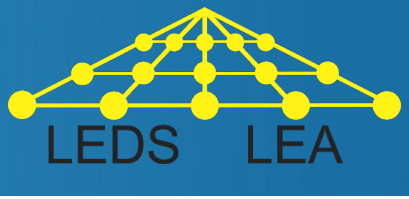
Arquitetura e Organização de Computadores

Conjunto de Instruções da Arquitetura – CompSim



Agenda

- Tipos de Instruções
- Operações Avançadas de Acesso à Memória

Tipos de Instruções

- Pseudo-Instruções do Montador (*Assembler*)
  - Segmento
    - .code, .data, .bss, .stack
  - Rótulo ou Nome
    - :
  - Delimitador de comentário
    - ;
  - Definição/Declaração de variáveis
    - DD, DB, RESD, RESB
- Conjunto de Instruções da Arquitetura (ISA)
  - Aritméticas
    - ADD, SUB
  - Lógicas
    - NAND, SHIFT
  - Transferência de dados
    - MOV, LDA, STA, LDI, STI, SOP
  - Transferência de controle
    - JMP, JN, JZ, CALL, RET, INT
  - Entrada/Saída
    - INT

Operações Avançadas de Acesso à Memória

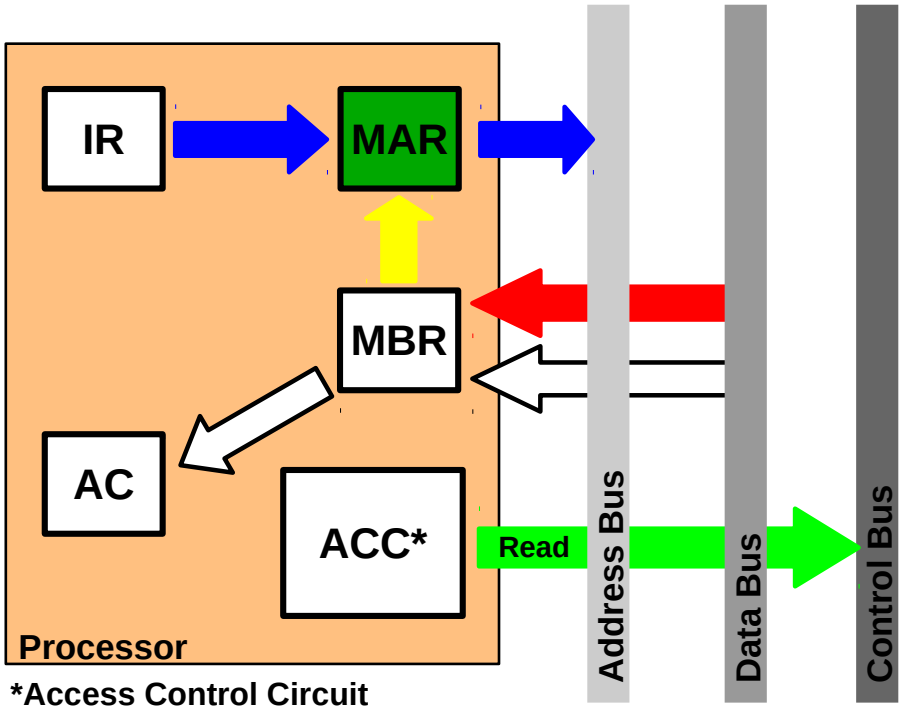
- Vimos que as instruções LDA e STA são utilizadas para leitura e armazenamento na memória prinicipal.
  - Elas implementam o **Endereçamento Direto**.
- Contudo, a manipulação de dados em estruturas de dados mais complexas, como em arrays, por exemplo, exige um novo modo de endereçamento à memória.
- Com **Endereçamento Indireto**, o valor lido da memória agora é utilizado como o novo endereço (e o definitivo) do dado.
  - Vantagens:
    - Pode-se implementar “Apontadores” (semelhantes aos da linguagem C)!
    - Pode-se expandir o limite de endereçamento de memória, pois o campo de endereço na instrução possui 12 bits e um endereço lido indiretamente pode ter 16 bits.
  - Desvantagem: mais de um acesso à memória.

Operações Avançadas de Acesso à Memória

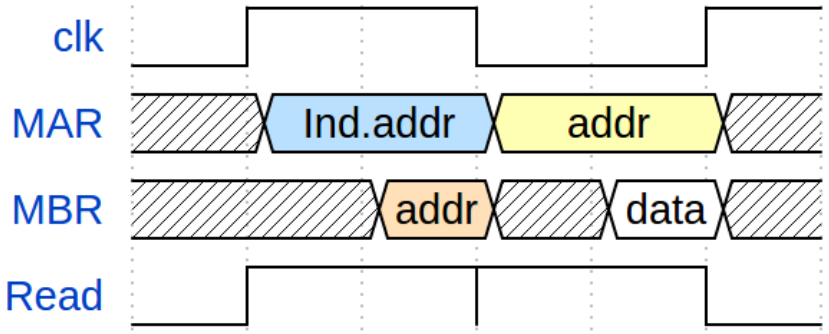
- Endereçamento indireto pode ser realizado basicamente por:
  - LDI *(LoaD from Indirect)* – utiliza o valor lido em um determinado endereço de memória para endereçar uma nova posição de memória para leitura de um dado.
  - STI *(Store To Indirect)* – utiliza o valor lido em um determinado endereço de memória para endereçar uma nova posição de memória para escrita de um dado.
- Sintaxe:
  - Leitura indireta: [<rotulo>] LDI <endereco-memoria>
  - Escrita indireta: [<rotulo>] STI <endereco-memoria>

Operações Avançadas de Acesso à Memória

- LDI (*LoaD from Indirect*)



Waveform



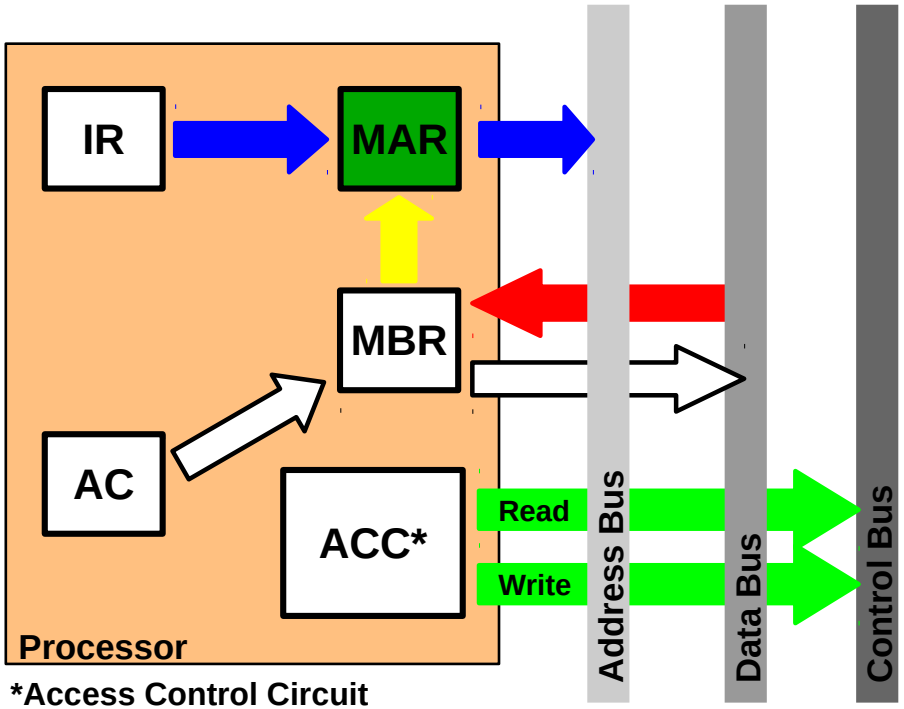
Microcode (RTN*)

```
MAR ← IR[11-0]
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
AC ← MBR
```

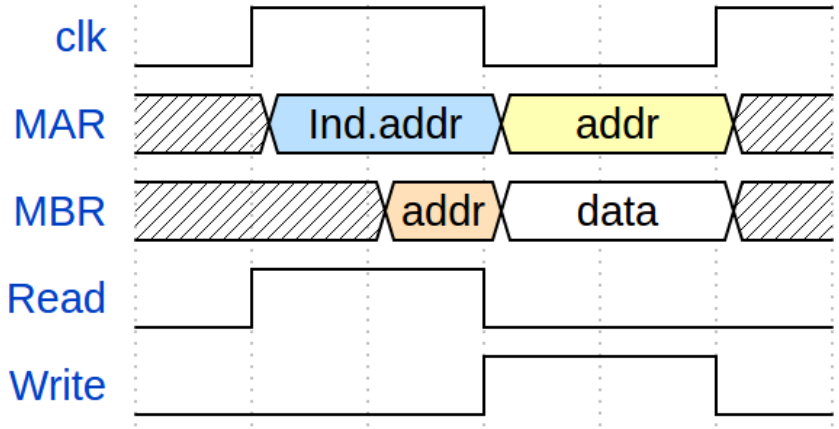
*Register Transfer Notation

Operações Avançadas de Acesso à Memória

- STI (*Store To Indirect*) - PUSH



Waveform



Microcode (RTN*)

```
MAR ← IR[11-0]
MBR ← M[MAR]
MAR ← MBR
MBR ← AC
M[MAR] ← MBR
```

*Register Transfer Notation

Operações Avançadas de Acesso à Memória

- Exemplo prático: Operações com Arrays.
- Procedimento:
  - Baixar e extrair o pacote:
    - 6.advanced_memory_access.zip
  - Menu “File” → “Open”
    - Ou Teclas “Ctrl+o”
  - Arquivo:
    - “array_operations.asm”

```
1       .code
2
3       select_pos:
4           LDI ptr_a ; a[i] = value;
5           ADD value
6           STI ptr_a
7
8           LDA value  ;value += 1
9           ADD one
10          STA value
11
12          LDA ptr_a ;i += 1
13          ADD one
14          STA ptr_a
15
16          ;if value == size(a)
17          LDA size_a
18          SUB value
19          ;then exit
20          JZ end
21          ;else
22          JMP select_pos
23
24      end:
25          INT exit
26
27      .data
28          ptr_a: DD a ; int *index_a = a
29          size_a: DD 10
30          value: DD 0
```

Atividade Prática

- Refaça o programa da atividade anterior, porém:
  - Utilizar um *loop* (iteração) para realizar a soma dos elementos em um array.