SEL0433/SEL0336/SEL0614
APLICAÇÃO DE MICROPROCESSADORES

Roteiros de Aulas, Atividades e Projetos
Parte 1 - Sistemas Embarcados e Microcontroladores

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, pilha, sub-rotinas, timer, counter, interrupt, display, I/O digital.

Motivação e objetivos: O propósito desta primeira parte do curso é compreender o funcionamento dos microcontroladores, especialmente dentro do contexto dos sistemas embarcados, diferenciando-os das aplicações de microprocessadores em computadores de propósito geral. O objetivo é entender a interação entre os elementos e circuitos internos dos microcontroladores, a disposição de sua arquitetura e sua programação, por meio do estudo em "baixo-nível", o qual envolve a manipulação direta de dados bit a bit em registradores e endereços de memória. A família de microcontroladores MCS-51 é destacada por sua aderência a essa abordagem. Os principais recursos abordados incluem a manipulação básica de dados, operações com instruções, utilização de pilha e sub-rotinas, manipulação de portas paralelas de entrada e saída, uso de ferramenta de simulação computacional e o desenvolvimento de um projeto prático que explore esses recursos.

Instruções e datas para entrega de atividades/projetos: todas as atividades podem ser realizadas em grupos de duas pessoas. Não é permitido realizá-las de forma individual. Instruções específicas e datas de entrega estarão contidas em documento específico na tarefa correspondente atribuída no e-Disciplinas (Moodle).
Material de apoio: além dos links disponibilizados diretamente no presente documento, o material principal estará também disponível no e-Disciplinas.
Instruções para as aulas: As ferramentas computacionais utilizadas no curso estão disponíveis nos computadores do laboratório. Durante o horário da aula, os computadores são de uso exclusivo para realização das atividades pertencentes à disciplina e não devem ser utilizados para outros fins. **Importante:** ao final da aula ou ao término do uso dos computadores, não esqueça de fazer logout de contas pessoais que eventualmente tenha acessado durante seu uso (email, e-Disciplinas etc), desligar os computadores e organizar mesas e cadeiras que utilizou.
Orientações específicas deste documento:

- **Pré-aulas:** atividades ou material de estudo preparatório para o conteúdo que será abordado na aula, os quais deverão ser acessados preferencialmente com antecedência ou no início da aula. Apesar disso, também servirão para acompanhamento durante a exposição da aula e como material de consulta pós-aula para realização de atividades/projetos.
- **Pós-aulas:** refere-se a atividades, tarefas ou projetos que devem ser realizados após a aula com objetivo de reforçar o conteúdo e consolidar conceitos. **Questões:** são apresentadas neste documento como recurso para reflexão, bem como para que o conteúdo não seja apenas absorvido de forma passiva pela aula expositiva. Podem ser resolvidas todas como uma lista de exercícios até a data de entrega atribuída no e-Disciplinas. Apesar da possibilidade de resolução como tarefa, de forma assíncrona, a sugestão é que seja abordado um grupo de questões a cada aula, seguindo o cronograma apresentado (já que todas elas serão, de algum modo, respondidas durante as aulas) e permitindo melhor distribuição de carga de trabalho quando da entrega da atividade.

Monitoria: será possível agendar esclarecimentos de dúvidas e atendimento para auxílio nas atividades/projetos em outros horários, com o professor e com monitores. Da mesma forma, podem utilizar os laboratórios (LEI Maior ou Lab. de Microprocessadores) em horários que não estão sendo usados para aulas para realizar atividades da disciplina (caso necessitem de computadores e uso das ferramentas computacionais), preferencialmente agendando previamente com técnicos responsáveis, monitores e/ou com professor.

Roteiros

Aula 1

Pré-aula

- **Leitura do Programa da disciplina (disponível no e-Disciplinas):** "Cap. 0 - Introdução ao curso" (assuntos tratados na disciplina, cronograma das aulas, projetos, ferramentas utilizadas, material de apoio, informações sobre monitoria, critérios de avaliação etc.)
- **Visão geral de algumas das ferramentas do curso:** EdSim51; kit EasyPIC; MikroC PRO for PIC; SimulIDE; Wokwi; ESP32.
- **Motivação para a disciplina e reflexões sobre sistemas embarcados (disponível no e-Disciplinas):** Cap. 1 - Sistemas Embarcados (características, requisitos, arquiteturas, projeto, destaques e aprofundamento).
- **Mapa mental: "Embedded Systems"; "Embedded Computing"**
- **Leitura dos relatórios de pesquisa sobre o mercado de sistemas embarcados no Brasil e no mundo:**
    - Relatório sobre o Mercado Brasileiro em Sistemas Embarcados em 2023 https://embarcados.com.br/relatorio-da-pesquisa-sobre-o-mercado-brasileiro-de-sistemas-embarcados-e-iot-2023/
    - Embedded Survey- "The current state of embedded development"- 2023 https://www.embedded.com/wp-content/uploads/2023/05/Embedded-Market-Study-For-Webinar-Recording-April-2023.pdf

Pós-aula

Questão 1

Apresentar a definição formal de um sistema embarcado, indicando a referência primária do IEEE (Institute of Electrical and Electronics Engineers) que subsidie esta definição. Em seguida fazer uma explanação breve e objetiva sobre as principais características, funcionalidades e o que difere um sistema embarcado de um computador de propósito geral.

Questão 2

De acordo com os Relatórios de Pesquisa sobre o Mercado de Sistemas Embarcados acima referidos (listar os itens abaixo com base nos 2 relatórios disponibilizados):

I. Das ferramentas para sistemas embarcados – quais as principais áreas de aplicação dos projetos no mercado brasileiro e o cenário internacional?
II. Quais as principais ferramentas de comunicação sem fio que estão sendo usadas no Brasil e no mundo?
III. Quais os principais kits/plataformas de prototipagem usados?
IV. Dos softwares para sistemas embarcados - qual a principal ferramenta de codificação, principal sistema de controle de versão, e principal linguagem de programação?
V. Dos microprocessadores/microcontroladores – quais os fabricantes/modelos mais citados na pesquisa?

Aula 2

Pré-aula

- **Material de aula: "**Cap. 2 - Microcontroladores" (Objetivos da aula: Revisar arquitetura de computadores, CISC vs. RISC, Harvard vs. Von Neumann, explorar a organização e periféricos, famílias e principais soluções de microcontroladores)
- **Conhecer o simulador EdSim51:** acessar a página http://www.edsim51.com (para download, guia de uso, exemplos etc.). Trata-se de uma ferramenta aberta e gratuita.
    - **OBS.** A ferramenta encontra-se disponível nos computadores do laboratório. Buscar pelo atalho "edsim51" ou "edsim51di", ou pelo ícone "edsim51di.jar" (java), geralmente na unidade de disco local "C"/ Arquivos de Programa; ou na área de trabalho. Para abrir o simulador, deve-se clicar no ícone "edsim51di.jar".
- **Diagramas: Mapa mental;**
- **Texto: Microcontroladores - Introdução**

Pós-aula

- **Material complementar sobre organização de computadores (apenas sugestão para aprofundamento e/ou revisão caso achar necessário)**
    - ➔ Mapa mental - microprocessador
    - ➔ Revisão de arquitetura e organização de computadores: https://edisciplinas.usp.br/course/view.php?id=80119 (curso Prof. Marcelo –aulas gravadas e material) http://inf.ufes.br/~zegonc/material/Introducao_a_Computacao/Microprocessadores%20-%20Parte%201.pdf (material sobre Org.Comp)

Questão 3

Recorra ao exemplo do microcontrolador aplicado ao controle de um elevador que foi apresentado em aula, disponível nas transparências do Cap. 2. Quais as vantagens de se utilizar um microcontrolador para aquele tipo de aplicação e qual deve ser o "perfil" de um microcontrolador ideal para aquela aplicação do elevador em termos de capacidade da CPU (baixa, média ou alta), quantidade de bits no barramento, e precisão no tratamento das informações (operação somente com inteiros ou ponto flutuante?)

Questão 4

Quanto às portas paralelas de um microcontrolador:
( ) São somente de entrada.

( ) São somente de saída.
( ) Cada palavra (A, B, C, P1, P2, P3…) pode ser configurada como entrada ou saída.
( ) Cada bit pode ser configurado como entrada ou saída.
( ) Cada palavra (A, B, C... P1, P2, P3…) pode ser configurada como entrada, saída ou bidirecional.
( ) Cada bit pode ser configurado como entrada, saída ou bidirecional.

Questão 5

Assinale V para verdadeiro e F para falso nas afirmações abaixo:
( ) No modelo de Von Neumann, o microprocessador segue as instruções armazenadas na memória ROM (programas), lê as entradas e envia comandos sobre os canais de saída, alterando as informações contidas na memória RAM.
( ) Os registradores Special Function Registers localizam-se sempre internos à CPU.
( ) O ciclo de máquina é composto pelo ciclo de busca mais o ciclo de execução, cada qual demorando um pulso de clock.
( ) A instrução "CLR A" não possui operando e gasta apenas 1 ciclo de máquina
( ) A arquitetura Von Neumann é considerada uma arquitetura mais simples do que a arquitetura Harvard porque utiliza o mesmo barramento para o tráfego de dados e de instruções.
( ) A técnica de pipeline é impossível de ser utilizada em computadores de arquitetura Von Neumann.

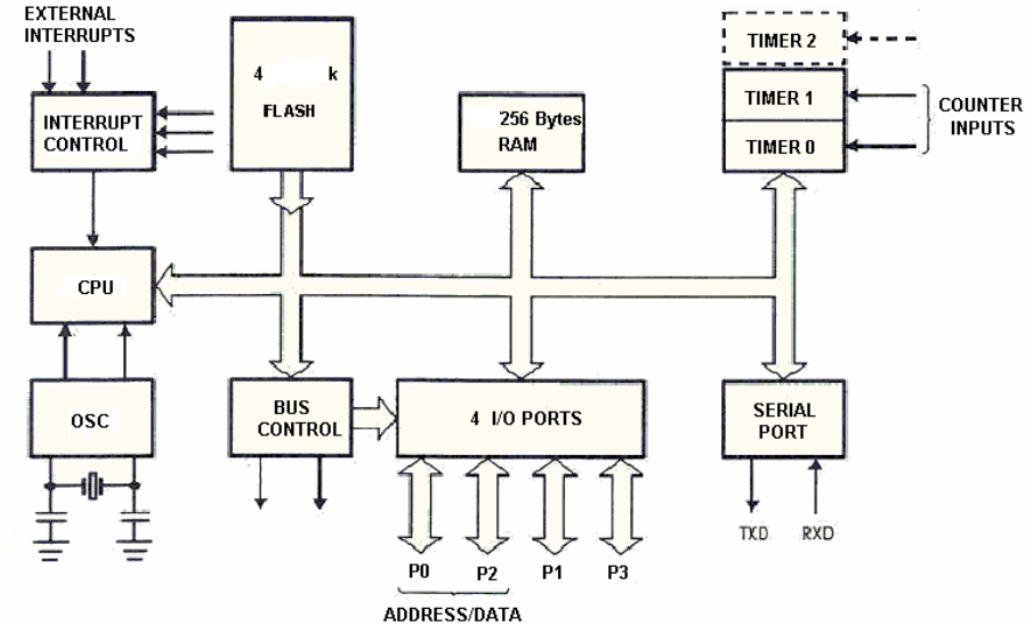
Questão 6

Indique quais afirmativas se aplicam a uma instrução CISC e quais a uma instrução RISC:

- ➔ Os programas são mais complexos
- ➔ A maioria das instruções tem a mesma duração
- ➔ Mais instruções disponíveis
- ➔ Programas menores
- ➔ Utiliza menos espaço na memória de programa
- ➔ Processamento de cada instrução é mais lento
- ➔ Microcontroladores PIC, AVR, ARM
- ➔ Tempo de execução das instruções depende da frequência do clock.

Questão 7

Abaixo é apresentado o diagrama de um microcontrolador. Qual a arquitetura utilizada e como chegamos a essa conclusão? Quantas portas I/O bidirecional e quantas linhas (bits/pinos) são endereçados de forma individual neste microcontrolador, com base no diagrama abaixo?


EXTERNAL
INTERRUPTS
INTERRUPT
CONTROL
4 k
FLASH
256 Bytes
RAM
TIMER 2
TIMER 1
TIMER 0
COUNTER
INPUTS
CPU
OSC
BUS
CONTROL
4 I/O PORTS
SERIAL
PORT
TXD
RXD
P0
P2
P1
P3
ADDRESS/DATA

Aula 3

Pré-aula

- **Material de aula: "**Cap. 3 - MCS-51" (Objetivos da aula: estudar arquitetura; analisar pinagem, organização de memória e set de instruções em microcontroladores com uso de simulador).
- Ver **Datasheet AT89S51 (Atmel)** https://www.keil.com/dd/docs/datashts/atmel/at89s51_ds.pdf
- **Set de instruções 8051**: set completo; tabela resumida
- Mapa mental 8051
- **Conhecer o simulador EdSim51:** acessar a página http://www.edsim51.com (para download, guia de uso, exemplos etc.). Trata-se de uma ferramenta aberta e gratuita.
    - **OBS.** A ferramenta encontra-se disponível nos computadores do laboratório. Buscar pelo atalho "edsim51" ou "edsim51di", ou pelo ícone "edsim51di.jar" (java). Para abrir o simulador, deve-se clicar no ícone "edsim51di.jar".
- **Códigos com conceitos iniciais (fazer download abrir no simulador durante a aula)**
- Apostila sobre MCS51

Pós-aula

Questão 8

No simulador EdSim51, digite e execute (clicando em "Assm") as instruções abaixo:

MOV R0, #22h
MOV 00h, #22h

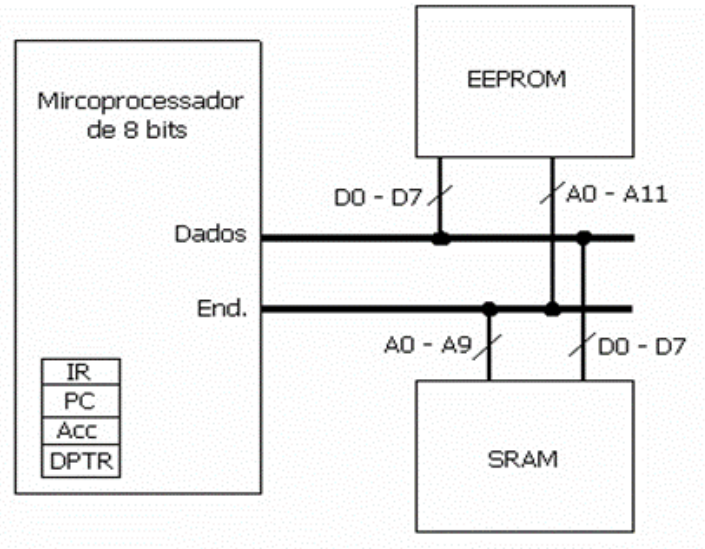
Qual a diferença entre as duas instruções acima? Tente refletir porque possuem ciclos de máquina diferentes se a operação é realizada na mesma posição de memória RAM (00h ou R0 usa o mesmo espaço).

MOV A, #22h
MOV ACC, #22h

Qual a diferença entre as duas instruções acima? Tente refletir sobre a diferença de usar A ou ACC e sobre porque possuem ciclos de máquina diferentes se a operação realizada é a mesma.

Questão 9

A Figura abaixo mostra um microcontrolador genérico de 8 bits com 4 registradores internos à CPU, os quais são: *Instruction Register* (IR), *Program Counter* (PC), *Accumulator* (ACC) e *Data Pointer* (DPTR). Baseado na Figura abaixo, responda às questões com verdadeiro (V) ou Falso (F):



( ) Trata-se de um microcontrolador de arquitetura Harvard.
( ) A memória EEPROM é de 4Kbytes e armazena as instruções que comandam o microcontrolador.
( ) A memória SRAM é de 512 bytes e armazena dados voláteis
( ) O registrador IR tem a função de armazenar a instrução lida da memória SRAM.
( ) Para esse microcontrolador, o registrador IR deve ser de 8 bits
( ) O registrador PC armazena o endereço da instrução lida da memória EEPROM.
( ) Para esse microcontrolador, o registrador PC deve ser de 10 bits.
( ) Para esse microcontrolador, o registrador ACC deve ser de 8 bits.
( ) O registrador DPTR é um ponteiro que aponta para a última instrução lida da memória.
( ) Para esse microcontrolador, o registrador DPTR deve ser de 10 bits**.**

Aula 4

Pré-aula

- **Material de aula: "**Cap. 3 - MCS-51" (Objetivos da aula: estudar arquitetura; analisar pinagem, organização de memória e set de instruções em microcontroladores com uso de simulador). "Cap. 4 - Programação em Assembly para MCS-51".
- 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)**

Aula e pós-aula (próxima página)

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

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 poderá ser realizada durante a aula e deverá ser concluída como tarefa caso o tempo de 1 aula não for suficiente. Deverá ser realizada em grupos de até 3 pessoas (não é permitido realizar de forma individual). 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" da Aula 4.

(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 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.

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 com nome de sua preferência (ex. inicio/main etc.)
- ➔ Mover de forma imediata um valor qualquer em hexadecimal (de 00 a FF) para o registrador acumulador (A);
- ➔ Mover de forma imediata o valor zero para A;
- ➔ 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 para seleção dos bancos conforme demonstrado em aula).
- ➔ 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: P0, P1, P2, P3).

  (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 essa 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. Exemplo: "Resposta da questão (a)," etc.

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:**
  - ◆ 1- 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).
  - ◆ 2 – Tente decrementar 1 unidade de algum registrador ou endereço de memória cujo valor é igual a zero (DEC A; ou DEC Rn; ou DEC 60h, por exemplo, sendo A, Rn ou 60h iguais a zero). Por que a operação resulta em FF?
- ➔ **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.

5 – Verificar sequencialmente o conteúdo das posições de memória de 20h até 23h e incrementar um registrador com a quantidade de valores menores do que #45h 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 esta correta)
- ➔ 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 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:

  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.**

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.

Aula 5

Pré-aula

- **Material de aula:** "Cap. 4 - Programação em Assembly para MCS-51" (objetivos da aula: programar em nível de instrução as interfaces I/O, pilhas, sub-rotinas e periféricos de microcontroladores)
- **Exemplos de códigos em Asssembly (fazer download e abrir no EdSim51 durante a aula)**

Pós-aula

Questão 10

Responder com Verdadeiro (V) ou Falso (F) às seguintes afirmações.
( ) A pilha é uma memória RAM sequencial do tipo FIFO.
( ) A pilha geralmente é utilizada para armazenar endereço de retorno de subrotinas e também de interrupções.
( ) O ponteiro de pilha (Stack Pointer) é um registrador que aponta para um endereço da memória ROM, que é o endereço de retorno do programa após o atendimento a uma interrupção ou sub-rotina.
( ) As instruções PUSH e POP são exclusivas para operações com pilha.
( ) A instrução CALL <endereço> deve ser usada para indicar qual endereço o programa deve desviar no caso de um atendimento à interrupção ou chamada de sub-rotina.
( ) A instrução RET, colocada no final de uma sub-rotina, faz com que o último endereço armazenado na pilha seja carregado no registrador PC (program counter).
( ) A área da RAM interna dedicada à pilha é determinada pelo ponteiro SP, um dos SFRs, que possui tamanho 8 bits, mesmo tamanho do barramento de endereço da CPU.
( ) Geralmente são baseadas em flip-flops tipo D

Questão 11

Refletir se existe diferença entre o endereço armazenado em um espaço de pilha e o endereço armazenado no Stack Pointer (SP)?

Questão 12

Colocou-se 3 LEDs nos endereços **P1.0**, **P1.1** e **P1.2** no microcontrolador e 3 chaves nos endereços **P2.0, P2.1 e P2.2.** Considerando que os LEDs acendem quando é colocado nível baixo na saída e as chaves, quando pressionadas, colocam nível baixo na porta, explique o funcionamento do programa abaixo quando cada uma destas 3 chaves são pressionadas.

```
ORG 0000H

Leitura:
        JNB P2.0, PX
        JNB P2.1, PY
        JNB P2.2, PZ
        LCALL Leitura
PX:
        MOV P1, #0
        RET

PY:
        MOV P1, #00000101b
        RET
PZ:
        MOV A, P1
        CPL A
        MOV P1, A
        RET

FIM:
        SJMP FIM
```

Projeto 1 - Cronômetro Digital usando Assembly e 8051

Objetivos

- Desenvolvimento de um projeto em linguagem Assembly para 8051 que explore os seguintes recursos no simulador EdSim51: registradores GPR e SFR, contagem de tempo, detecção de eventos, pilha, sub-rotinas, portas de entradas e saídas, e interfaces externas (botões, LEDs e displays de 7 segmentos).

Requisitos do projeto

Escrever um programa em Assembly para 8051 no simulador EdSim51 que atenda os seguintes requisitos:

- Ao pressionar um botão/chave (por ex.: SW0), um Display de 7 segmentos deve mostrar a contagem de números na sequência de 0 a 9 em loop (ao chegar em 9, a contagem é reiniciada automaticamente), com intervalo de tempo de 0,25 s.
- Quando um segundo botão/chave (por ex. SW1) for pressionado, tal ação deve alterar o intervalo de tempo da contagem deste mesmo display para **1s**, isto é, a contagem de 0 a 9 em loop continuará, porém, o display passará a contar em um período de tempo mais rápido.
- Caso SW0 seja pressionado novamente, a contagem de números retorna ao intervalo de tempo de 0,25s e vice-versa, mantendo a execução do programa em loop.
- Ao executar o programa pela primeira vez (após clicar em “Run”), a contagem não se inicia automaticamente e o display estará desligado. Somente quando uma das chaves SW0 ou SW1 for pressionada é que a contagem se inicia e o display passa a mostrar os valores de 0 a 9 nos intervalos de tempo mencionados anteriormente atribuídos para cada chave.
- Usar sub-rotinas de delay para gerar as bases de tempo do cronômetro e promover a mudança de período de tempo solicitada, bem como instruções condicionais de verificação de acionamento das chaves (não é necessário utilizar a programação de temporizadores e interrupções, pois esses conceitos serão abordados em aulas futuras).
- Usar um dos displays de 7 segmentos e dois switches disponíveis no EdSim51.

III - Formato de entrega

- Seguir as orientações específicas contidas em documento diretamente na tarefa atribuída no e-Disciplinas. Apresentar o programa desenvolvido e devidamente comentado. Cada linha de código deve ser brevemente comentada. Adicionalmente, uma explicação/discussão sobre o programa deve ser fornecida (no máximo 1 página de texto), explicando os blocos, lógica e quais recursos foram usados e manipulados no programa (registradores, interfaces externas, portas, bases de tempo adotadas, como foi feita a varredura no display etc.). Para complementar a explicação textual, apresentar um diagrama esquemático do microcontrolador 8051 com a ligação das interfaces de entrada e saída usadas no projeto (segundo a estrutura disponível no EdSim51), e um diagrama ou tabela de como é feita a varredura no display de 7 segmentos disponível no EdSim51 para acender números de 0 a 9 usando os 8 bits do registrador da Porta P1.

- Caso preferir, ao invés de apresentar em documento de texto, poderá ser gravado um vídeo curto com a explicação breve e objetiva sobre os requisitos solicitados acima, no formato "screencast" (gravação da tela do computador com narração) compartilhando a tela do computador que mostre o programa desenvolvido e sua a execução no simulador EdSim51 enquanto explica (neste caso, as partes referentes ao diagrama do projeto e a varredura do display de 7 segmentos podem ser mostradas usando os recursos do próprio EdSim51). A gravação da tela pode ser feita via algum software diretamente no computador ou pode ser feita pelo celular.
- Enviar também o código fonte funcional desenvolvido e simulado no EdSim51 (programa em Assembly: arquivo ".asm").
- Fazer o upload dos arquivos na respectiva tarefa atribuída no e-Disciplinas até a data especificada.
- A atividade deve ser feita em grupo de até 3 pessoas (identificar devidamente no documento de entrega os nomes e Nº USP).
- Entregas atrasadas não serão consideradas ou, consideradas com o devido desconto de pontos proporcional ao tempo de atraso. O canal oficial para entrega de tarefas é por meio do e-Disciplinas. Não enviar arquivos por e-mail.
- Qualquer dúvida sobre o formato de envio ou sobre a implementação da atividade prática, entrar em contato com o professor ou com o monitor.

Critérios de avaliação

Item	Pontuação
Entrega no formato: (arquivo em PDF com programa, discussão/diagramas; ou Readme no Github; ou video) + código fonte ".asm"	1
Programa com as linhas de código devidamente comentadas Explicação e discussão **textual** sobre o programa, suportada com diagramas (ou vídeo com essa explicação)	2
Correção lógica do programa: atendimento ao enunciado e uso dos recursos solicitados, como interfaces I/O (botões, display), rotinas de delay, instruções de verificação dos eventos, programação das bases de tempo, contagem em loop no display etc.	7

OBS.: não será considerado como entrega somente o envio do programa (arquivo ".asm"), sem algum arquivo com a explicação do projeto no formato solicitado.