## Processadores BIP

A família de processadores BIP foi desenvolvida por pesquisadores do Laboratório de Sistemas Embarcados e Distribuídos (LSED) da Universidade do Vale do Itajaí – UNIVALI com o objetivo de desenvolver uma série de processadores de arquitetura simplificada que auxiliassem no aprendizado de Arquitetura e Organização de Computadores por alunos de fases iniciais de cursos de graduação em Computação. Outro objetivo do BIP é permitir relacionar os conteúdos de programação com suas representações em hardware, provendo a base necessária para compreensão das abstrações adotadas nas disciplinas da área de Algoritmos e Programação (MORANDI *et al.*, 2006).

Nesse contexto, ao buscar estabelecer uma relação entre a programação de alto nível e a sua representação em hardware, podem ser mencionadas as relações entre: (a) declaração de variável e alocação de memória; (b) constantes e operandos imediatos; (c) atribuição de variáveis e operações de acesso a memória; e (d) operações aritméticas e a utilização de unidades de soma/subtração (MORANDI; RAABE; ZEFERINO, 2006).

Conforme Morandi *et al* (2006), durante o desenvolvimento do BIP, foram definidas três diretrizes de projeto:

- A arquitetura do processador deveria ser simples o suficiente para facilitar o entendimento do seu funcionamento e a sua construção a partir dos conceitos estudados na disciplina de circuitos digitais;
- A arquitetura deveria ser extensível; e
- As ferramentas de apoio desenvolvidas no contexto do projeto deveriam priorizar o uso de estruturas de linguagem compreendidas no primeiro ano do curso.

De acordo com estas diretrizes, foram definidos inicialmente dois modelos de processador (MORANDI; RAABE; ZEFERINO, 2006):

- BIP I: com foco no suporte ao entendimento de conceitos como níveis de linguagem, constantes, variáveis, representação de dados e de instruções,

conjuntos de instruções, programação em linguagem de montagem e geração de código em linguagem de máquina; e

- BIP II: extensão do BIP I, com foco na inclusão de suporte aos conceitos de estruturas de controle para desvios condicionais e incondicionais e laços de repetição.

Uma terceira versão de processador, denominada μBIP (lê-se microBIP), foi desenvolvida. Este processador estende a arquitetura das versões anteriores do BIP e acrescenta suporte a instruções de lógica booleana, deslocamento lógico, manipulação de vetores, instruções de controle, suporte a procedimentos e funcionalidades típicas de microcontroladores como, por exemplo, pinos de E/S e temporizador programável (PEREIRA, 2008).

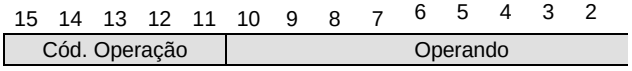
Como o foco deste trabalho é auxiliar o entendimento de conceitos básicos aprendidos nas fases iniciais do ensino de programação, será utilizado somente os modelos de arquitetura simplificada dos processadores BIP I e BIP II.

### 1.1.1 BIP I

O BIP I utiliza uma arquitetura baseada na arquitetura RISC do microcontrolador PIC (Programmable Intelligent Computer). Ele é uma máquina orientada a acumulador e não possui banco de registradores. Todas as operações de transferência e aritmética envolvem o acumulador e, em algumas operações aritméticas, é necessário buscar operandos posicionados na memória (MORANDI *et al.*, 2006*)*.

A arquitetura do BIP I possui um conjunto de instruções com poucos modos de endereçamento e todas as instruções são baseadas neste formato. Como pode ser visto na Tabela 1, o formato de instrução esta dividido em dois campos: 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 (MORANDI; RAABE; ZEFERINO, 2006).

Tabela 1. Arquitetura do BIP I

| | |
|---|---|
| Tamanho da palavra de dados | 16 bits |
| Tipo de dados | Inteiro de 16 bits com sinal: –32768 a +32767 |
| Tamanho da palavra de instrução | 16 bits |
| Formato de instrução |  |
| Modos de endereçamento | Direto: o operando é um endereço da memória<br>Imediato: o operando é uma constante |
| Registradores | ACC: acumulador<br>IR: registrador de instrução<br>PC: contador de programa |
| Classes de instrução | Transferência (acesso à memória): STO, LD, LD<br>Aritmética: ADD, ADDI, SUB e SUBI<br>Controle: HLT |

Fonte: Adaptado de Morandi *et al* (2006).

A arquitetura do BIP I inclui três registradores: PC, IR e ACC. O contador de programa (PC) aponta para o endereço da próxima instrução que será executada. O registrador de instrução (IR – Instruction Register) armazena a instrução que está em execução. O acumulador (ACC – Accumulator) é utilizado para armazenamento de dados durante uma operação aritmética ou de atribuição.

O conjunto de instruções do BIP I, conforme apresentado na Tabela 2, inclui uma instrução de controle, três instruções de transferência (duas de acesso à memória) e quatro instruções de aritmética. Em cada instrução, exceto na instrução Halt (`HLT`), o PC é incrementado em uma unidade no final do ciclo de execução da instrução. Na tabela, o termo `Memory[operand]` representa uma variável armazenada na posição `operand` da memória de dados.

Tabela 2. Conjunto de instruções do BIP I

| **Código da operação** | **Instrução** | **Operação** | **Classe** |
|---|---|---|---|
| 00000 | `HLT` | Paralisa a execução | Controle |
| 00001 | `STO operand` | `Memory[operand]← ACC` | Transferência |
| 00010 | `LD operand` | `ACC ← Memory[operand]` | Transferência |
| 00011 | `LDI operand` | `ACC ← operand` | Transferência |
| 00100 | `ADD operand` | `ACC ← ACC + Memory[operand]` | Aritmética |
| 00101 | `ADDI operand` | `ACC ← ACC + operand` | Aritmética |
| 00110 | `SUB operand` | `ACC ← ACC –` | Aritmética |

| | | | |
|---|---|---|---|
| | | `Memory[operand]` | |
| 00111 | `SUBI operand` | `ACC ← ACC - operand` | Aritmética |
| 01000 - 11111 | Reservado para futuras gerações | | |

Fonte: Adaptado de Morandi *et al* (2006).

Como pode ser observado na Tabela 2, O BIP I realiza operações básicas de soma e subtração com variáveis e constantes. Apesar de limitada, sua arquitetura permite ilustrar a implementação em hardware de diversas abstrações estudadas nas disciplinas da área de Algoritmos e Programação (MORANDI *et al.*, 2006).

### 1.1.2 BIP II

O BIP II trata-se de uma extensão do BIP I e possui as mesmas características arquiteturais. Uma das mudanças realizadas na extensão do BIP I para o BIP II foi a inclusão da classe de instruções de desvio visando a implementação de desvios condicionais, incondicionais e laços de repetição (MORANDI; RAABE; ZEFERINO, 2006).

Com relação aos registradores, o BIP II possui de três a quatro: PC, ACC, STATUS e IR, conforme ilustrado na Tabela 3.

Tabela 3. Arquitetura do BIP II

| | |
|---|---|
| Tamanho da palavra de dados | 16 bits |
| Tipo de dados | Inteiro de 16 bits com sinal: –32768 a +32767 |
| Tamanho da palavra de instrução | 16 bits |
| Formato de instrução | 15 14 13 12 11 10 9 8 7 6 5 4 3 2<br>Cód. Operação \| Operando |
| Modos de endereçamento | Direto: o operando é um endereço da memória<br>Imediato: o operando é uma constante |
| Registradores | ACC: acumulador<br>IR: registrador de instrução<br>PC: contador de programa<br>STATUS: registrador de estado com dois *flags* (Z |
| Classes de instrução | Transferência (acesso à memória): STO, LD, LD<br>Aritmética: ADD, ADDI, SUB e SUBI<br>Controle: HLT<br>Desvio: BEQ, BNE, BGT, BGE, BLT, BLE e JM |

Fonte: Adaptado de Morandi, Raabe e Zeferino (2006).

Para suportar as instruções de comparação e desvio condicional foi acrescentado um registrador de estado (STATUS). Toda instrução de comparação e desvio condicional deve ser precedida por uma instrução de subtração (SUB ou SUBI). O registrador STATUS possui dois *flags* que são ativados dependendo do resultado da operação anterior: (i) Z, que indica se o resultado da última operação da ULA foi igual a zero ou não; e (ii) N, que indica se o último resultado da ULA foi um número negativo ou não (MORANDI; RAABE; ZEFERINO, 2006).

A Tabela 4 apresenta o conjunto de instruções do processador BIP II. Na tabela, é possível observar as alterações nos registradores ACC (coluna 3) e PC (coluna 4) em cada instrução. Na tabela, o termo `Memory[operand]` representa uma variável alocada em memória na posição `operand`.

Tabela 4. Conjunto de instruções do BIP II

| **Código da operação** | **Instrução** | **Operação e atualização do PC** | |
|---|---|---|---|
| 00000 | HLT | Paralisa a execução | PC ← PC |
| 00001 | STO operand | Memory[operand]← ACC | PC ← PC + 1 |
| 00010 | LD operand | ACC ← Memory[operand] | PC ← PC + 1 |
| 00011 | LDI operand | ACC ← operand | PC ← PC + 1 |
| 00100 | ADD operand | ACC ← ACC + Memory[operand] | PC ← PC + 1 |
| 00101 | ADDI operand | ACC ← ACC + operand | PC ← PC + 1 |
| 00110 | SUB operand | ACC ← ACC – Memory[operand] | PC ← PC + 1 |
| 00111 | SUBI operand | ACC ← ACC - operand | PC ← PC + 1 |
| 01000 | BEQ operand | | Se (STATUS.Z=1<br>PC ← endereç<br>Senão<br>PC ← PC + 1 |
| 01001 | BNE operand | | Se (STATUS.Z=0<br>PC ← endereç<br>Senão<br>PC ← PC + 1 |
| 01010 | BGT operand | | Se (STATUS.Z=0<br>(STATUS.N=0) e<br>PC ← endereç<br>Senão<br>PC ← PC + 1 |
| 01011 | BGE operand | | Se (STATUS.N=0<br>PC ← endereç<br>Senão<br>PC ← PC + 1 |
| 01100 | BLT operand | | Se (STATUS.N=1<br>PC ← endereç<br>Senão<br>PC ← PC + 1 |
| 01101 | BLE operand | | Se (STATUS.Z=1<br>(STATUS.N=1) e<br>PC ← endereç<br>Senão<br>PC ← PC + 1 |
| 01110 | JMP operand | | PC ← endereço |
| 01111-11111 | Reservado para as futuras gerações | | |

Fonte: Adaptado de Zeferino (2007).

### 1.1.3 ORGANIZAÇÃO DO BIP I E BIP II

Segundo Morandi, Raabe e Zeferino (2006), a organização do BIP I utiliza a estrutura Harvard, com memórias separadas para dados e instruções, como mostra a

Figura 1. Conforme a figura, o processador é dividido em dois blocos: (i) Controle, responsável por gerar sinais de controle para o caminho de dados e atualização do PC e; (ii) Caminho de Dados, que inclui os circuitos necessários para executar a instrução.

Essa organização é bastante simples e seus blocos constituintes podem ser descritos apenas pela sua funcionalidade, o que atende o objetivo de facilitar o entendimento por parte de alunos de fases iniciais da área de Arquitetura e Organização de Computadores (MORANDI; RAABE; ZEFERINO, 2006).

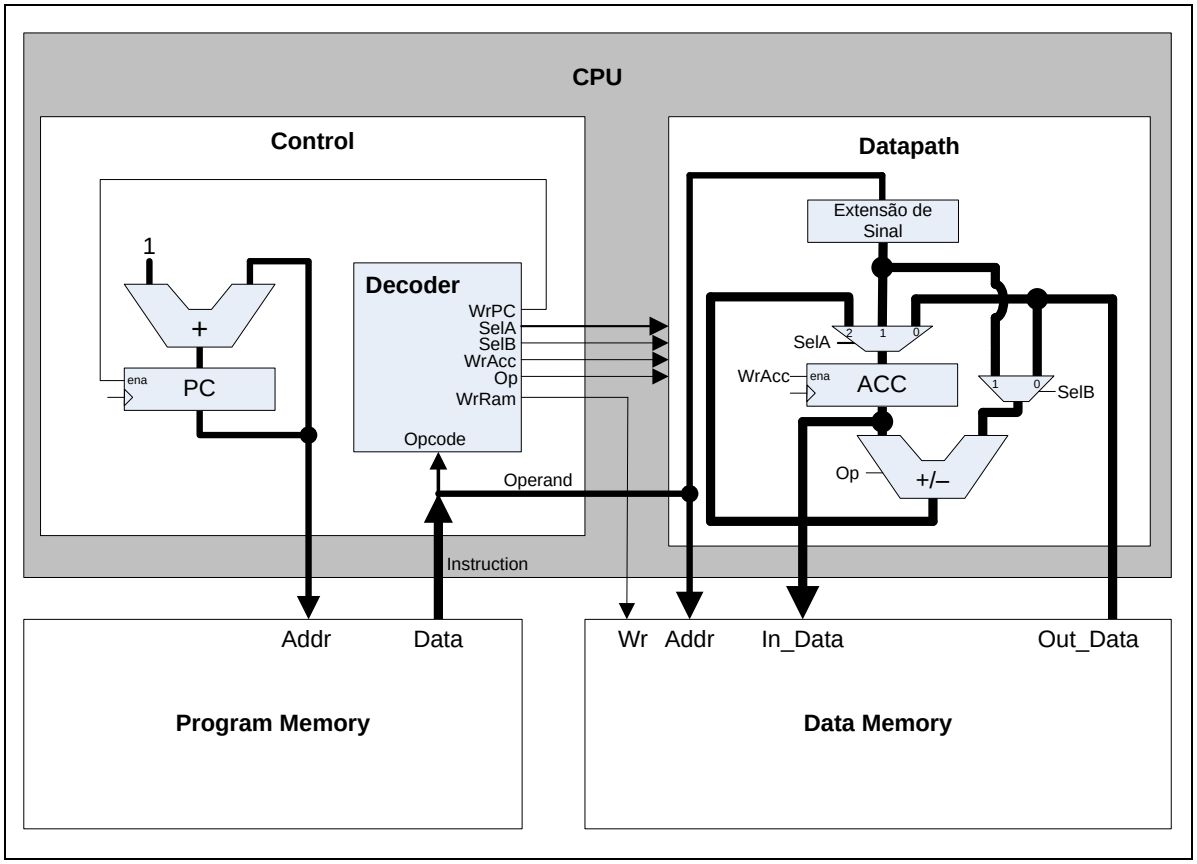



Figura 1. Organização do processador BIP I

Fonte: Zeferino (2007)

A organização do BIP II, conforme mostra a Figura 2, extende a organização do BIP I, incluindo o registrador STATUS e modificações no circuito de atualização do PC. Nas instruções de desvio, a fonte de atualização do PC é definida em função do tipo de desvio e dos valores dos *flags* N e Z do registrador STATUS (MORANDI; RAABE; ZEFERINO, 2006).

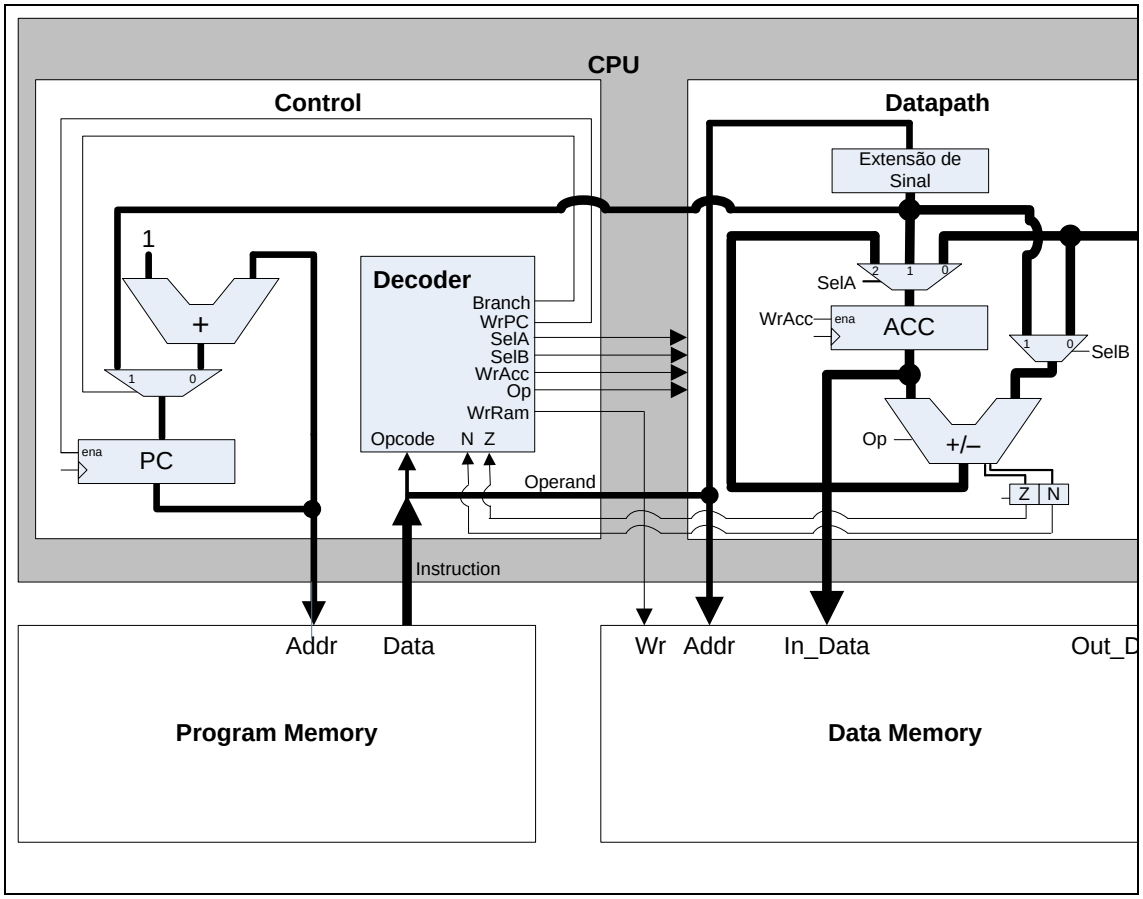



Figura 2. Organização do Processador BIP II

Fonte: Pereira (2008)

## Referências

MORANDI, Diana ; RAABE, André Luis Alice ; ZEFERINO, Cesar Albenes. Processadores para Ensino de Conceitos Básicos de Arquitetura de Computadores. In: WORKSHOP DE EDUCAÇÃO EM ARQUITETURA DE COMPUTADORES, 1., 2006, Ouro Preto. **Proceedings of the 18th International Symposium on Computer Architecture and High Performance Computing - Workshops**. Porto Alegre: SBC, 2006. p. 17-24.

MORANDI, Diana et al. Um processador básico para o ensino de conceitos de arquitetura e organização de computadores. **Hífen,** Uruguaiana, v. 30, p. 73-80, 2006.

PEREIRA, Maicon Carlos. **µBIP:** Microcontrolador Básico para o Ensino de Sistemas Embarcados. 2008. Trabalho de Conclusão de Curso – Faculdade de Ciência da Computação, Universidade do Vale do Itajaí, Itajaí, 2008.

ZEFERINO, Cesar A. **Especificação da Arquitetura e da Organização do Processador BIP**. Itajaí, 2007. (Relatório Técnico).