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)

IRM
Lab. de Robótica Móvel
CROB
Centro de Robótica da USP

Aula 03s

Apresentação da Aula

Agenda:

1. NEANDER
   - Arquitetura do Processador
   - Organização dos Componentes
   - Conjunto de Instruções
2. Programando o NEANDER

4. Arquiteturas Didáticas

Neander - Computador Hipotético [Weber 2001*]

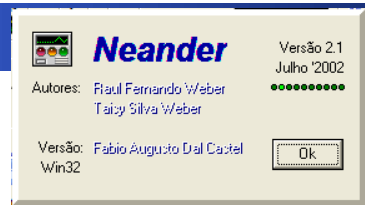
Arquitetura: características gerais

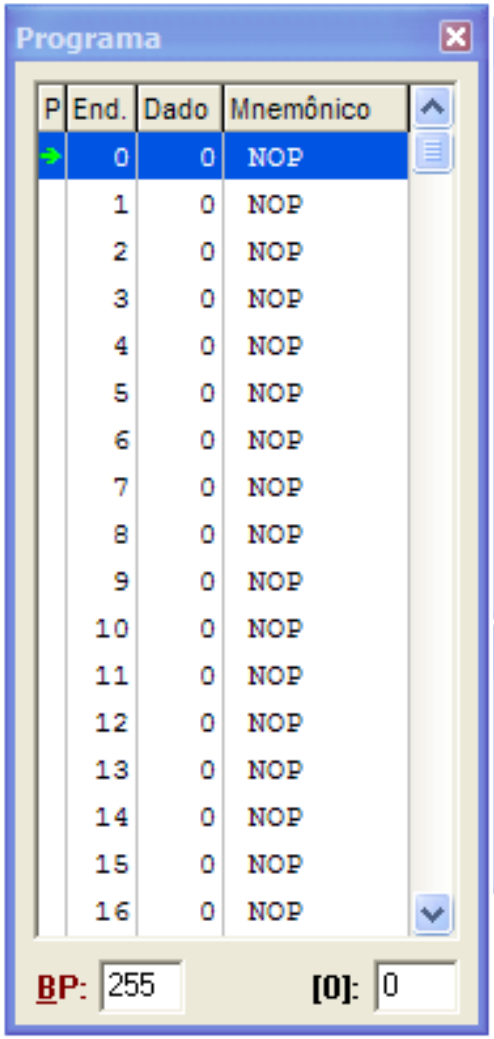
- Largura de dados e endereços de 8 bits (bus)
- Dados representados em complemento de 2
- Acumulador de 8 bits (AC - Accumulator)
- Apontador de programa de 8 bits (PC - Program Counter)
- Registrador de Instruções de 8 bits (IR - Instruction Reg.)
- Registrador de estado (flags) com 2 códigos de condição: Negativo (N) e Zero (Z)
- Endereçamento de memória total de 256 bytes

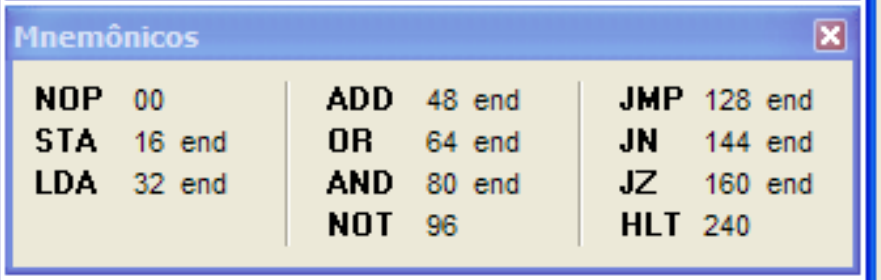
* ftp://ftp.inf.ufrgs.br/pub/inf107/

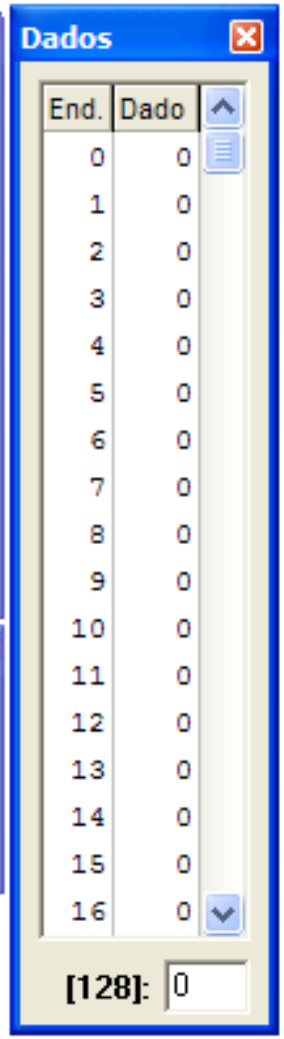
4. Arquiteturas Didáticas

Neander => Simulador WNeander





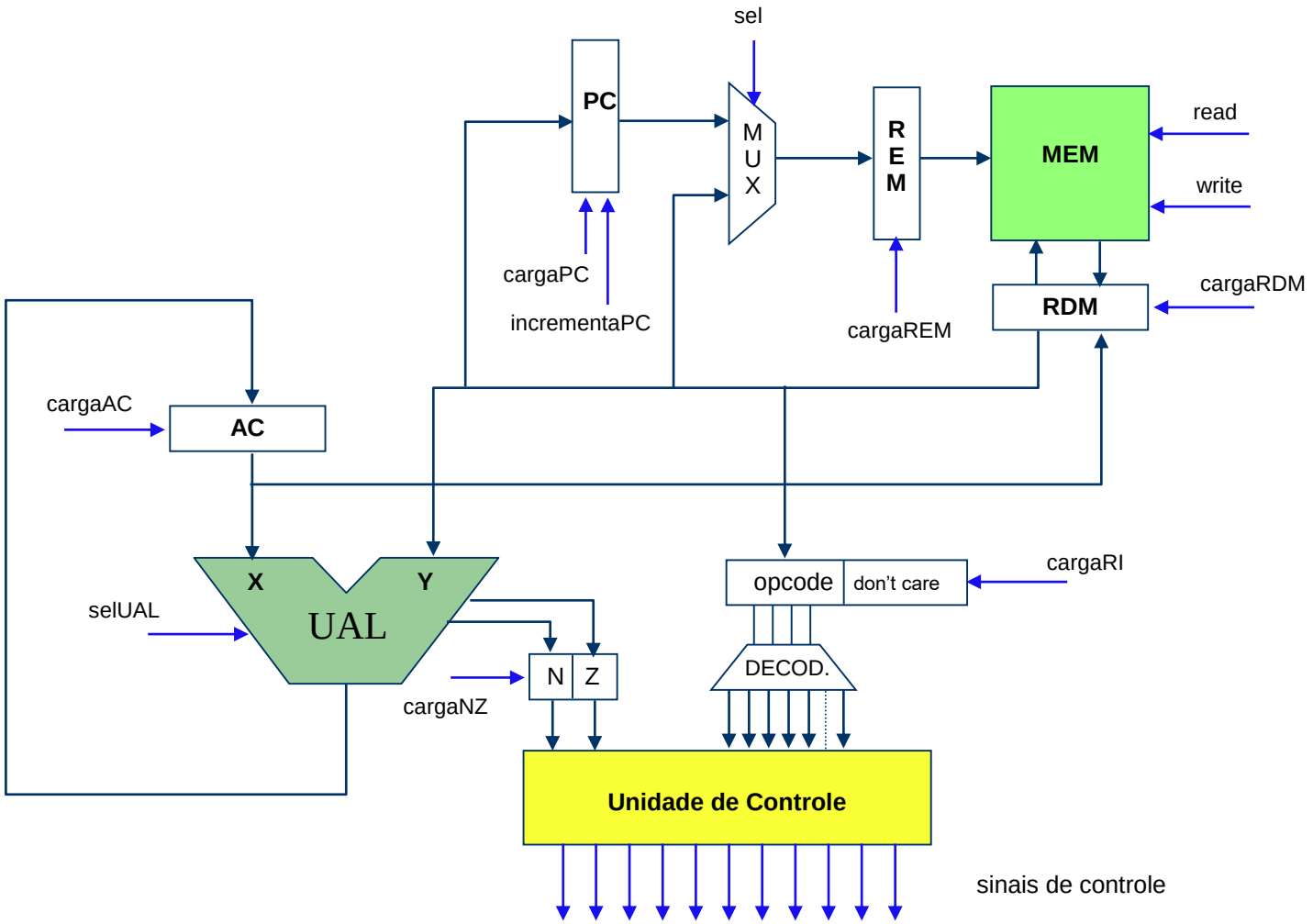




Simuladores Didáticos
ftp://ftp.inf.ufrgs.br/pub/inf107/
ftp://ftp.inf.ufrgs.br/pub/inf108/

4. Arquiteturas Didáticas

Neander - Computador Hipotético [Weber 2001*]

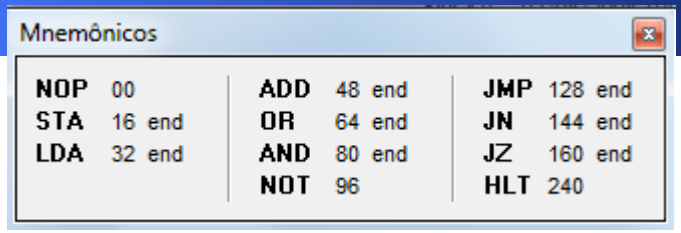


* ftp://ftp.inf.ufrgs.br/pub/inf107

Versão Web: http://inf.ufrgs.br/~vbuaraujo/sw/neander/

2. Programação do Neander

Programação do Neander



Código Binário	Instrução em Hexa	Instr. em Dec.	Instrução (mnemônico)	comentário
0000	00 ---	0	NOP	Nenhuma operação (no op.)
0001	10 XX	16	STA end	MEM(end) ← AC Store
0010	20 XX	32	LDA end	AC ← MEM(end) Load
0011	30 XX	48	ADD end	AC ← MEM(end) + AC
0100	40 XX	64	OR end	AC ← MEM(end) OR AC
0101	50 XX	80	AND end	AC ← MEM(end) AND AC
0110	60 ---	96	NOT	AC ← NOT AC
1000	80 XX	128	JMP end	PC ← end Jump
1001	90 XX	144	JN end	IF N=1 THEN PC ← end
1010	A0 XX	160	JZ end	IF Z=1 THEN PC ← end
1111	F0 ---	240	HLT	Para processamento Halt

2. Programação do Neander

Programação do Neander :

Programação do Neander – Exercícios

1) Somar vários valores de 8 bits
   Soma: Valor1 + Valor2 + Valor3 + Valor4 + Valor5
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 positivos compostos de 2 bytes (16 bits!)
6) Multiplicar 2 valores (somas sucessivas)
7) Pesquisar um dado em uma tabela

2. Programação do Neander

Programação do Neander : Somar vários Valores

Linguagem de Montagem (Código Comentado)

```
Valor1   EQU 100  ; Endereço da variável Valor1 definido como 100  (64h)
Valor2   EQU 101  ; Endereço da variável Valor2 definido como 101  ($65)
Valor3   EQU 102  ; Endereço da variável Valor3 definido como 102  ($66)
Valor4   EQU 103  ; Endereço da variável Valor4 definido como 103  ($67)
Valor5   EQU 104  ; Endereço da variável Valor5 definido como 104  ($68)
Result   EQU 112  ; Endereço da variável Result definido como 112  ($70)

         ORG $00  ; Endereço inicial da execução do Prog. (Inicia com PC:00)

Ini:     LDA Valor1  ; Acumulador AC recebe conteúdo de Valor1 (End. 100)
         ADD Valor2  ; Soma AC = AC + conteúdo do end.  Valor2 (End. 101)
         ADD Valor3  ; Soma AC = AC + conteúdo do end.  Valor2 (End. 102)
         ADD Valor4  ; Soma AC = AC + conteúdo do end.  Valor2 (End. 103)
         ADD Valor5  ; Soma AC = AC + conteúdo do end.  Valor2 (End. 104)
         STA Result  ; Salva o resultado do AC na memória(End. 112)
Fim:     HLT         ; Termina a execução;

         END         ; Fim do código de Montagem
```

2. Programação do Neander

Programação do Neander : Somar vários Valores

Linguagem de Montagem (Código sem comentários)

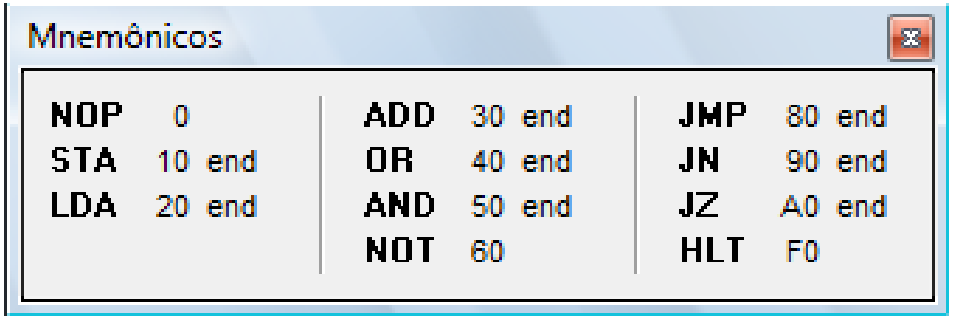
```
Valor1      EQU $64
Valor2      EQU $65
Valor3      EQU $66
Valor4      EQU $67
Valor5      EQU $68
Result      EQU $70

            ORG $00

Ini:        LDA Valor1
            ADD Valor2
            ADD Valor3
            ADD Valor4
            ADD Valor5
            STA Result
Fim:        HLT

            END
```

2. Programação do Neander

Programação do Neander : Somar vários Valores

Linguagem de Montagem (Código Básico – Sem variáveis, constantes e labels)

LING. DE MONTAGEM	MONTADOR	CÓDIGO DE MÁQUINA
ASSEMBLY	ASSEMBLER	BINARY/MACHINE CODE
LDA $64	=>	$20 $64
ADD $65	=>	$30 $65
ADD $66	=>	$30 $66
ADD $67	=>	$30 $67
ADD $68	=>	$30 $68
STA $70	=>	$10 $70
HLT	=>	$F0



2. Programação do Neander

Programação do Neander : Soma Valores – Código em Memória

Memória	Instrução em Hexa	Instrução
$00	$20 $64	LDA Valor1
$02	$30 $65	ADD Valor2
$04	$30 $66	ADD Valor3
$06	$30 $67	ADD Valor4
$08	$30 $68	ADD Valor5
$0A	$10 $70	STA Result
$0C	$F0	HLT
$64	01	Valor1
$65	02	Valor2
$66	03	Valor3
$67	04	Valor4
$68	05	Valor5
$70	00	Result

2. Programação do Neander

Programação do Neander :

Programação do Neander – LISTA de Exercícios (Internet)

1) Crie um programa para tornar um dado na memória negativo, usando complemento de dois.
2) Faça um programa para realizar a multiplicação de dois números a * b (somas sucessivas)
3) Faça um programa para realizar a divisão de dois números a / b (subtrações sucessivas)
4) Faça um programa para realizar a operação a > b
   Se a > b, o programa deverá escrever 1 em uma posição da memória.
   Caso contrário, deverá escrever 0
5) Faça um programa para decidir se um número natural é par ou não.
   Caso o número seja par, o programa deverá escrever 1 na memória.
   Caso contrário, deverá escrever 0

2. Programação do Neander

Programação do Neander : Gera número em Complemento de 2

Linguagem de Montagem (Código Comentado)

```
        ORG $00  ; Endereço inicial da execução do Prog. (Inicia com PC:00)

Ini:    LDA Entrada ; Acumulador AC recebe conteúdo de Entrada (End. $A1)
        NOT         ; Inverte os bits do Acumulador
        ADD ValorUm ; Soma AC = AC + conteúdo do end.  ValorUm (End. $A0)
        STA Saida   ; Salva o resultado do AC na memória (End. $A2)
Fim:    HLT         ; Termina a execução;

        ORG $A0
CteUm   EQU  #01          ; Valor constante #01
ValorUm DB   CteUm        ; Cria uma variável com o valor constante #01
Entrada DB   #75          ; Reserva o espaço de memória para 1 byte
Saida   DB   1            ; Reserva o espaço de memória para 1 byte

        END         ; Fim do código de Montagem
```

2. Programação do Neander

Programação do Neander : Complemento de 2

Memória	Instrução em Hexa	Instrução
$00	$20 $A1	LDA Entrada
$02	$60	NOT
$03	$30 $A0	ADD ValorUm
$05	$10 $A2	STA Saida
$07	$F0	HLT
$08	$00	NOP
$09	$00	NOP
$A0	01	ValorUm
$A1	75	Entrada
$A2	00	Saida

```
            ORG $00
Ini:        LDA Entrada
            NOT
            ADD ValorUm
            STA Saida
Fim:        HLT

            ORG $A0
CteUm       EQU  #01
ValorUm     DB   CteUm
Entrada     DB   #75
Saida       DB   1

            END
```

2. Programação do Neander

Programação do Neander :

Programação do Neander – Exercícios

1) Somar vários valores de 8 bits

   Soma: Valor1 + Valor2 + Valor3 + Valor4 + Valor5
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 positivos compostos de 2 bytes (16 bits!)
6) Multiplicar 2 valores (somas sucessivas)
7) Pesquisar um dado em uma tabela

Mnemônicos Hexadecimal

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	

Mnemônicos Decimal

NOP	00		ADD	48 end		JMP	128 end	
STA	16 end		OR	64 end		JN	144 end	
LDA	32 end		AND	80 end		JZ	160 end	
			NOT	96		HLT	240	

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)

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 ou fosorio [at] gmail. com

Disciplina de Organização de Computadores Digitais / BSI
Web disciplina: Wiki ICMC - Http://wiki.icmc.usp.br
> Programa, Material de Aulas, Critérios de Avaliação,
> Lista de Exercícios, Trabalhos Práticos, Datas das Provas