SEL0433
APLICAÇÃO DE MICROPROCESSADORES

Parte 1 - Sistemas Embarcados e Microcontroladores
Atividade Semanal 4

Pré-aula

- **Material de aula: "**Cap. 3" (Objetivos da aula: estudar arquitetura; analisar pinagem, organização de memória e set de instruções em microcontroladores com uso de simulador). "Cap. 4" (objetivos da aula: programar em nível de instrução as interfaces I/O, pilhas, sub-rotinas e periféricos de microcontroladores)
- Ver **Datasheet AT89S51 (Atmel)** https://www.keil.com/dd/docs/datashts/atmel/at89s51_ds.pdf
- Set de instruções 8051: set completo; tabela resumida

- **Códigos com conceitos iniciais (fazer download abrir no simulador durante a aula)**
- **Exemplos de códigos em Assembly (fazer download e abrir no EdSim51 durante a aula)**

Aula e pós-aula

- A atividade prática abaixo deverá ser realizada durante a aula (que será destinada somente para essa finalidade) e o que faltar deverá ser realizado como tarefa (ou o contrário, caso a atividade seja atribuída na semana anterior a aula em questão).

Objetivos

- Revisão de conceitos
- 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.
- Realizar operações com instruções de transferência de dados, lógicas, aritméticas, booleanas, incondicionais e condicionais usando a família MCS-51.

Instruções para realização da atividade

A atividade prática deve ser realizada em duplas. O primeiro passo é abrir o simulador **EdSim51** no PC do laboratório (ou no seu computador após ter feito o download aqui conforme instruções das aulas anteriores). Além dos conceitos das aulas anteriores, o material de suporte para realização da atividade está indicado no tópico "Pré-aula".

(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* para comandos, 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 início, 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.

(b) Roteiro

Atividade 1

1. **Criação e Inicialização do Programa**
   - Criar um novo programa clicando em "New".
   - Colocar a origem no endereço 0000h.
   - Inicializar o programa com a label main.
2. **Transferência de Dados**
   - Mover de forma imediata o valor 12 em hexadecimental (formato default – quando não for especificado qual formato, sempre usar este) para o acumulador (A).
   - Mover de forma imediata o valor zero para A.
   - Mover de forma imediata o valor 34 em hexadecimental para o registrador R2 no banco 0.
   - Mover de forma imediata o valor 56 em decimal para o registrador B.
   - Mover a porta P1 para o endereço de memória 0x40.
   - Mover de forma direta o conteúdo da posição de memória 0x40 para o registrador R4 do banco 1.
   - Mover o conteúdo de R4 para o endereço de memória 0x50.
   - Mover de forma imediata o valor 0x50 em binário para R1.
   - Mover R1 de forma indireta para o acumulador (usar @R1).
   - Mover de forma imediata o valor 0x9A5B para o registrador DPTR.
3. **Instruções Aritméticas**
   - Mover de forma imediata o valor 2 para o ACC.
   - Mover de forma imediata o valor 3 para B.
   - Mover de forma imediata o valor 7 para R4
   - Somar o conteúdo de R4 com o ACC.
   - Decrementar 3 unidades do 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 nos endereços de memória 0x70 e 0x71.

4. **Instruções Lógicas e Booleanas**
   - Mover de forma imediata os valores 0b11001100 e 0b10101010 para A e B, respectivamente.
   - 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.
   - Em "*bit field information*" no simulador EdSim51 (onde geralmente é exibido PSW no formato binário), colocar ACC no lugar de PSW e observar os valores binários resultantes das operações neste registrador.
5. **Instruções de Desvio Condicional e Incondicional**
   - Criar uma label de referência para início
   - Limpar o ACC.
   - Mover de forma imediata o valor 0x10 para R0.
   - Saltar SE A = 0 para o bloco2 (label).
   - Saltar SE A ≠ 0 para o bloco3.
   - Consumir tempo de 1 µs sem nenhuma operação.
   - Inicializar bloco2:
     - Mover R0 para A.
     - Saltar de forma incondicional para bloco1.
   - Inicializar bloco3:
     - Decrementar e Saltar SE R0 ≠ 0 para bloco3.
     - Saltar de forma incondicional para inicio
6. **Encerramento do Programa**
   - Segurar o programa na última linha.
   - Encerrar o programa.
   - Todos os programas podem ser estruturados em um único código e pode-se usar instruções de salto ou sub-rotinas para navegar entre eles (caso preferir, escrever os programas dos 5 itens anteriores em scripts separados).

Depuração:

1. **Montar e Executar o Programa:**
   - Clique em "Assm" para montar o código.
   - Execute cada linha clicando em "Step" e observe os resultados na memória RAM e nos registradores.
   - Salve o programa.

2. **Questões:**

   Sobre Transferência de Dados:

   - **(a)** Qual foi o tempo gasto em cada linha de instrução e quantos ciclos de máquina esse programa contém? Justifique sua resposta.
   - **(b)** O que aconteceu ao mover uma porta inteira de 8 registradores (ex.: MOV A, P1) para um destino e por que seu valor é FF?

- **(c)** Qual valor apareceu no acumulador após ter movido R1 de forma indireta para ele?
- **(d)** Por que foi possível mover um valor de 4 dígitos para DPTR? Em quais registradores especiais do simulador foi possível verificar mudanças quando essa operação foi realizada? Qual o maior valor que pode ser movido para DPTR em hexadecimal?

Sobre Instruções Aritméticas:

- **(e)** Faça os seguintes testes em um programa a parte:
    1. Por que 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?
    2. Tente decrementar 1 unidade de algum registrador ou endereço de memória cujo valor é igual a zero (ex.: DEC A, DEC Rn, ou DEC 60h, sendo A, Rn, ou 60h iguais a zero). Por que a operação resulta em FF?

Atividade 2

Verificar sequencialmente o conteúdo das posições de memória de 20h até 23h (por exemplo) e incrementar um registrador com a quantidade de valores menores do que #45h (por exemplo) contidos nestas posições de memória.

- ➔ Criar um novo programa
- ➔ Colocar a origem no endereço 00h
- ➔ Saltar para a label do programa principal (main)
- ➔ Colocar a origem em 33h
- ➔ Na label principal, inicializar R0 com valor #20h; e R1 com #0;
- ➔ Criar uma label chamada LOOP (ou com qualquer outro nome – será um ponto de retorno)
- ➔ Mover R0 de forma indireta para A;
- ➔ Subtrair #45h de A
- ➔ Saltar de forma condicional, se não houver carry, para uma terceira label (operação com bit - verificar o bit de flag carry de PSW e salta se = 0)
- ➔ Incrementar 1 unidade em R1 (ou seja, se #45h for maior do que A, o resultado da subtração é negativo e carry será = 1, portanto o salto da linha anterior não irá acontecer e o programa executará essa linha, sinalizando a quantidade de valores maiores do que #45h)
- ➔ Atribuir a terceira label, para onde o programa irá saltar segundo a condição atribuída anteriormente
- ➔ Incrementar 1 unidade em R0 (incrementa o ponteiro para próxima posição de memória)
- ➔ Compara R0 com #24h e salta para LOOP se não forem iguais (verificar se chegou na última posição de memória, 23h +1, a ser testada)
- ➔ Nenhuma operação
- ➔ Segurar o programa nesta linha

- ➔ Depurar o programa e descrever seu comportamento (atribua manualmente valores aleatórios, maiores e menores do que #45h, nas posições de memória de 20h à 23h para testar o programa no simulador EdSim51, verificando ao final se a quantidade em R1 está correta)
- ➔ Salvar o programa
- ➔ Formato de resposta: apresentar as linhas de código comentadas.

(c) Formato para entrega

Apresentar as linhas de código (programa devidamente comentado e, quando for caso, as respostas às perguntas específicas ao final do programa), em um documento conforme orientações específicas disponíveis na tarefa correspondente atribuída no e-Disciplinas. O documento de respostas deve seguir a ordem do roteiro. Por exemplo:

Atividade 1

Transferência de dados:
<resposta> ***;****(linhas de código comentadas)*
…

<resposta para as questões>
…

Atividade 2
…

OBS. Não serão consideradas as soluções em que as linhas de código não estejam devidamente comentadas. 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.**

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