



ENGENHARIA DA COMPUTAÇÃO

Organização e Organização Básica de Computadores

Relatório de Laboratório

CPU - Visual Studio Code

Henrique Campana de Moraes – RA: 180200
Paulo Marcos Araújo da Rocha – RA: 200872
Victor Soares Nunes Pires de Oliveira – RA:223585
Vitor Araújo Agostino – RA: 190886

Prof. Marcos F.Jardini

Sorocaba / SP

SUMÁRIO

1. Objetivo	3
2. Introdução.....	3
3. Materiais utilizados	3
4. Procedimentos	3
5. Análise de dados	5
6. Conclusão	6
Referências.....	6

1. Objetivo

Os objetivos da atividade consistem no aprendizado de dispositivos de lógica programável e dessa forma familiarizar-se com uso do ambiente de simulação aplicada “Visual Studio Code”, desenvolvendo software em Assembly.

2. Aplicação

Este procedimento é aplicável a todos os colaboradores do curso de mecatrônica que participam.

3. Responsabilidade

É de responsabilidade dos colaboradores dos laboratórios de manter este procedimento atualizado e seguir todas as orientações para a arquitetura

4. Introdução

O visual Studio Code (figura 1.1) é um editor de código desenvolvido pela Microsoft para edição de código fonte. Nesse sentido, o mesmo é customizado permitindo que os usuários possam utilizar inúmeras linguagens de programação e um conjunto de recursos do software. Com isso através do software aplicamos o conceito da linguagem Assembly essa mesma é uma linguagem de máquina que é utilizada para comunicação ou transferência de dados.

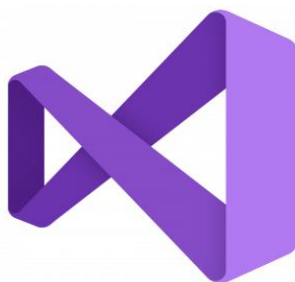


Image 1.1 Fonte: <https://visualstudio.microsoft.com/pt-br/>

5. Material Utilizado

Computador com a aplicação Visual Studio Code.

6. Procedimentos

- **Utilização do Software**

Acessar a aba de criar projeto (figura 1.2) selecionando a linguagem C++, em seguida adicionar o nome do projeto (figura 1.3) e clicar em criar projeto.

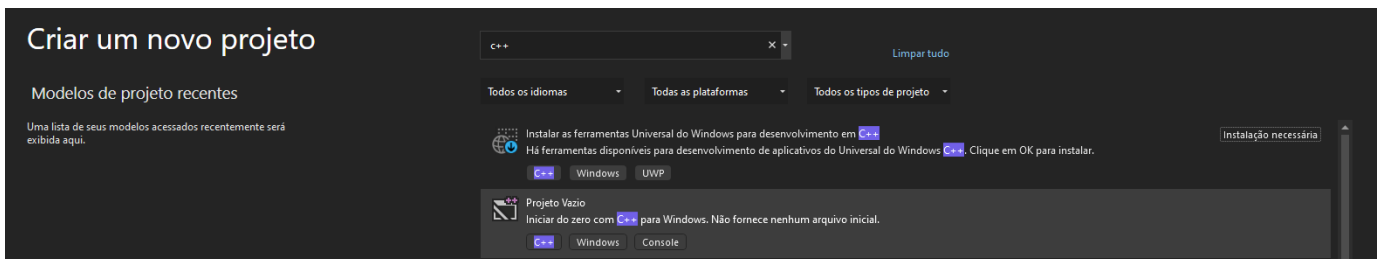


Figura 1.2 Fonte: Software Visual Studio Code

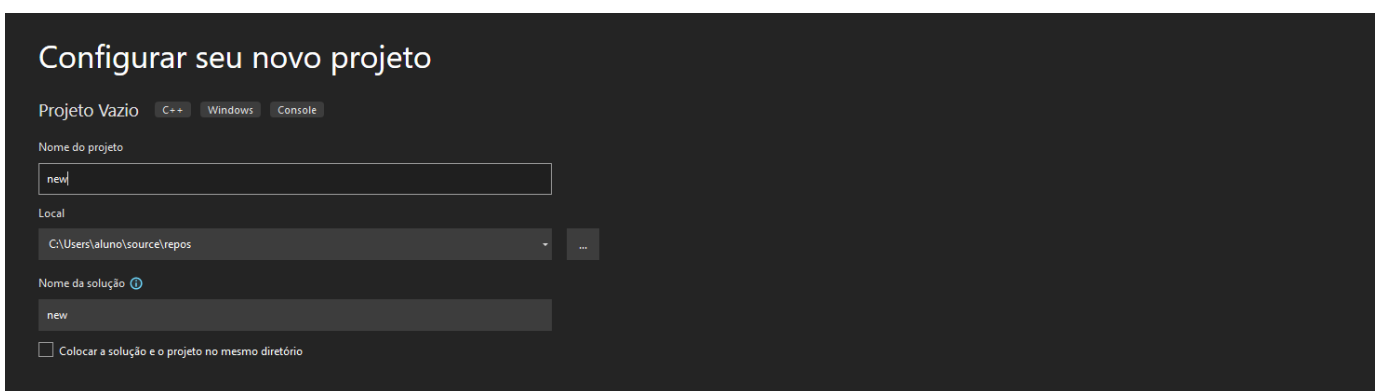
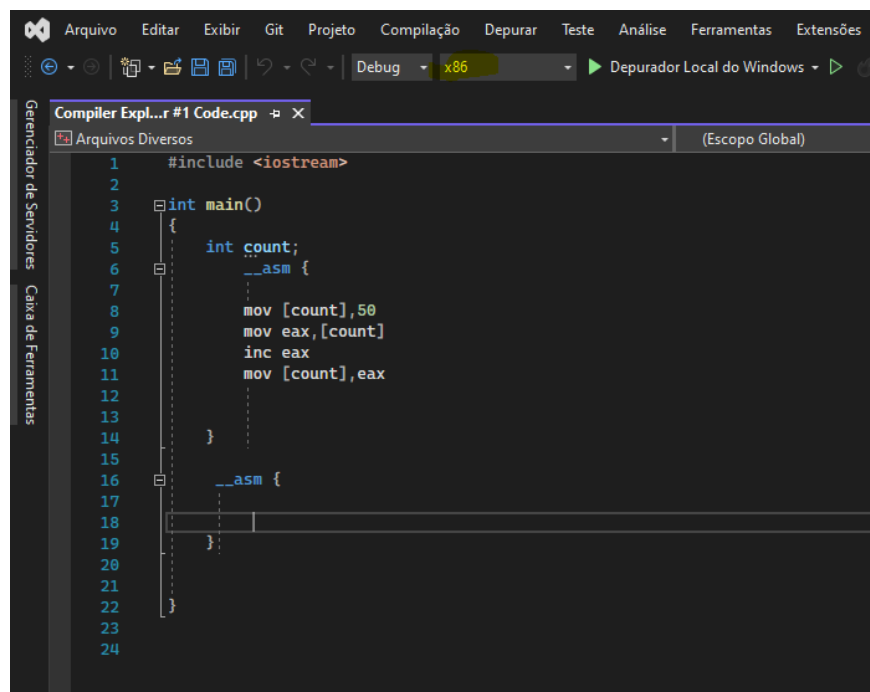


Figura 1.3 Fonte: Software Visual Studio Code

- **Motagem do Circuito**

O circuito a seguir foi construído utilizando conceitos da linguagem assembly conectada com a utilização de funções elaboradas em C++, conforme demonstrado na (figura 1.4) aonde utilizamos a importação da função `int main ()`. Nesse sentido, seguimos com o detalhamento do código e sua aplicabilidade no projeto.



The image shows a screenshot of the Visual Studio IDE. The main window displays a C++ source file named 'Compiler Expl...r #1 Code.cpp'. The code is as follows:

```
1  #include <iostream>
2
3  int main()
4  {
5      int count;
6      __asm {
7
8          mov [count],50
9          mov eax,[count]
10         inc eax
11         mov [count],eax
12
13     }
14
15     __asm {
16
17     }
18
19 }
20
21
22
23
24
```

The interface includes a menu bar at the top with options like 'Arquivo', 'Editar', 'Exibir', 'Git', 'Projeto', 'Compilação', 'Depurar', 'Teste', 'Análise', 'Ferramentas', and 'Extensões'. Below the menu is a toolbar with icons for file operations and debugging. The left sidebar shows the 'Gerenciador de Servidores' and 'Caixa de Ferramentas' panels. The status bar at the bottom indicates 'Debug' mode and 'x86' architecture.

Image 1.4 Fonte: Software Visual Studio

- Programação

Diante da explicação acima, seguimos com a declaração da biblioteca **<stdio.h>** e em seguida com a definição das variáveis segundos e minutos. Por conseguinte, dividimos o software em 4 etapas: início, que tem como função principal chamar a função **call [Print]** para exibir os valores e incrementação de dados de segundos, compara, que realiza a comparação, minuto, onde o mesmo é utilizado para adicionar minutos e a função por final a função reset, utilizada para resetar os dados.

```
1  #include <stdio.h>
2
3  int segundos = 0;
4  int minutos = 0;
5
6  void print()
7  {
8      printf(_Format: "%d:%d\n", minutos, segundos);
9  }
10
11 int main()
12 {
13     __asm {
14         inicio:
15             call [print]
16             mov eax, [segundos] ; carrega valor de segundos
17             inc eax             ; incrementa de 1
18             mov [segundos], eax ; salva valor em segundos
19             cmp eax, 59         ; compara com o limite de segundos : eax - 59
20             jle inicio          ; se o resultado anterior for negativo volta pro começo
21             jmp compara        ; se o resultado anterior for positivo avança para a comparação
22
23         compara:
24             mov eax, [minutos] ; carrega o valor de minutos
25             cmp eax, 59         ; compara com o limite de minutos : eax - 59
26             jl minuto          ; se o resultado anterior for negativo avança para a contagem de minutos
27             jmp reset          ; se o resultado anterior for positivo avança para o reset do tempo
28
29         minuto:
30             mov eax, [minutos] ; carrega valor de minutos
31             inc eax             ; incrementa de 1
32             mov [minutos], eax ; salva valor em minutos
33             mov eax, 0          ; carrega o valor de 0 para eax
34             mov [segundos], eax ; volta o contador de segundos para 0
35             jmp inicio          ; volta para o início
36
37         reset:
38             mov eax, 0          ; carrega eax com o valor 0
39             mov [segundos], eax ; volta o contador de segundos para 0
40             mov [minutos], eax ; volta o contador de minutos para 0
41             jmp inicio          ; volta para o início
42     }
43 }
```

Image 1.5 Fonte: Software Visual Studio

- Compilação

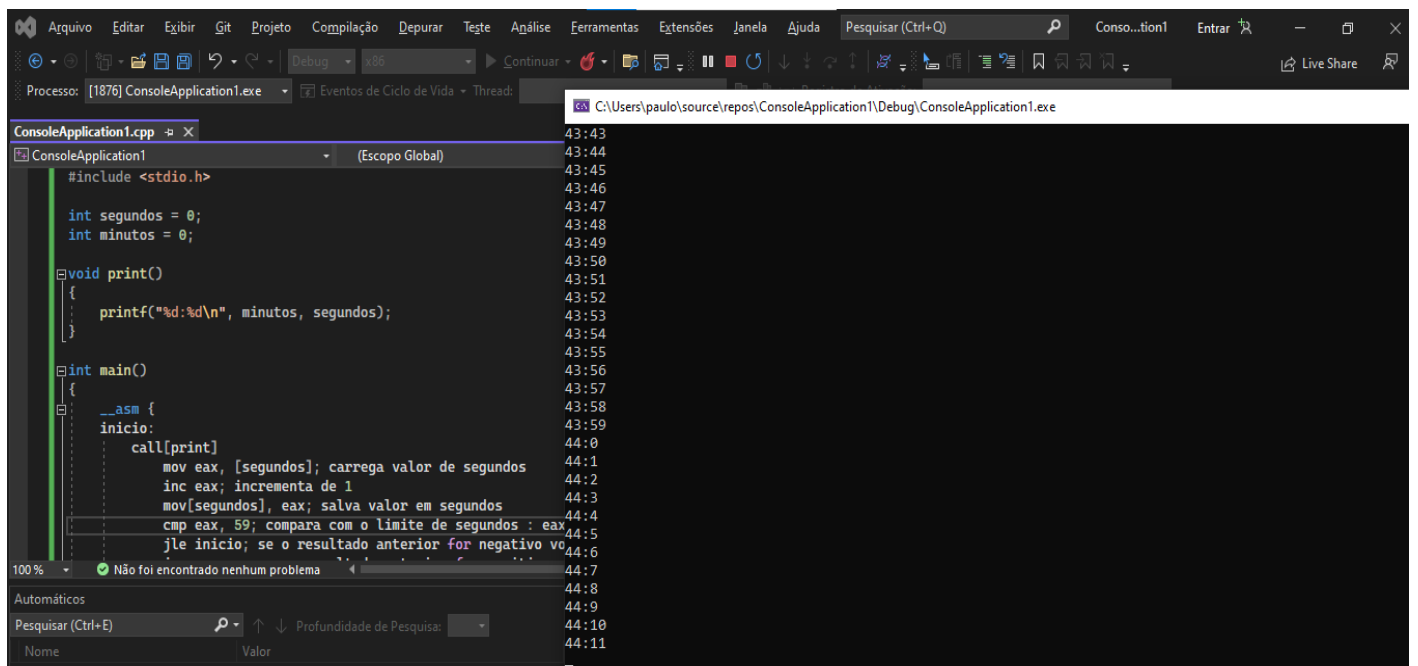


Image 1.6 Fonte: Software Visual Studio

7.Conclusão

Baseado em todos os dados, conclui-se que os objetivos foram atingidos e todas as atividades propostas foram realizadas por completo, realizando os procedimentos de contagem de tempo de 0 a 59 através de um loop organizado em linguagem Assembly.

10.Referências

Visual Studio Code - Wikipedia < https://pt.wikipedia.org/wiki/Ou_exclusivo>. Acesso em 19 Maio. 2022.

C++ - Wikipedia < https://pt.wikipedia.org/wiki/Porta_AND>. Acesso em 09 Maio. 2022.