|  |  |  |  |
| --- | --- | --- | --- |
| Nome: | Vitor Henrique de Oliveira Costa | R.A.: | 22121078-4 |
| Nome: | Denis Bento Lima | R.A.: | 22121090-9 |

Projeto de Arquitetura de Computadores

# Descrição do Projeto

A dupla concordou em realizar um projeto que consiste em criar um relógio digital, no qual irá aparecer em seu display o horário e será atualizado constantemente, utilizando a tabela ASCII para poder imprimir os valores do relógio e o display LCD de modo que mostrare os valores presentes dentro do data memory.

Este relógio funciona por sistema de AM e PM, que faz com que toda vez que chegue em 12:00:00 AM(período matutino), ele zera o relógio e inicie o período da vespertino e noturno(PM).

O desenvolvimento do código e lógica para criação de tais funcionalidades do projeto foram implementadas no programa Edsim51, compilado na linguagem Assembly 8051.

# Desenhos esquemáticos

O desenho esquemático do edsim51, mostrando as partes que estão sendo utilizadas. Exemplo:

Diagrama, Esquemático

Descrição gerada automaticamente

Diagrama, Esquemático

Descrição gerada automaticamente

# Fluxograma ou Diagrama

Fluxograma ou desenho descrevendo as posições de memória, sub-rotinas, etc.;

Exemplo mostrando os caracteres **a, b, c, d, e** na memória:

Uma imagem contendo Tabela

Descrição gerada automaticamente



**Marcação em vermelho**: Os valores que possuem destaque em vermelho são responsáveis pelo armazenamento do relógio digital no data memory, sendo eles posicionados no 71 até o 78h.

**Marcação em verde:** Os valores que possuem destaque em verde são partes do data memory que armazenam atividades rotineiras do código.

# Imagens da simulação realizada na IDE

Algumas Imagens da simulação realizada na IDE, com telas apresentando os resultados obtidos.

Interface gráfica do usuário

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente

# Discussões e conclusões

O projeto foi um desafio para os integrantes do grupo, devido a linguagem assembly ser baixo nível, sendo a primeira vez que realizamos um projeto nesse tipo de linguagem. As principais dificuldades que foram encontradas durante a realização do projeto foi conseguir criar a lógica de implementação dos valores em ASCII no display e acrescentar as unidades e dezenas das horas para retornar do am para o pm.

Além disso, foi uma excelente oportunidade de entendermos o funcionamento do hardware, pois o Assembly necessita que os programadores tenham um conhecimento mais aprofundado no hardware e aprendemos como funciona o LCD, bem como trabalhar com uma linguagem mais próxima a forma como a máquina se comunica.

# Código-fonte

; --- Mapeamento de Hardware (8051) ---

RS equ P1.3 ;Reg Select ligado em P1.3

EN equ P1.2 ;Enable ligado em P1.2

org 0000h;inicio no endereço 00

acall lcd\_init;inicia o LCD

MOV 71H, #'1';Endereço da dezena da hora

MOV 72H, #'1';Endereço da unidade da hora

MOV 73H, #'5';Endereço da dezena da minuto

MOV 74H, #'8';Endereço da unidade do minuto

MOV 75H, #'2';Endereço da unidade do segundo

MOV 76H, #'1';Endereço da dezena da segundo

MOV 77H, #'0';dois pontos

MOV 78H, #'0';dois pontos

MOV 79H, #0 ;MARCA O NULL NO FIM DA STRING

acall base

mov A, #48h;move o valor de 48h para o acumulador que será utilizado para printar a letra A no display, indicando que é o período da manhã

ACALL posicionaCursor;executa o posiciona cursor

MOV A, #'A';Símbolo para representar o período da manhã juntamente com o M

ACALL sendCharacter ;Envia o caracter

LJMP main; Chama a função principal

org 0030h; Começar o main no 30h

main:;programa principal

acall segundos\_un;sub rotina responsável por incrementar a unidade dos segundos

acall print;sub rotina responsável por printar no lcd

JMP main; volta para o main

segundos\_un:; subrotina que faz com que inicie o valor da unidade do segundo começe em 0 e vai até 9 utilizando as strings da tabela ASCII,depois que der 9, passa para a próxima subrotina, que é responsável pela dezena dos segundos

inc 77h;incrementa o valor no endereço de memória 77h

MOV A,77H;move o valor presente no endereço de memória 77h para o acumulador

mov psw,#00h;zera o psw(para não alterar o valor do a)

subb a, #3Ah;caso o acumulador for igual a 9, sendo verificado pela subtração do acumulador

jz segundo\_dez;vai para a subrotina do

RET;retorna

segundo\_dez:; subrotina que faz com que inicie o valor da dezena do segundo começe em 0 e vai até 6 utilizando as strings da tabela ASCII,depois que der 6, passa para a próxima subrotina, que é responsável pela unidade dos minutos

MOV 77H, #'0';a unidade dos segundos volta a ser 0

inc 76H;incrementa o valor no endereço de memória 76h

mov a,76H;move o valor presente no endereço de memória 76h para o acumulador

mov psw,#00h;zera o psw(para não alterar o valor do a)

subb a,#36h;caso o valor no acumulador for 6, entra na sub rotina

jz minuto\_un;subrotina que incrementa a unidade dos minutos

ret;retorna

minuto\_un:; sub rotina que faz com que inicie o valor da unidade do minuto começe em 0 e vai até 9 utilizando as strings da tabela ASCII,depois que der 9, passa para a próxima subrotina, que é responsável pela dezena dos minutos

MOV 76H, #'0';zera a dezena do segundo

INC 74H;incrementa o valor da unidade do minuto

MOV A, 74h;move o valor do 74h para o acumulador

mov psw,#00h;zera o psw

subb a, #3Ah; faz a subtração por 9 para poder entrar na subrotina

jz minuto\_dez;subrotina que incrementa as dezenas dos minutos

RET;retorna

minuto\_dez:;subrotina que faz com que inicie o valor da dezena do minuto começe em 0 e vai até 6 utilizando as strings da tabela ASCII,depois que der 6, passa para a próxima subrotina, que é responsável pela unidade das horas

MOV 74H, #'0';zera a unidade do minuto

INC 73H;incrementa o valor da dezena do minuto

MOV A, 73h;move o valor de 73h para o acumulador

mov psw,#00h;zera o psw(para não alterar o valor do a)

subb a,#36h;faz a subtração do acumulador e se der 6 vai para a subrotina

jz hora\_un;subrotina que faz incrementar o valor da unidade da hora

ret;retorna

hora\_un:;subrotina que faz com que inicie o valor da unidade da hora começe em 0 e vai até 9 utilizando as strings da tabela ASCII,depois que der 9, passa para a próxima subrotina, que é responsável pela dezena das horas

MOV 73H, #'0';zera a dezena do minuto

inc 72h;incrementa o valor da

MOV A, 72H;move 72h para o acumulador

mov psw,#00h;zera o psw(para não alterar o valor do a)

subb a,#3Ah;faz a subtração para identificar se o valor da unidade for 9, caso seja, vai para a subrotina

jz hora\_dez;subrotina que incrementa o valor na dezena da hora

MOV A,72H;move o valor em 72h para o acumulador

; Entra nessa parte para verificar se chegou a 12h(mais especificamente se na unidade é 2 e a dezena é 1), para entrar na subrotina para trocar o am para pm e zerar o relógio

mov psw,#00h;zera o psw(para não alterar o valor do a)

subb a,#32h;caso der 2 no acumulador, entra na subrotina

JZ fuso1;subrotina que verifica se o valor na dezena da hora é 1

ret

hora\_dez:;subrotina responsável por incrementar o valor da dezena da hora e zerar sua unidade

MOV 72H, #'0';zera a unidade da hora

inc 71h;incrementa o valor da dezena da hora

RET;retorna

fuso1:;subrotina responsável pela identificação se caso chegou a 12 para trocar o am para pm e zerar o relógio

mov a,71H;move o endereço da dezena da hora para o acumulador

mov psw,#00h;zera o psw(para não alterar o valor do a)

subb a,#31h;caso o valor seja igual a 1 no endereço 71h, ele entra na subrotina, caso o contrário, volta para

jz fuso2;

RET;retorna

fuso2:;subrotina responsável por zerar o relógio e trocar o AM para PM

mov A, #48h;move para a posição do a para trocar para P

ACALL posicionaCursor;posiciona cursor

MOV A, #'P';Insere o valor P no acumulador,segundo a tabela ASCII

ACALL sendCharacter ;envia o caracter

MOV 71H, #'0';|

MOV 72H, #'0';|

MOV 73H, #'0';|

MOV 74H, #'0';|

MOV 75H, #'0';|

MOV 76H, #'0';|

MOV 77H, #'0';|

MOV 78H, #'0';Reseta o relógio para o valor inicial

RET;retorna

print:; subrotina que printa o relógio

mov A, #40h;posição da dezena da hora

ACALL posicionaCursor ;posiciona o cursor

MOV A, 71h;pega o valor armazenado na posição 71h

ACALL sendCharacter;envia o caracter

mov A, #41h;posição da unidade da hora

ACALL posicionaCursor ;posiciona o cursor

MOV A, 72h;pega o valor armazenado na posição 72h

ACALL sendCharacter;envia o caracter

mov A, #43h;posição da dezena do minuto

ACALL posicionaCursor;posiciona o cursor

MOV A, 73h;pega o valor armazenado na posição 73h

ACALL sendCharacter;envia o caracter

mov A, #44h;posição da unidade do minuto

ACALL posicionaCursor

MOV A, 74h;pega o valor armazenado na posição 74h

ACALL sendCharacter

mov A, #46h;posição da dezena do segundo

ACALL posicionaCursor ;posiciona o cursor

MOV A, 76h;pega o valor armazenado na posição 76h

ACALL sendCharacter;envia o caracter

mov A, #47h;posição da unidade do segundo

ACALL posicionaCursor ;posiciona o cursor

MOV A, 77h;pega o valor armazenado na posição 77h

ACALL sendCharacter;envia o caracter

RET

base:

mov A, #42h;move o valor de 42h para o acumulador

ACALL posicionaCursor ; posiciona o cursor

MOV A, #3aH;printa : na tela do lcd

ACALL sendCharacter ; send data in A to LCD module

mov A, #45h;move o valor de 45h para o acumulador

ACALL posicionaCursor ; posiciona o cursor

MOV A, #3aH;printa : na tela do lcd

ACALL sendCharacter ; send data in A to LCD module

mov A, #49h;move o valor de 49h para o acumulador

ACALL posicionaCursor; posiciona o cursor

MOV A, #'M';seta o M do display, representando o horário

ACALL sendCharacter ; send data in A to LCD module

;ACALL retornaCursor

ret

; initialise the display

; see instruction set for details

lcd\_init:

CLR RS ; clear RS - indicates that instructions are being sent to the module

; function set

CLR P1.7 ; |

CLR P1.6 ; |

SETB P1.5 ; |

CLR P1.4 ; | high nibble set

SETB EN ; |

CLR EN ; | negative edge on E

CALL delay ; wait for BF to clear

; function set sent for first time - tells module to go into 4-bit mode

; Why is function set high nibble sent twice? See 4-bit operation on pages 39 and 42 of HD44780.pdf.

SETB EN ; |

CLR EN ; | negative edge on E

; same function set high nibble sent a second time

SETB P1.7 ; low nibble set (only P1.7 needed to be changed)

SETB EN ; |

CLR EN ; | negative edge on E

; function set low nibble sent

CALL delay ; wait for BF to clear

; entry mode set

; set to increment with no shift

CLR P1.7 ; |

CLR P1.6 ; |

CLR P1.5 ; |

CLR P1.4 ; | high nibble set

SETB EN ; |

CLR EN ; | negative edge on E

SETB P1.6 ; |

SETB P1.5 ; |low nibble set

SETB EN ; |

CLR EN ; | negative edge on E

CALL delay ; wait for BF to clear

; display on/off control

; the display is turned on, the cursor is turned on and blinking is turned on

CLR P1.7 ; |

CLR P1.6 ; |

CLR P1.5 ; |

CLR P1.4 ; | high nibble set

SETB EN ; |

CLR EN ; | negative edge on E

SETB P1.7 ; |

SETB P1.6 ; |

SETB P1.5 ; |

SETB P1.4 ; | low nibble set

SETB EN ; |

CLR EN ; | negative edge on E

CALL delay ; wait for BF to clear

RET

sendCharacter:

SETB RS ; setb RS - indicates that data is being sent to module

MOV C, ACC.7 ; |

MOV P1.7, C ; |

MOV C, ACC.6 ; |

MOV P1.6, C ; |

MOV C, ACC.5 ; |

MOV P1.5, C ; |

MOV C, ACC.4 ; |

MOV P1.4, C ; | high nibble set

SETB EN ; |

CLR EN ; | negative edge on E

MOV C, ACC.3 ; |

MOV P1.7, C ; |

MOV C, ACC.2 ; |

MOV P1.6, C ; |

MOV C, ACC.1 ; |

MOV P1.5, C ; |

MOV C, ACC.0 ; |

MOV P1.4, C ; | low nibble set

SETB EN ; |

CLR EN ; | negative edge on E

CALL delay ; wait for BF to clear

RET

;Posiciona o cursor na linha e coluna desejada.

;Escreva no Acumulador o valor de endereï¿½o da linha e coluna.

;|--------------------------------------------------------------------------------------|

;|linha 1 | 00 | 01 | 02 | 03 | 04 |05 | 06 | 07 | 08 | 09 |0A | 0B | 0C | 0D | 0E | 0F |

;|linha 2 | 40 | 41 | 42 | 43 | 44 |45 | 46 | 47 | 48 | 49 |4A | 4B | 4C | 4D | 4E | 4F |

;|--------------------------------------------------------------------------------------|

posicionaCursor:

CLR RS ; clear RS - indicates that instruction is being sent to module

SETB P1.7 ; |

MOV C, ACC.6 ; |

MOV P1.6, C ; |

MOV C, ACC.5 ; |

MOV P1.5, C ; |

MOV C, ACC.4 ; |

MOV P1.4, C ; | high nibble set

SETB EN ; |

CLR EN ; | negative edge on E

MOV C, ACC.3 ; |

MOV P1.7, C ; |

MOV C, ACC.2 ; |

MOV P1.6, C ; |

MOV C, ACC.1 ; |

MOV P1.5, C ; |

MOV C, ACC.0 ; |

MOV P1.4, C ; | low nibble set

SETB EN ; |

CLR EN ; | negative edge on E

CALL delay ; wait for BF to clear

RET

;Retorna o cursor para primeira posiï¿½ï¿½o sem limpar o display

retornaCursor:

CLR RS ; clear RS - indicates that instruction is being sent to module

CLR P1.7 ; |

CLR P1.6 ; |

CLR P1.5 ; |

CLR P1.4 ; | high nibble set

SETB EN ; |

CLR EN ; | negative edge on E

CLR P1.7 ; |

CLR P1.6 ; |

SETB P1.5 ; |

SETB P1.4 ; | low nibble set

SETB EN ; |

CLR EN ; | negative edge on E

CALL delay ; wait for BF to clear

RET

;Limpa o display

clearDisplay:

CLR RS ; clear RS - indicates that instruction is being sent to module

CLR P1.7 ; |

CLR P1.6 ; |

CLR P1.5 ; |

CLR P1.4 ; | high nibble set

SETB EN ; |

CLR EN ; | negative edge on E

CLR P1.7 ; |

CLR P1.6 ; |

CLR P1.5 ; |

SETB P1.4 ; | low nibble set

SETB EN ; |

CLR EN ; | negative edge on E

CALL delay ; wait for BF to clear

RET

delay:;subrotina do delay

mov r2, #10h

again2: mov r3, #100h

again1: djnz r3, again1

djnz r2, again2;decrementa de r2

RET;retorna