PABLO JEAN ROZARIO

PLANO DE AULA
MICROCONTROLADORES - TURMA 1

- **Aula** : 1
- **Turma** : 1
- **Duração** : 4 Horas
- **Tema** : Apresentação e Conceitos

Cambé - PR
Dezembro de 2021
Revisão 1

SUMÁRIO

1 Objetivos **5**
1.1 Objetivo Geral 5
1.2 Objetivos Específicos 5

2 Metodologia **5**
2.1 Introdução 5
2.2 Microcontrolador ou Microprocessador 9
2.3 8bits ou 32bits 10
2.4 Arquitetura de Microcontroladores 11
2.4.1 Von Neumann 11
2.4.2 Harvard 11
2.4.3 Diferenças 12
2.5 Registradores 13
2.5.1 Program Counter e a Stack 15
2.6 Set de Instruções 17
2.7 Kit de Desenvolvimento NUCLEO-G0B1RE 18
2.8 Sistema de Clocks 19
2.9 IDE de Desenvolvimento - STM32CubeIDE 19

3 Recursos Didáticos **20**

4 Forma de Avaliação **20**

5 Histórico de Revisão **21**

A Anexos **22**
A.1 Lista de Exercícios - Resolvida 22

LISTA DE FIGURAS

2.1 Exemplos de aplicação de sistemas embarcados. 7
2.2 Diagrama de blocos de um microcontrolador. 8
2.3 Diagrama do barramento de um microcontrolador. 8
2.4 Esquemático de uma arquitetura Von Neumann e Harvard, respectivamente. 12
2.5 Esquema de três SFRs. 14
2.6 Esquemático de uma arquitetura Von Neumann e Harvard, respectivamente. 15
2.7 Exemplo de operação de uma stack. 16
2.8 NUCLEO-G0B1RE, kit de desenvolvimento da ST Microeletronics. . . . 18
2.9 Logo do STM32CubeIDE. 20

1 OBJETIVOS

1.1 OBJETIVO GERAL

Apresentar aos estudantes os conceitos básicos do microcontrolador e o kit que será utilizado durante o Módulo.

1.2 OBJETIVOS ESPECÍFICOS

- Compreender a diferença entre as principais arquiteturas de microcontroladores e a aplicação em Sistemas Embarcados;
- conhecer alguns modelos de microcontroladores de diferentes fabricantes;
- apresentar o kit da STMicroeletronics *NUCLEO-G0B1RE* que será trabalhado na disciplina.

2 METODOLOGIA

2.1 INTRODUÇÃO

No inicio da disciplina, me apresentarei, de forma bem curta e rápida, e em sequência resumirei o que pretendemos trabalhar durante o curso. Sendo utilizado como base a tabela 2.1.

Tabela 2.1: Cronograma planejado para o curso de Microcontroladores.

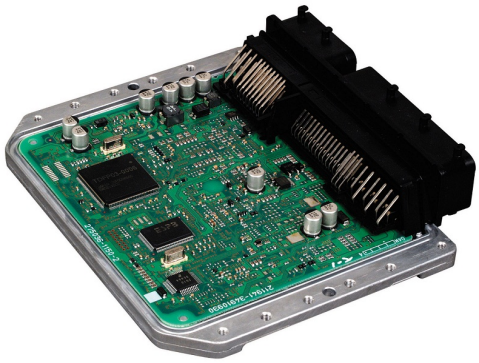
Aula	Descrição	Aula	Descrição
1	Apresentação de Conceitos Apresentação do kit STM32	**7**	Atividades
2	Tipos especiais de (u)int Registradores e Clock do uC Entradas e Saídas IOs	**8**	Comunicação Serial I²C
3	Estruturas em C Interrupções em IOs	**9**	Timers e RTC
4	Conversor A/D e Interrupções	**10**	PWM
5	Comunicação Serial UART	**11**	Desenvolvimento de Projetos
6	Comunicação Serial SPI	**12**	

Apresentarei algumas questões questões aos alunos, tais como, "O que é o microcontrolador?"e "Qual a importância dos microcontroladores em nossas vidas". Após as resposta da turma, será respondido que, respectivamente:

"O microcontrolador é um componente eletrônico composto de um processador e periféricos em um único encapsulamento, capaz de executar um programa previamente gravado".

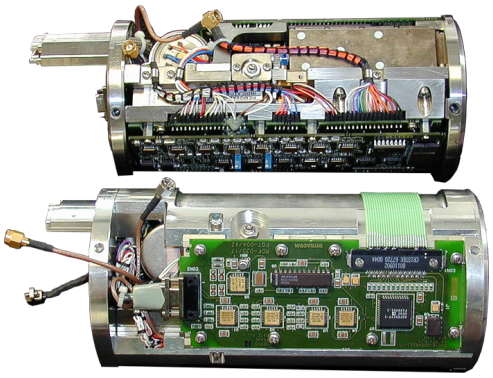
"A importância dos microcontroladores em nossas vidas está na automatização de diversos equipamentos que utilizamos e ainda de dispositivos que carregamos no dia a dia que são dotados de certa autonomia e inteligência, tais como *smartwatch's* e *smartphones*."

Figura 2.1: Exemplos de aplicação de sistemas embarcados.

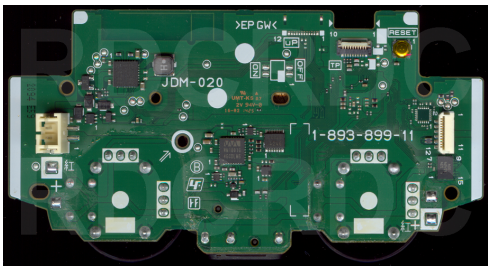


(a) Central Eletrônica (ECU) de um carro.

(b) Fechadura eletrônica.



(c) Módulo de Controle de um Míssil.



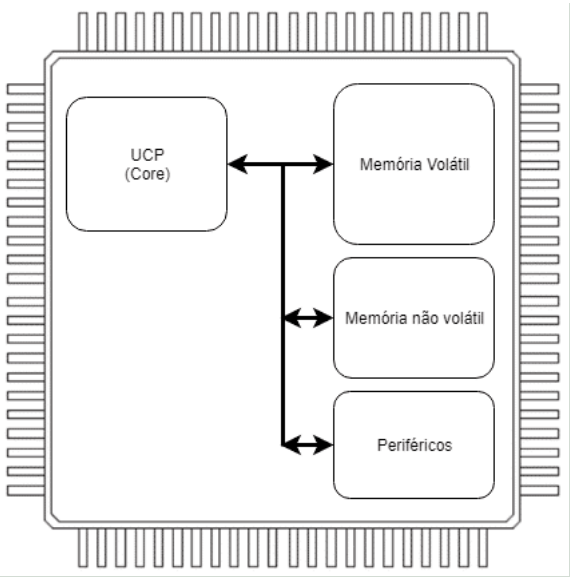
(d) Placa eletrônica de um controle de PS4.

Fonte: Google imagens.

Com estas resposta, será apresentado exemplos de produtos e aplicações dos microcontroladores, desde eletrodomésticos e brinquedos, até sistemas bélicos. Pois é importante que o estudante tenha noções de onde pode ser aplicado estes dispositivos.

Será apresentado também a figura , com finalidade de exemplificar como é a estrutura interna de um microcontrolador.

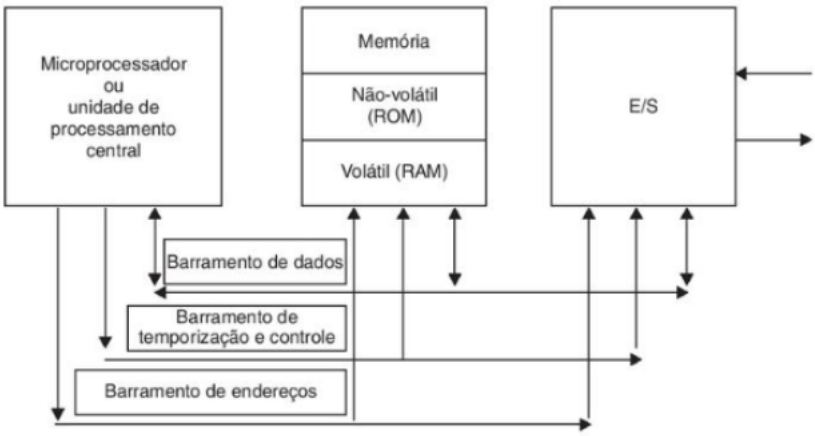
Figura 2.2: Diagrama de blocos de um microcontrolador.



Fonte: Google imagens.

Na figura 2.3 tem-se por objetivo exibir de forma um pouco mais detalhada a estrutura de um microcontrolador, buscando fornecer aos estudantes uma compreensão de como os dados fluem internamente.

Figura 2.3: Diagrama do barramento de um microcontrolador.



Fonte: (GIMENEZ, 2002).

Deverá ser explicado que o **Barramento de temporização e controle** é responsável por definir o endereço do dispositivo de IO no barramento de endereços por um período de tempo definido, além do sentido (leitura ou escrita). O **Barramento de endereços** tem por objetivo informar ao periférico o endereço da informação que de-

seja alterar ou ler. E por fim, o **Barramento de dados** é por onde é disponibilizada a informação.

Dica Didática : Utilizar como exemplo, uma conversa com os alunos, onde falo com um e específicos pelo seu nome, e solicito,por exemplo, quantas bolinhas tem na caixa X, e da mesma forma, para outro aluno, digo para guardar Y bolinhas na caixa Z, como forma de ilustrar o barramento.

2.2 MICROCONTROLADOR OU MICROPROCESSADOR

Durante a aula, será apresentada uma tabela 2.2 para apresentar os grandes pontos de diferença entre microcontroladores e microprocessadores.

Tabela 2.2: Escolha entre microcontrolador e microprocessador.

Característica	Microprocessadores	Microcontrolador
Periféricos	Necessita de periféricos externos	Periféricos integrados no chip
Memória	Permite vários formatos de dados	Poucos tipos de dados (8, 16 ou 32 bits)
Processamento	ALU Complexa e possui coprocessador	ALU limitada e ausência de coprocessamento
Custo	Custo elevado	Baixo custo, a depender da plataforma
Consumo	Alto consumo de enegeria	Possui métodos para economia de energia

É importante ressaltar a importância de ter os parâmetros do projetos, não apenas para decidir entre Microcontrolador ou Microprocessador, mas também para definir a escolha do modelo do microcontrolador.

Como existem vários fabricantes de microcontroladores e ainda, várias linhas dentro dos fabricantes, é importante saber definir qual microcontrolador será escolhido. Como forma de ilustrar isto, a tabela 2.3 irá indicar 4 exemplos de microcontroladores de diferentes capacidades.

Tabela 2.3: Comparativo entre microcontroladores de várias linhas.

	PIC16F1824	**MSP430FR2433**	**STM32G0B1RE**	**CC2642R**
Fabricante	Microchip	Texas	ST	Texas
Core	PIC16	MSP430	ARM M0+	ARM M4F
Bits	8bits	16bits	32bits	32bits
Flash	7KB	15.5K	512KB	352KB
RAM	256B	4K	144KB	80KB
I/Os	12	19	60	31
ADC	10bits	10bits	12bits	12bits
SPI	X	X	X	X
I2C	X	X	X	X
UART	X	X	X	X
EEPROM	X	-	-	-
Extra	-	FRAM	USB OTG	Bluetooth

Será importante apontar as diferenças entre as interfaces, principalmente com respeito as capacidades de memórias, que é um dos parâmetros que mais saltam aos olhos. E além destes pontos que estão na tabela, esclarecer que existem mais parâmetros, como quantidade e capacidade dos periféricos, clock máximo, recursos especiais e consumo energético.

2.3 8BITS OU 32BITS

Irei explicar o caso de como quase sempre se desenvolve em C, as diferenças ficam muito transparentes, pois o C irá abstrair muitas etapas das instruções dos microcontroladores, mas quando se trabalha com assembly, fica evidente alguns aspectos, como por exemplo, somar números de 32bits, em nível de assembly é bem simples em um microcontrolador de 32 bits, mas árduo em 8bits.

Informar também aos estudantes, que as diferenças estão mais no tamanho dos registradores, e do barramento, que será do tamanho da arquitetura do microcontrolador. Mas é importante ressaltar que, em tese um microcontrolador de 8 bits poderia endereçar até no máximo 256 bytes de ram/flash, no entanto, alguns microcontroladores possuem uma estratégia de paginação, permitindo ter um barramentos de 16bits para a flash, permitindo endereçar até 64KB.

2.4 ARQUITETURA DE MICROCONTROLADORES

É importante desde o começo os estudantes ja terem em mente que existem duas arquiteturas de microcontroladores, para tal, será explicado de forma mais básica sobre cada uma das duas.

Em (GEEKS, 2021) é explanado um pouco sobre cada uma das arquiteturas, podendo sugerir aos alunos, acesso a esta referencia, para buscar esta e mais informações.

2.4.1 Von Neumann

A arquitetura Von Neumann foi desenvolvida pelo Matemático e Físico John Von Neumann em 1945, é uma arquitetura que tem como base o conceito de armazenar os dados e as instruções em uma mesma memória. ou seja, a memória RAM e a Flash estão no mesmo barramento.

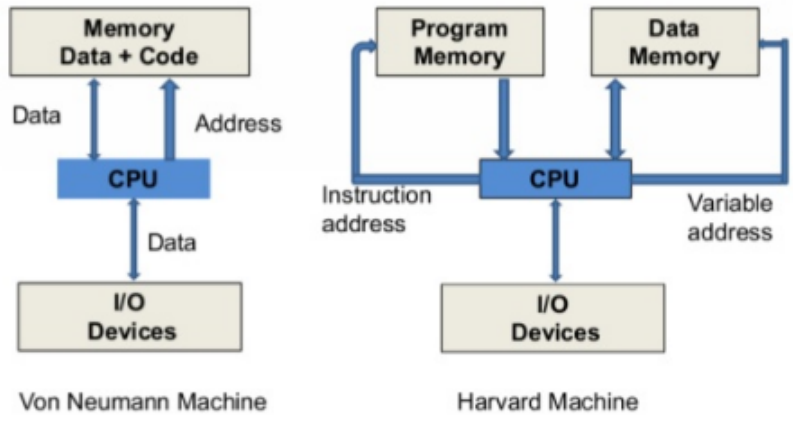
2.4.2 Harvard

Diferente da Von Neumann, na arquitetura Harvard, o barramento da memória de dados e da memória de programa é separada, ou seja, a RAM é separada da Flash.

Foi desenvolvida para superar o gargalo causado pela arquitetura Von Neumann na época.

Aprestarei a figura 2.4 onde é exibido o esquemático básico de uma arquitetura Von Neumann e uma Harvard.

Figura 2.4: Esquemático de uma arquitetura Von Neumann e Harvard, respectivamente.



Fonte: Google Imagens.

Pela figura, fica bem evidenciado a diferença das arquiteturas, onde o barramento de dados é compartilhado no Von Neumann, e separado na Harvard.

2.4.3 Diferenças

Não existe uma arquitetura melhor que a outra, ambas possuem vantagens e desvantagens, a depender da aplicação em que são empregados os microcontroladores. Em resumo, as diferenças são exibidas na tabela 2.4, que foi adaptada de (GEEKS, 2021).

Tabela 2.4: Comparativo entre microcontroladores de várias linhas.

Von Neumann	**Harvard**
O mesmo endereço físico é utilizado para memória de programa e memória de dados	Diferentes endereços físicos são utilizados pelas memórias
O barramento da memória de programa e da memória de dados é compartilhado	O barramentos das memórias é isolado
É necessário pelo menos dois ciclos de clock para executar uma única instrução	Uma instrução pode ser executada em um ciclo de clock
Possui custo menor	Possui um custo mais elevado se comparado a Von Neumann
Não é possível acessar uma instrução e ler/escrever ao mesmo tempo	E possível ler/escrever ao mesmo tempo que acessa uma instrução
Muito utilizado em computadores e microcontroladores ARM	Utilizado em microcontroladores e processamento de sinais

2.5 REGISTRADORES

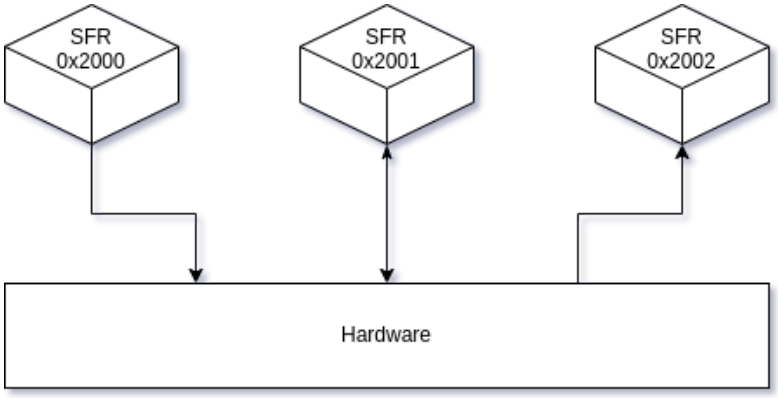
Segundo (WILDER, 2015) registradores nada mais são do que locais de memória em que é possível realizar as operação de Escrita e/ou Leitura, como por exemplo a memória RAM, que nada mais é do vários registradores que podem ser lidos e escritos, que tem como característica serem velozes e perderem seus dados ao serem desenergizados.

Mas os microcontroladores possuem os SFRs, ou *Special Function register*, que são, de forma similar a RAM, locais de memória que podem controlar diretamente o hardware do componente, ou o próprio hardware do microcontrolador pode controla-los, para indicar status.

Cada bit do SFR é designado a uma função, este podendo ser um *Control bit* ou um *Flag bit*. Os *Control bits* são como se fossem chaves que acionam determinadas funções do hardware, ou configuram, alguns necessitando apenas de um bit, como um **enable** para uma interface serial, ou vários bits, como definir um *prescaler*. Um *Flag bit* é controlado diretamente pelo hardware, e seu objetivo é indicar alguma status ou evento que ocorreu no microcontrolador, como, por exemplo, indicar que uma interrupção em um pino de entrada.

Na figura 2.5 será exibido aos estudantes um diagrama para exemplificar três SFRs de um determinado microcontrolador.

Figura 2.5: Esquema de três SFRs.

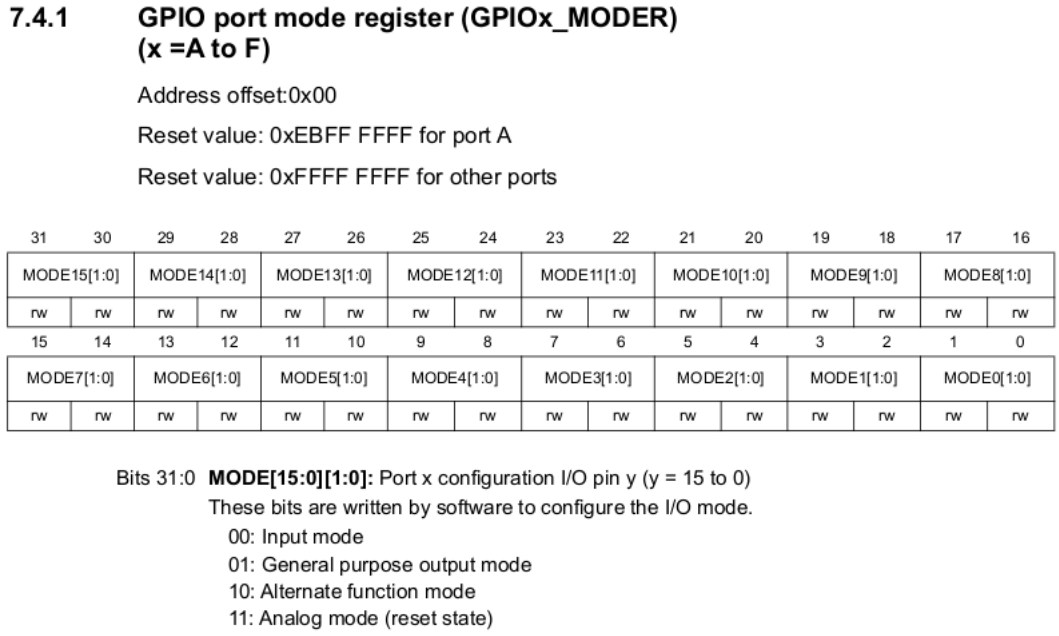


Fonte: Autoria própria.

Todas as informações dos SFRs estão nos datasheets dos componentes, explicitando cada bit de cada registrador, se podem ser lidos e/ou escritos.

Na figura 2.6 tem-se um exemplo de registrador que configura o modo de operação de uma GPIO de um microcontrolador, neste exemplo estamos tomando o registrador do STM32G0B1RE, que está disponível em (ST MICROELETRONICS, 2020).

Figura 2.6: Esquemático de uma arquitetura Von Neumann e Harvard, respectivamente.



7.4.1 GPIO port mode register (GPIOx_MODER) (x =A to F)

Address offset:0x00

Reset value: 0xEBFF FFFF for port A

Reset value: 0xFFFF FFFF for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODE[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Fonte: (ST MICROELETRONICS, 2020)

Nesta figura, será relevante apontar que o documento especifica cada bit do registrador, se suporta leitura e/ou escrita, descrição dos elementos do registrador e o valor do reset do registrador.

2.5.1 Program Counter e a Stack

Ainda tomando como base (WILDER, 2015), é importante citar que o contador de programa (*Program Conter* ou simplesmente PC) é um ponteiro que diz para a CPU onde se encontra a próxima instrução a ser executada.

No momento do POR (*Power On Reset*) o contador de programa inicia no endereço $0x0000$, e é incrementado automaticamente ao carregar a instrução, sendo então $0x0001$.

O PC é incrementado automaticamente em 1 a cada instrução executada, até o momento em que o mesmo é modificado pelo programa (causando um *jump*) ou em chamadas de funções, que também realizam um salto no programa, mas que faz uso da *Stack*.

Neste ponto, é interessante apresentar o problema antes de falar da *stack*, após apresentar e questionar sobre, partir para explanação.

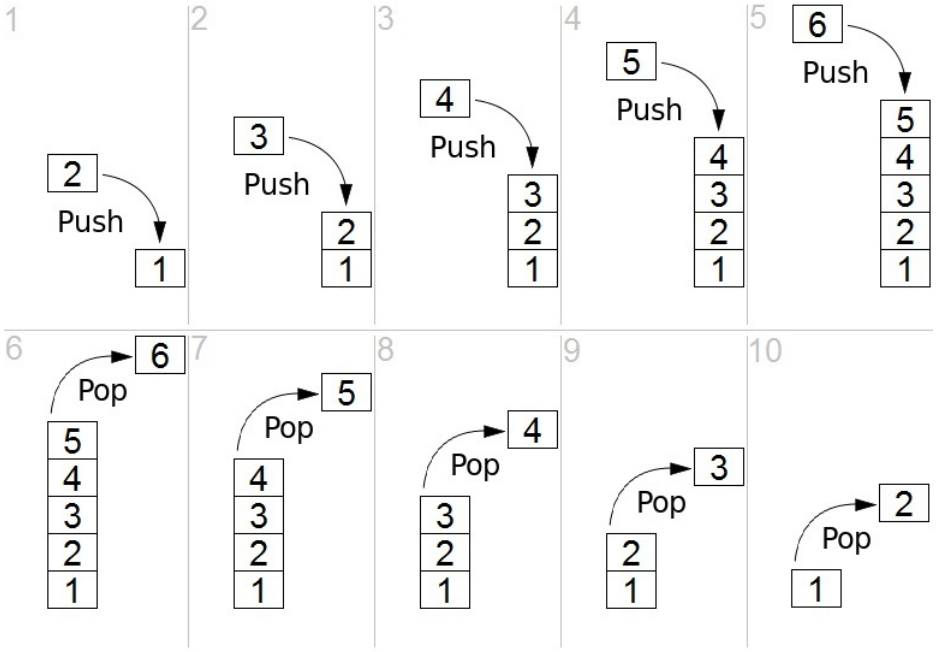
A *stack* é uma pilha que é utilizada pelas chamadas de subrotinas, realizando o armazenamento do próximo endereço da chamada da subrotina. A mesma função na lógica de LIFO (*Last In, First Out*) para armazenar os valores para retorno.

Durante a operação, ocorre os seguintes passos:

1. É feito o PUSH na stack do valor do PC na chamada de subrotina ou interrupção;
2. o PC é modificado com o endereço de onde inicia a execução da subrotina ou interrupção;
3. o programa então pula e executa a subrotina;
4. quando ocorre a chamada de *return*, é realizado o POP na pilha e o valor atribuído ao PC;
5. o PC retorna a próxima instrução antes da chamada da função ou da ocorrência da interrupção.

Na figura 2.7 é exemplificado o funcionamento de uma Stack.

Figura 2.7: Exemplo de operação de uma stack.



Fonte: Google imagens.

Um microcontrolador pode implementar a stack de duas formas: Por software e por Hardware. Por software, como é o caso de linhas como STM8, STM32 e entre outras,

a stack é armazenada na memória RAM e pode ser definida pelo desenvolvedor no programa, tem por vantagem ser be flexível, mas parte da memória RAM é perdida.

Já por hardware, a stack é implementada através de registradores dedicados à esta funcionalidade, como é o caso da linha PIC da Microchip, a grande vantagem é que não é necessário reservar memória RAM para a funcionalidade, mas o tamanho da *stack* é limitado ao que o hardware oferece.

2.6 SET DE INSTRUÇÕES

É importante falar aos estudantes que o conjunto de instruções ou *instruction set* engloba todas as instruções que são compreendidas pelo microcontrolador, que nada mais são do que as operações que o microcontrolador pode fazer e que possui implementado em seu hardware.

Quando um programa é compilado e gravado no microcontrolador, o binário gerado nada mais é do que a sequência de instruções geradas para atingir a finalidade desejada. Abaixo tem-se um exemplo de instruções para o *core* PIC18.

```
MOVLW 10H         ; Set 0x10 to the WREG
MOVF  20H, 1, 0   ; Move 0x10 to the address 0x20
MOVLW 5H          ; Set 0x5 to the WREG
ADDWF 20H, 1, 0   ; Sum WREG with value on 0x20
                  ; (0x10+0x5) and stores in 0x20
```

Cada microcontrolador possui seu *set* de instruções específicos da *CPU*, estes sets são divididos em dois principais grupos, **RISC** (*Reduced Instruction Set Computer*) e **CISC** (*Complex Instruction Set*).

- **RISC** : Arquitetura com pequeno conjunto de instruções mais simples, sendo mais baratos de produzir e permitem um *clock* mais alto pela simplicidade dos circuitos internos. Geralmente é associado a arquitetura Harvard.
- **CISC** : Arquitetura com uma grande gama de instruções, sendo algumas instruções feita por "microprogramas"gravados na cpu, possuindo ainda unidades de controle poderosas. Geralmente é associado a arquitetura Von Neumann.

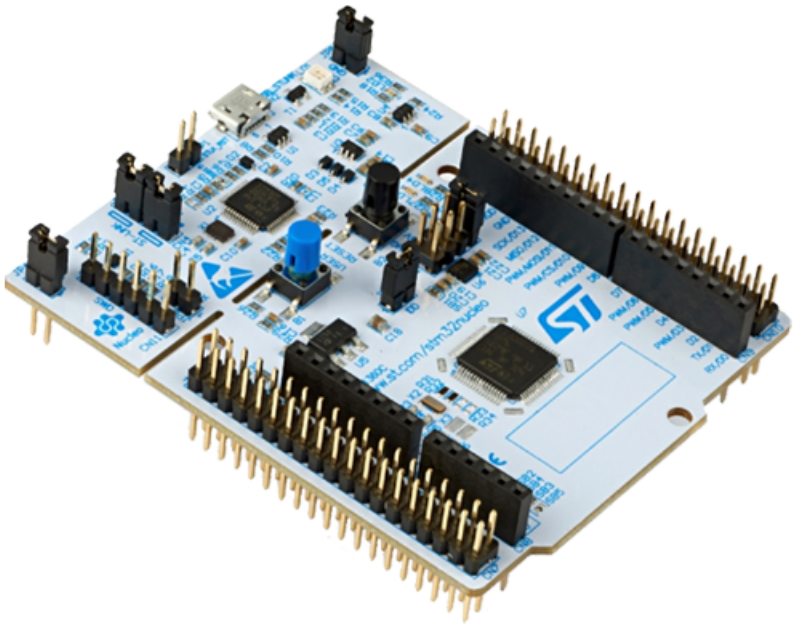
Como exemplos, temos a linha PIC e ARM com arquiteturas RISC, e em contraponto, temos a linha 8051 que é baseada em arquitetura RISC. É interessante informar aos estudantes que em (THE. . . , 2021) é possível ter acesso as instruções do núcleo.

2.7 KIT DE DESENVOLVIMENTO NUCLEO-G0B1RE

Após apresentar um básico sobre microcontroladores, apresentar o kit que iremos utilizar para as aulas o kit da ST Microeletronics, a **NUCLEO-G0B1RE**, que possui um STM32G0B1RE com um debugger já na placa. Possuindo também terminais de I/Os para ligar com circuitos externos, como botões, LEDs, circuitos integrados e etc.

na figura 2.8 temos a foto de um exemplar do kit de desenvolvimento.

Figura 2.8: NUCLEO-G0B1RE, kit de desenvolvimento da ST Microeletronics.



Fonte: Google imagens.

Possui como principais características, além das citadas na tabela 2.3:

- Unidade de cálculo de CRC;
- RTC alimentado por bateria;
- controlador DMA;
- DACs de 12bits;
- 15 timers;
- três interfaces I2C;

- seis interfaces USART;
- três UART de baixo consumo;
- três SPI;
- HDMI interface;
- USB OTG.

Outras informações podem ser acessadas em (ST MICROELETRONICS, 2021), e neste momento distribuir uma versão impressora do documento para cada estudante, para que sempre tenham em mãos informações do *hardware* da placa.

2.8 SISTEMA DE CLOCKS

Será abordado sobre o sistema de clocks do microcontrolador, sendo apresentado através do arquivo CubeMX (.ioc) na própria IDE, descrevendo apenas os blocos mais relevantes.

Aqui, destacar que o **ciclo de máquina** corresponde a um múltiplo do dos ciclos de clock, o que representa quantos ciclos são necessários para executar uma instrução. Por exemplo, microcontroladores PIC requerer 4 ciclos de clock para executar uma instrução.

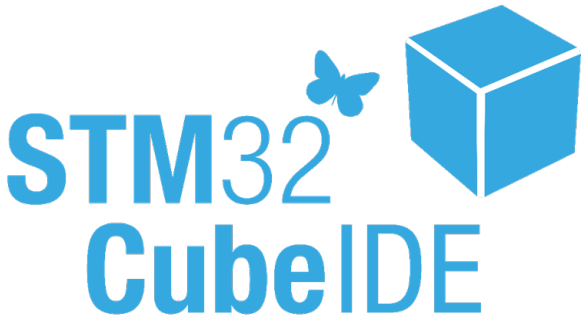
2.9 IDE DE DESENVOLVIMENTO - STM32CUBEIDE

Será realizada a instalação da ferramenta **STM32CubeIDE**, que será utilizada para desenvolver as atividades no kit.

A instalação será feita na distro Ubuntu, do Linux, que é o Sistema Operacional que está instalado nas máquinas.

Se possível, será rodado um firmware exemplo apenas para verificar se o software e as ferramentas de debugger estão funcionando adequadamente e para apresentar como é realizado o debug.

Figura 2.9: Logo do STM32CubeIDE.



Fonte: Google imagens.

3 RECURSOS DIDÁTICOS

Será utilizado lousa e canetas apropriadas, computador com os softwares necessários e projetor.

4 FORMA DE AVALIAÇÃO

Avaliação será feita através das listas de exercícios para serem resolvidas durante as aulas ou no horário reservado.

5 HISTÓRICO DE REVISÃO

Revisão 1

Primeiro documento.

A ANEXOS

A.1 LISTA DE EXERCÍCIOS - RESOLVIDA

Lista de Exercícios #1
Pado Labs - Microcontroladores
Introdução aos microcontroladores

Tips and Tricks : Utilizar o *User Manual UM2324* para resolver as questões.

Requirements : Resolva todos os exercícios.

1: Além dos exemplos citados em sala, poderia citar mais exemplos de aplicação de microcontroladores?

R: Mouse, teclado, microondas, televisão, roteador, etc.

2: Qual a principal diferença entre as memórias voláteis e memórias não voláteis?

R: A memória volátil perde os dados armazenados ao ser desligada, enquanto que as não voláteis retém os dados mesmo após ser desenergizado.

3: Explique a diferença do barramento de endereço para o barramento de controle de um microcontrolador.

R: O barramento de endereços indica qual a posição da informação a ser alterada ou requerida do periférico, enquanto que o barramento de controle seleciona o periférico desejado e se a operação é leitura ou escrita.

4: Você é um projetista de uma empresa que está desenvolvendo um novo produto que necessitará de um dispostivo programável para efetuar um controle. Este produto tem como alicerce o baixo custo e processamento relativamente pequeno. Tomando isto, é possível que o microcontrolador que melhor se encaixar na sua aplicação seja um de arquitetura *Von Neumann* ou *Harvard*?

R: Von Neumann, pois são mais baratos de serem produzidos.

5: Aproveitando a questão anterior, fale sobre a arquitetura Harvard.

R: A Arquitetura Harvard tem por grande diferença, ter a memória de dados e a memória de programas separadas, o que implica em melho performance das instruções, mas em um custo mais elevado, além de impossibilitar executar instruções da

memória RAM.

6: É verdade que o tamanho do barramento de dados (bits) de um microcontrolador é suficiente para escolher um modelo para aplicar em um projeto? Justifique.

R: Não, pois é necessário também observar outros parâmetros como quantidade de memórias, periféricos e entre outros parâmetros.

7: Explique o que são os SRFs.

R: SFRs são os registrados de funções especiais, que estão conectados ao hardware do microcontrolador, e tem por função controlar funcionalidades do microcontrolador (*control bit* ou indicar algum status (*flag bit*)).

8: O que ocorre com o microcontrolador caso a *stack* estoure?

R: O microcontrolador, em geral, força um reset quando ocorre o estouro da *stack*.

9: O contador de programa pode ser alterado durante a execução dos programas, cite em que pontos que podem ocorrer a alteração do contador de programa.

R: O PC pode ser alterado na ocorrência de uma interrupção, na chamada de uma subrotina ou ao chamar a função de *return*, que realiza o *pop* da stack.

10: Qual a ordem de entradas e saídas da stack?

R: Pelo sistema de LIFO, *Last In First Out*, ou seja, quem entrou por último, sai primeiro.

11: Tomando como base a questão anterior, esboce o funcionamento de uma stack.

R: Diagrama similar ao apresentado nos slides.

12: Qual a grande vantagem de se utilizar uma *stack* po software?

R: É mais flexível, pois permite que o tamanho seja configurado de acordo com a necessidade.

13: A arquitetura RISC permite o microcontrolador operar com *clocks* mais elevados, qual o motivo deste fato?

R: Como os circuitos internos de uma arquitetura RISC são mais simples, é possível atingir maiores velocidades de processamento.

14: Quais conhecimentos você possuia previamente sobre o tema microcontroladores?

R: Pessoal.

15: Já trabalhou com microcontroladores? Conte quais e o qual sua opinião sobre eles?

R: Pessoal.

16: O kit contém LEDs que podem ser controlados pelo programa, quantos este kit nos disponibiliza

R: O kit disponibiliza um LED para ser controlado pelo programa, indicado pela legenda LD4.

17: Vemos que o kit possui dois botões, um azul e um preto, qual o uso de cada um deles?

R: O botão USER é ligado em uma entrada do microcontrolador, e sua função é definida pelo desenvolvedor, enquanto que o RESET é ligado ao pino de reset do microcontrolador.

18: A memória flash é onde o microcontrolador armazena os comandos a serem executados, parâmetros de configuração e ainda pode ser utilizada para armazenar dados (memória não volatil). Quantos de capacidade nosso microcontrolador possui?

R: Na página 8 do documento UM2324, podemos ver que o microcontrolador possui 512KB de memória flash.

19: Ao observar a placa, vemos que a mesma possui dois microcontroladores, um sendo o STM32G0B1RE e o outro se trata de um STM32F103CB. Qual a função do ultimo microcontrolador?

R: O STM32F103 é o microcontrolador que tem implementar o ST-Link V2, se tratando do debugger que é utilizado para gravar o microcontrolador e realizar o debug.

20: O kit possui um debugger integrado, que possui um interface serial auxiliar e a interface de gravação e debug do microcontrolador, qual o nome desta interface e quais o terminais dessa interface?

R: A interface e chamada de SWD (Serial Wire Debug), os terminais do SWD são o SWCLK (clock) e SWDIO (entrada e saída de dados), além do RESET e dos pinos de alimentação do conector CN11.

21: Este kit permite que utilizemos debugger para gravar um microcontrolador externamente (em outra placa, por exemplo), no entanto o que é necessário ser feito e qual conector que é utilizado para realizar esta função?

R: Remover os Jumpers do CN4 e remover o SB19, que fica no bottom layer da placa.

22: O kit vem de fábrica com um programa exemplo. Descreva o que este programa teste faz.

R: O programa exemplo pisca o LED LD4 em uma frequência constante, que é alterada ao pressionar o botão USER.

23: É possível alimentar o kit com 4 formas de alimentação, cite-as e indique como liga-lás de forma adequada.

R: 5V_USB_STLK, alimentação oriunda do conector USB, o jumper JP2 deve estar nos pinos 1 e 2. VIN, alimentação que suporte de 7 à 12V, o jumper deve estar no pino 3 e 4 do JP2, e a alimentação deve entrar no pino 24 do CN7 ou pino 8 do

CN6. O E5V é a alimentação de 5V que pode vir externamente, o Jumper deve estar no pino 5 e 6 do JP2 e alimentar pelo pino 6 do CN7. 5V_USB_CHARGER, alimentação oriunda do USB, assim como o 5V_USB_STLK, a diferença é que o debugger é desabilitado e, como não há o controle de corrente, o USB do dispositivo que esta provendo a alimentação pode ser danificado, para habilitar o JP2 deve estar com o Jumper na posição 7 e 8.

24: Cite as fontes de clock que o kit permite utilizar e seus respectivos usos.

R: LSE, cristal de 32.758kHz utilizado pelo RTC. MCO, sinal de 8MHz gerado pelo ST-Link. HSE, cristal de 8MHz, não implementado no kit, sendo necessário adquiri-lo e solda-lo manualmente.

25: Cite as fontes de reset que o kit permite utilizar para resetar o microcontrolador.

R: Botão B2 (RESET), através do ST-Link, pelo pino 3 do CN6 e através do pino 14 do CN7.

26: O ST-Link do kit implementa uma porta COM virtual (serial) através do USB, está porta virtual consome qual periférico do STM32? O Manual UM2324 ainda informa que é possível isolar o periférico, qual é o procedimento?

R: Utiliza o periférico UART2. Para isolar a UART2 da porta COM é necessário desabilitar o SB16 e SB18.

27: O kit possui 4 LEDs, denote a função de cada um dos LEDs.

R: O LED LD1 é um led bi-color que indica o status do debugger. O Led LD2 é um LED vermelho que acende ao detectar uma sobrecorrente. LED LD3 é ligado ao barramento de 5V e acende ao alimentar a placa. Enquanto que o LED LD4 pode ser utilizado pelo desenvolvedor, através do GPIO PA5.

28: O jumper JP3 presente no kit, referido como I_{DD} possui qual finalidade.

R: Permite que seja ligado um amperímetro para medir a corrente de consumo do microcontrolador.

29: Quantos terminais de I/O possui o kit?

R: 62 terminais de I/O, incluídos de BOOT0 e de osciladores.

30: Note que na tabela de IOs do microcontrolador, temos por exemplo várias informações assocaidas ao pino 26: $PC5, ARD\_D0||UART\_1\_RX$. Explique o porque este e outros pinos tem essa informação associada a tabela.

R: Os pinos do microcontrolador são multiplexado com outras funções, ou seja, o terminal pode ser alterado para outras funções, além de entrada/saída, quando estes estão conectados a perfiéricos do dispositivo.

REFERÊNCIAS

GEEKS, Geeks for. **Difference between Von Neumann and Harvard Architecture**. 2021. https://www.geeksforgeeks.org/difference-between-von-neumann-and-harvard-architecture/. Acesso em 10 de Dezembero de 2021.

GIMENEZ, Salvador Pinillos. **Microcontroladores 8051: teoria do Hardware e do Software: aplicações em controle digital: laboratório e simulação**. 1. ed. [S.l.]: Microcontroladores 8051: teoria do Hardware e do Software: aplicações em controle digital: laboratório e simulação, 2002. ISBN 9788587918284.

ST MICROELETRONICS. **RM0444 - Reference Manual**. 5. ed. [S.l.], 2020. STM32G0x1 advanced Arm ® -based 32-bit MCUs.

____. **UM2324 - User Manual**. 4. ed. [S.l.], 2021. STM32 Nucleo-64 boards (MB1360).

THE Cortex-M0 Unstruction Set. 2021. https://developer.arm.com/documentation/dui0497/a/the-cortex-m0-instruction-set. Acesso em 10 de Dezembro de 2021.

WILDER, Jon. **A beginner's guide to microcontrollers**. 2015. https://www.microcontrollertips.com/a-beginners-guide-to-microcontrollers-faq/. Acesso em 10 de Dezembro de 2021.