, a tradução do código de montagem em código de máquina não é chamada compilação, mas montagem.

Consegue-se transformar a linguagem de montagem em linguagem de máquina recorrendo a um montador (também chamado “assembler”), e a transformação inversa faz-se recorrendo a um desmontador (também chamado “disassembler”, originado do termo em inglês).

Cada arquitetura de computador tem a sua própria linguagem de máquina e, portanto, sua própria linguagem de montagem. Essas linguagens de montagem diferem no número e tipo de operações que suportam. Também têm diferentes tamanhos e números de registros, e diferentes representações dos tipos de dados armazenados. Enquanto todos os computadores de utilização genérica são capazes de desempenhar essencialmente as mesmas funções, o modo como o fazem é diferente.

Além disso, podem existir conjuntos múltiplos de mnemônicas, ou sintaxes de linguagem de montagem, para um único conjunto de instruções. Nestes casos, o conjunto mais popular é aquele que é utilizado pelo fabricante na sua documentação.

#### III. DESENVOLVIMENTO EM ASSEMBLY

Para a realização do experimento, serão necessários um editor para escrevermos o código, um montador e um simulador. Para a edição do código, podemos utilizar o qualquer editor de texto simples, que seja capaz de salvar o arquivo no formato de texto puro (“plain text”, isto é, sem formatação). Por exemplo, pode ser utilizado o próprio “bloco de notas” do Windows, ou o VI no Linux. Recomenda-se a utilização do software Notepad++, pois é disponível em versão livre, multi-plataforma, e fornece ferramentas avançadas como múltiplas abas e highlight em diversas linguagens de programação. É importante que o texto seja salvo em formato de texto puro (“plain text”), isto é, não deve ser salvo como arquivo do MS Word ou formatos que insiram código de formatação de documento.

O montador a ser utilizado será o DOS8051, disponível no Moodle. Para a simulação, é recomendável a utilização do TS Controls Emulator 8051, ou de qualquer outro simulador de sua preferência. Neste roteiro, utilizaremos o TS Controls Emulator 8051 como tutorial.

### 3.1 Somador

Como primeiro exemplo, vamos desenvolver um sistema que leia os valores apresentados nas portas P0 e P1 e apresente a soma destes dois valores na porta P2. O primeiro conjunto de código que deve ser inserido neste sistema deve ser a indicação de que estamos trabalhando com a arquitetura 8051, portanto usamos no cabeçalho do arquivo, o código:

```
$MOD51
```

Precisamos então indicar onde, na memória de código, colocaremos o código de máquina que será gerado pelo montador. Nós selecionamos o início da memória para isso. Portanto, devemos utilizar o código:

```
ORG 0
```

A seguir limpamos o acumulador para que não apareçam valores indesejáveis no resultado da nossa soma, para isso, nós movemos o valor 0 para o acumulador, através do comando:

```
MOV A, #0H
```

A seguir passamos para o algoritmo da soma propriamente dita. Para isso, transferimos o conteúdo da porta P0 ao acumulador com o comando MOV e em seguida somamos o conteúdo da porta P1 ao valor do acumulador. Para mostrar o resultado desta soma, transferimos o conteúdo do acumulador para a porta P2. Abaixo segue o código deste algoritmo:

```
MOV A, P0  
ADD A, P1  
MOV P2, A
```

Para que esta operação se repita inúmeras vezes, podemos colocar esta execução dentro de um laço de controle ou loop. Para isto, damos um nome (label) para o início desta execução e colocamos uma indicação ao fim desta execução para voltar à linha de execução desejada através do comando:

```
JMP LOOP
```

Após a escrita do código, deve-se acrescentar o comando a seguir para indicar que o programa chegou ao fim.

```
END
```

É recomendado colocar comentários onde for relevante, para facilitar os “debugs” e análises do código. Para isso, utiliza-se ponto e vírgula (;) antes do comentário. Comentários não são analisados pelo montador durante o processo de montagem do programa. O código resultante para o somador está apresentado a seguir:

---

```
;Soma de P0 com P1, resultado apresentado em P2
```

```
$MOD51          ;início  
ORG 0           ;para gravar a partir de 0  
  
MOV A, #0H      ;para zerar o acumulador A  
LOOP: MOV A, P0  ;para mover o valor de P0 para o acumulador A  
ADD A, P1       ;realiza a soma do valor do acumulador com P1  
MOV P2, A       ;apresenta o resultado em P2  
JMP LOOP        ;volta ao loop  
END
```

Escreva o código apresentado e salve o arquivo como “soma.asm”. Evite colocar o arquivo em pastas que tenham nomes muito extensos ou com espaço ou acentos e cedilhas, pois isto atrapalha o montador a encontrar o arquivo.

Para montar o seu programa, abra uma janela do DOS (prompt de comando) e navegue até a localização do seu montador (por exemplo, “C:\eduardo\workspace\Assembly\dos8051\”). De lá, chame montador com o comando:

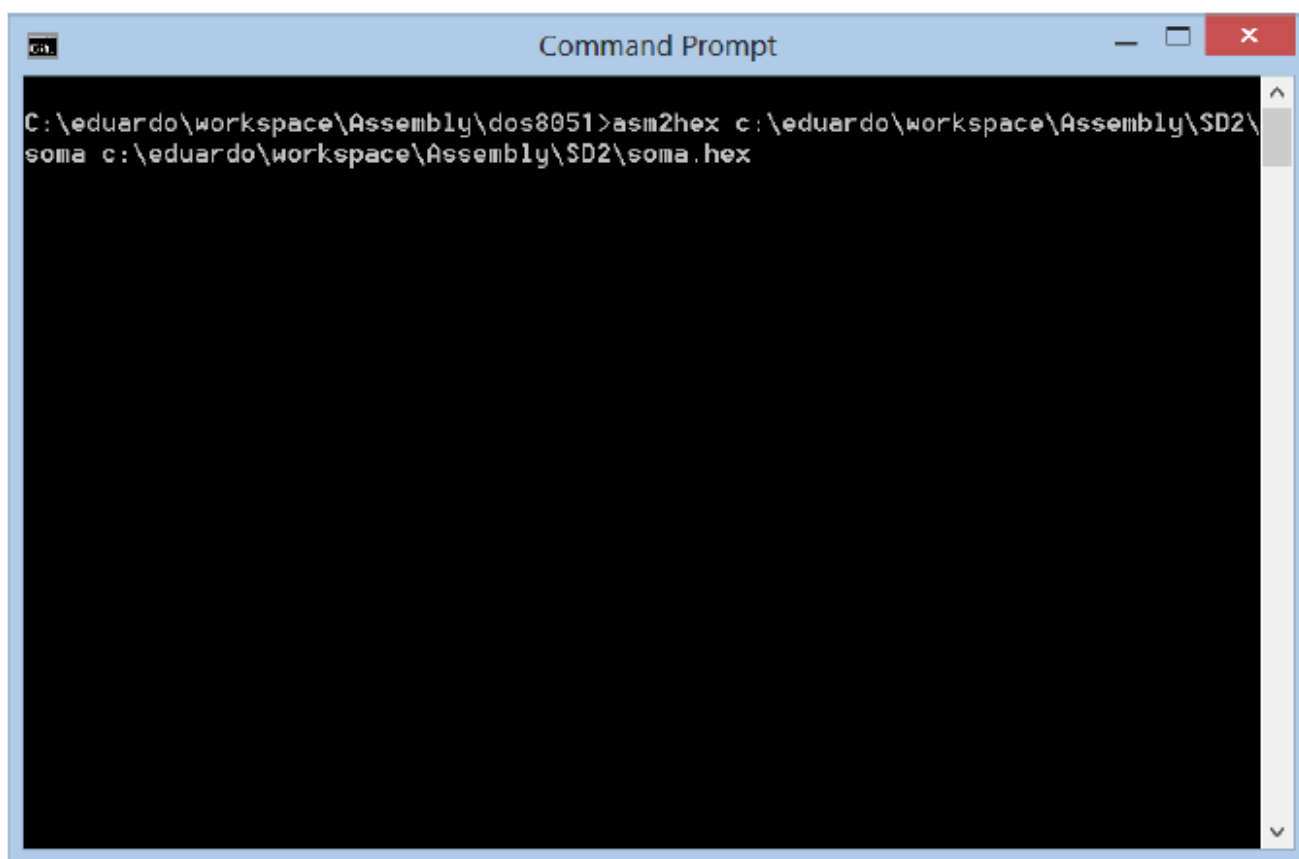
```
asm2hex <path_asm> <path_hex>
```

Onde <path\_asm> é o endereço completo do seu arquivo .asm (sem a extensão .asm, por exemplo: C:\eduardo\workspace\Assembly\SD2\soma) e <path\_hex> é o endereço completo onde você quer o resultado da montagem (o arquivo “.hex” – no caso, C:\eduardo\workspace\Assembly\SD2\soma.hex).

Assim, o comando pode ser:

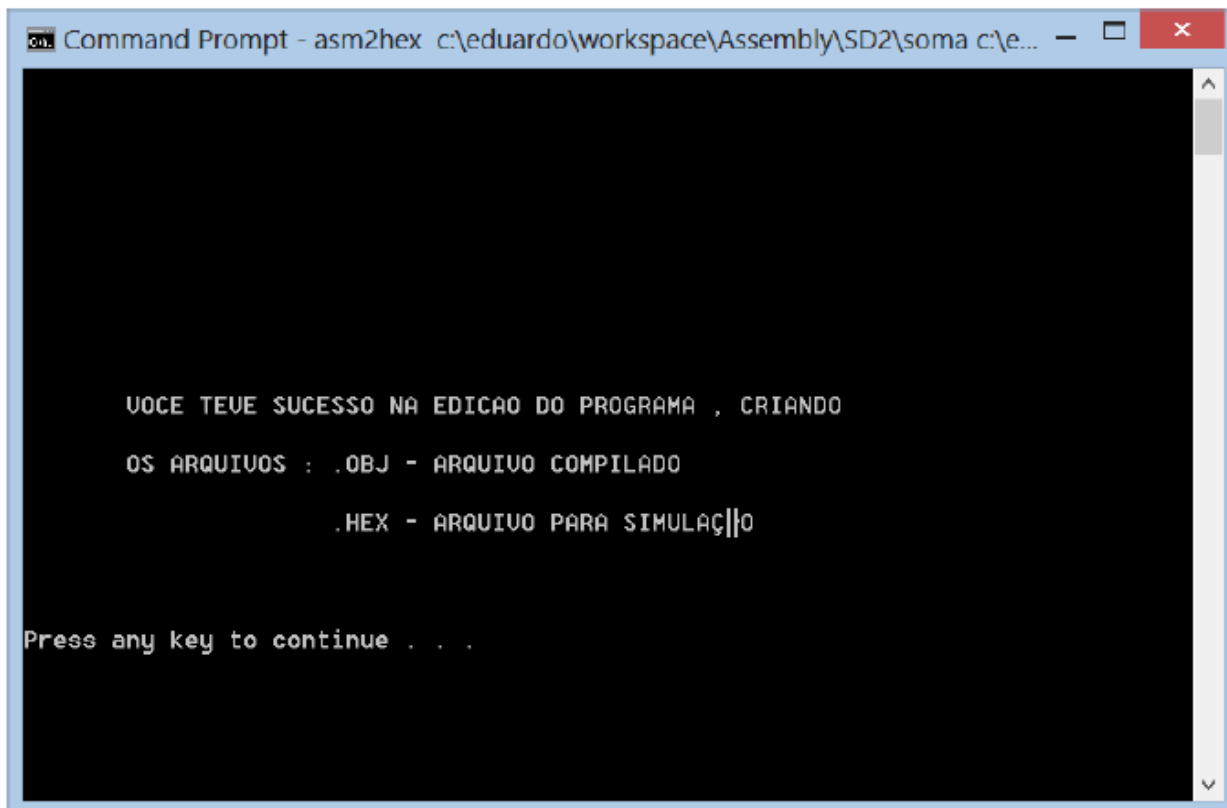
```
asm2hex  
C:\eduardo\workspace\Assembly\SD2\soma  
C:\eduardo\workspace\Assembly\SD2\soma.hex
```

Que irá montar o arquivo soma.asm para o arquivo com op codes soma.hex. Se houverem erros, eles serão descritos no arquivo de extensão .LST que é criado após a tentativa de compilação do arquivo.



```
Command Prompt  
C:\eduardo\workspace\Assembly\dos8051>asm2hex c:\eduardo\workspace\Assembly\SD2\soma c:\eduardo\workspace\Assembly\SD2\soma.hex
```

Figura 1 - Montador AM51



```

Command Prompt - asm2hex c:\eduardo\workspace\Assembly\SD2\soma c:\e...

VOCE TEVE SUCESSO NA EDICAO DO PROGRAMA , CRIANDO

OS ARQUIVOS : .OBJ - ARQUIVO COMPILADO

               .HEX - ARQUIVO PARA SIMULAÇ||O

Press any key to continue . . .
  
```

Figura 2 - Execução do montador a partir de uma janela de comando.

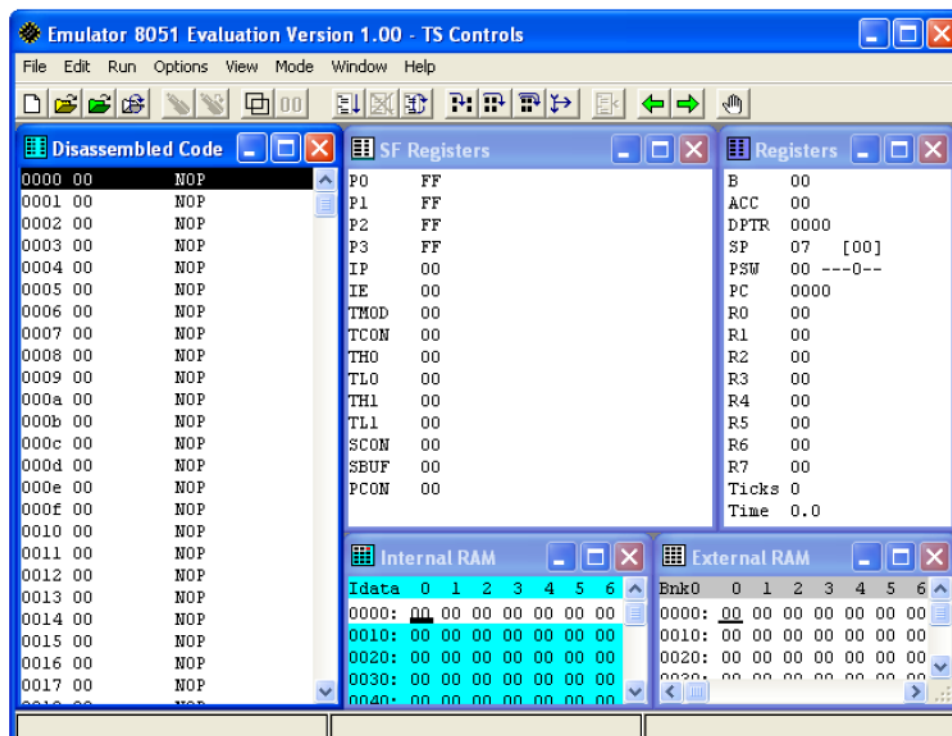


Figura 3 - Tela do simulador TS Controls 8051.

Após montagem com sucesso do programa desenvolvido, encontraremos um arquivo soma.hex no path escolhido. O arquivo .hex será usado para realizar a simulação. Caso tente abrir esse arquivo com um editor de texto, verificará que ele não é muito legível para nós. Abaixo segue o resultado da abertura deste arquivo com o editor de texto:

```
:0A0000007400E5802590F5A080F85B
:00000001FF
```

Para a simulação, abra o programa TS Controls 8051 através de duplo clique no arquivo Emul8051.exe. Deve-se encontrar a tela apresentada na figura 3. Caso alguma das janelas dos registradores ou do código não apareça, pode acrescentá-la pelo menu View (View Registers, View SF Registers, View Disassembled Code). Utilize a opção “Load Hex File” do menu File ou através do seu botão na barra de ferramentas para carregar o arquivo soma.hex gerado com o montador. A partir de então, será verificado que aparece o código na janela “Disassembled Code”. Para executar o código, pode se utilizar os botões “Run”, “Stop” e “Reset” da barra de ferramentas. Estes botões estão apresentados na figura 5. É importante, para verificar os resultados, utilizar o botão “Update Display While Running” apresentado na figura 6 durante a execução do código para verificar seu real funcionamento.

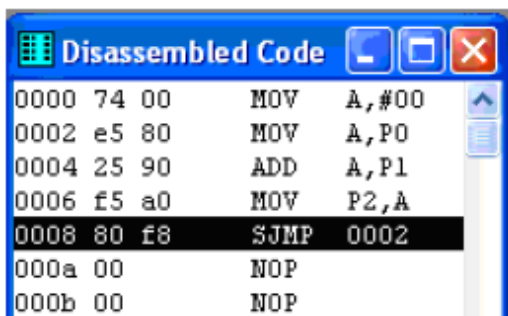


Figura 4 - Código do soma.hex carregado.



Figura 5 - Botões de execução

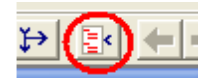


Figura 6 - Botão "Update Display while Running"

Outra forma de acompanhar a execução é através dos botões da execução passo a passo apresentados na figura 7. Um fator interessante da utilização destes botões é que a simulação através deles faz a atualização das informações automaticamente, não sendo necessário pressionar o botão de update.



Figura 7 - Botões para execução em passo-a-passo

Para inserir os valores desejados das portas P0 e P1, dê duplo clique sobre estes registradores na janela “SF Registers” e aparecerá uma janela de edição do valor semelhante à da figura 8. A edição dos valores das portas pode ser realizada durante a simulação, não sendo necessário encerrar a simulação para fazer as mudanças de valores. Após a edição, a simulação deve alterar sozinha o valor de P2, e o resultado será visto também na janela “SF Registers”, como apresentado na figura 9.

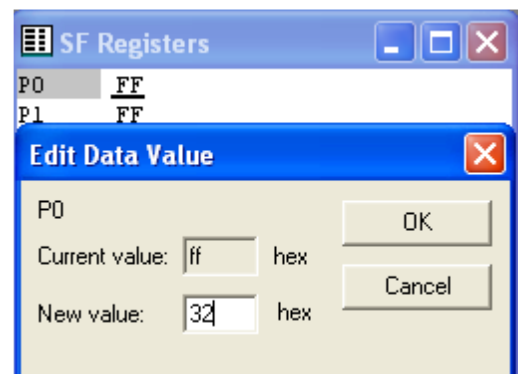


Figura 8 - Edição do valor da porta P0.

SF Registers	
P0	32
P1	1A
P2	4C
P3	FF
IP	00

Figura 9 - Registradores após a simulação.

### 3.2 Contador de pulsos

Um pulso pode ser definido como um sinal que sai de um nível baixo e vai para o nível alto durante um pequeno período de tempo, e então retorna ao nível baixo, como representado na figura 10. O contador de pulsos deve incrementar uma determinada variável sempre que for detectado um pulso em um determinado pino de uma porta. Para isso, usaremos como pino de entrada o pino '0' da porta 'P0' e armazenaremos o valor da contagem no registrador 'R0'.



Figura 10 – Pulso

Para detectar um pulso completo, pode-se esperar o sinal sair de '0' para '1' e esperar ele voltar de '1' para '0' novamente, assim, o algoritmo de detecção de pulsos pode ser verificado da seguinte forma:

```
ENQUANTO Entrada = 0
COMEÇA
```

```
Continua no laço
FIM
ENQUANTO Entrada = 1
COMEÇA
Continua no laço
FIM
Ocorreu pulso
```

Se colocarmos este código dentro de um laço para contar mais de um pulso e substituirmos a detecção por um incremento do registrador teremos o algoritmo completo da contagem de pulsos:

```
R0 recebe '0'
ENQUANTO Programa executando
COMEÇA
ENQUANTO Entrada = 0
COMEÇA
Continua no laço
FIM
```

```
ENQUANTO Entrada = 1
COMEÇA
Continua no laço
FIM
INCREMENTA R0
FIM
```

Traduzindo este algoritmo para assembly, teremos o seguinte código:

```
$MOD51 ; início
ORG 0 ; para gravar a partir de 0
MOV R0, #0H ; para zerar o registrador R0
JNB P0.0, CASO0 ; se o primeiro bit de P0 for 0,
; vai para a rotina 'CASO0'
JMP CASO1 ; senão vai para a rotina 'CASO1'
CASO0: JNB P0.0, CASO0 ; enquanto o primeiro bit de P0 não
; mudar para 1, ele permanece aqui.
CASO1: JB P0.0, CASO1 ; enquanto o primeiro bit de P0 não mudar
; de volta para 0, ele permanece aqui.
INC R0 ; completou um ciclo de 0 e 1,
; indicando que foi feito um pulso.
JMP CASO0 ; Volta para 'CASO0' para reiniciar
; o processo
END
```



Esse programa também pode ser simulado no TS Controls 8051 e o resultado vai ser verificado no registrador R0 na janela 'Registers'. Para verificar a mudança do registrador, altere o valor da porta P0 de forma que o primeiro bit mude de 0 para 1 (pode-se alternar entre os valores '00' e '01') e deve-se clicar no botão de atualização dos valores para ver a mudança no registrador R0.

### 3.3 Conclusão

Estes dois programas apresentam como são feitas operações simples, comparações e fluxos de controle em programação assembly. Se desejar, altere alguns dos comandos desses programas para ver como o assembly se comporta, ou para testar outros comandos e funcionalidades dessa linguagem.

## IV. PROJETOS A SEREM IMPLEMENTADOS

### 4.1 Contador de pulsos com controle

O grupo deve desenvolver um contador de pulsos que observa pulsos na porta P0.1 e utiliza a porta P0.0 como controle. A contagem deve ser realizada nos registradores R0 e R1. O programa deve funcionar da seguinte forma:

A cada pulso completo ocorrido na porta P0.1 (isto é, transição de 0 para 1 e 1 para 0), o registrador R0 deve ser incrementado. Quando ocorrer uma borda de subida na porta P0.0 (isto é, transição de 0 para 1), a contagem em R0 deve ser adicionada à contagem em R1, e o registrador R0 deve ser zerado.

- Sugestão: desenhe o diagrama de estados do seu contador antes de implementá-lo!

### 4.2 Calculadora

Faça uma calculadora que realize a soma, subtração, multiplicação e divisão de dois números. Utilize para entrada dos números as portas P0 e P1. Para definir qual será a execução a ser feita, utilize alguns bits da porta P2. A resposta deve ser colocada nos registradores R0 e R1 (se necessário). Podem ser utilizadas as instruções já existentes na arquitetura do 8051 para realizar as operações propriamente ditas. Cabe aos alunos pesquisarem o material de referência para conhecê-los. A forma dos números de entrada e saída (isto é, se apenas números positivos, negativos, de 8 bits, 16 bits, etc.) fica a critério do aluno (ver relatório).

Sugestão: desenhe o diagrama de estados dos bits de controle antes de implementar seu programa.

## V. INSTRUÇÕES PARA A REALIZAÇÃO DO EXPERIMENTO

### 5.1 Pré-relatório

Não haverá pré-relatório para este experimento.

### 5.2 Obtenção do Visto

O visto será dado se o grupo implementar com sucesso códigos assembly para os itens 4.1 e 4.2, isto é, realizar o contador de ciclos e a calculadora. O contador de ciclos valerá 4,5 pontos. A calculadora valerá 4,5 pontos. O ponto restante será dado como nota de pré-relatório após a obtenção de ambos os vistos. A obtenção do visto será mediante explicação do código implementado pela dupla e da simulação no software escolhido pelos mesmos para tal.

## VI. RELATÓRIO

O relatório é **individual**, e consiste em responder ao questionário a seguir.

- 1) Apresente o código utilizado para implementar o contador de ciclos com controle, **comentando as partes mais importantes**. (2 pontos)
- 2) Apresente o diagrama de estados do seu contador de ciclos (1 ponto).
- 3) Apresente o código utilizado para implementar a calculadora, **comentando as partes mais importantes**. (2 pontos)
- 4) Apresente o diagrama de estados do código da calculadora. (1 ponto)
- 5) Sobre a operação de **soma** implementada na calculadora, responda: Qual foi a representação numérica utilizada para os números de entrada? E de saída? Quais registradores (se algum) foi modificado pelo seu código? Como (e quando) eles foram modificados? (1 ponto)
- 6) Sobre a operação de **subtração** implementada na calculadora, responda: Qual foi a representação numérica utilizada para os números de entrada? E de saída? Quais registradores (se algum) foi modificado pelo seu código? Como (e quando) eles foram modificados? (1 ponto)
- 7) Sobre a operação de **multiplicação** implementada na calculadora, responda: Qual foi a representação numérica utilizada para os números de entrada? E de saída? Quais registradores (se algum) foi modificado pelo seu código? Como (e quando) eles foram modificados? (1 ponto)

- 8) Sobre a operação de **divisão** implementada na calculadora, responda: Qual foi a representação numérica utilizada para os números de entrada? E de saída? Quais registradores (se algum) foi modificado pelo seu código? Como (e quando) eles foram modificados? (1 ponto)

### • AVISOS IMPORTANTES

→ **ALUNOS QUE APRESENTAREM CÓDIGOS COPIADOS DE OUTROS GRUPOS DO MESMO SEMESTRE OU DE SEMESTRES ANTERIORES SERÃO PENALIZADOS COM NOTA ZERO NO EXPERIMENTO!**

→ **A OBTENÇÃO DO VISTO SERÁ MEDIANTE EXPLICAÇÃO POR PARTE DO ALUNO DO CÓDIGO IMPLEMENTADO, NÃO APENAS A SIMULAÇÃO! LOGO, NÃO OBTENHA O VISTO UTILIZANDO CÓDIGOS DE OUTROS GRUPOS!**

→ **O RELATÓRIO É INDIVIDUAL, CADA ALUNO DA DUPLA DEVE APRESENTAR SEU ENTENDIMENTO PRÓPRIO DO EXPERIMENTO! EVITE ENTREGAR RESPOSTAS E CÓDIGOS COM COMENTÁRIOS IDÊNTICOS AO DA SUA DUPLA!**