Exercício 1

Objectivo

Implementação de operações aritméticas

Introdução

Efectuar a multiplicação de um registo interno por 2.

A multiplicação é uma operação aritmética complexa, que demora por isso mais tempo a ser executada do que a adição. No caso do 8051 a 12 MHz, a multiplicação demora o quadruplo do tempo a ser executada, pelo que há situações muito particulares, em que é útil recorrer a alguns artifícios para evitar o recurso a esta operação.

Funcionamento 1

Multiplicar o conteúdo do registo R0 por 2.
Corra o código para vários valores de R0.

Funcionamento 2

Faça um programa que desloque à esquerda, sem utilização da flag de carry, o conteúdo do registo R0.

Corra o código para os valores que utilizou no ponto anterior.

O que é que pode concluir?

Exercício 2

Objectivo

Implementação de operações aritméticas. Efectuar a soma de valores a oito e dezasseis bits, contidos em registos internos da máquina

Funcionamento

Somar o conteúdo do registo R0 com o conteúdo do registo R1, e colocar o resultado no registo R7.

Sugestão

Corra o programa para o seguinte conjunto de valores:

R0	**R1**
03	02
04	05
09	02
09	09

Observação 1

Não se esqueça que para obter resultados correctos, tem de inicializar a flag de Carry a zero. Observe o resultado da operação para os dois últimos casos e comente o seu funcionamento.

Observação 2

Repare que na rotina que escreveu, efectua uma adição de oito bits, pelo que o estado da máquina no final da operação (ie o resultado da adição) no que toca ao valor lógico da flag de Carry, é irrelevante.
O mesmo não se passa, se quisermos utilizar operandos com mais de oito bits.

Altere agora o seu programa de forma a efectuar a seguinte operação de adição, e comprove a afirmação que se acabou de fazer.

Parcela 1: R3:R2
Parcela 2: R5:R4

Resultado: R7:R6

Para o efeito, corra o programa para o seguinte conjunto de valores:

R0:R2	R1:R3
00:01	01:05
01:02	01:09
00:A8	00:59
00:CD	00:D0

Exercício 3

Objectivo:
Implementação de operações de **Block Move**

Introdução:

Este tipo de operações é conhecido por Block Move. Elas permitem copiar blocos de informação de uma zona de memória para outra.

Funcionamento 1

Escrever uma rotina que mova um bloco de informação com um número de bytes especificado em R7, da posição de memória apontada por R0, para a posição de memória apontada por R1.

O valor do acumulador não deve ser destruído.

Ter em atenção que é necessário preservar o espaço destinado à Stack, sob pena de perder o controlo da execução.

Funcionamento 2

Mover um bloco de N bytes de uma zona de memória externa para uma zona de memória interna.

Neste caso, o interface da rotina é diferente do descrito inicialmente.

ssim, considere:

R7 - tamanho do bloco de dados a mover
DPTR - Endereço do buffer origem
R5 - Endereço do buffer destino

OBS: O valor do Acumulador deve ser preservado

Exercício 4

Objectivo

Exemplificar a utilização de endereçamento indirecto

Introdução

A utilização de endereçamento indirecto, revela-se útil em inúmeras situações.
Este tipo de endereçamento recorre ao conceito já familiar de apontador, que consiste na utilização de um registo da máquina, cujo conteúdo indica o endereço de memória que queremos processar.

Funcionamento

O programa deve somar dois valores de 16 bits, localizados respectivamente a partir dos endereços 20h e 22h da memória interna, colocando o resultado a partir da posição 24h.

Observação

Ao usar a técnica sugerida, com pequenas alterações o programa estará apto a efectuar somas de operandos de qualquer comprimento, localizados em qualquer local da memória.

Exercício 5

Objectivo:
Escrever uma rotina para empacotar dígitos BCD, colocados num buffer de memória interna, deixando o resultado num outro buffer também situado em memória interna.

Introdução:
A utilização de representação numérica em formato BCD empacotado, tem como principal vantagem o facto de a informação ser legível por um operador humano, além do facto de representar uma poupança do espaço de memória necessário de 50%.

Funcionamento 1:
O buffer origem, que contém os dígitos a empacotar, apenas contém dígitos na gama [0..9], expressos em BCD, e de tal forma organizados que cada dígito ocupa uma posição de memória.

Desta forma, o valor 1234, está armazenado em quatro posições de memória consecutivas, da seguinte forma:

01, 02, 03, 04

A rotina a escrever, deve endereçar o buffer origem, utilizando para o efeito um apontador para memória interna, e compactar a informação de cada duas posições consecutivas de memória, num único byte situado num buffer destino, ficando então a informação organizada em dois dígitos BCD por cada posição de memória.

O buffer destino encontra-se igualmente situado em memória interna, e deverá ser endereçado por um segundo apontador de 8 bits.
Após a operação de compactação, a informação do buffer destino deverá ser (para o caso do exemplo anterior):

12, 34

Para testar o funcionamento da rotina, introduza valores manualmente a partir do endereço base do buffer origem .

O comprimento do buffer origem (expresso em número de bytes a compactar) é tratado através de um registo interno do micro controlador, sendo o processo de conversão terminado, quando o número de bytes a compactar fôr igual a zero.

Funcionamento 2
Escreva um programa funcionalmente equivalente ao anterior, com a diferença de que o buffer destino, encontra-se situado num endereço de memória de dados externa, à sua escolha.

Funcionamento 3
Escreva um programa funcionalmente equivalente aos anteriores, mas em que ambos os buffers, (de origem e destino) se encontram instalados em memória de dados externa.

Exercício 6

Objectivo
Escrever um programa para comparação de strings

Introdução
A comparação de strings é uma operação que executa a comparação byte a byte entre duas cadeias de caracteres.

Funcionamento 1:
A rotina base de comparação a desenvolver, deve efectuar a comparação de duas cadeias de caracteres, uma localizada em memória interna e outra localizada em memória externa, as quais têm o mesmo tamanho e são terminadas por um caracter nulo.

O interface com a rotina deve ser o seguinte:

entradas:
R7 - Início da cadeia de caracteres em memória interna
R5:R6 - Início da cadeia de caracteres em memória externa
saídas:
Cy = 1 SSE as strings forem diferentes

Funcionamento 2:
Se o resultado da operação indicar que as strings são diferentes, então a string inicialmente apontada por R5:R6 deve ser copiada para a zona de memória apontada inicialmente por R7. Isto pressupõe por isso, que terá de guardar em lugar seguro a informação existente nestes registos. Note que pode utilizar para este efeito a rotina de block move escrita anteriormente.

Caso o resultado indique as strings são iguais, então deve ser colocado na porta 1, o valor ffh.

Funcionamento 3:
Altere o seu programa por forma a que as duas strings em questão estejam ambas em memória externa.