

**USP - ICMC - SSC  
SSC 0610 - Eng. Comp. - 2o. Semestre 2010**

## **Disciplina de Organização de Computadores I**

**Prof. Fernando Santos Osório**

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

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

**Estagiário PAE Maurício Dias - Email: [maccddias \[at\] gmail.com](mailto:maccddias@gmail.com)**

**Material on-line Wiki ICMC - <http://wiki.icmc.usp.br/index.php/Ssc-610>**

*Aula 04s*

## **Apresentação da Disciplina**

### **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

3

Agosto 2009

## 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

4

Agosto 2009

## 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_C 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

5

Agosto 2009

## 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 \$D2
\$0C	F0	HLT

Memória	Valor
\$C8	01
\$C9	02
\$CA	03
\$CB	04
\$CC	05
\$D2	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

6

Agosto 2009

## 2. Programação do Neander

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

Linguagem de Montagem:

```

Ini:      ORG $00
          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
    
```

Mnemônicos					
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

7

Agosto 2009

## 2. Programação do Neander

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

Linguagem de Montagem: >> Solução Alternativa <<

```

Ini:      ORG $00
          LDA Val_A
          ADD Val_B
          STA Result

Fim:     HLT

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

Mnemônicos					
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	F0	

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

```

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

8

Agosto 2009

## 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
          DB $FF
          Contador DB $0A
          END
    
```

Mnemônicos					
<b>NOP</b>	0	<b>ADD</b>	30 end	<b>JMP</b>	80 end
<b>STA</b>	10 end	<b>OR</b>	40 end	<b>JN</b>	90 end
<b>LDA</b>	20 end	<b>AND</b>	50 end	<b>JZ</b>	A0 end
		<b>NOT</b>	60	<b>HLT</b>	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

## 2. Programação do Neander

### Programação do Neander : (4) Somar os dados de um vetor

```

EndValor EQU Início+1
          ORG $00
          ORG $A0
; Soma valores do vetor em total
Início:   LDA Valores
          ADD Total
          STA Total
; Altera endereço do valor somado
          LDA EndValor
          ADD Valor01
          STA EndValor
; Contador de nros. somados
          LDA Contador
          ADD Vminus1
          STA Contador
          JZ Fim:
          JMP Início

Fim:      HLT
          END
    
```

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

## 2. Programação do Neander

### Programação do Neander : (5) Somar valores com mais de 8 bits (!)

```

                ORG $00
Inicio: LDA V1L    ; Soma Low Bytes
        ADD V2L
        JN TrataN: ; Trata vai-um (negativo)
        STA RESL

                ORG $A0
                ; Bytes LOW e HI de V1
A0: V1L  DB $7F
A1: V1H  DB $00
                ; Bytes LOW e HI de V2
A2: V2L  DB $01
A3: V2H  DB $01
                ; Bytes LOW e HI Result
A4: RESL DB $00
A5: RESH DB $00
                ; Valores auxiliares
A6: Valor01 DB $01
A7: VaiUm  DB $00
A8: ZeraB7  DB $7F

SomaHi: LDA V1H    ; Soma High Bytes
        ADD V2H
        ADD VaiUm  ; Soma Vai um
        STA RESH
        HLT

TrataN: AND ZeraB7  ; Zera bit 7 (sinal)
        STA RESL   ; Guarda resultado
        LDA Valor01 ; Sinaliza Vai um
        STA VaiUm
        JMP SomaHi: ; Prossegue a soma
```

11

Agosto 2009

## 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)

12

Agosto 2009

## 2. Arquiteturas Didáticas

### Evolução do Neander... Ahmes, Ramses, Cesar

#### Quadro comparativo

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

Simuladores Didáticos

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

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

[http://pt.wikipedia.org/wiki/Máquinas\\_hipotéticas\\_da\\_Universidade\\_Federal\\_do\\_Rio\\_Grande\\_do\\_Sul](http://pt.wikipedia.org/wiki/Máquinas_hipotéticas_da_Universidade_Federal_do_Rio_Grande_do_Sul)

13

Agosto 2009

## 2. Arquiteturas Didáticas

### Evolução do Neander... Ahmes, Ramses, Cesar

The screenshot displays the Ahmes simulator interface. On the left, the 'Programa' window shows a list of instructions (NOP) with addresses from 0 to 16. The central 'Ahmes' window shows the current state of the processor, including the Accumulator (AC) and Program Counter (PC) registers, and status flags (N, Z, V, C, B). Below these are fields for 'Execução', 'Acessos', and 'Instr.'. On the right, the 'Dados' window shows a memory dump with addresses from 129 to 144. At the bottom, a 'Mnemônicos' window lists various instructions and their addresses, such as NOP (00), STA (16 end), LDA (32 end), ADD (48 end), OR (64 end), AND (80 end), NOT (96), SUB (112 end), JMP (128 end), JN (144 end), JP (148 end), JV (152 end), JNV (156 end), JZ (160 end), JNZ (164 end), JC (176 end), JNC (180 end), JB (184 end), and JNB (188 end). Other instructions like SHR, SHL, ROR, ROL, and HLT are also listed.

Simuladores Didáticos

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

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

14

Agosto 2009

## 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

**Ramses v1.2**

RA: 0000 RB: 0000 RX: 0000 PC: 0000

N: 0 Z: 1 C: 0

Execução: 00000000 Instrução: 00000000

Acessos: 00000000 RI: 0

Instr.: 00000000 Mnem: NOP

0..9 0..F [0] [128]

**Códigos das instruções**

Nome	Código	Modo
NOP	0	
STR	16 r end	0: Dir: n
LDR	32 r end	1: Ind: n,l
ADD	48 r end	2: lmd: #n
OR	64 r end	3: ldc: n,x
AND	80 r end	
NOT	96 r	
SUB	112 r end	
JMP	128 end	
JN	144 end	
JZ	160 end	
JC	176 end	
JSR	192 end	
NEG	208 r	
SHR	224 r	
HLT	240	

**Ramses** Versão 1.2 Outubro 2003  
Autor: Prof. Fernando Osório  
Taty Sá e Wilson  
Versão: Fátima Augusta D.J. Casali  
Win32

Simuladores Didáticos  
<ftp://ftp.inf.ufgrs.br/pub/inf107/>  
<ftp://ftp.inf.ufgrs.br/pub/inf108/>

15

Agosto 2009



### 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 I / Eng. Comp.**

**Estagiário PAE: Maurício A. Dias**

**Web disciplina:** <http://wiki.icmc.usp.br/index.php/Ssc-610>

**> Programa, Material de Aulas, Critérios de Avaliação,**

**> Lista de Exercícios, Trabalhos Práticos, Datas das Provas**

16

Agosto 2010