SEL0614- APLICAÇÃO DE MICROPROCESSADORES

Atividade prática de uso de set de instruções e manipulação de dados em registradores e endereços de memória em microcontroladores

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.

Conceitos: sistemas embarcados, microprocessadores, microcontroladores, arquitetura e organização de computadores, set de instruções, CISC, RISC, memória de dados, memória de programa, registradores, clock, ciclos de máquina, linguagem Assembly, *timer*, *counter*, *interrupt*, display, I/O.

Material relacionado: Cap. 1 - Rev. de Org.Comp. e Microcontroladores; Cap. 2 - MSC-51

I - **Motivação e aplicação prática**

O propósito desta atividade é o exercício da compreensão sobre o funcionamento dos microcontroladores, isto é, como ocorre a interação entre seus elementos e circuitos internos, como sua arquitetura é disposta, e como os microcontroladores são programados. Neste contexto, uma forma de obter essas respostas é por meio do estudo em “baixo-nível”, isto é, a manipulação direta de dados bit a bit em registradores e endereços de memória, o estudo de arquiteturas de 8 bits, do set de instruções e a programação em Assembly (linguagem de baixo nível). Ainda que seja uma forma primitiva de obter informações, ela permite conhecer as particularidades de cada hardware e o funcionamento do sistema computacional em um nível de detalhamento não disponível em sistemas “alto-nível”. Ademais, o uso de uma ferramenta de simulação computacional, tal como o EdSim51, potencializa essa vantagem, uma vez que permite visualizar o funcionamento seccionado do microcontrolador e sua

estrutura interna (uma analogia a uma vista em “corte”). Por fim, cumpre frisar que a família de microcontroladores MCS-51 é totalmente aderente a essa “filosofia” e permite materializar os conceitos acima referidos de forma simplificada ao mesmo tempo em que não abstrai conceitos importantes de arquitetura e organização de computadores.

II - Atividade prática

Portanto, essa atividade irá versar sobre a manipulação básica de dados em registradores e endereços de memória utilizando o simulador EdSim51 e valendo-se do set de instruções do MCS-51. O objetivo é exercitar o uso de registradores de propósito geral e de função especial, instruções lógicas, aritméticas, booleanas, incondicionais e condicionais. Tal exercício irá subsidiar as etapas subsequentes de desenvolvimento de projetos práticos de sistemas embarcados utilizando funcionalidades de microcontroladores (temporização, contagem de eventos, interrupção, ativação de entradas e saídas etc.).

A atividade prática poderá ser realizada durante a próxima aula, podendo ser feita em duplas.

O primeiro passo é abrir o simulador **EdSim51** no PC do laboratório (ou no seu computador após ter feito o download aqui).

(a) - Considerações iniciais sobre a estrutura de programas em nível de instrução (Assembly)

- **Label:** um rótulo ou etiqueta para identificar um bloco de instruções, a qual pode ser usada como ponto de referência para salto, para um loop e para realizar funções: função principal (main), auxiliar, específica (para realizar determinado cálculo, por exemplo). Desde que não seja uma palavra reservada (MOV, por exemplo) e não use caracteres especiais da língua portuguesa, qualquer palavra pode ser usada para formar uma Label, conforme exemplo a seguir:

```
        org         0000h       ; Origem
inicio:                         ; Label chamada “inicio” (poderia ser qualquer outro nome)
        MOV         R0, #02h    ; Move o valor 2 para  R0
        MOV         022h, R0    ; Move o conteúdo de R0 para o endereço de memória 022h
        DEC         R0          ; Decrementa R0 em 1 unidade
        JMP         inicio      ; Salto para label inicio, ou seja, será executado..
                                ;…novamente o programa a partir da primeira linha de
                                ;...instrução após a label init, deixando o ;programa em
                                ;…loop.

        end                     ; Necessário para indicar o final do programa
```

- Note que a programação em nível de instrução não é *case sensitive*, isto é, **Mov, mov, MOV, MoV**, são equivalentes, **org, ORG**, ou **End** e **END**, também. Da mesma forma, os endereços podem ser informados: **022h**, ou **022H**.

- A operação **NOP** é usada para consumir tempo de 1 µs, indicando que nenhuma operação será realizada naquela linha de código (uma forma primitiva de causar delays, por exemplo). Enquanto o símbolo "**$**", refere-se ao endereço atual. Logo:

```
        org         0000h
main:
        MOV         R0, #02h    ; Move o valor 2 para R0 - duração: 1 µs (1 ciclo)
        MOV         022h, R0    ; Move o conteúdo de R0 para a posição 22h - 2 µs (2 ciclos)
        ADD         A, 022h     ; Move o conteúdo da posição 22h para o ACC - 1 µs (1 ciclo)
        INC         A           ; Incrementa o ACC em 1 unidade -1 µs (1 ciclo)
        SUBB        A, R0       ; Subtrai o valor de R0 do ACC-1 µs (1 ciclo)
        RL          A           ; Rotaciona A à esquerda em 1 bit - 1 µs (1 ciclo)
        NOP                     ; Nenhuma operação executada (1 µs)
        JMP         $           ; Ao invés de retornar para a label inicio, neste exemplo…
                                ;...o programa será executado e "segurado" nesta linha…
                                ;Ou seja:  "jump to current address" -  2 µs (2 ciclos)
        end                     ;Fim do programa
```

- **Sugestão para organização programas**: primeira coluna para as labels, a segunda (separar com TAB) para instruções, a terceira para valores, endereços e registradores (destino, origem, byte etc.), e a quarta coluna para comentários. Contudo, não é obrigatório o uso de indentação, pois não interfere no funcionamento do programa.

III - Roteiro da atividade prática

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

- ➔ Criar um novo programa clicando em "New".
- ➔ Colocar a origem no endereço 0000h
- ➔ Inicializar o programa com uma label (ex. inicio/main etc.)
- ➔ Mover de forma imediata um valor qualquer em hexadecimal (de 00 a FF) para o registrador acumulador (A ou ACC)
- ➔ Mover de forma imediata o valor zero para ACC;
- ➔ Mover de forma imediata um valor para um registrador qualquer (escolher um entre R0 a R7) no banco 00 (primeiro banco - verificar bits seletores dos bancos em PSW: RS0 e RS1).
- ➔ Mover de forma imediata um valor qualquer em hexadecimal para o registrador B

- ➔ Mover a porta P1 para um endereço de memória RAM qualquer (entre 00 a 7F).
- ➔ Mover de forma direta o conteúdo da posição de memória escolhida na linha anterior para um registrador qualquer do Banco 01 (segundo banco).
- ➔ Mover o conteúdo do registrador escolhido para um outro endereço de memória qualquer (ex.: MOV endereço, Rx)
- ➔ Apontar o endereço de memória escolhido na linha anterior como valor para R1, ou seja: Mover de forma imediata (com #) o endereço de memória escolhido na linha anterior (ex.: Se o endereço foi 65h, então #65h será movido, passando a ser um valor dentro de R1 e não mais um endereço).
- ➔ Mover R1 de forma **indireta** para o acumulador (ou seja, usar R1 como um "ponteiro").
- ➔ Mover de forma imediata o valor 9A5B (4 dígitos) para o registrador DPTR.
- ➔ Consumir tempo de 1 µs sem nenhuma operação e segurar o programa na próxima linha.
- ➔ Encerrar o programa.
- ➔ Depurar o programa clicando em "Assm" (assembly source code - montagem das instruções executando linha por linha) e executar cada "Step". Visualizar o resultado da manipulação de dados na memória RAM ("Data Memory") e nos Registradores.
- ➔ Salvar o programa e responder as questões a seguir:

  (a) - Qual foi o tempo gasto em cada linha de instrução e o tempo total em µs?

  (b) - Quantos ciclos de máquina esse programa contém ? (Justifique sua resposta);

  (c) - O que aconteceu ao mover uma porta inteira de 8 registradores (como: "MOV A, P1", no exemplo) para um destino e porque seu valor é FF ? (consulte a página 7 do datasheet AT89S51 Atmel que versa sobre a inicialização de registradores - lembrando que o MCS-51 possui 4 portas: P1, P2, P3, P4).

  (d) - Qual valor apareceu no acumulador após ter movido R1 de forma indireta para ele?

  (e) - Por que foi possível mover um valor de 4 dígitos para DPTR? Em quais registradores especiais do simulador foi possível verificar mudanças quando a operação foi realizada? Qual o maior valor que pode ser movido para DPTR em hexadecimal?
- ➔ **Formato da resposta:** apresentar as linhas de código comentadas, colocando o tempo de duração de cada linha (ex: 1µs; 2µs..) . Ao final, responder às questões acima.. Ex.: "Resposta da questão (a) …".

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

- ➔ Criar um novo programa
- ➔ Colocar a origem no endereço 00h
- ➔ Inicializar o programa com uma label
- ➔ Mover de forma imediata o valor 2 em **decimal** para o ACC.
- ➔ Mover de forma imediata o valor 3 em **decimal** para B
- ➔ Mover para um endereço de memória qualquer o valor imediato 7 em decimal
- ➔ Somar o conteúdo do endereço de memória escolhido na linha anterior com ACC.
- ➔ Decrementar 3 unidades de ACC
- ➔ Incrementar 1 unidade em B
- ➔ Subtrair A por B
- ➔ Multiplicar A por B
- ➔ Incrementar 2 unidades em B
- ➔ Dividir A por B
- ➔ Armazenar os conteúdos de A e B em dois endereços de memória quaisquer na RAM.
- ➔ Saltar para label declarada no "inicio"
- ➔ Encerrar o programa.
- ➔ Depurar o programa e observar os valores em ACC e B a medida que as operações são executadas.
- ➔ Salvar o programa.
- ➔ **Realiza o seguinte teste:** em um novo programa, mover de forma imediata o valor 4 para o ACC; na linha seguinte mover de forma imediata o valor 3 para o ACC. Execute as duas linhas clicando em "Assm", observando PSW. Porque ao mover o valor 4 para ACC, o bit menos significativo de PSW resulta em 1; e ao mover o valor 3 esse bit resulta em 0?(**OBS**. Não é necessário salvar esse novo programa, somente execute a operação para responder a questão).
- ➔ **Formato de resposta:** Apresentar as linhas de código comentadas e a resposta à questão acima ao final.

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

- ➔ Criar um novo programa
- ➔ Colocar a origem no endereço 00h
- ➔ Inicializar o programa com uma label
- ➔ Mover de forma imediata para o ACC e para B dois valores em **binário** (8 bits), distintos (evitar escolher valores em que todos os dígitos são iguais: 11111111 ou 00000000, por ex., e escolher de forma que pelo menos 1 bit seja igual na mesma posição entre os dois valores. Por ex.: ambos compartilham bit 1 no dígito menos significativo ou em alguma outra posição).
- ➔ Realizar AND lógico entre A e B
- ➔ Rotacionar A à direita em 2 bits.

- ➔ Realizar o complemento de A
- ➔ Rotacionar A à esquerda em 2 bits.
- ➔ Realizar OR lógico entre A e B
- ➔ Realizar XOR entre A e B
- ➔ Realizar SWAP de A;
- ➔ Saltar para a label inicial
- ➔ Encerrar o programa.
- ➔ Em "*bit field information*" no simulador EdSim51 (onde geralmente é exibido PSW no formato binário), colocar ACC no lugar de PSW.
- ➔ Depurar o programa e observar os valores em ACC em binário à medida que as operações lógicas são executadas.
- ➔ Salvar o programa
- ➔ **Formato de resposta**: apresentar as linhas de código comentadas.

4- Manipulação de dados em registradores e endereços de memória por meio de instruções de desvio incondicional e condicional:

- ➔ Criar um novo programa
- ➔ Colocar a origem no endereço 00h
- ➔ Saltar para a label do programa principal (main)
- ➔ Colocar a origem em 33h
- ➔ Inicializar o programa principal com a label informada anteriormente na operação de salto.
- ➔ Limpar o ACC
- ➔ Mover de forma imediata um valor qualquer para R0
- ➔ A partir daqui, separar o programa em blocos de códigos por meio de outras labels. Primeiramente, inicializar o primeiro bloco com uma nova label (ex.: "bloco1")
- ➔ Saltar **SE** A = 0 para um o segundo bloco do programa (informar a label do segundo bloco nesta instrução. Ex.: bloco2)
- ➔ Na próxima linha (ainda no primeiro bloco), Saltar SE A ≠ 0 para um terceiro bloco (passar a label do terceiro bloco nesta instrução. Ex.: bloco3)
- ➔ Na próxima linha (ainda no primeiro bloco), consumir tempo de 1 μs (não realizar operação).
- ➔ Inicializar o segundo bloco com a label chamada anteriormente (para onde o programa irá saltar se A =0)
- ➔ Mover R0 para A
- ➔ Saltar de forma incondicional para o bloco 1 (passar a label do primeiro bloco).
- ➔ Inicializar o terceiro bloco com label chamada anteriormente (para onde o programa irá saltar se A ≠ 0)
- ➔ Decrementar e Saltar SE R0 ≠ 0 para a label do terceiro bloco (isto é, irá ficar em loop enquanto R0 ≠ 0, e sempre no terceiro bloco)
- ➔ Na próxima linha (no terceiro bloco), saltar de forma incondicional para a label do programa principal para reiniciar toda a operação.

➔ Encerrar o programa.
➔ Depurar o programa e descrever seu comportamento.
➔ Salvar o programa
➔ **Formato de resposta:** apresentar as linhas de código comentadas.

- **Formato de entrega da atividade:** apresentar as linhas de código (programa devidamente comentado e, quando for caso, as respostas às perguntas específicas ao final do programas), em um documento (pode ser: arquivo PDF, ou "Readme.md" do Github, ou um vídeo curto com explicação breve e objetiva das linhas de código e instruções de cada um dos 4 exercícios acima, compartilhando a tela do computador com a execução dos programas no EdSim51). O documento de respostas deve seguir a ordem do roteiro. Por exemplo:

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

  <resposta> ***;****(linhas de código comentadas)*

  2 - Manipulação de dados em registradores e endereços de memória por meio de instruções Aritméticas:
  <resposta> ***;****(linhas de código comentadas)*

  3 - " " …

OBS. Não serão consideradas as soluções em que as linhas de código não estejam devidamente comentadas com uma explicação de cada operação (vide programa de exemplo acima). Não será necessária a entrega dos códigos fontes em arquivos separados (.asm), referente aos programas gerados para cada exercício acima**. A entrega deverá ocorrer pelo e-Disciplinas até a data especificada na tarefa atribuída. Os trabalhos poderão ser realizados em duplas.**

Critérios de avaliação: entrega no formato solicitado, sequência lógica dos códigos conforme o roteiro, comentários nas linhas, e uso correto das instruções e modos de endereçamento nas diferentes operações: transferência de dados, aritméticas, lógicas e de desvio.