

Conjuntos de Instruções de Arquitetura BIP



Introdução

(VIEIRA; RAABE; ZEFERINO, 2009)

A escolha de processadores para o ensino da lógica de programação e de conceitos de arquitetura de computadores deve facilitar o estabelecimento de relações entre:

- Abstrações lógicas necessárias à programação.
- Implementação dessas abstrações em hardware.

Os modelos de processadores tipicamente utilizados são muito abstratos e dificultam o estabelecimento dessas relações.

Processadores BIP

(VIEIRA; RAABE; ZEFERINO, 2009)

Discussões entre professores na UNIVALI levaram à concepção de uma família de processadores com arquitetura simplificada.

Esta família é composta por uma série de processadores denominados **BIP** (***Basic Instruction-set Processor***).

Foi desenvolvida pelos pesquisadores do Grupo de Sistemas Embarcados e Distribuídos da Universidade do Vale do Itajaí (UNIVALI).

Conjunto de Instruções

Um conjunto de instruções refere-se ao conjunto de todas as instruções reconhecidas por um computador.

Uma instrução indica ao processador uma sequência de micro-operações que deverá ser executada.

De acordo com seu propósito e formato, as instruções podem ser classificadas como:

- Instruções de transferência de dados
- Instruções aritméticas e lógicas
- Instruções de desvio (condicional ou incondicional)
- Instruções de teste (ou de comparação)

Conjunto de Instruções BIP

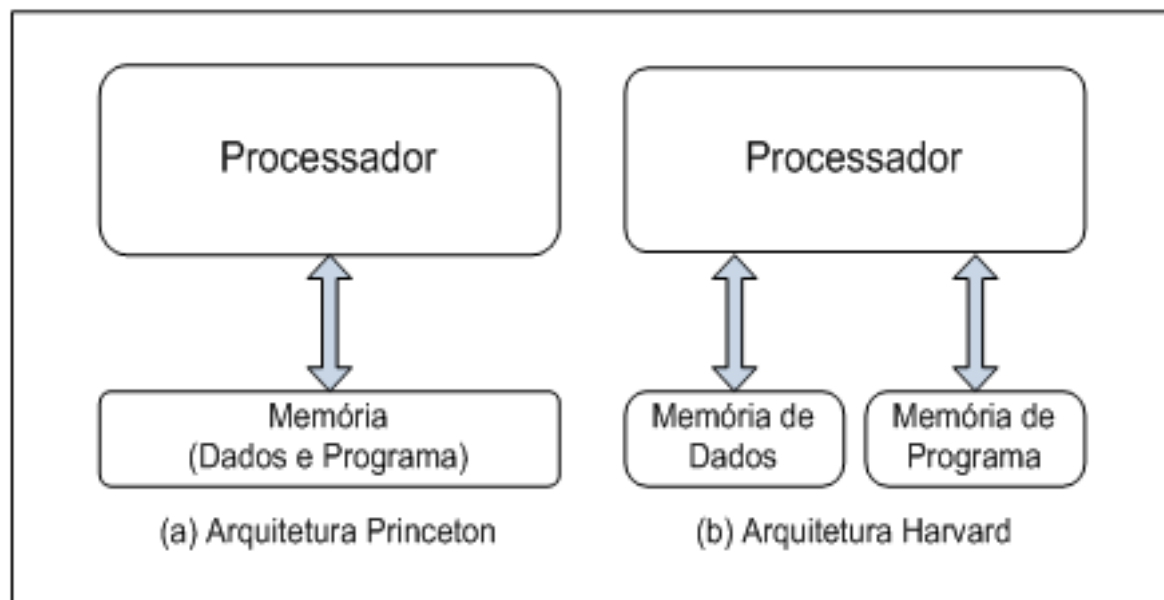
- **BIP I** - Inclui instruções aritméticas e de acesso às variáveis armazenadas na memória de dados.
- **BIP II** - Adiciona instruções de desvio, oferecendo suporte para estruturas de controle.
- **BIP III** - Inclui instruções de lógica, suportando operações bit-a-bit.
- **BIP IV** - Acrescenta suporte para operações de entrada e saída, sub-rotinas com passagem de parâmetros, deslocamento lógico e manipulação de vetores.

Conjunto de Instruções BIP

Características	BIP I	BIP II	BIP III	BIP IV
Instruções de transferência	X	X	X	X
Instruções de aritmética	X	X	X	X
Instruções de controle	X	X	X	X
Instruções de desvio		X	X	X
Operações bit-a-bit			X	X
Operações de E/S				X
Manipulação de vetores				X
Suporte a sub-rotinas				X
Deslocamento lógico				X

Organização dos processadores BIP

A organização do BIP I utiliza a estrutura Harvard, com memórias separadas para dados e instruções.



Processador BIP I

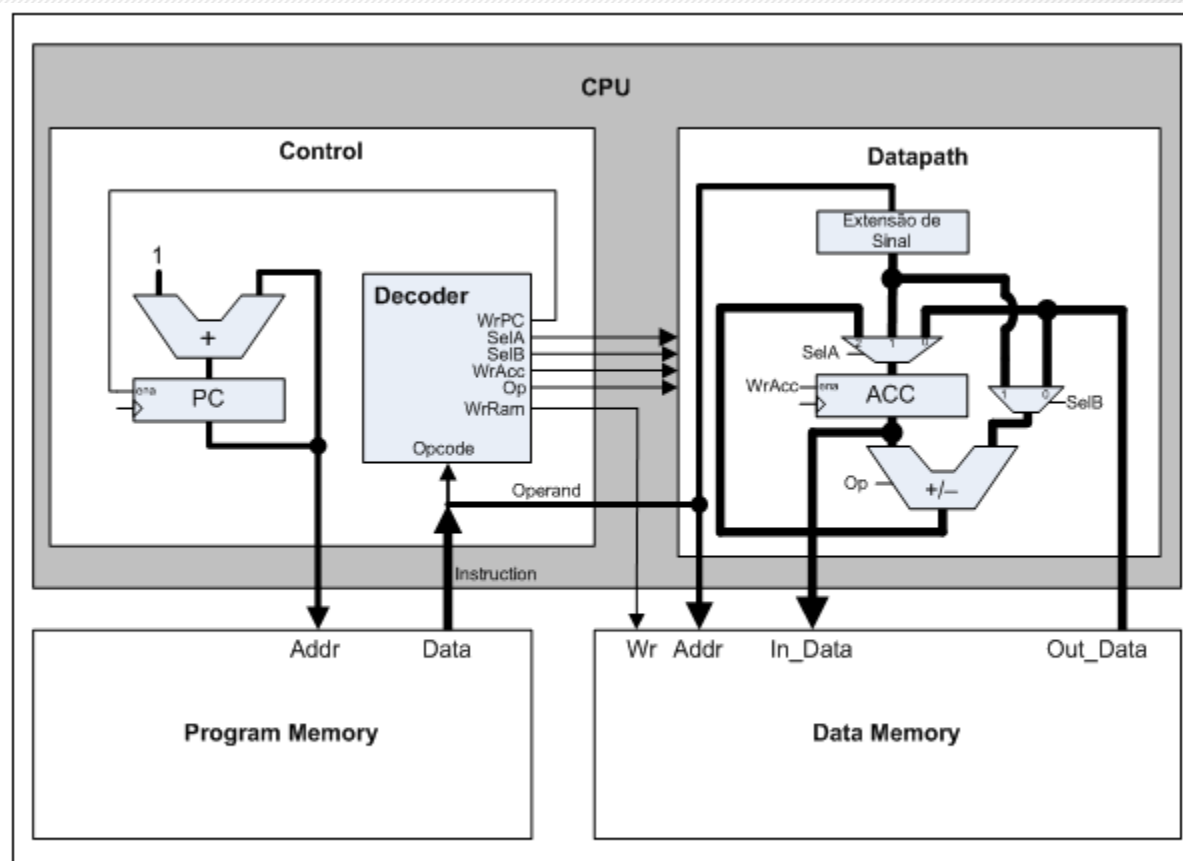


Figura 3. Organização do processador BIP I

Fonte: Zeferino (2007)

Processador BIP II

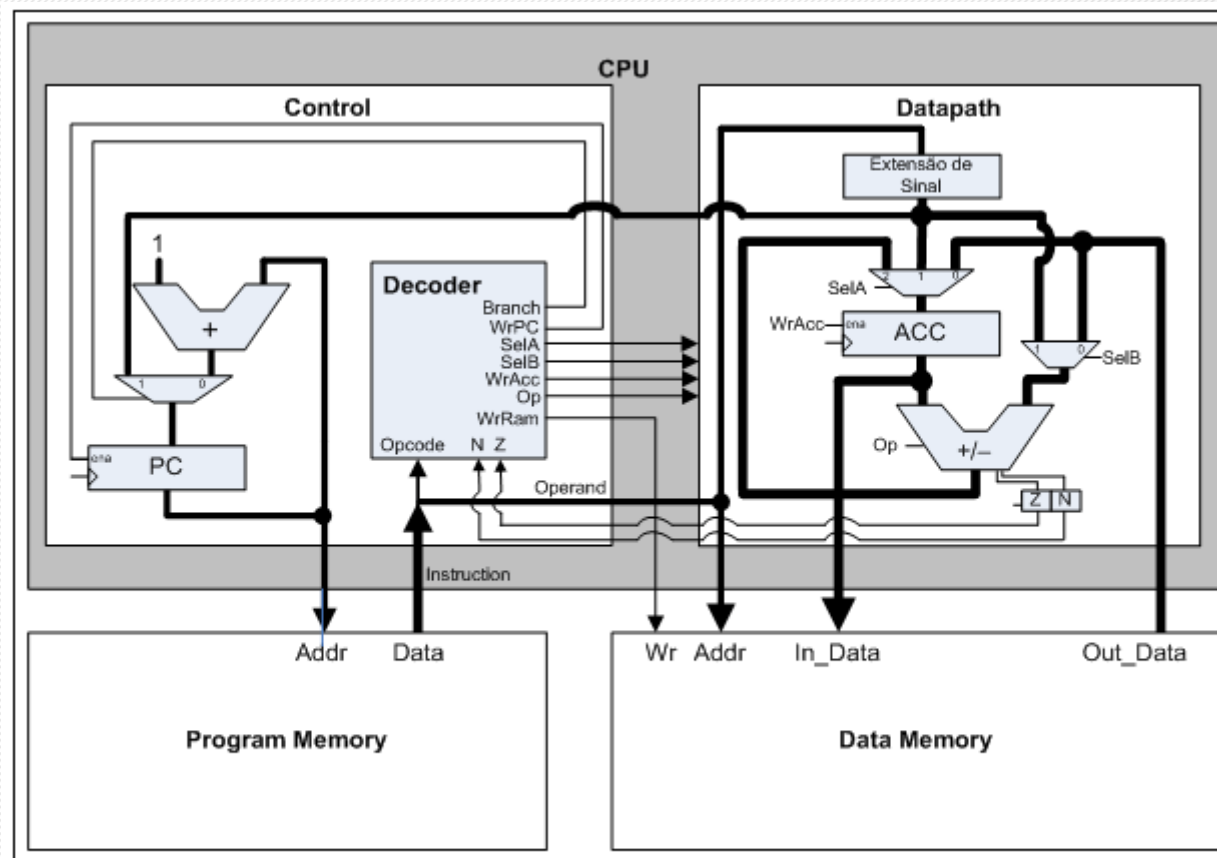
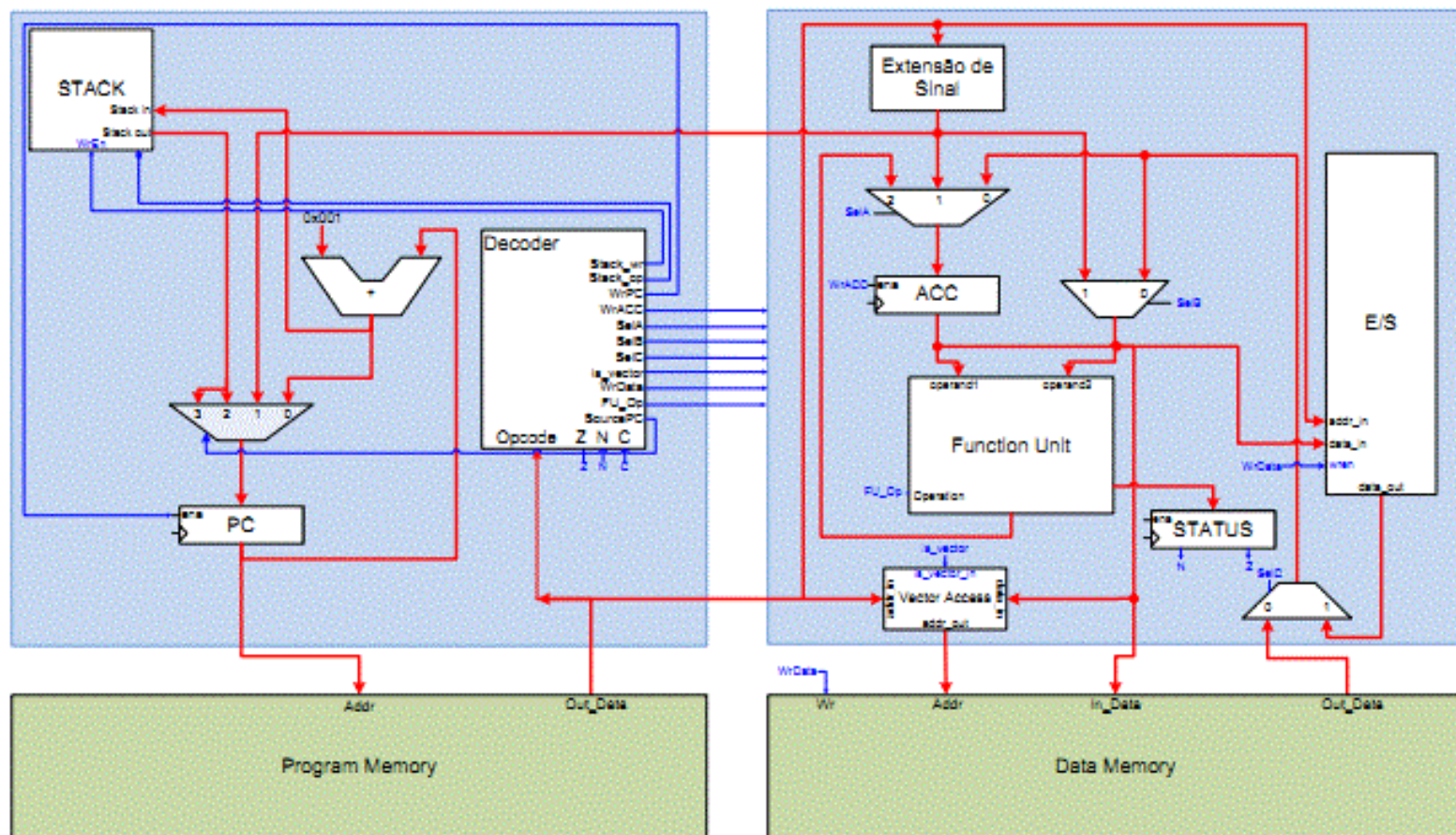


Figura 4. Organização do Processador BIP II

Fonte: Pereira (2008)

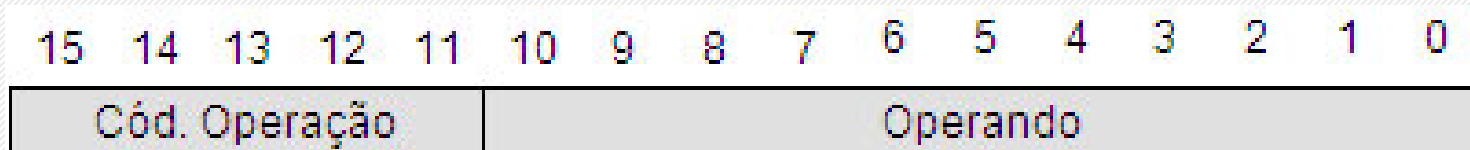
Processador BIP IV



Formato de instruções BIP

A arquitetura do BIP possui um conjunto restrito de instruções com poucos modos de endereçamento.

Todas as instruções são baseadas no formato:



- **5 bits** para o código de operação, o que permite implementar até 32 instruções; e
- **11 bits** reservados para o operando, o que possibilita representar até 2048 posições de endereçamento ou qualquer constante com valores entre -1024 até +1023

Formato de instruções BIP

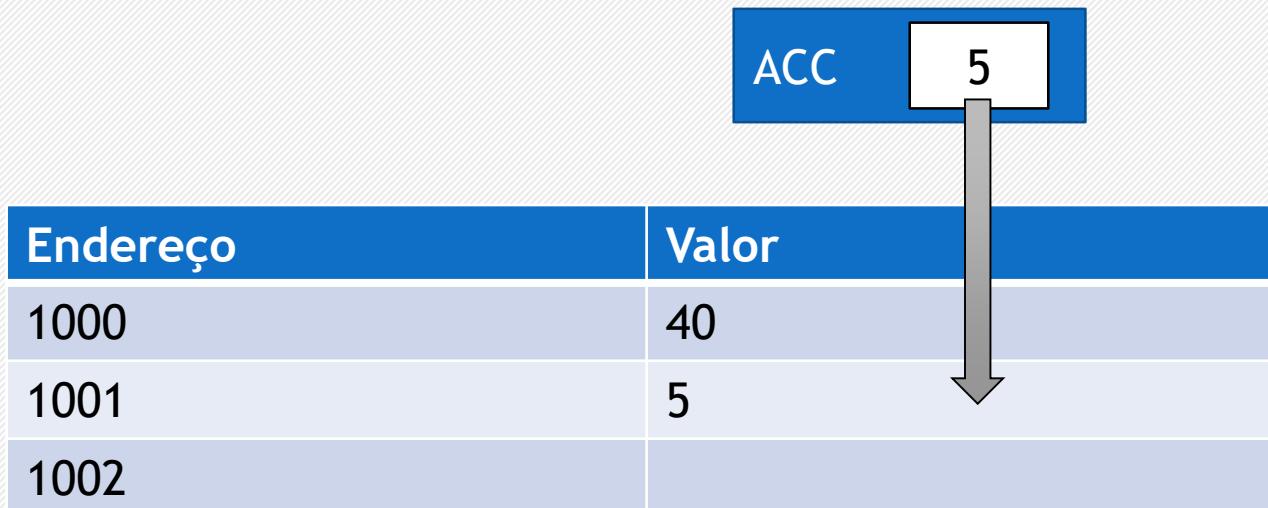
Exemplo: **LDI 5** carrega o valor do operando imediato 5 no registrador acumulador.



Endereço	Valor
1000	40
1001	
1002	

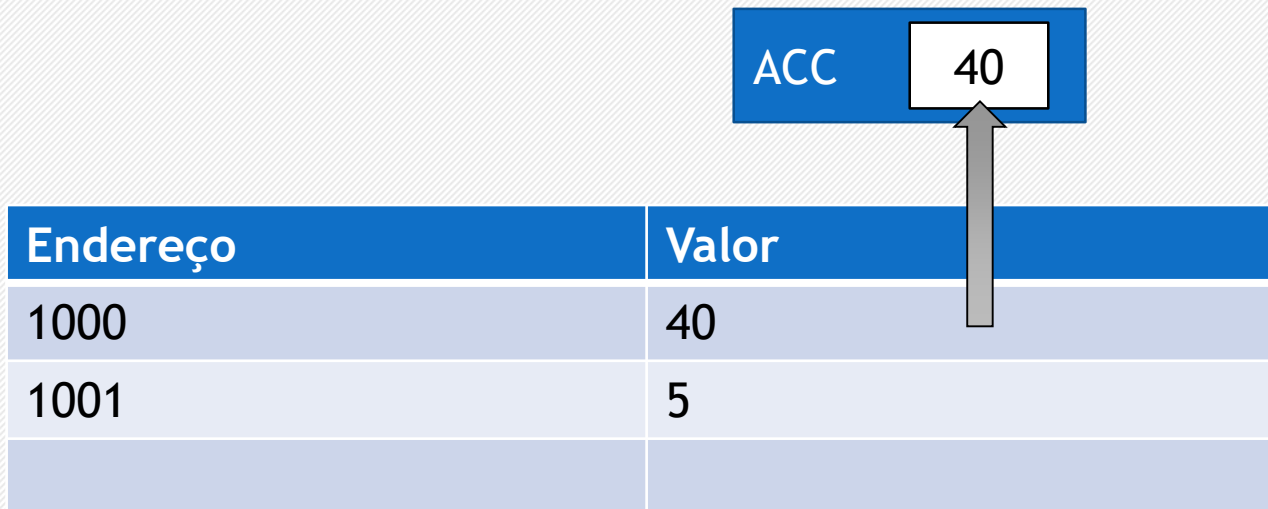
Formato de instruções BIP

Exemplo: **STO 1001** armazena o conteúdo do registrador acumulador no endereço 1001 de memória.



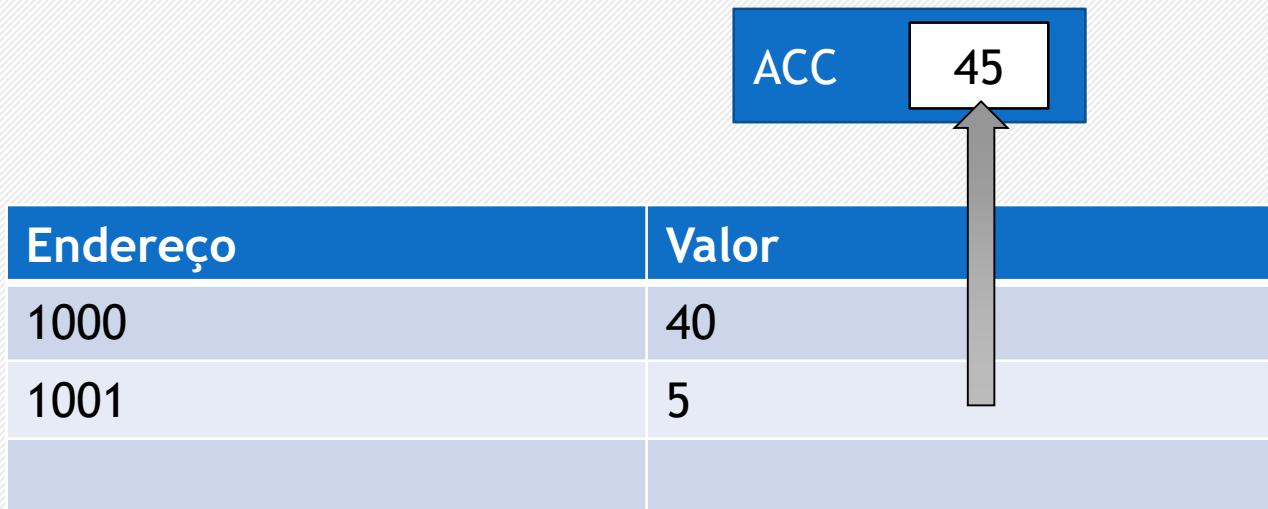
Formato de instruções BIP

Exemplo: **LD 1000** carrega o valor armazenado na posição 1000 de memória no registrador acumulador.



Formato de instruções BIP

Exemplo: **ADD 1001** carrega o valor armazenado na posição 1001 de memória e soma com o valor armazenado no registro acumulador.



Conjunto de instruções BIP

Instruções de controle

HLT (*halts*) - Desabilita a atualização do PC.



PC

Endereço	Instrução
0	LDI 1
1	STO 1001
2	LD4
...	...

Memória de programa

Conjunto de instruções BIP

Instruções de armazenamento

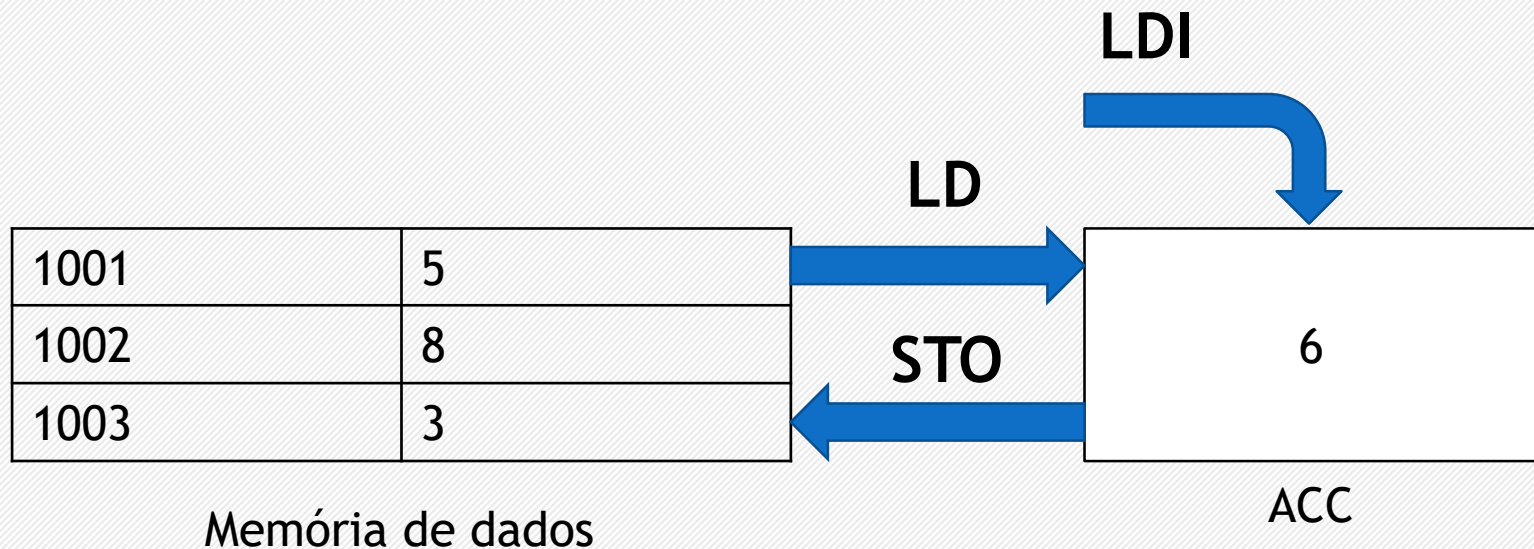
LD (*load*) - Carrega um valor armazenado em uma posição de memória indicada pelo operando para o registrador ACC .

LDI (*load immediate*) - Carrega o valor de um operando imediato para o registrador ACC.

STO (*store*) - Armazena o conteúdo do ACC em uma posição da memória indicada pelo operando.

Conjunto de instruções BIP

Instruções de armazenamento



Conjunto de instruções BIP

Instruções aritméticas

ADD - O valor do registrador ACC é somado com o valor armazenado na posição de memória indicado pelo campo operando e o resultado armazenado no ACC.

ADDI (*immediate*) - O valor do registrador ACC é somado com o valor operando imediato e o resultado armazenado no ACC.

SUB - O valor do registrador ACC é subtraído pelo valor armazenado na posição de memória indicado pelo campo operando e o resultado armazenado no ACC.

SUBI (*immediate*) - O valor do registrador ACC é subtraído pelo valor do operando imediato e o resultado armazenado no ACC.

Conjunto de instruções BIP

Instruções de desvio

JMP (*jump*) - Atualiza o valor do PC com o valor do campo operando, ou seja, realiza um desvio incondicional.

BEQ (*branch equal*) - Atualiza o valor do PC com o valor do campo operando caso o resultado da operação anterior na ULA foi zero.

BNE (*branch non-equal*) - Atualiza o valor do PC com o valor do campo operando caso o resultado da operação anterior na ULA tenha sido diferente de zero.

Conjunto de instruções BIP

Instruções de desvio

BGT (*branch greater than*) - Atualiza o valor do PC com o valor do campo operando caso o resultado da operação anterior na ULA tenha sido maior que zero.

BGE (*branch greater equal*) - Atualiza o valor do PC com o valor do campo operando caso o resultado da operação anterior na ULA tenha sido maior ou igual a zero.

BLT (*branch less than*) - Atualiza o valor do PC com o valor do campo operando caso o resultado da operação anterior na ULA tenha sido menor que zero.

BLE (*branch less equal*) - Atualiza o valor do PC com o valor do campo operando caso o resultado da operação anterior na ULA tenha sido menor ou igual a zero.

Conjunto de instruções BIP

Lógicas - NOT, AND, ANDI, OR, ORI, XOR, XORI

Deslocamento lógico - SLL, SRL

Manipulação de vetor - LDV, STOV

Suporte a procedimentos - CALL, RETURN

Exemplo

Portugol

```
procedimento principal()  
declaracoes  
    inteiro x  
inicio  
    x<-1  
    se (2 = 2) entao  
        x <- 30  
    fimse  
fim
```

Assembly

```
.data  
    x : 0  
.text  
_PRINCIPAL:  
    LDI    1  
    STO    x  
    LDI    2  
    STO    1000  
    LDI    2  
    STO    1001  
    LD     1000  
    SUB    1001  
    BNE    FIMSE1  
    LDI    30  
    STO    x  
FIMSE1:  
    HLT    0
```

Exemplo

Portugol

```
procedimento principal()  
declaracoes  
    inteiro x  
inicio  
    x<-1  
    se (2 = 2) entao  
        x <- 30  
    fimse  
fim
```

Assembly

```
.data  
    x : 0  
.text  
_PRINCIPAL:  
    LDI    1  
    STO    x  
    LDI    2  
    STO    1000  
    LDI    2  
    STO    1001  
    LD     1000  
    SUB    1001  
    BNE    FIMSE1  
    LDI    30  
    STO    x  
FIMSE1:  
    HLT    0
```


Exemplo

Portugol

```
procedimento principal()  
declaracoes  
  inteiro x  
inicio  
  x<-1  
  se (2 = 2) entao  
    x <- 30  
  fimse  
fim
```

Assembly

```
.data  
  x : 0  
.text  
_PRINCIPAL:  
  LDI    1  
  STO    x  
  LDI    2  
  STO    1000  
  LDI    2  
  STO    1001  
  LD     1000  
  SUB    1001  
  BNE    FIMSE1  
  LDI    30  
  STO    x  
FIMSE1:  
  HLT    0
```

Armazena os
valores a
serem
comparados

Exemplo

Portugol

```
procedimento principal()  
declaracoes  
    inteiro x  
inicio  
    x<-1  
    se (2 = 2) entao  
        x <- 30  
    fimse  
fim
```

Assembly

```
.data  
    x : 0  
.text  
_PRINCIPAL:  
    LDI    1  
    STO    x  
    LDI    2  
    STO    1000  
    LDI    2  
    STO    1001  
    LD     1000  
    SUB    1001  
    BNE    FIMSE1  
    LDI    30  
    STO    x  
FIMSE1:  
    HLT    0
```

Se #1000-#1001
for diferente de
zero, vai para
fimse

Exemplo

Portugol

```
procedimento principal()  
declaracoes  
    inteiro x  
inicio  
    x<-1  
    se (2 = 2) entao  
        x <- 30  
    fimse  
fim
```

Assembly

```
.data  
    x : 0  
.text  
_PRINCIPAL:  
    LDI    1  
    STO    x  
    LDI    2  
    STO    1000  
    LDI    2  
    STO    1001  
    LD     1000  
    SUB    1001  
    BNE    FIMSE1  
    LDI    30  
    STO    x  
FIMSE1:  
    HLT    0
```

Senão executa
a próxima
instrução

Exemplo

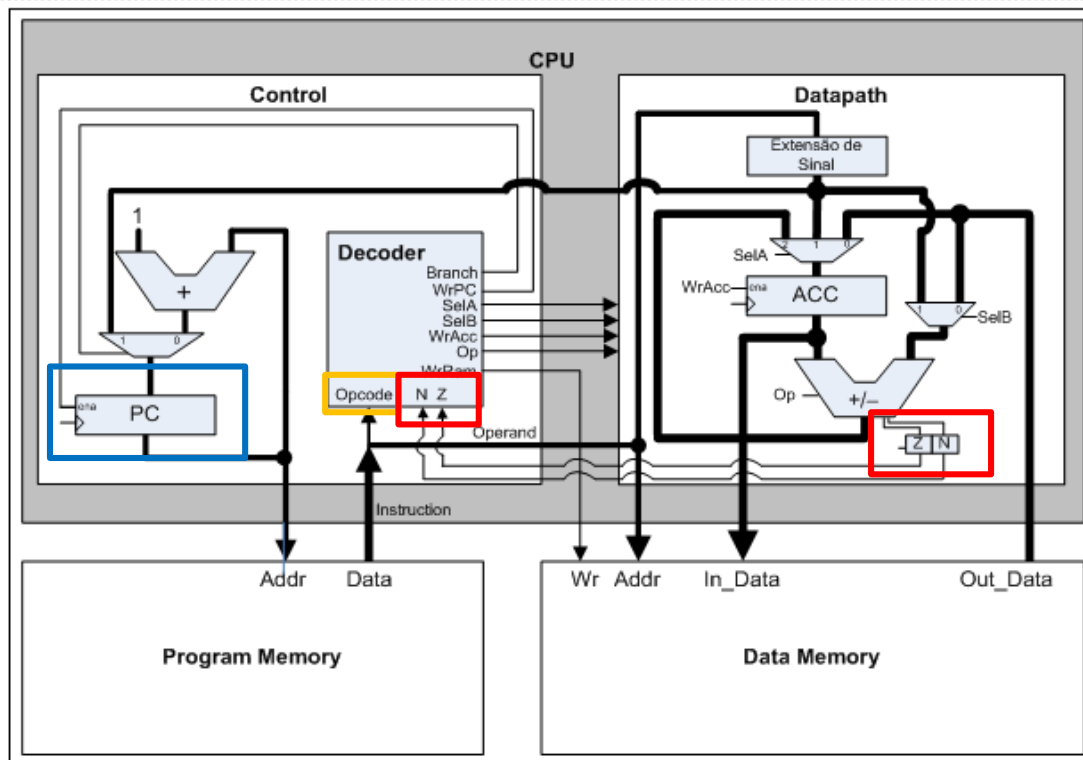


Figura 4. Organização do Processador BIP II

Fonte: Pereira (2008)

SUB	Subtract
Sintaxe:	SUB operand
Descrição:	O valor do registrador ACC é subtraído pe memória indicado pelo campo operand e
Opcode:	00110
Status:	Z e N são afetados por esta operação
PC:	PC ← PC + 1
Operação:	ACC ← ACC - Memory[operand]
BEQ	Branch Equal
Sintaxe:	BEQ operand
Descrição:	Atualiza o valor do PC com o valor do cam operação anterior na ULA foi zero.
Opcode:	01000
Status:	Nenhum bit é afetado
PC:	Se (STATUS.Z=1) então PC ← endereço Se não PC ← PC + 1
Operação:	Nenhuma Operação Realizada

Ambiente Bipide 3.0

The screenshot displays the Bipide 3.0 software interface, which is used for simulating a custom processor. The interface is divided into several sections:

- Top Bar:** Contains tabs for 'Programação', 'Simulação', 'Instruções', and 'Ajuda'. Below these are buttons for 'Simular', 'Pausar', 'Repetir', 'Próximo', 'Continuar', and 'Parar'. There are also checkboxes for 'Portugol', 'Assembly', and 'Simulador', and a 'Configurações' button.
- Portugol Code Editor:** Displays the following code:

```
vetor[2]<-4
vetor[3]<-2
vetor[4]<-1
para i <- 0 ate 4 passo 1
  para j <- i+1 ate 4 passo 1
    se (vetor[i] > vetor[j]) entao
      aux <- vetor[i]
      vetor[i] <- vetor[j]
      vetor[j] <- aux
fimse
```
- Assembly Code Editor:** Displays the following assembly code:

```
LDI 0
STO i
LDI 4
STO 1000
LDI 1
STO 1001
LD i
PARA1:
SUB 1000
BGT FIMPARA1
```
- Hardware Diagram:** A schematic diagram of the processor, labeled 'BIP IV'. It shows the internal components, including the 'Control' and 'Data Path' sections. The diagram includes a 'Decodificador' (Decoder) and various registers and ALUs.
- Registers and Status:** A table on the right side of the interface shows the current state of the processor's registers and status flags:

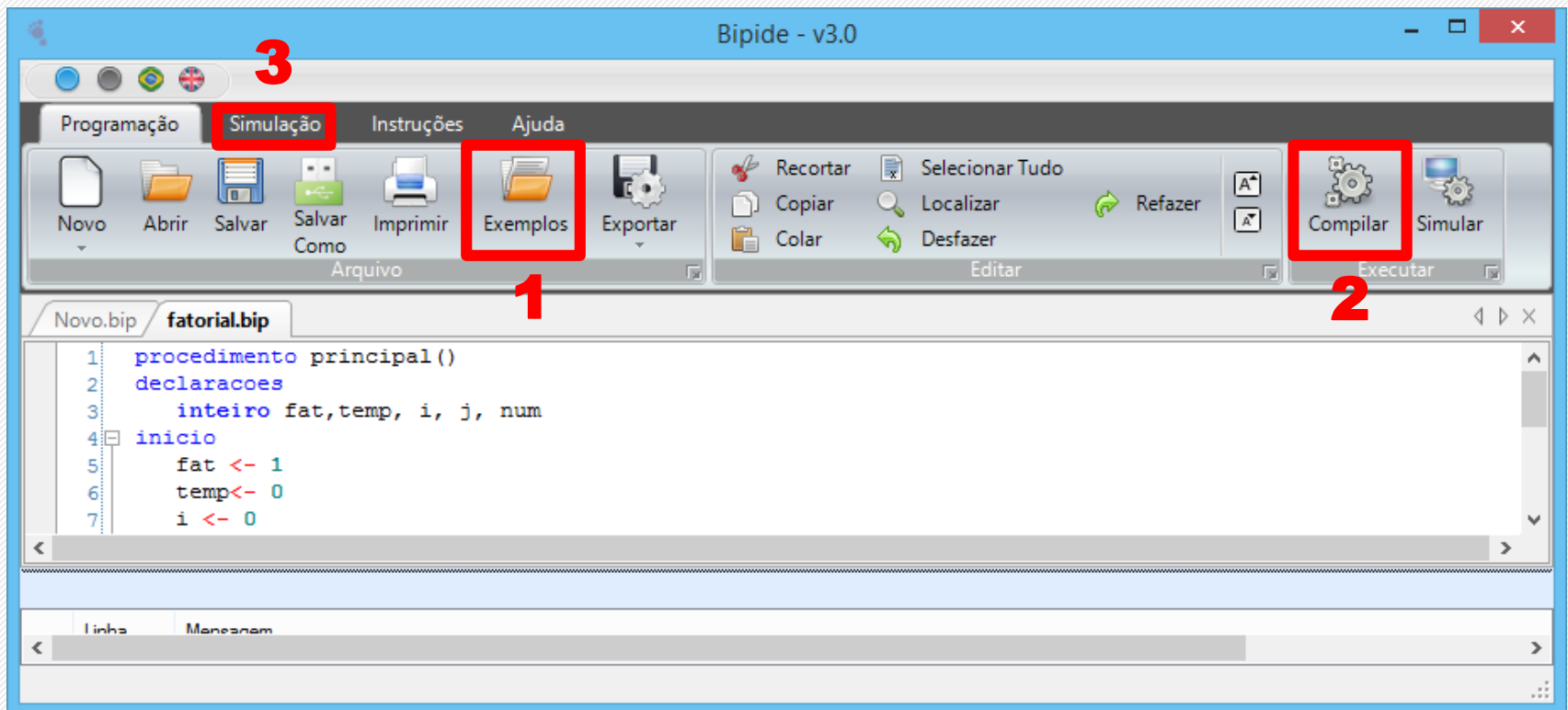
PC	Acumulador	StatusZ	StatusN	SP	INDR
47	1	0	0	0	4
- Instruction Table:** A table at the bottom left shows the current instruction being executed:

Rótulo	Endereço	Instrução
45	45	STO 1001
46	46	LDI 1
PARA1	47	SUB 1000
- Serial Ports:** At the bottom, there are two serial port status indicators: 'Sout_port' and 'Sin_port', each with a series of green LEDs and a numerical value (0).

www.bipide.com.br

Atividade

Compilar os exemplos em Portugol disponíveis no Bipide e analisar, através do simulador, a execução dos códigos *assembly* gerados.



Referências

1. VIEIRA, Paulo Viniccus; RAABE, André Luís Alice; ZEFERINO, Cesar Albenes . Bipide: Ambiente de Desenvolvimento Integrado para Utilização dos Processadores BIP no Ensino de Programação. In: XX Simpósio Brasileiro de Informática na Educação, 2009, Florianópolis. Anais do XX Simpósio Brasileiro de Informática na Educação, 2009. v. 1.
2. www.bipide.com.br