

UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET

BACHARELADO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

**BRUNA DE KASSIA**

**JOSE VICTOR GAIOSO**

**NERVAL DE JESUS SANTOS JUNIOR**

TRABALHO REFERENTE A AVALIAÇÃO 1-TURMA 1

São Luís

2023

**BRUNA DE KASSIA**

**JOSE VICTOR GAIOSO**

**NERVAL DE JESUS SANTOS JUNIOR**

TRABALHO REFERENTE A AVALIAÇÃO 1-TURMA 1

Trabalho referente a 70% da primeira nota da disciplina de introdução à arquitetura de computadores.

Orientador: Prof. Dr. Denivaldo Lopes

São Luís

2023

**Sumário**

[**1.**](#_heading=h.30j0zll) **INTRODUÇÃO 3**

[**1.1.**](#_heading=h.1fob9te) **Problemática 4**

[**1.2.**](#_heading=h.3znysh7) **Justificativa 4**

[**1.3.**](#_heading=h.2et92p0) **Objetivo 4**

[**1.3.1.**](#_heading=h.tyjcwt) **Geral 4**

[**1.3.2.**](#_heading=h.3dy6vkm) **Específico 4**

[**1.4.**](#_heading=h.1t3h5sf) **Divisão do texto 4**

[**2.**](#_heading=h.4d34og8) **METODOLOGIA 4**

[**2.1.**](#_heading=h.2s8eyo1) **Circuito 4**

[**2.2.**](#_heading=h.17dp8vu) **Componentes 4**

[**2.2.1.**](#_heading=h.3rdcrjn) **Microcontrolador 4**

[**2.2.2.**](#_heading=h.26in1rg) **Componentes 4**

[**3.**](#_heading=h.lnxbz9) **TECNOLOGIAS UTILIZADAS PARA SOLUÇÃO 5**

[**3.1.**](#_heading=h.35nkun2) **Linguagem de Programação 5**

[**3.2.**](#_heading=h.1ksv4uv) **Ambiente de desenvolvimento 5**

[**4.**](#_heading=h.44sinio) **RESULTADOS E DISCUSSÕES 5**

[**5.**](#_heading=h.2jxsxqh) **CONSIDERAÇÕES FINAIS 5**

[**REFERÊNCIAS 6**](#_heading=h.z337ya)

# **INTRODUÇÃO**

# **Problemática**

* 1. **questão 1**

Pede-se que seja feito um programa em assembly para PIC16F84A correspondente ao programa em linguagem de alto nível a seguir.

int func(short int *a*){

return 4\**a* - 3;

}

void main(void ){

short int a, b, c, d, e, f, g;

a=2;

b=3;

c= -3;

d = readBit(portaA,RA1); // utilizar a PORTA A (RA1)

e= a + 8;

if(d){

f= func(c);

}else{

f = func(e);

}

if(f > 0){

write(portB,5); // escreve na porta B o valor 0x5

}else{

write(portB,4); // escreve na porta B o valor 0x4

}

}

//obs: deve-se implementar uma sub-rotina de multiplicação.

// a função readBit deve corresponder a leitura de 1 bit da porta especificada.

// a função write deve colocar os valores especificados como segundo parâmetro na porta A (configurada como saída)

A, B, C, D , E , F e G são algumas das variáveis ​​que são declaradas primeiro no código . Ele dá a algumas dessas variáveis ​​valores particulares , incluindo:

A recebe o valor 2 , b recebe o valor 3 e c recebe o valor -3.

A chamada função d = readBit(portaA, RA1); presumivelmente lê um bit da PORTA A no pino RA1 na linha seguinte . A variável d contém o valor que foi lido.

O código então executa a instrução if (d) . O código executa a função func(c) e define o valor de retorno como f se d for diferente de zero (verdadeiro). Caso contrário, atribui o retorno e chama func(e) .

Para fazermos isso em Assembly precisamos seguir os passos da programação a seguir:

; Definições de registradores

#define portaA 5

#define RA1 1

#define portaB 6

; Variáveis

a equ 0x20

b equ 0x21

c equ 0x22

d equ 0x23

e equ 0x24

f equ 0x25

g equ 0x26

; Vetor de multiplicação

mul equ 0x27

org 0x00

goto main

org 0x04

call multiplicacao

return

multiplicacao:

movlw 0x04 ; Multiplicador (4)

movwf mul ; Armazena no registrador temporário

movf a, W ; Carrega o valor de a

call multiplicar

sublw 0x03 ; Subtrai 3 do resultado

return

multiplicar:

addwf a, F ; Soma a com a

decfsz mul, F ; Decrementa o multiplicador

goto multiplicar

return

main:

bsf STATUS, RP0 ; Banco de registradores 1

movlw 0x07 ; Configura os pinos RA0 e RA1 como entrada

movwf TRISA

movlw 0x00 ; Configura o pino RB0 como saída

movwf TRISB

bcf STATUS, RP0 ; Retorna ao banco de registradores 0

movlw 0x02 ; Atribui 2 a "a"

movwf a

movlw 0x03 ; Atribui 3 a "b"

movwf b

movlw 0xFD ; Atribui -3 a "c"

movwf c

bsf STATUS, RP0 ; Banco de registradores 1

btfsc portaA, RA1 ; Lê o bit em RA1

goto d\_true

movf a, W ; Carrega o valor de "a" em "e"

addlw 0x08 ; Soma 8 a "e"

movwf e

call func ; Chama a função func

movwf f ; Armazena o resultado em "f"

goto check\_f

d\_true:

movf c, W ; Carrega o valor de "c" em "f"

call func ; Chama a função func

movwf f ; Armazena o resultado em "f"

check\_f:

btfss STATUS, Z ; Verifica se f é igual a zero

goto f\_positive

movlw 0x04 ; Atribui 0x04 a "f" (f < 0)

goto write\_portB

f\_positive:

movlw 0x05 ; Atribui 0x05 a "f" (f > 0)

write\_portB:

movwf PORTB ; Escreve o valor de "f" em PORTB

end:

goto end

esse código roda em arquivo .nasm, pois só testei nessa IDE, falta transformar para o formato que roda dentro da IDE MPLAB X

* 1. **questão 2**

Um sistema hardware/software é baseado no PIC16F84A e deve verificar um sensor A continuamente (utilize a Porta A, bit RA1), caso o sensor retorne o bit 1, então deve enviar o valor 0x75 pela porta B, caso contrário 0x31 pela porta B. O mesmo sistema deve enviar pela porta A, bit RA0, um sinal quadrado com frequência de 8 Hz. O clock do PIC16F84A deve ser de 4 MHz. Programa deve ser feito em assembly

; Definições de registradores

#define portaA 5

#define RA0 0

#define RA1 1

#define portaB 6

org 0x00

goto main

org 0x04

call check\_sensor

return

check\_sensor:

btfss portaA, RA1 ; Verifica se o bit RA1 é igual a 1

goto bit\_0

movlw 0x75 ; Atribui 0x75 a W (bit RA1 = 1)

goto write\_portB

bit\_0:

movlw 0x31 ; Atribui 0x31 a W (bit RA1 = 0)

write\_portB:

movwf PORTB ; Escreve o valor de W em PORTB

return

main:

bsf STATUS, RP0 ; Banco de registradores 1

movlw 0x02 ; Configura o pino RA1 como entrada

movwf TRISA

movlw 0x01 ; Configura o pino RA0 como saída

movwf TRISB

bcf STATUS, RP0 ; Retorna ao banco de registradores 0

clrf PORTB ; Zera o valor inicial de PORTB

; Configuração do Timer0 para gerar um sinal quadrado com frequência de 8 Hz

movlw 0x67 ; Valor inicial do Timer0

movwf TMR0

movlw 0x00 ; Habilita o Timer0 com prescaler 1:256

option

bsf INTCON, T0IE ; Habilita a interrupção do Timer0

bsf INTCON, GIE ; Habilita todas as interrupções

loop:

; O programa entra em loop infinito

goto loop

org 0x0B

bcf INTCON, T0IF ; Limpa a flag de interrupção do Timer0

movlw 0x67 ; Recarrega o valor inicial do Timer0

movwf TMR0

xorwf PORTA, F ; Inverte o valor do bit RA0 (sinal *quadrado*)

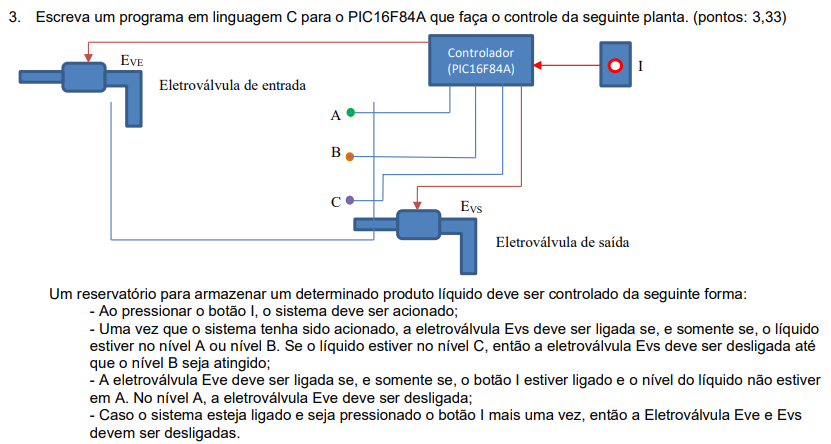
retfie

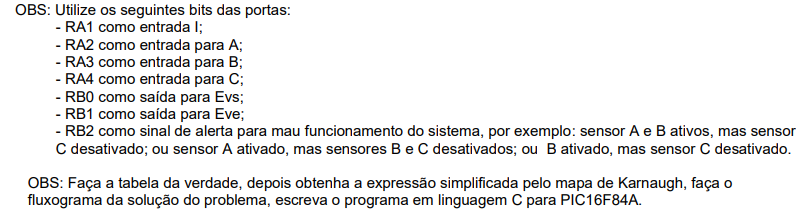
end

é necessário fazer as mesmas operações para rodar no mplab aqui

tem que verificar a questão da frequência tbm

* 1. **questão 3**





# **Justificativa**

# **Objetivo**

# **Geral**

# **Específico**

# **Divisão do texto**

# **METODOLOGIA**

# **Circuito**

# **Componentes**

# **Microcontrolador**

O microcontrolador é uma junção de Software e Hardware que através da programação conseguimos controlá-los para desempenhar tarefas. É um tipo especial de circuito integrado, visto que conseguimos programar os microcontroladores para realizar tarefas específicas.

Microcontrolador computa dados, mas com capacidade reduzida se comparada aos microprocessadores. Possui elevada capacidade de fazer entrada e saída. Deve ser pequeno, barato e auto-contido. Funciona em temperaturas extremas. Consome pouca energia se comparado ao microprocessador. Possui memória reduzida.

# **Componentes**

# 

# **TECNOLOGIAS UTILIZADAS PARA SOLUÇÃO**

# **Linguagem de Programação**

A linguagem de programação utilizada pode ser assembly ou c

Entre instruções de máquina e linguagens de programação de alto nível , Assembly é uma linguagem de programação de baixo nível . Ela é chamada de linguagem de montagem ou linguagem simulada .

Escrever instruções que um processador ou unidade de processamento central (CPU) de um computador pode entender diretamente faz parte da programação do Assembly . Cada instrução Assembly denota uma operação específica que a CPU é capaz de realizar , como mover dados , realizar cálculos ou gerenciamento do fluxo do programa .

Cada arquitetura de processador possui sua própria linguagem Assembly específica , portanto, cada família de processadores possui sua própria linguagem Assembly . Cada instrução Assembly é representada por um código mnemônico relacionado à operação específica (como ADD , SUB, JMP, MOV, etc.).

# **Ambiente de desenvolvimento**

O MPLAB X IDE (Integrated Development Environment ) é um software que pode ser utilizado com Windows, MAC e Linux para desenvolver aplicações que utilizam microcontroladores e controladores digitais da Microchip . A frase " ambiente de desenvolvimento integrado " é usada porque o MPLAB X permite a criação de um ambiente único para o desenvolvimento de código para sistemas marítimos .

Um sistema embarcado é um projeto que faz uso dos recursos de um microcontrolador , como o Microchip PIC® (MCU) ou o controlador digital de sinal dsPIC® (DSC) . Esses microcontroladores combinam uma unidade de microprocessador (como a CPU de um computador) com alguns circuitos auxiliares . Este componente pode ser integrado a outros sistemas de controle digital eletrônico e mecânico de baixo custo .

# **RESULTADOS E DISCUSSÕES**

# **CONSIDERAÇÕES FINAIS**

# **REFERÊNCIAS**