**Universidade de São Paulo**

**Escola de Engenharia de São Carlos**

****

**Atividade prática de uso de set de instruções e manipulação de dados em microcontroladores**

**SEL0614 - Aplicação de Microprocessadores**

**Prof. Pedro Oliveira**

Bárbara Fernandes Madera, 11915032

Matheus dos Santos Inês, 12546784

Victor Gabriel Miranda Rosa, 11232114

**São Carlos**

**2023**

**Sumário**

[**1. Objetivos**](#_j82qxkrr1r32) **2**

[**2. Explicação Teórica**](#_l7b0fm9zmsws) **2**

**3**[**. Resolução Comentada**](#_pfd3le7ii8qv) **3**

**4**[**. Resposta às perguntas**](#_yic7a5bx83qj) **7**

## **Objetivos**

* Revisar conceitos relativos às primeiras aulas do curso e o propósito da disciplina no âmbito de sistemas embarcados e microcontroladores.
* Realizar manipulação básica de dados em registradores e endereços de memória e exercitar o uso do set de instruções por meio de ferramenta de simulação computacional visando potencializar a compreensão sobre o funcionamento dos microcontroladores.
* Exercitar o uso do EdSim51, dos registradores de propósito geral e de função especial, e do set de instruções de transferência de dados, lógicas, aritméticas, booleanas, incondicionais e condicionais usando a família MCS-51.

## **Explicação Teórica**

**1- Manipulação de dados em registradores e endereços de memória por meio de instruções de transferência de dados:**

O programa começa com uma label e é inicializado. Em seguida, são realizadas várias operações, incluindo a movimentação de valores para o registrador acumulador, registradores gerais, registrador B e memória RAM. Há também a utilização de um registrador como "ponteiro" para mover dados indiretamente para o acumulador. O programa consome 1 μs de tempo sem operações e, por fim, é encerrado. Esses passos representam uma sequência de instruções em assembly que envolvem manipulação de dados, movimentação entre registradores e memória, e controle de tempo.

**2- Manipulação de dados em registradores e endereços de memória por meio de instruções aritméticas:**

O exercício envolve a criação de um programa em linguagem assembly que realiza diversas operações com registradores e memória. Inicialmente, os valores 2 e 3 são carregados em registradores específicos (ACC e B). Em seguida, ocorrem operações de soma, subtração, multiplicação e divisão entre esses registradores, bem como manipulações em endereços de memória. O programa encerra com a armazenagem dos resultados em locais de memória designados. Além disso, o exercício inclui um teste que explora como as operações de carga afetam o bit menos significativo (LSB) do Program Status Word (PSW) em situações de valores positivos e negativos no ACC. O objetivo é desenvolver um programa funcional e entender como as operações afetam os registradores e o PSW. Ademais, o código descrito está em anexo ao relatório em duas partes e apresenta comentários ao longo do mesmo a fim do melhor entendimento do que foi feito.

**3 - Manipulação de dados em registradores e endereços de memória por meio de instruções lógicas e booleanas:**

O código começa com a definição da origem do programa (ORG 0000h) e a marcação da label "inicio". O programa então executa uma série de operações lógicas e manipulações de bits utilizando os registradores A e B. Vale ressaltar ainda que esse programa é uma sequência de operações lógicas e manipulações de bits que são aplicadas nos registradores A e B. O código continua executando indefinidamente, pois há um salto (JMP inicio) de volta ao início. O programa só encerra se interrompido manualmente ou se houver uma condição de parada implementada posteriormente no código.

**4- Manipulação de dados em registradores e endereços de memória por meio de**

**instruções de desvio incondicional e condicional:**

No último exercício, foi desenvolvido um programa em linguagem assembly com o objetivo de demonstrar o uso de saltos condicionais e incondicionais, bem como operações de controle de fluxo. O programa começa definindo a origem em 00h e salta para a label "main". Em seguida, define a origem em 33h e inicia o programa principal com o mesmo label. O programa consiste em três blocos de código identificados por labels "bloco1", "bloco2" e "bloco3". O primeiro bloco inclui saltos condicionais com base no valor em A, levando a diferentes blocos. O segundo bloco move o valor de R0 para A e retorna ao primeiro bloco de forma incondicional. O terceiro bloco executa um loop enquanto R0 ≠ 0 e salta incondicionalmente para o terceiro bloco. O programa encerra quando o fluxo é retornado à label do programa principal, reiniciando a operação. Esse exercício demonstra habilidades de controle de fluxo e manipulação de registradores em assembly. De forma análoga aos demais exercícios, o mesmo encontra-se anexado ao arquivo do relatório com comentários ao longo de sua resolução.

## **Resolução Comentada**

**1- Manipulação de dados em registradores e endereços de memória por meio de instruções de transferência de dados:**

org 0000h

inicio:

; Carrega o valor hexadecimal 5A em A (ou ACC)

MOV ACC, #0x3A ; 2us

; Move de forma imediata um valor qualquer em hexadecimal (de 00 a FF) para o registrador acumulador

MOV ACC, #00h ; 4us

; Move o valor hexadecimal 0x42 para o registrador R3

MOV R3, #0x42 ; 5us

; Move o valor hexadecimal 0x5A para o registrador B

MOV B, #0x5A ;7us

;Move a porta P1 para um endereço de memória RAM qualquer (entre 00 a 7F)

MOV R0, P1 ; 9us

MOV 0x20, R0 ; 11us

;Move de forma direta o conteúdo da posição de memória escolhida na linha anterior para um registrador qualquer do Banco 01 (segundo banco)

MOV ACC, #0x30 ; 13us

MOV R2, ACC ; 15 us

; Move o conteúdo do registrador R2 para o endereço de memória 0x200

MOV 0x70, R2 ; 17us

; Move o valor imediato 0x200 para o registrador R1

MOV R1, #0x70 ; 18us

; Move o conteúdo da memória apontada por R1 para o acumulador A

MOV ACC, @R1 ; 20us

; Move o valor para O DPTR

MOV DPTR, #0x9A5B ; 22us

NOP ; 23us

end

**2- Manipulação de dados em registradores e endereços de memória por meio de instruções aritméticas:**

**Primeira Parte**

org 00h ; #Origem

inicio: ; #Label de início

MOV A,#2 ; #Move o valor 2 para o acumulador ACC

MOV B,#3 ; #Move o valor 3 para o registrador B

MOV 20h,#7 ; #Move o valor 7 para o endereço de memória 20h

ADD A, 20h ; #Soma o conteúdo de 20h ao ACC

DEC A ; #Decrementa 3 unidade do ACC

INC B ; #Incrementa 1 unidade em B

SUBB A,B ; #Subtrai B de A

MUL AB ; #Multiplica A por B

INC B,#2 ; #Incrementa B em 2 unidades

DIV AB ; #Divide A por B

MOV 30h, A ; #Armazena o conteúdo de A no endereço 30h

MOV 31h, B ; #Armazena o conteúdo B no endereço 31h

SJMP inicio ; #Salta para o início do programa

end

**Segunda Parte- Teste**

rg 00h ; #Origem

inicio: ; Label de início

MOV A, #4 ; Move o valor 4 para ACC

MOV A, #3 ; Move o valor 3 para ACC

end ; #Fim do programa

**3 - Manipulação de dados em registradores e endereços de memória por meio de instruções lógicas e booleanas:**

ORG 0000h ; Escolhendo a origem em 0000h

inicio:

MOV A, #10110111b ; Movendo de imediato o número binário 10110111 para o ACC

MOV B, #10000111b ; Movendo de imediato o número binário 10000111 para o B

ANL A, B ; Aplicando lógica AND entre A e B

RR A ; Rotacionando A para a direita em um bit

RR A ; Rotacionando A para a direita em um bit

CPL A ; Complemento de A

RL A ; Rotacionando A para a esquerda em um bit

RL A ; Rotacionando A para a esquerda em um bit

ORL A, B ; Aplicando lógica or e guardando em A

XRL A, B ; Aplicando lógica xor e guardando em A

SWAP A ; Realizando SWAP de A

JMP inicio ; Saltar para a label inicial

END ; Encerrando o programa

**4- Manipulação de dados em registradores e endereços de memória por meio de**

**instruções de desvio incondicional e condicional:**

rg 00h ; Origem

; Saltar para o programa principal(main)

SJMP main

org 33h; Inicializar o programa principal com label informada anteriormente

main: ; Label principal

CLR A; Limpar o ACC

MOV RO, #5;Mover um valor qualquer 5,por exemplo, para RO

; Primeiro bloco do código (bloco1)

bloco1:

CJNE A, #0, bloco3; Saltar para bloco3 se A ≠ 0

NOP; Consumir 1μs

SJMP bloco2; Saltar para bloco2

; Segundo bloco de código (bloco2)

bloco2:

MOV A, RO; Mover o valor de RO para A

SJMP bloco1; Saltar incondicionalmente para bloco1

; Terceiro bloco de código (bloco3)

bloco3:

DJNZ RO, bloco3; Decrementar RO e saltar para bloco3 se R0 ≠ 0

SJMP main; Saltar incondicionalmente para o programa principal para reiniciar

end; Fim do programa

## **Respostas às Perguntas**

**1- Manipulação de dados em registradores e endereços de memória por meio de instruções de transferência de dados:**

a) O tempo gasto em cada linha estão indicados como comentários no código.

b)MOV A, 0x5A - 1 ciclo

MOV ACC, #00h - 3 ciclos

MOV R3, #0x42 - 4 ciclos

MOV B, #0x5A - 6 ciclos

MOV R0, P1 - 8 ciclos

MOV 0x20, R0 - 10 ciclos

MOV A, #0x30 - 11 ciclos

MOV R2, A - 12 ciclos

MOV 0x70, R2 - 14 ciclos

MOV R1, #0x70 - 15 ciclos

MOV A, @R1 - 16 ciclos

MOV DPTR, #0x9A5B - 18 ciclos

NOP - 1 ciclo

Total: 109 ciclos de máquina

c) Ao mover uma porta inteira de 8 registradores, como P1, para um registrador, como R0, o valor resultante é frequentemente 0xFF. Isso ocorre porque, por padrão, os pinos da porta P1 são configurados como entradas de alta impedância, e quando nenhum dispositivo externo os está puxando para um estado específico, eles tendem a ficar em um estado de "1" devido a resistores de pull-up internos.

d) Continuou o mesmo valor anterior (0x30) -> não tenho certeza se funcionou essa parte do meu código

e) Foi possível mover um valor de 4 dígitos (16 bits) para DPTR porque DPTR é um registrador especial de 16 bits usado para armazenar endereços de memória. O DPTR é formado pelos registradores DPH (byte mais significativo) e DPL (byte menos significativo). Ele pode armazenar valores de endereço de memória de até 0xFFFF em hexadecimal (65535 em decimal), e é possível verificar as mudanças nos registradores DPH e DPL no simulador ao realizar a operação de carga em DPTR.

**2- Manipulação de dados em registradores e endereços de memória por meio de instruções aritméticas:**

Para o teste adicional de mover os valores 4 e 3 para o ACC em sequência e observar o bit menos significativo do registrador PSW, o algoritmo acima foi escrito. Neste sentido,ao executar essas linhas e observar o registrador PSW,nota-se que o bit menos significativo (bit de carry)será 0 após a segunda operação de movimento.Isso ocorre porque o segundo movimento sobrescreve o valor anterior em ACC, e o bit de carry é zerado porque não houve nenhuma operação que o alterasse.

**4- Manipulação de dados em registradores e endereços de memória por meio de**

**instruções de desvio incondicional e condicional:**

O programa principal começa na label main.No primeiro bloco de código (bloco1), verifica-se se o valor em A é diferente de zero. Se for, salta-se para o terceiro bloco (bloco3), caso contrário, consome-se 1 μs (não realizamos operação) e salta-se para o segundo bloco (bloco2).No segundo bloco (bloco2), move-se o valor de R0 para A e salta-se incondicionalmente de volta para o primeiro bloco (bloco1).No terceiro bloco (bloco3), decrementa-se o valor em R0 e salta-se de volta para o terceiro bloco enquanto R0 for diferente de zero. Quando R0 se tornar zero, salta-se incondicionalmente de volta para o programa principal para reiniciar todo o processo.