ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES I

Exercícios Referentes à Prova P2

prof. Dr. César Augusto M. Marcon
prof. Dr. Edson Ifarraguirre Moreno

Exercícios

1. Faça um programa VHDL de uma máquina reconhecedora de padrões
   - Esta máquina tem como entradas os sinais:
     - **reset** (std_logic), que quando tiver o valor ‘1’ faz com que os registradores e/ou contadores da máquina sejam zerados;
     - **clock** (std_logic), que informa os momentos de carga dos registradores e/ou contadores. Considere que, tanto os registradores quanto contadores são *assíncronos* e *sensíveis à borda de subida* do clock;
     - **entr** (std_logic_vector(7 downto 0)), vetor de 8 bits que contém uma seqüência randômica de padrões que devem ser reconhecidos pela máquina.
   - Para reconhecimento dos padrões a máquina dispõe das seguintes saídas:
     - **numClocks** (std_logic_vector(15 downto 0)), vetor que contém o número de vezes que a entrada **entr** teve o valor x”FF” desde que o sinal de **reset** foi ativado até o momento atual;
     - **numRep** (std_logic_vector(15 downto 0)), vetor que contém o número de vezes que o vetor **entr** teve padrões iguais consecutivos desde que o sinal de **reset** foi ativado;
     - **Ov**, flip-flop que, com valor 1, indica se algum dos contadores (**numClocks** ou **numRep**) atingiu o máximo da contagem de 16 bits. Uma vez **Ov** obtendo valor igual a 1, este se mantém enquanto não ocorrer um novo sinal de reset.
   - Faça o VHDL tanto da *entidade*, como da *arquitetura*. *OBS.: Não é necessário fazer o testbench!*

Exercícios

2. Faça um contador sensível à borda de subida e com um reset assíncrono. O contador deve ter também um valor para inicialização. A contagem deve ser em passos de 5 e o contador deve ter a capacidade de contar até 315. Depois do contador atingir este valor, ele deve ser zerado. Mostre a entidade e como ficaria o diagrama de blocos implementado na arquitetura
3. Faça um testbench para o item acima. Considere 2 momentos de reset e um clock de 333.33 MHz
4. Faça uma subrotina (com label **Mult**) para multiplicar dois valores que estão nas posições de memória apontadas pelos labels **a** e **b**. O resultado desta multiplicação deve ir para a posição de memória apontada pelo label **c**. Note que o valor da multiplicação pode ultrapassar a capacidade de armazenamento de **c**. **Ignore este problema!**
5. Faça um programa que leia um vetor de 5 valores apontado pelo label **vet** e, utilizando a rotina **Mult** implementada acima, gere um novo vetor, apontado pelo label **vetMul**. Este último vetor deve ser igual ao vetor **vet** com todos os seus valores multiplicados pela constante que está no endereço **cte**

Exercícios

6. Dado o programa ao lado, descrito em linguagem de montagem do processador Cleópatra
   a) Faça uma descrição em linguagem mais abstrata deste programa
   b) Mostre como fica o endereçamento de código e dados
   c) Coloque ao lado das instruções o número de ciclos de relógio gasto
   d) Gere o código objeto do mesmo
   e) Qual o tamanho em bytes do programa?
   f) Suponha que o label **b** tenha associado ao invés da constante 07h, a variável **n** (considere que a área de dados dá suporte a qualquer valor de **n**). Então, expresse o número de ciclos para executar este programa em termos de **n**, usando a forma: **n_ciclos** = a1***n** + a2, onde a1 e a2 são constantes que devem ser calculadas a partir do programa e da definição de quantos ciclos de relógio cada instrução leva para ser executada. Ex. n_ciclos = 80***n** + 45
   g) Considerando que o tempo de execução desejado para o programa deva ser o mais próximo possível de 1 milissegundo, e que se fixa o valor de n = 8, calcule a freqüência de relógio do processador

```
 LDA b        a: DB #c
INI:          b: DB #07h
  jz FIM,R    c: DB #01h
  LDA a,I        DB #02h
  ADD #3         DB #03h
  STA a,I        DB #04h
  LDA a          DB #05h
  ADD #1         DB #06h
  STA a          DB #07h
  LDA b
  ADD #-1
  STA b
  JMP INI
FIM:
  HLT
```

Exercícios

6. Preencher com o endereçamento (b), código objeto (d) e número de ciclos (c)

Linha	End	Código	Objeto	ciclos
1.		LDA b		
2.		INI:		
		jz FIM,R		
3.		LDA a,I		
4.		ADD #3		
5.		STA a,I		
6.		LDA a		
7.		ADD #1		
8.		STA a		
9.		LDA b		
10.		ADD #-1		
11.		STA b		
12.		JMP INI		
13.		FIM:		
		HLT		

Linha	End	Dados	Objeto
14.		a: DB #c	
15.		b: DB #07h	
16.		c: DB #01h	
17.		DB #02h	
18.		DB #03h	
19.		DB #04h	
20.		DB #05h	
21.		DB #06h	
22.		DB #07h	

Exercícios

7. **Complete a tabela abaixo com as microinstruções referente às instruções das linhas 5 e 6 do programa exemplo anterior (instruções STA e LDA)**

ALU	write_r	read_r	lnz	lcv	ce	rw	MAR	MDR	IR	PC	AC	N	Z	C	V	Microcódigo

RESPOSTAS

Resposta de Exercícios

1. Faça um programa VHDL de uma máquina reconhecedora de padrões ...

```
entity Cont is
   port ( reset, clock: in std_logic;
      entr: in std_logic_vector(7 downto 0);
      numClocks: out std_logic_vector(15 downto 0);
      numRep: out std_logic_vector(15 downto 0);
      Ov: out std_logic );
end Cont;

architecture Cont of Cont is
   signal numClocksInt: std_logic_vector(15 downto 0);
   signal numRepInt: std_logic_vector(15 downto 0);
   signal entrOld: std_logic_vector(7 downto 0);
   signal OvInt: std_logic;
begin
   Ov <= OvInt;
   numClocks <= numClocksInt;
   numRep <= numRepInt;
   process(reset, clock)
   begin
      if reset = '1' then
         numClocksInt <= (others=>'0');
         numRepInt <= (others=>'0');
         OvInt <= '0';
      elsif clock'event and clock ='1' then
         if entr = x"FF" then
            numClocksInt <= numClocksInt + 1;
         end if;
         if entr = entrOld then
            numRepInt <= numRepInt + 1;
         end if;
         entrOld <= entr;
         if numClocksInt = x"FFFF" or
            numRepInt = x"FFFF" then
            OvInt <= '1';
         end if;
      end if;
   end process;
end Cont;
```

Resposta de Exercícios

4. Faça uma subrotina (com label Mult) para multiplicar dois valores que estão nas posições de memória apontadas pelos labels a e b. ...

```
.code
   lda #4
   sta a
   lda #5
   sta b
   jsr Multip
   lda c
   hlt
Multip:
   lda #0
   sta c
   lda a
```

```
LoopMultip:
   jz FimMultip
   lda b
   add c
   sta c
   lda a
   add #-1
   sta a
   jmp
LoopMultip
FimMultip:
   rts
.endcode
```

```
.data
   a: db #0h
   b: db #0h
   c: db #0h
.enddata
```

Resposta de Exercícios

5. Faça um programa que leia um vetor de 5 valores apontado pelo label vet e, utilizando a rotina Mult implementada acima ...

```
.code
    lda cte
    sta b
Inicio:
    lda size
    jz Fim
    lda pVet,i
    sta a
    jsr Multip
    lda c
    sta pVetMult,i
    lda pVetMult
    add #1
    sta pVetMult
    lda pVet
    add #1h
    sta pVet
    lda size
    add #-1
    sta size
    jmp Inicio
Fim:
    hlt
```

```
Multip:
    lda #0
    sta c
    lda a
LoopMultip:
    jz FimMultip
    lda b
    add c
    sta c
    lda a
    add #-1
    sta a
    jmp LoopMultip
FimMultip:
    rts
.endcode
```

```
.data
    pVet: db #vet
    pVetMult: db #vetMult
    size: db #5h
    vet:
        db #3h
        db #4h
        db #5h
        db #6h
        db #7h
    vetMult:
        db #0h
        db #0h
        db #0h
        db #0h
        db #0h
    cte: db #3h
    a: db #0h
    b: db #0h
    c: db #0h
.enddata
```

Resposta de Exercícios

6. Linguagem mais abstrata (a)

```
int c[] = { 1, 2, 3, 4, 5, 6, 7 };
int *a = &c[0];
int b = 7;

while(b != 0)
{
    *a = *a + 3;
    a++;
    b--;
}
```

Resposta de Exercícios

6. Preencher com o endereçamento (b), número de ciclos (c), código objeto (d)

Linha	End	Código	Objeto	ciclos
1.	00	LDA b	44 1A	8
2.	02	INI:		
		jz FIM,R	BC 14	5/6
3.	04	LDA a,I	48 19	10
4.	06	ADD #3	50 03	6
5.	08	STA a,I	28 19	9
6.	0A	LDA a	44 19	8
7.	0C	ADD #1	50 01	6
8.	0E	STA a	24 19	7
9.	10	LDA b	44 1A	8
10.	12	ADD #-1	50 FF	6
11.	14	STA b	24 1A	7
12.	16	JMP INI	84 02	6
13.	18	FIM:		
	00	HLT	F0	4

Linha	End	Dados	Objeto
14.	19	a: DB #c	1B
15.	1A	b: DB #07h	07
16.	1B	c: DB #01h	01
17.	1C	DB #02h	02
18.	1D	DB #03h	03
19.	1E	DB #04h	04
20.	1F	DB #05h	05
21.	20	DB #06h	06
22.	21	DB #07h	07

Resposta de Exercícios

6. Tamanho do código (e)

```
25 bytes de código
9 bytes de dados
34 bytes no total
```

6. Equação (f)

```
n_ciclos = (5 + 10 + 6 + 9 + 8 + 6 + 7 + 8 + 6 + 7 + 6) * N +
                (8 + 6 + 4)
n_ciclos = 78N + 18
```

6. Tempo (g)

```
n_ciclos = 78 * (8) + 18
         = 642 ciclos
         = 642 T (períodos de relógio)
642 T    = 1ms
         = 0,001s → T = 0,001 / 642 = 1,558 us
f = 1 / T
         = 1 / 1,558 us = 642 KHz
```

Exercícios

7. Complete a tabela abaixo com as microinstruções referente às instruções das linhas 5 e 6 do programa exemplo anterior (instruções STA e LDA)

alu_op	write_re	read_re	ce	rw	lnz	lcv	mar	mdr	ir	pc	ac	N Z C V	MicroInst
7	0	3	0	0	0	0	08	05	50	08	06	0 0 0 0	mar<-pc
1	6	3	1	1	0	0	08	28	50	09	06	0 0 0 0	mdr<-pmem(mar)
4	2	1	0	0	0	0	08	28	28	09	06	0 0 0 0	ir<-mdr
7	0	3	0	0	0	0	09	28	28	09	06	0 0 0 0	mar<-pc
1	6	3	1	1	0	0	09	19	28	0A	06	0 0 0 0	mdr<-pmem(mar)
4	0	1	0	0	0	0	19	19	28	0A	06	0 0 0 0	mar<-mdr
7	1	0	1	1	0	0	19	1B	28	0A	06	0 0 0 0	mdr<-pmem(mar)
4	0	1	0	0	0	0	1B	1B	28	0A	06	0 0 0 0	mar<-mdr
7	7	4	1	0	0	0	1B	1B	28	0A	06	0 0 0 0	pmem(mar)<-ac
7	0	3	0	0	0	0	0A	1B	28	0A	06	0 0 0 0	mar<-pc
1	6	3	1	1	0	0	0A	44	28	0B	06	0 0 0 0	mdr<-pmem(mar)
4	2	1	0	0	0	0	0A	44	44	0B	06	0 0 0 0	ir<-mdr
7	0	3	0	0	0	0	0B	44	44	0B	06	0 0 0 0	mar<-pc
4	0	1	0	0	0	0	19	19	44	0C	06	0 0 0 0	mar<-mdr
7	1	0	1	1	0	0	19	1B	44	0C	06	0 0 0 0	mdr<-pmem(mar)
4	4	1	0	0	1	0	19	1B	44	0C	1B	0 0 0 0	ac<-mdr