ENGENHARIA MECATRÔNICA

ESCOLA DE ENGENHARIA DE SÃO CARLOS

UNIVERSIDADE DE SÃO PAULO

Aplicação de Microprocessadores
Prática 1 - Introdução aos Sistemas Embarcados e Microcontroladores

Alunos:
Ana Beatriz da Silva Araujo - 11232538
Vítor Fernando Rinaldini - 11232305

Professor:
Pedro de Oliveira Conceicao Junior

19 de abril de 2023

Conteúdo

1 Questões de Revisão das Primeiras Aulas

1.1 Questão 1 - Definição formal de um sistema embarcado

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

Resposta: De acordo com o Institute of Eletrical and Electronics Engineers (IEEE), um sistema embarcado é (*"um sistema computacional que faz parte de um sistema maior e implementa alguns dos requerimentos deste sistema"*.

Um sistema embarcado é um sistema microprocessado onde o computador é totalmente encapsulado (ou dedicado) ao sistema que ele controla, como por exemplo um equipamento eletrodoméstico.

O sistema embarcado usualmente executa apenas um programa, e repetidamente. Enquanto que um PC Desktop poderia executar variados programas.

1.2 Questão 2 - Aspectos importantes no projeto de um sistema embarcado

Pergunta: Liste os aspectos que devem ser considerados no projeto de um sistema embarcado para produtos. Responder essa questão levando em consideração que em sistemas embarcados é comum a utilização de diversos tipos de sistemas computacionais, desde os mais simples até aqueles mais sofisticados, além de um seleto grupo de sistemas embarcados que utilizam sistemas operacionais, sistemas de tempo real (soft ou hard), comunicação sem fio etc. Justifique em que situação é mais conveniente usar cada um deles.

Resposta: O primeiro aspecto que deve ser levado em consideração é a escala de operação na qual usaremos o nosso sistema embarcado. Sendo eles : **Pequena Escala:** Onde a aplicação não é crítica com relação ao desempenho. (Ex: editor e IDE) **Média Escala:** Maior complexidade, normalmente possuem Sistema Operacional de tempo real. (Ex: Robótica, Máquinas Industriais, etc) **Sofisticados :** Altíssima complexidade, maior capacidade de memória e maior velocidade.

O segundo aspecto seria a necessidade de um sistema embarcado de tempo real, onde este possui restrição de tempo, e responde a eventos externos ou estímulos de entrada no tempo que foi previamente especificado. Por fim, tarefas *Hard Real Time* que são tarefas intolerantes a atrasos e devem ser realizadas em um tempo preciso ou haverá falhas. Ou tarefas *Soft Real Time* onde esta é mais tolerante com relação a atrasos na resposta em determinado intervalo de tempo.

1.3 Questão 3 - Relatório de Mercado de Sistemas Embarcados e IoT em 2021

De acordo com o Relatório de Pesquisa sobre o Mercado Brasileiro de Sistemas Embarcados e IoT em 2021, elaborado pela empresa Embarcados:

1.3.1 Item A) Áreas de aplicação

Pergunta: Das ferramentas para sistemas embarcados – quais as principais áreas de aplicação dos projetos no mercado brasileiro?

Resposta: De acordo com a pesquisa, publicada em 2021, as principais áreas são : IoT em primeiro, e Sistemas Industriais em segundo.

1.3.2 Item B) Principais ferramentas

Pergunta: Quais as principais ferramentas de comunicação sem fio usadas?

Resposta: WI-Fi, Bluetooth LE/Smart, e 3G/4G.

1.3.3 Item C) Principais kits de prototipagem rápida

Pergunta: Quais os principais kits de prototipagem rápida usados?

Resposta: Primeiramente, Kits Fornecidos pelos fabricantes do processador, seguido por ESP32, Arduino e por fim, Raspberry PI.

1.3.4 Item D) Principal ferramenta de codificação e controle de versão

Pergunta: Dos softwares para sistemas embarcados - qual a principal ferramenta de codificação e o principal sistema de controle de versão ?

Resposta: Visual Studio Code é a principal ferramenta de codificação enquanto que o Git é o sistema principal sistema de controle.

1.3.5 Item E) Fabricantes mais citados

Pergunta: Dos microprocessadores para sistemas embarcados – quais os fabricantes mais citados na pesquisa?

Resposta: Os fabricante mais citados foram : ST, Microchip/Atmel e Espressif.

1.4 Questão 4 - Definição de Microcontroladores

Pergunta: Apresente a definição formal para microcontroladores (MCU). Sintetize sua resposta comparando os MCUs com os microprocessadores nos quesitos foco,

periféricos, capacidade, aplicação, e exemplos.

Resposta: Microcontrolador é um circuito integrado que contém núcleo de um processador, memória e periféricos programáveis de entrada e de saída, para programar totalmente ou algumas funções de um dispositivo eletrônico. Apresentam menor desempenho e foco em maior economia e baixo custo se comparados com microprocessadores e são destinados a aplicações embarcadas.

1.5 Questão 5 - Vantagens

Pergunta: Recorra ao exemplo do microcontrolador aplicado ao controle de um elevador que foi apresentado em aula e que encontra-se disponível nas transparências do Cap. 1 no e-Disciplinas. Quais as vantagens de se utilizar um microcontrolador para aquele tipo de aplicação? Reflita (sem se preocupar em responder corretamente, apenas expondo seus pontos de vista) 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 ?)

Resposta: A vantagem de usarmos um microcontrolador nesse tipo de projeto se dá pois teremos facilidade de uso, já que a programação de um microcontrolador é relativamente mais fácil quando usada as ferramentas de desenvolvimento adequadas. Além disso, microcontroladores consomem pouca energia, é de baixo custo, e pode ser programado para realizar uma variedade de funções.
Pensando agora no perfil ideal que um microprocessador teria, para aplicação no projeto de um elevador, consideramos como ideal que o microprocessador tenha uma grande quantidade de bits no barramento, já que quanto maior o número de bits, maior será a precisão com que o microcontrolador irá manipular as informações necessárias para o funcionamento desse elevador. Além disso, deveríamos ter também um microcontrolador que suporte operações com número de ponto flutuante para garantirmos uma boa precisão nas operações matemáticas necessárias para o controle do elevador.

2 Atividades Práticas

2.1 Prática 1 - Manipulação por instruções de transferência de dados

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

Listing 1: Código utilizado na prática 1

```
1                       ;(1) criar novo arquivo
2       org 0000h       ;Origem do programa (2)
3   inicio:             ;Label de inicio (3)
4       mov a, #5Fh     ;Mover imediato do hexa para o acumulador (4)
5       mov a, #00h  ;Mover imediato o zero para o acumador (5)
6
7       mov PSW, #00000000b ;Colocar banco '00'(6.1)
8       mov R0, #0AFh  ;Mover imediato de um valor para R0 (6.2)
9
10      mov B, #0BAh   ;Mover imediato de um valor para B (7)
11      mov 7Fh, P1    ;Mover porta P1 para o endereço 7F (8)
12
13      mov PSW, #00001000b ;Colocar banco '01' (9.1)
14      mov R5, 7Fh    ;Mover direto do valor de 7Fh para R5 (9.2)
15
16      mov 70h, R5    ;Mover R5 para o endereço 70h (10)
17
18      mov R1, #70h   ;Mover imediato do valor 70h para R1 (11.1)
19      mov a, @R1     ;Mover indireto de R1 para o acumulador (11.2)
20
21      mov DPTR, #9A5Bh ;Mover imediato (12)
22
23      nop            ;Consumir tempo (13.1)
24      jmp $          ;Segurar o programa nesta linha (13.2)
25
26      end            ;Encerrar (14)
```

Passo-a-passo:

1. Criar um novo programa clicando em “New”;

2. Colocar a origem no endereço 00h;

3. Inicializar o programa com uma label (ex. inicio/main etc.);

4. Mover de forma imediata um valor qualquer em hexadecimal (de 00 a FF) para o registrador acumulador (A ou ACC);

5. Mover de forma imediata o valor zero para ACC;

6. 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);

7. Mover de forma imediata um valor qualquer em hexadecimal para o registrador B;

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

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

10. Mover o conteúdo do registrador escolhido para um outro endereço de memória qualquer (ex.: MOV Rx, endereço);

11. Usar o registrador R1 como um ponteiro, ou seja, apontar de forma imediata (com #) o endereço de memória escolhido na linha anterior. Na linha subsequente, mover R1 de forma indireta para o acumulador;

12. Mover de forma imediata o valor 9A5B (4 dígitos) para o registrador DPTR;

13. Consumir tempo de 1 µs sem nenhuma operação e segurar o programa na próxima linha;

14. Encerrar o programa.

Algumas Questões:

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

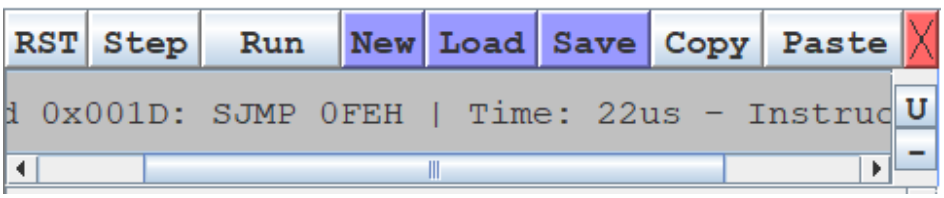

Figura 1: Resultados obtidos no simulador da prática 1

Os tempos calculados podem ser consultados na tabela 1. Os resultados são equivalentes ao obtido no simulador, como podemos observar na figura 1. Vale ressaltar que foi utilizado um clock de 12 MHz durante a simulação.

Instrução	**Tempo (µs)**	**Ciclos de Máquina**
mov a, #5Fh	1	1
mov a, #00h	1	1
mov PSW, #00000000b	2	2
mov R0, #0AFh	1	1
mov B, #0BAh	2	2
mov 7Fh, P1	2	2
mov PSW, #00001000b	2	2
mov R5, 7Fh	2	2
mov 70h, R5	2	2
mov R1, #70h	1	1
mov a, @R1	1	1
mov DPTR, #9A5Bh	2	2
nop	1	1
jmp $	2	2
Total	**22**	**22**

Tabela 1: Duração e quantidade de ciclos de máquina por instrução.

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

Como foi mostrado na tabela 1, este programa tem **22 ciclos de máquina**. O cálculo foi baseado na seguinte afirmação encontrada nas notas de aula: Considerando um cristal oscilador de 12 MHz, como o usado no simulador, 1 Ciclo de máquina será igual à 12 períodos de Clock. 12 périodos de Clock por sua vez será igual à 1 MHz, ou 1 µs. Portanto, nota-se uma relação de 1 pra 1 entre o tempo gasto por instrução e a quantidade de ciclos utilizada.

(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 PORTs: P1, P2, P3, P4).

Ao consultar a tabela 5.1 do *datasheet* AT89S51 Atmel, notamos que a porta P1 é como se fosse um registrador localizado no endereço 90H que ao fazer o reset/inicializar o dispositivo, o mesmo assume o valor FFh. Em suma, essa característica foi definida pelo fabricante.

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

Nada aconteceu com o registrador R1, pois ele serviu apenas como um ponteiro. Ao solicitar a operação 'mov a, @R1', o dispositivo consulta o valor guardado em R1 e vai na memória indicada por ele para buscar o seu valor e assim ao acumulador. Sendo assim, o valor que deveria e apareceu no acumulador foi FFh.

(e) Por que foi possível mover um valor de 4 dígitos para DPTR? O que aconteceu no simulador quando a operação foi realizada? Qual o maior valor que pode ser movido para DPTR?

O registrador DPTR é na verdade composto por 2 registradores de 8 bits e, por isso, é possível fazer esta instrução. Ao realizar a operação 'mov DPTR, #9A5Bh', a parte mais significativa da palavra de 16 bits foi movida para o registrador DPH e a parte menos significativa para o registrador DPL (Veja a figura 2). O maior valor que pode ser movido é FFFFh.

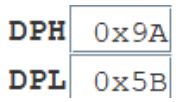

Figura 2: Comportamento da instrução 'mov DPTR, #9A5Bh',

2.2 Prática 2 - Manipulação por instruções aritméticas

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

Listing 2: Código utilizado na prática 2

```
                        ;Criar novo programa (1)
        org 0000h       ;Colocar origem em 00h (2)
inicio:                 ;Label inicio (3)
        mov a, #2       ;mov imediato o 2 para A (4)
        mov b, #3       ;mov imediato o 3 para B (5)
        mov 7Fh, #7     ;mov imediato do 7 para 7Fh (6)
        add a, 7Fh      ;A = A + 'conteudo do 7Fh' (7)
        ;decrementar 3 unidades de A (8)
        dec a           ;A = A -1
        dec a           ;A = A -1
        dec a           ;A = A -1
        inc b           ;incrementar 1 unidade em B (9)
        clr C           ;Limpar Carry para o uso do subb
        subb A, B       ;Subtrair A por B (10)
        mul AB          ;Multiplicar A por B (11)
        ;incrementar 2 unidades em B (12)
        inc b           ;B = B + 1;
        inc b           ;B = B + 1;
        div AB          ;Dividir A por B (13)
        ;Armazenar A e B na RAM (14)
        mov 6Fh, B      ;Mover conteudo de B para 6Fh
        mov 5Fh, A      ;Mover conteude de A para 5Fh
        jmp inicio      ;Saltar para inicio (15)
        end             ;Encerrar (16)
```

Passo-a-passo:

1. Criar um novo programa;
2. Colocar a origem no endereço 00h;
3. Inicializar o programa com uma label;
4. Mover de forma imediata o valor 2 em decimal para o ACC;
5. Mover de forma imediata o valor 3 em decimal para B;
6. Mover para um endereço de memória qualquer o valor imediato 7 em decimal;
7. Somar o conteúdo do endereço de memória escolhido na linha anterior com ACC;

8. Decrementar 3 unidades de ACC;
9. Incrementar 1 unidade em B;
10. Subtrair A por B;
11. Multiplicar A por B;
12. Incrementar 2 unidades em B;
13. Dividir A por B;
14. Armazenar os conteúdos de A e B em dois endereços de memória quaisquer na RAM;
15. Saltar para label declarada no "inicio";
16. Encerrar 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).

Listing 3: Testando Bit de Paridade do PSW

```
1         org 0000h
2 inicio:
3         mov a, #4 ;100b -> Paridade: 1
4         mov a, #3 ;011b -> Paridade: 0
5         end
```

O PSW é na verdade um registrador especial onde cada bit tem um significado. O Bit em questão (o menos significativo) é o bit de paridade. Ele indica se a quantidade de 1s no valor binário movido é par (valor 1) ou impar (valor 0). No exemplo vemos que 4 em binário tem um único algarismo 1, ou seja, ele possui um número impar de 1s e, por isso, o bit de paridade assume o valor 1. Em contrapartida, o número 3 em binário apresenta uma quantidade par de 1s e, portanto, assume valor 0 no bit de paridade.

2.3 Prática 3 - Manipulação por instruções lógicas e booleanas

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

Listing 4: Código utilizado na prática 3

```
                    ;(1) Criar um novo programa
    org 0000h       ;Origem (2)
inicio:             ;Label (3)
    mov A, #01h     ;Mover imediato para A (4.1)
    mov B, #51h     ;Mover imediato para B (4.2)

    ;(5) Realizar AND entre A e B;
    anl A, B        ;AND e guardar no R0

    RR A            ;Rotacionar A a direita 1x (6.1)
    RR A            ;Rotacionar A a direita 2x (6.2)

    cpl A           ;Complemento de A (7)

    RL A            ;Rotacionar A a esquerda 1x (8.1)
    RL A            ;Rotacionar A a esquerda 2x (8.2)

    ;(9) OR entre A e B
    orl A, B        ;OR e guardar em A

    ;(10) XOR entre A e B
    xrl A, B        ;XOR e guardar em A

    swap A          ;(11) swap de A
    jmp inicio      ;Saltar para a label inicial(12)

    end             ;Encerrar o programa (13)
```

Passo-a-passo:

1. Criar um novo programa
2. Colocar a origem no endereço 00h
3. Inicializar o programa com uma label
4. 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).

5. Realizar AND lógico entre A e B
6. Rotacionar A à direita em 2 bits.
7. Realizar o complemento de A
8. Rotacionar A à esquerda em 2 bits.
9. Realizar OR lógico entre A e B
10. Realizar XOR entre A e B
11. Realizar SWAP de A;
12. Saltar para a label inicial
13. Encerrar o programa

2.4 Prática 4 - Manipulação por instruções de desvio

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

Listing 5: Código utilizado na prática 4

```
1                       ;Criar um novo programa (1)
2       org 0000h       ;Colocar origem em 00h (2)
3       jmp main        ;Saltar para a Label Main (3)
4
5       org 0033h       ;Origem em 33h (4)
6   main:               ;Rotulo principal (5)
7       clr A           ;Limpar ACC (6)
8       mov R0, #5Ah    ;Mov imediato para R0 (7)
9
10  bloco1:             ;Separar em blocos (8)
11      jz bloco2       ;Saltar se A=0 (9)
12      jnz bloco3      ;Saltar se A!=0 (10)
13      nop             ;Consumir tempo (11)
14
15  bloco2:             ;Inicializar bloco 2 (12)
16      mov A, R0       ;Mov direto de R0 para A (13)
17      jmp bloco1      ;Saltar para o bloco 1 (14)
18
19  bloco3:             ;Inicializar bloco 3 (15)
20      ;Decrementar e saltar se R0 != 0 (16)
21      djnz R0, bloco3
22      jmp main        ;Saltar para Main (17)
23      end             ;Encerrar (18)
```

Passo-a-passo:

1. Criar um novo programa
2. Colocar a origem no endereço 00h
3. Saltar para a label do programa principal (main)
4. Colocar a origem em 33h
5. Inicializar o programa principal com a label informada anteriormente na operação de salto.
6. Limpar o ACC

7. Mover de forma imediata um valor qualquer para R0

8. 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”)

9. Saltar SE A = 0 para um o segundo bloco do programa (informar a label do segundo bloco nesta instrução. Ex.: bloco2)

10. Na próxima linha (ainda no primeiro bloco), Saltar SE A $\neq$ 0 para um terceiro bloco (passar a label do terceiro bloco nesta instrução. Ex.: bloco3)

11. Na próxima linha (ainda no primeiro bloco), consumir tempo de 1 µs (não realizar operação).

12. Inicializar o segundo bloco com a label chamada anteriormente (para onde o programa irá saltar se A =0)

13. Mover R0 para A

14. Saltar de forma incondicional para o bloco 1 (passar a label do primeiro bloco).

15. Inicializar o terceiro bloco com label chamada anteriormente (para onde o programa irá saltar se A $\neq$ 0)

16. Decrementar e Saltar SE R0 $\neq$ 0 para a label do terceiro bloco (isto é, irá ficar em loop enquanto R0 $\neq$ 0, e sempre no terceiro bloco)

17. Na próxima linha (no terceiro bloco), saltar de forma incondicional para a label do programa principal para reiniciar toda a operação.

18. Encerrar o programa.

Descrição do funcionamento:

Este programa começa por um bloco inicializador (’main’), onde ele limpa o acumulador e move um valor para o registrador R0. Em seguida ele inicia o *loop* principal (’bloco1’) que executa um bloco (’bloco2’) caso A = 0 ou outro bloco (’bloco3’) caso A != 0. O ’bloco2’ move o valor de R0 para A e volta pro loop principal. O ’bloco3’ decrementa o valor de A até chegar em zero e volta para a *label* de inicialização.

Como resultado, veremos um *loop* sendo executado permanentemente, onde o programa inicializa o valor de A copiando o valor de R0 e, em seguida, decrementa o acumulador de um em um até zerar e, após zerar, ele recomeça o ciclo.

Referências

Atmel Corporation. 8-bit microcontroller with 4k bytes in-system programmable flash - at89s51, 2008. URL https://edisciplinas.usp.br/pluginfile.php/7621375/mod_resource/content/1/AT89S51.pdf. Último acesso em 15 de abril de 2023.

Pedro de Oliveira Conceicao Junior. Aula - microcontroladores, mcs-51 e set de instruções, 2023. URL https://edisciplinas.usp.br/pluginfile.php/7621363/mod_resource/content/1/SEL0336_SEL0433_Aplic_Micros_Cap1_Microcontroladores.pdf. Último acesso em 15 de abril de 2023.

Portal Embarcados. Relatório da pesquisa sobre o mercado brasileiro de sistemas embarcados e iot 2021, 2021. URL https://embarcados.com.br/relatorio-da-pesquisa-sobre-o-mercado-brasileiro-de-sistemas-embarcados-e-iot-2021/. Último acesso em 15 de abril de 2023.

EdSim51DITM Simulator. The 8051 simulator for teachers and students, 2005-2022. URL http://www.edsim51.com/. Último acesso em 15 de abril de 2023.