

USP - ICMC - SSC

SSC 0511 - Sist. Informação - 2o. Semestre 2014

Disciplina de Organização de Computadores Digitais

Prof. Fernando Santos Osório

Email: fosorio [at] { icmc. usp. br , gmail. com }

Página Pessoal: <http://www.icmc.usp.br/~fosorio/>

Material on-line: Wiki ICMC:

[http://wiki.icmc.usp.br/index.php/SSC-511-2014\(fosorio\)](http://wiki.icmc.usp.br/index.php/SSC-511-2014(fosorio))



Lab. de Robótica Móvel



Centro de Robótica da USP

Agenda:

- 1. Programação usando o NEANDER**
- 2. Limitações do Neander**
- 3. Arquiteturas: Ahmes e Ramses**

1. Programação do Neander

Programação do Neander

Código Binário	Instrução em Hexa	instrução	comentário
0000	00	NOP	Nenhuma operação
0001	10 XX	STA end	MEM(end) ← AC
0010	20 XX	LDA end	AC ← MEM(end)
0011	30 XX	ADD end	AC ← MEM(end) + AC
0100	40 XX	OR end	AC ← MEM(end) OR AC
0101	50 XX	AND end	AC ← MEM(end) AND AC
0110	60	NOT	AC ← NOT AC
1000	80 XX	JMP end	PC ← end
1001	90 XX	JN end	IF N=1 THEN PC ← end
1010	A0 XX	JZ end	IF Z=1 THEN PC ← end
1111	F0	HLT	pára processamento

1. Programação do Neander

Programação do Neander :

Programação do Neander – Exercícios

- 1) Somar vários valores de 8 bits ($A + B + C + D + E$)
- 2) Subtrair valores de 8 bits ($A - B$)
- 3) Contador: Laço de contagem até 10
- 4) Somar os dados de um vetor
- 5) Somar valores com mais de 8 bits (!)
- 6) Multiplicar 2 valores
- 7) Pesquisar um dado em uma tabela

NOP	0	ADD	30 end	JMP	80 end
STA	10 end	OR	40 end	JN	90 end
LDA	20 end	AND	50 end	JZ	A0 end
		NOT	60	HLT	F0

1. Programação do Neander

Programação do Neander : (1) Soma 5 Valores (Result = A+B+C+D+E)

Linguagem de Montagem:

```
Val_A    EQU  $C8
Val_B    EQU  $C9
Val_C    EQU  $CA
Val_D    EQU  $CB
Val_E    EQU  $CC
Result   EQU  $D2

                ORG  $00

Ini:        LDA  Val_A
            ADD  Val_B
            ADD  Val_C
            ADD  Val_D
            ADD  Val_E
            STA  Result

Fim:        HLT

                END
```

Memória	Instrução em Hexa	Instrução
\$00	20 C8	LDA \$C8
\$02	30 C9	ADD \$C9
\$04	30 CA	ADD \$CA
\$06	30 CB	ADD \$CB
\$08	30 CC	ADD \$CC
\$0A	10 D2	STA \$D2
\$0C	F0	HLT

1. Programação do Neander

Programação do Neander : (1) Soma 5 Valores (Result = A+B+C+D+E)

Linguagem de Montagem:

```

                ORG $00
Ini:            LDA Val_A
                ADD Val_B
                ADD Val_C
                ADD Val_D
                ADD Val_E
                STA Result
Fim:            HLT

                ORG $C8
Val_A           DB $01
Val_B           DB $02
Val_C           DB $03
Val_D           DB $04
Val_E           DB $05
Result         DB $00
                END
```

Memória	Instrução em Hexa	Instrução
\$00	20 C8	LDA \$C8
\$02	30 C9	ADD \$C9
\$04	30 CA	ADD \$CA
\$06	30 CB	ADD \$CB
\$08	30 CC	ADD \$CC
\$0A	10 D2	STA \$CD
\$0C	F0	HLT

Memória	Valor
\$C8	01
\$C9	02
\$CA	03
\$CB	04
\$CC	05
\$CD	00

ORG = Início da área de montagem

EQU = Define um valor constante (Label = Valor)

DB = Aloca uma variável do tipo byte (Label = End. de Valor)

END = Fim do código de montagem

2. Programação do Neander

Programação do Neander : (2) Subtrair valores de 8 bits (A – B)

Linguagem de Montagem:

```

        ORG $00
Ini:    LDA Val_B
        NOT
        ADD Val_01
        ADD Val_A
        STA Result

Fim:    HLT

        ORG $A0
Val_01  DB $01
Val_A   DB $56
Val_B   DB $0A
Result  DB $00
        END
    
```

NOP	0	ADD	30 end	JMP	80 end
STA	10 end	OR	40 end	JN	90 end
LDA	20 end	AND	50 end	JZ	A0 end
		NOT	60	HLT	F0

Memória	OpCode	Operando
\$00	20	A2
\$02	60	
\$03	30	A0
\$05	30	A1
\$07	10	A3
\$09	F0	

Memória	Valor
\$A0	01
\$A1	56
\$A2	0A
\$A3	00
Run:	4C

2. Programação do Neander

Programação do Neander : (2) Subtrair valores de 8 bits (A – B)

Linguagem de Montagem: >> Solução Alternativa <<

```

                ORG $00
Ini:           LDA Val_A
                ADD Val_B
                STA Result
Fim:           HLT
    
```

```

                ORG $A0
Val_A          DB $56
Val_B          DB $F6
Result        DB $00
                END
    
```

```

; Valor B já está representado
; em complemento de 2
; $0A => $F6 em C2
    
```

NOP	0	ADD	30 end	JMP	80 end
STA	10 end	OR	40 end	JN	90 end
LDA	20 end	AND	50 end	JZ	A0 end
		NOT	60	HLT	F0

Memória	OpCode	Operando
\$00	20	A0
\$02	30	A1
\$04	10	A2
\$06	FO	

Memória	Valor
\$A0	56
\$A1	F6
\$A2	00
Run:	4C

2. Programação do Neander

Programação do Neander : (3) Contador: Laço de contagem até 10

Linguagem de Montagem:

```

Início:      ORG $00
              LDA Contador
              ADD VMinus1
              STA Contador
              JZ  Fim:
              JMP Início:

Fim:         HLT

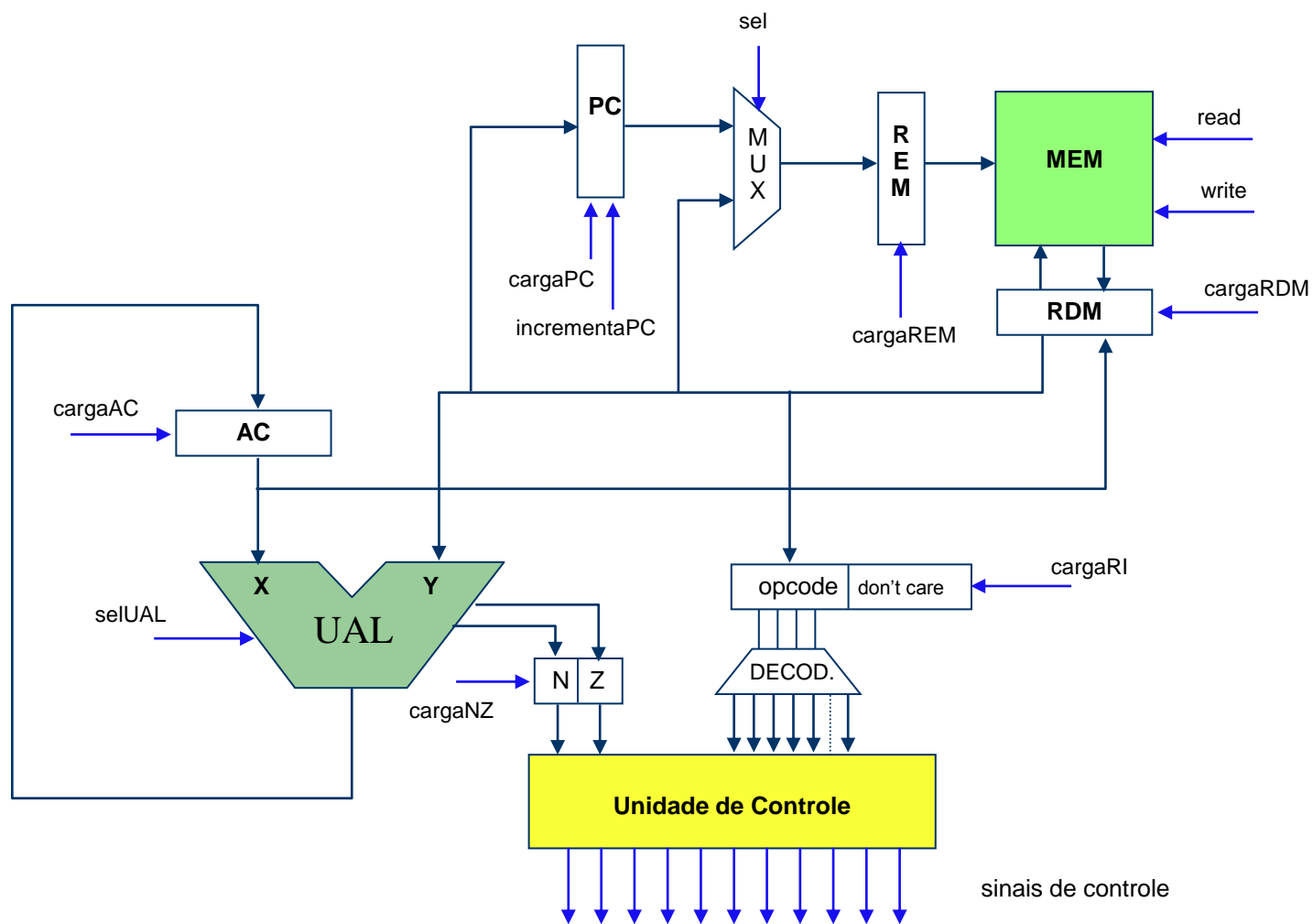
              ORG $A0
VMinus1     DB  $FF
Contador    DB  $0A
              END
    
```

NOP 0	ADD 30 end	JMP 80 end
STA 10 end	OR 40 end	JN 90 end
LDA 20 end	AND 50 end	JZ A0 end
	NOT 60	HLT F0

Memória	OpCode	Operando
\$00	20	A1
\$02	30	A0
\$04	10	A1
\$06	A0	0A
\$08	80	00
\$0A	F0	

Memória	Valor
\$A0	FF
\$A1	0A

Neander - Computador Hipotético [Weber 2001*]



2. Programação do Neander

Programação do Neander :

Arquitetura do Neander – Críticas?

- Possui apenas 1 modo de endereçamento (Direto Absoluto)
- Possui apenas 1 registrador de uso geral (Acumulador)
- Possui apenas 2 flags de status da ULA (Flip-flops N e Z)
- Possui apenas 11 instruções de máquina (incluindo NOP e HLT)
- Não possui flags de “vai-um” (Carry In, Carry Out)
- Não possui instruções de desvio/retorno de sub-rotina (JSR, RTS)
- Não possui uma pilha auxiliar para dados/endereços (Push, Pop)
- Não possui instruções de acesso imediato a memória (LDA #)
- Não possui instruções de acesso indexado a memória (LDA \$,X)
- Não possui instruções dedicadas de E/S (In, Out)

2. Arquiteturas Didáticas

Evolução do Neander... Ahmes, Ramses, Cesar

Quadro comparativo

Arquitetura	Endereços	Dados	Nro. Instruções	Registradores
NEANDER	8 bits 256 bytes	8 bits Compl.2	11 instruções (OpCode: 4bits)	AC, PC, IR, Flags (N,Z) REM, RDM
AHMES	8 bits	8 bits	24 instruções (Neander ext.)	PC, IR, REM, RDM Flags (N, Z, C, B, V)
RAMSES	8 bits	8 bits	Modos de End. 4 modos x 16 instr.	PC, IR, RA, RB, RX Flags (N, Z, V, C)
CESAR	16 bits 64 Kbytes	16 bits	Inúmeras	R0 a R6 (uso geral) R7 (PC)

Simuladores Didáticos

<ftp://ftp.inf.ufrgs.br/pub/inf107/>

<ftp://ftp.inf.ufrgs.br/pub/inf108/>

http://pt.wikipedia.org/wiki/Máquinas_hipotéticas_da_Universidade_Federal_do_Rio_Grande_do_Sul

2. Arquiteturas Didáticas

Evolução do Neander... Ahmes, Ramses, Cesar

The screenshot displays the Ahmes simulator interface with three main windows:

- Programa:** A table showing memory addresses (End.) and instructions (Mnemônico).

P	End.	Dado	Mnemônico
0	0	0	NOP
1	0	0	NOP
2	0	0	NOP
3	0	0	NOP
4	0	0	NOP
5	0	0	NOP
6	0	0	NOP
7	0	0	NOP
8	0	0	NOP
9	0	0	NOP
10	0	0	NOP
11	0	0	NOP
12	0	0	NOP
13	0	0	NOP
14	0	0	NOP
15	0	0	NOP
16	0	0	NOP
- Ahmes:** The main CPU window showing registers (AC, PC), status flags (N, Z, V, C, B), and instruction details.
 - AC: 000
 - PC: 000
 - Flags: N (off), Z (on), V (off), C (off), B (off)
 - Execução: Acessos: 00000, Instr.: 00000
 - Instrução: R I: 0, Mnem: NOP
- Dados:** A table showing memory addresses (End.) and data (Dado).

End.	Dado
128	0
129	0
130	0
131	0
132	0
133	0
134	0
135	0
136	0
137	0
138	0
139	0
140	0
141	0
142	0
143	0
144	0

Below the Ahmes window is a **Mnemônicos** table:

NOP	00	JMP	128 end	SHR	224
STA	16 end	JN	144 end	SHL	225
LDA	32 end	JP	148 end	ROR	226
ADD	48 end	JV	152 end	ROL	227
OR	64 end	JNV	156 end	HLT	240
AND	80 end	JZ	160 end		
NOT	96	JNZ	164 end		
SUB	112 end	JC	176 end		
		JNC	180 end		
		JB	184 end		
		JNB	188 end		

N = Negativo
 Z= Zero
 V = Overflow
 (estouro)
 C = Carry
 (vai-um)
 B = Borrow
 (vem-um)

Simuladores Didáticos

<ftp://ftp.inf.ufrgs.br/pub/inf107/>

<ftp://ftp.inf.ufrgs.br/pub/inf108/>

Ahmes Versão 2.1
 Julho 2002

Autores: Fabul Fernando Weber
 Taicy Silva Weber

Versão: Fabio Augusto Dal Castel
 Win32

OK

2. Arquiteturas Didáticas

Evolução do Neander... Ahmes, Ramses, Cesar

Programa

P	End.	Dado	Mnemônico
0	0	0	NOP
1	0	0	NOP
2	0	0	NOP
3	0	0	NOP
4	0	0	NOP
5	0	0	NOP
6	0	0	NOP
7	0	0	NOP
8	0	0	NOP
9	0	0	NOP
10	0	0	NOP
11	0	0	NOP
12	0	0	NOP
13	0	0	NOP
14	0	0	NOP
15	0	0	NOP

BP: 255 [0]: 0

Ramses v1.2

Arquivo Editar Visualizar Executar ?

RA: 000 RB: 000 RX: 000 PC: 000

N Z C

Execução:

Acessos: 0000 Instr.: 0000

Instrução:

RI: 0 Mnem: NOP

0..9 0..F [F1] [F2]

Ok.

Códigos das instruções

NOP	0	JMP	128	end	Modo: 0: Dir: n 1: Ind: n,l 2: Imd: #n 3: Idx: nX
STR	16	JN	144	end	
LDR	32	JZ	160	end	
ADD	48	JC	176	end	
OR	64	JSR	192	end	Registrador: 0: A 2: X 1: B 3: ?
AND	80	NEG	208	r	
NOT	96	SHR	224	r	
SUB	112	HLT	240		

Dados

End.	Dado
128	0
129	0
130	0
131	0
132	0
133	0
134	0
135	0
136	0
137	0
138	0
139	0
140	0
141	0
142	0
143	0

[128]: 0

Ramses Versão 1.2 Outubro 2003

Autores: Raul Fernando Weber
 Talay Siva Weber

Versão: Win32 Fábio Augusto Dal Castel

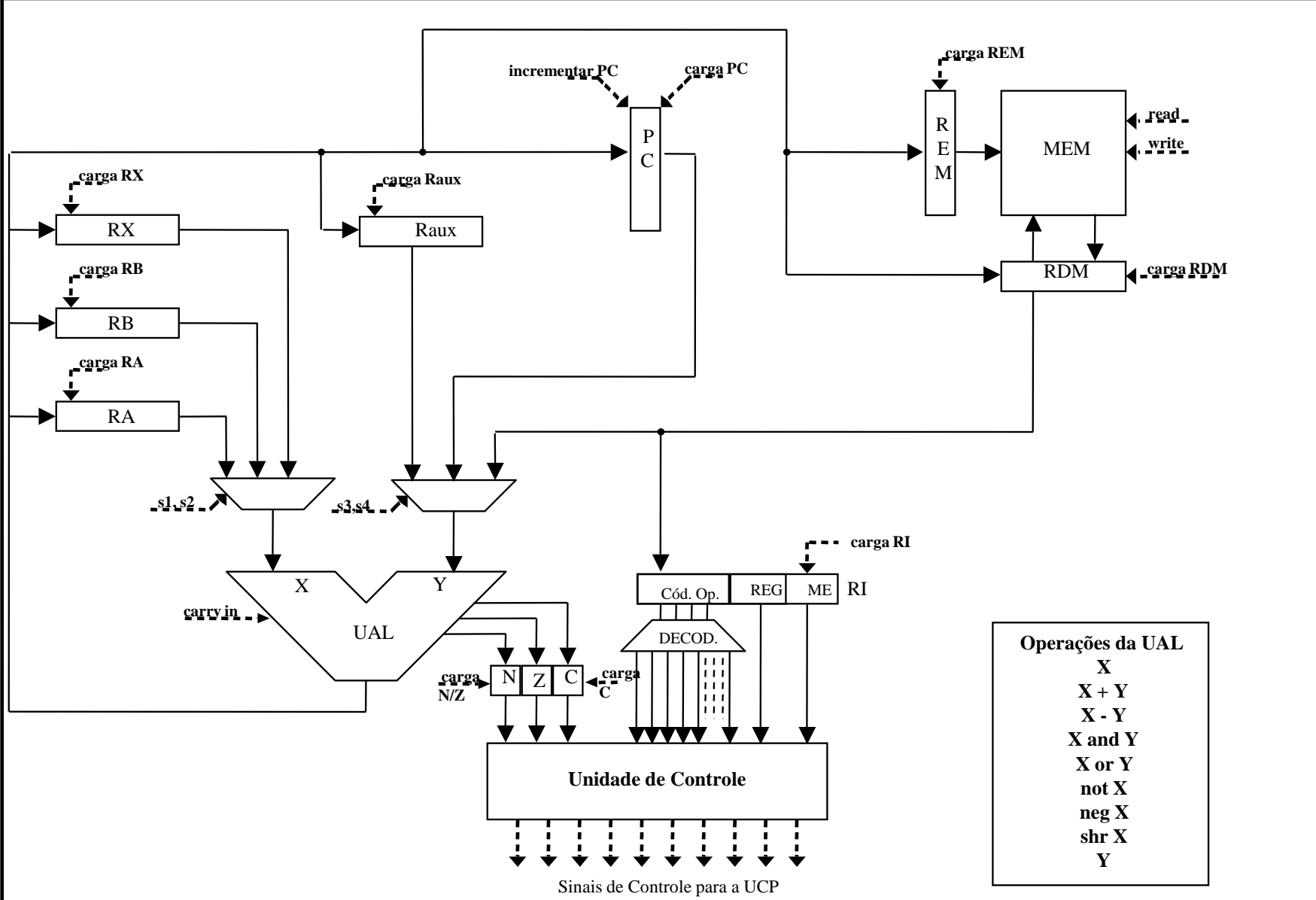
Ok

Simuladores Didáticos

<ftp://ftp.inf.ufrgs.br/pub/inf107/>

<ftp://ftp.inf.ufrgs.br/pub/inf108/>

Arquiteturas Didáticas: Ramses



- Operações da UAL**
- X
 - X + Y
 - X - Y
 - X and Y
 - X or Y
 - not X
 - neg X
 - shr X
 - Y

INFORMAÇÕES SOBRE A DISCIPLINA

USP - Universidade de São Paulo - São Carlos, SP

ICMC - Instituto de Ciências Matemáticas e de Computação

SSC - Departamento de Sistemas de Computação

Prof. Fernando Santos OSÓRIO

Web institucional: <http://www.icmc.usp.br/ssc/>

Página pessoal: <http://www.icmc.usp.br/~fosorio/>

E-mail: [fosorio \[at\] icmc. usp. br](mailto:fosorio@icmc.usp.br) ou [fosorio \[at\] gmail. com](mailto:fosorio@gmail.com)

Disciplina de Organização de Computadores Digitais / BSI

Web disciplina: Wiki ICMC - [Http://wiki.icmc.usp.br](http://wiki.icmc.usp.br)

> Programa, Material de Aulas, Critérios de Avaliação,

> Lista de Exercícios, Trabalhos Práticos, Datas das Provas