SEL0415
Introdução à Organização de Computadores

Prof. Dr. Marcelo Andrade da Costa Vieira

Lista 09 - Microcontrolador Intel 8051

1) Abaixo temos o esquema de um microcontrolador 80C51. Qual a arquitetura utilizada e como chegamos a essa conclusão?

A arquitetura é Von Neumann, pois o barramento de dados (RAM) é o mesmo do de endereços (ROM Flash).

2) Explique e diferencie ciclo de busca, execução e máquina. Quando um ciclo de instrução não será equivalente a um ciclo de máquina? Explique com exemplos.

Ciclo de busca: Ciclo de busca de um Opcode de uma instrução a partir da posição de memória definida pelo Program Counter (PC). Esse Opcode é então armazenado no Instruction Register (IR) para ser executado.
Ciclo de execução: Decodifica a instrução e a executa a instrução (caso ela ocupe apenas uma posição). Ou busca os demais bytes da instrução na memória do programa, iniciando um novo ciclo de busca. O conteúdo do PC é incrementado uma, duas ou três unidades dependendo do tamanho da instrução.
Ciclo de máquina: Ciclo de busca + ciclo de execução.
Ciclo de instrução: O tempo necessário para a execução de uma instrução completa. Pode ser composto por vários ciclos de máquina como nas instruções "INC DPTR" (2 ciclos) ou "MUL A, B" (4 ciclos).

3) Abaixo há várias alternativas a respeito do microcontrolador Intel 8051. Assinale V ou F, retificando as afirmações falsas.

(V) O microcontrolador em questão possui 4 portas de E/S bidirecionais, cada uma com 8 bits. São 32 linhas endereçadas de forma individual.
(F) Podemos controlar os bits das portas citadas anteriormente endereçando por bit os bytes 80h, 90h, A0h ou B0h da memória RAM. Por exemplo: SETB A0h.7 deixa o bit 7 da porta 3 em nível baixo.
A instrução SETB deixa o bit em nível alto, e não baixo.
(V) Nesse caso, estamos manipulando registradores de função especial, o que só é possível através de endereçamento direto.
(F) Os Special Function Registers não possuem relação alguma com o estado dos General Purpose Registers.
Algumas vezes possuem. O registrador PSW possui nos bits 3 e 4, RS0 e RS1. Esses dois bits decidem qual banco de registradores de uso geral será usado para R0 a R7.
(F) As flags são bits que indicam estado de alguma operação do microncontrolador e encontram-se mapeadas entre a posição 00h e 7Fh do Intel 8051.
Os flags são Special Function Registers e estão entre 80h e FFh.
(F) O endereçamento por bit e por byte são possíveis tanto para os GPRs quanto para os SFRs, porém em posições determinadas: no primeiro caso, podemos endereçar por bit os registradores 80h e CFh, por exemplo. No segundo caso, o endereçamento por bit ocorre para os bytes 20h a 2Fh.
20h a 2Fh são GPR que podem ser endereçados por bit (não SFR). 80h é um SFR (o P0) que pode ser endereçado por bit. CFh não é um SFR.
(F) Set de instruções do tipo CISC, como ocorre para a maioria das arquiteturas Harvard.

CISC é comum para arquitetura Von Neumann.
(V) 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.
(V) Caso PSW (Program Status Word) receba o valor B2h, o banco de registradores 2 é ativado, ou seja, usamos os bytes 10H a 17H da RAM interna para tal propósito.

4) A partir do programa abaixo, determine qual valor será armazenado no acumulador nos seguintes casos:

a)

```
ORG 0000h
CLR A
MOV R0, #30h
MOV 30h, #20h
ADD A, @R0
```

A = 20h

b)

```
ORG 0000h
CLR A
MOV R0, #30h
MOV 30h, #20h
ADD A, R0
```

A = 30h

c)

```
ORG 0000h
CLR A
MOV 40h, #15h
ADD A, #40h
ADD A, 40h
MOV 3Bh, #40h
MOV R2, #3Bh
ADD A, @R2
```

A = 95h

5) Faça um fluxograma que some o valor 34h com o valor que estiver no registrador R0 e salve o resultado no endereço 28h. Em seguida, faça o código em Assembly.

```
ORG 0000h
CLR A
ADD A, #34h
MOV R0, #dado
ADD A, R0
MOV 28h, A
END
```

6) Calcule o tempo total que o microcontrolador leva para executar o programa feito na questão anterior e o valor de PC (Program Counter) ao final da execução do programa.

Program Counter:
Avanço em bytes de cada instrução: 1 + 2 + 2 + 1 + 2 = 8
PC: 0008h

Tempo de execução:
Ciclos de máquina de cada instrução: 1 + 1 + 1 + 1 + 1 = 5
Tempo: 5 x 1µ = 5µ

7) Complete a tabela abaixo e calcule o tempo total para execução do programa.

Programa	**Descrição do Programa**	**Valores recebidos**				
ORG 0	Define que as instruções serão escritas em 0000h	PC	SP	R0	A	23H
MOV A, #73h	Insere 73h no acumulador	0002h	07h	00h	00h	00h
ADD A, #19h	Soma 19h no acumulador	0002h	07h	00h	73h	00h
MOV R0, #23h	Insere 23h no registrador R0	0004h	07h	00h	8Ch	00h
LCALL Rotina	Chama a sub-rotina ”Rotina”	0006h	07h	23h	8Ch	00h
MOV R0, 23h	Insere o conteúdo do end. 23h no registrador R0	0009h	07h	23h	8Ch	8Ch
Loop:						
SJMP Loop	Pula o programa para ”Loop”	000bH	07h	8Ch	8Ch	8Ch
Rotina:						
ORG 41h	Define que as instruções serão escritas em 0041h					
MOV @R0, A	Insere o valor do acumulador no end. apontado por R0	0041h	08h	23h	8Ch	00h
RET	Retorna da sub-rotina	0041h	08h	23h	8Ch	8Ch

Tabela 1: Tabela do programa

8) Descreva o que faz o seguinte programa, bem como os valores dos registradores modificados durante sua execução:

```
ORG   0000h
      MOV R2, #4DH
      CLR A
SOMA: ADD A, @R2
      INC R2
      CJNE R2, #51H, SOMA
OVER: SJMP OVER
```

40H	01H	3EH	32H	14H	20H	D3H	9AH	1BH	**47H**
48H	03H	64H	EEH	C7H	15H	48H	08H	11H	**4FH**
50H	0FH	FFH	7CH	18H	4EH	CBH	3DH	92H	**57H**
58H	A2H	2DH	40H	05H	ECH	56H	2FH	71H	**5FH**

Tabela 2: Tabela com os conteúdos de cada posição de memória da RAM interna.

O programa joga 4Dh no registrador R2 e limpa o acumulador. Em seguida, entra em um ciclo, jogando o valor do endereço indicado em R2 no acumulador (inicialmente o end. 4Dh), incrementando R2 e verificando após isso se R2 é igual a 51h, caso em que o ciclo se encerra. O acumulador é igual a soma dos valores contidos nos endereços 4Dh, 4Eh, 4Fh e 50h. Pela tabela:

A = 48h + 8h + 11h +0Fh = 70h
R2 = 51h ao fim da operação (mas o valor contido em 51h não é somado).

9)Colocou-se 3 LEDs nos endereços P1.0 até P1.2 no microcontrolador e 3 chaves nos endereços P2.0 até P2.2, considerando que os mesmos foram ligados de maneira apropriada e que os LEDs acendem quando é colocado nível baixo na saída e as chaves colocam nível baixo na porta quando são pressionadas, explique o funcionamento do programa abaixo. Os LEDs iniciam em qual estado?

```
         ORG 0000H
Leitura: JNB P2.0, PX
         JNB P2.1, PY
         JNB P2.2, PZ
         LCALL Leitura
   PX:   MOV P1, #0
         LJMP Leitura
   PY:   MOV P1, #00000101b
         LJMP Leitura
   PZ:   MOV A, P1
         CPL A
         MOV P1, A
         LJMP Leitura
   FIM:  SJMP FIM
```

O programa irá controlar o comportamento dos LEDs conforme o pressionamento das chaves.

Chave 2.0 pressionada: Todos os LEDs acendem.
Chave 2.1 pressionada: Somente o LED 1.1 acende.
Chave 2.2 pressionada: Os LEDs acesos se apagam, e os apagados acendem.

10) O que são diretivas do compilador? Caso eu inicie a questão 5 com ORG 0010H, qual a diferença no tempo de execução do programa e no valor final do ponteiro PC?

Diretivas são mnemônicos que indicam informações no programa, como endereço inicial, reserva da área de dados, definir equivalência entre identificadores e valores. Contudo, não geram código de máquina, portanto, não alteram o tempo de execução do programa.

Caso seja feita essa mudança na questão 5, a única alteração será no registrador PC que inicia em 0010h e seu valor final é, portanto, acrescido em 0010h.

11: Mostre os valores finais dos endereços modificados e o tempo total para execução do programa abaixo.

```
ORG 0000H
MOV R0, #0CH
MOV A, #0
MOV @R0, A
INC A
DJNZ R0, Prog
END
```

O programa insere valores em ordem crescente em endereços em ordem decrescente. Começa inserindo 0 em 0Ch, 1 em 0Bh, 2 em 0Ah, 3 em 09h e assim por diante, até é inserir 0Bh no endereço 01h, quando o registrador decrementa, vai para 00h e sai do loop “prog”.

12) Considere o seguinte programa em assembly com o 8051. Insira comentários em todas as linhas do programa. Calcule o valor final que estará em cada um dos seguintes registradores: A, R0, R1 e DPTR e também nas posições da memória RAM interna 30H, 31H e 32H e na RAM externa 3000H.

```
ORG 0000h           Inicia o programa em 0
MOV A, #10H         Move para o acumulador o dado 10H   A = 10H
MOV 30H, #33H       Move para o endereço 30H o dado 33H   30H = 33H
MOV 31H, #37H       Move para o endereço 31H o dado 37H   31H = 37H
MOV R0, #30H        Move para o registrador R0 o dado 30H   R0 = 30H
ADD A, @R0          Soma ao acumulador o conteúdo apontado pelo registrador R0   A = 43H
INC R0              Incrementa em 1 o registrador R0   R0 = 31H
ADD A, @R0          Soma ao acumulador o conteúdo apontado pelo registrador R0   A = 7AH
MOV R1, 30H         Move para o registrador R1 o conteúdo da posição de memória 30H   R1 = 33H
DEC R1              Decrementa em 1 o registrador R1   R1 = 32H
MOV @R1, A          Move o acumulador para o endereço apontado por R1   32H = 7AH
ADD A, #05H         Soma ao acumulador o dado 05H   A = 7FH
INC R0              Incrementa em 1 o registrador R0   R0 = 32H
MOV DPTR, #3000H    Move os 2 bytes 3000H para o data pointer   DPTR = 3000H
MOV A, 31H          Move para o acumulador o conteúdo da posição de memória 31H   A =37H
MOVX @DPTR, A       Move para o endereço apontado por DPTR o conteúdo do acumulador   3000H = 37H
SJMP $              $ Indica a linha atual
END
```

A = 7FH
R0 = 32H
R1 = 32H
DPTR = 3000H
30H = 33H
31H = 37H
32H = 7AH
Ram externa 3000H = 37H