Mestrado Integrado em Eng. Electrónica Industrial e Computadores

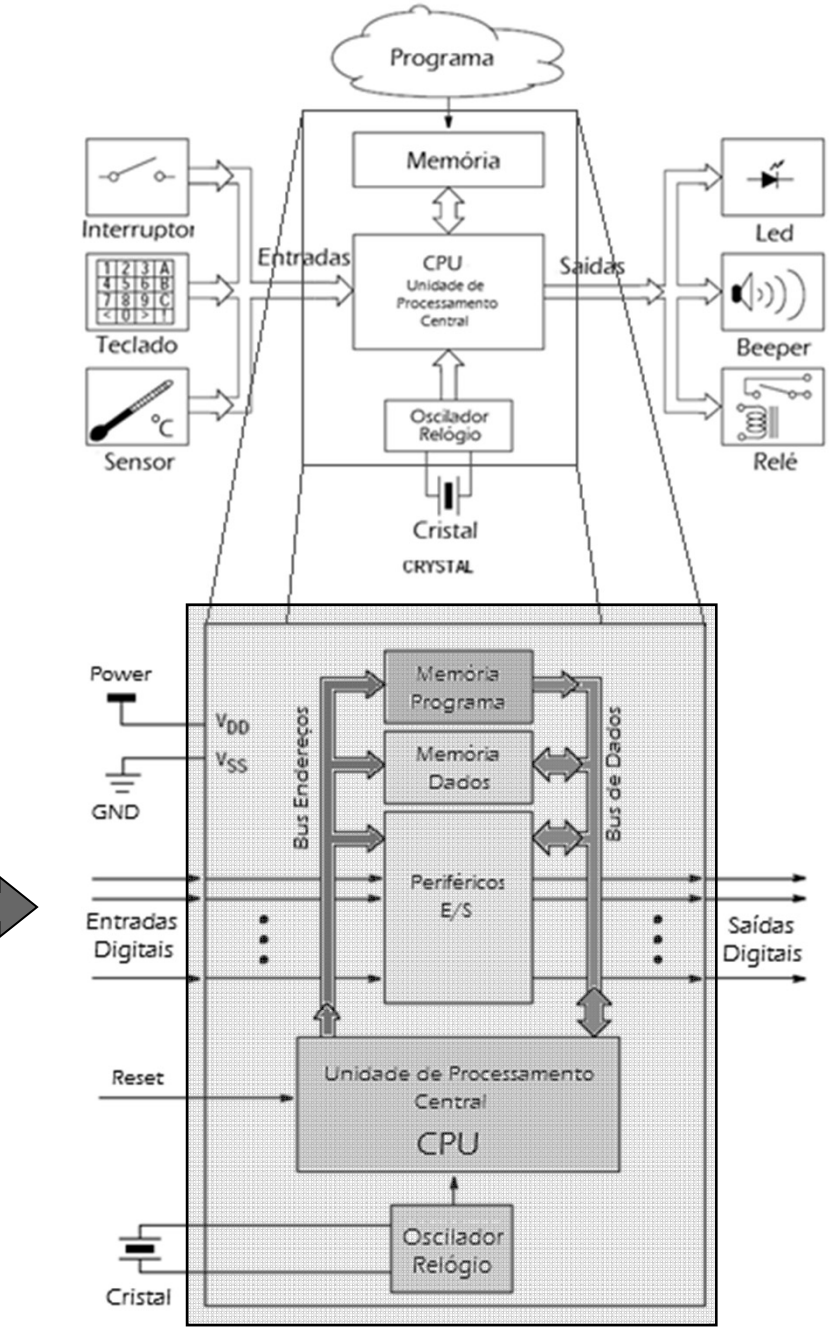
Sistema Computadorizado

Microprocessadores I

2º Ano – A03

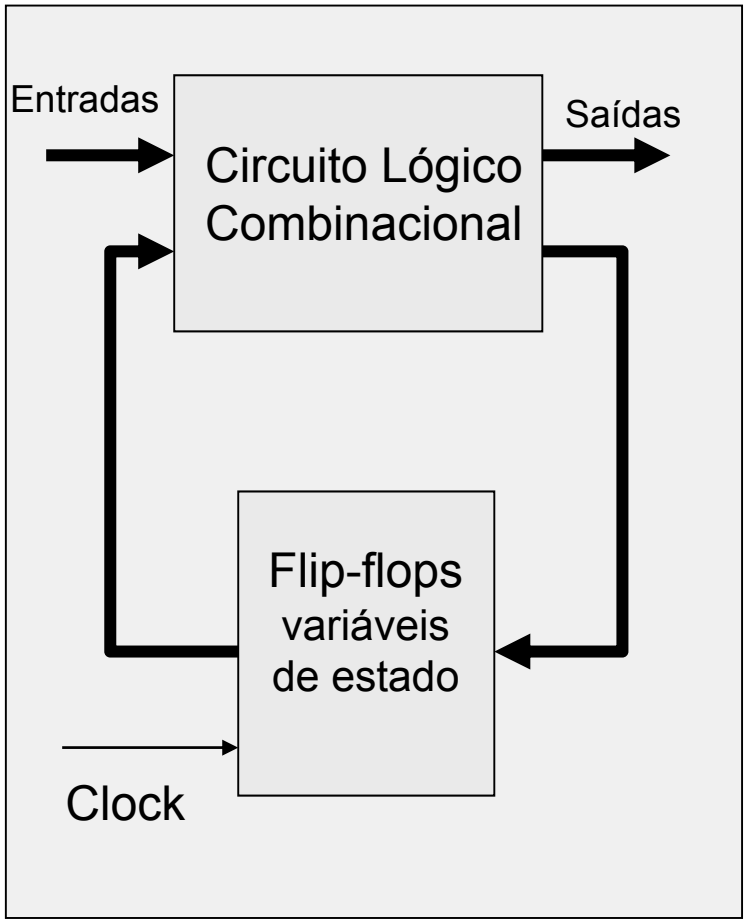
- O que é um microcontrolador ?

Microcontrolador

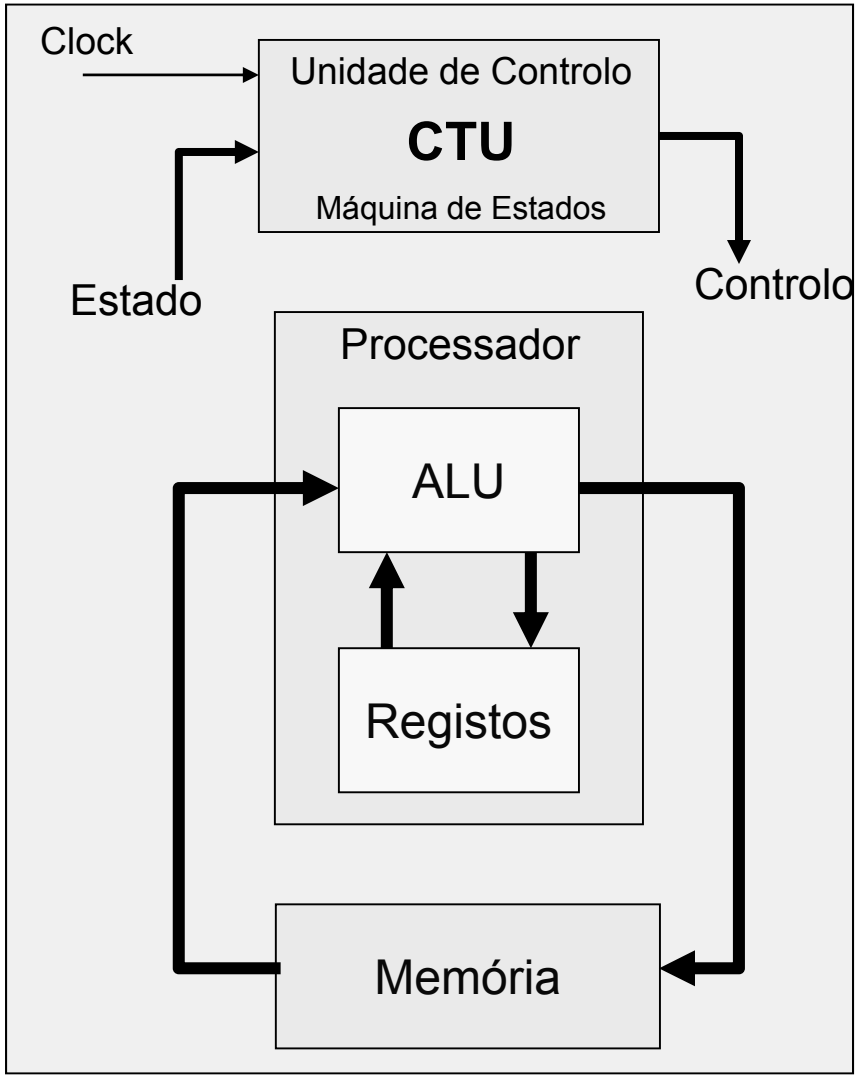


O que é um processador ?

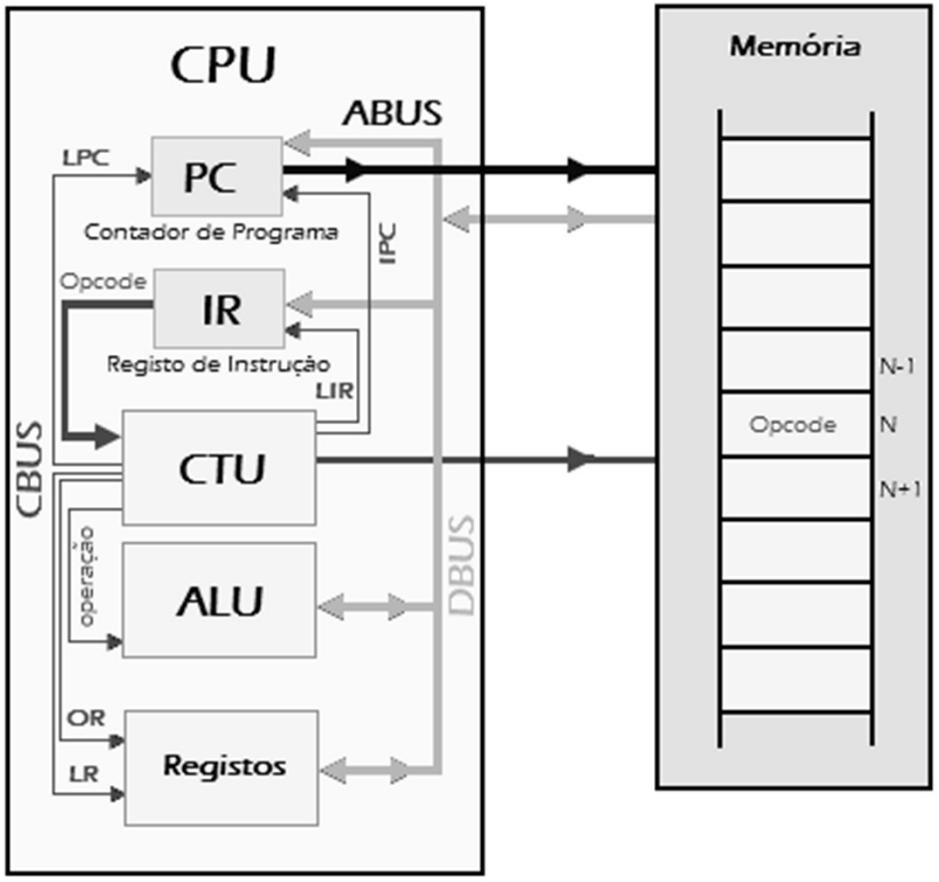
Máquinas de estados



Modelo computacional elementar



Fetch de um opcode

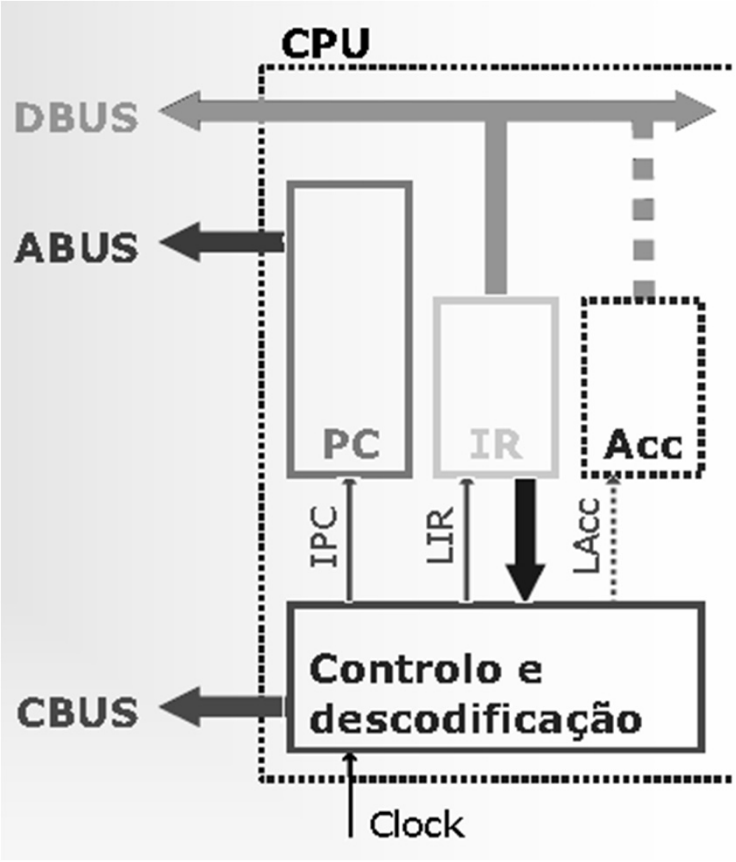


Passos

1. O conteúdo do PC é colocado no barramento de endereços;
2. O PC é incrementado para o *fetch* da próxima instrução;
3. A linha de controlo RD é activada;
4. A memória lê o conteúdo do endereço fornecido (código da instrução) e coloca-o no barramento de dados;
5. O *opcode* da instrução é transferido do barramento de dados para o registo interno IR;
6. O CTU descodifica o *opcode* e controla a execução das instruções a executar para realizar a operação.

Ler um *opcode*

- Leitura de uma instrução
  - O CPU envia à memória:
    - Endereço: ABUS;
    - Sinal de leitura: CBUS.
  - Recebe:
    - O *opcode* armazenado na posição ABUS em DBUS;
    - O *opcode* é armazenado no IR.
  - Hardware:
    - Registo de endereço – PC;
    - Registo de Instrução – IR;
    - Unidade de Controlo e de descodificação das instruções – CTU.
  - Sinais de Controlo:
    - IPC – Incrementar PC;
    - LIR – Load IR;
    - LAcc – Load Acumulador.



Executar uma instrução

- Colocar no Acumulador (A ou Acc) um valor constante:

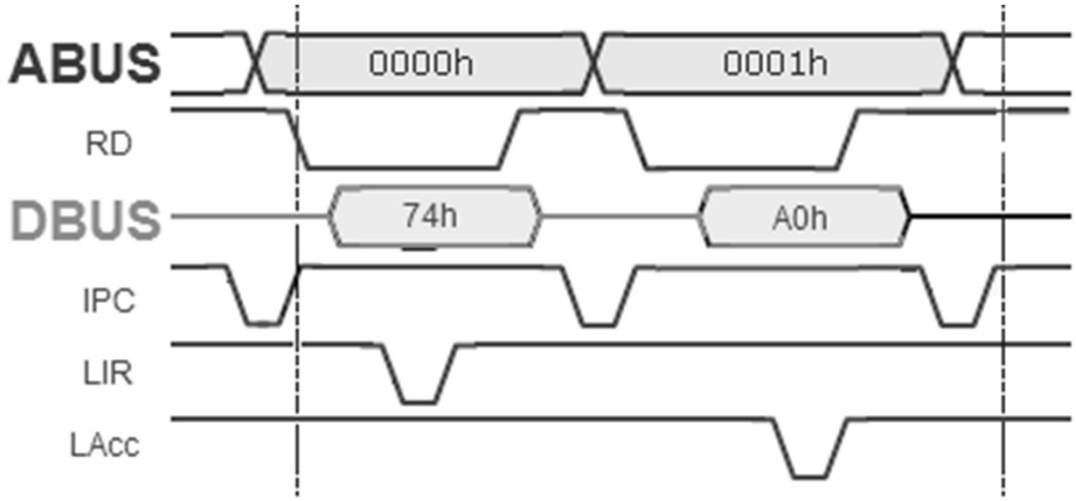
Instrução no 8051	***Opcode* da Instrução**
MOV A,#160	74

- A instrução será armazenada na memória de código. Há directivas *assembly* (instruções para o assemblador) que permitem definir em que endereço pretendemos armazenar a informação;

- Na instrução MOV A,#160 são armazenados dois bytes na memória de programa, o 74h que identifica a instrução “MOV A” e o A0h que é o valor constante que queremos carregar para o Acumulador. Estes dois bytes são armazenados sequencialmente na memória.

Executar um *opcode*

- **Comum a todas as instruções:**
  - Endereçar a instrução a ser executada na memória de código/programa (ABUS=PC);
  - Dar ordem de leitura da instrução à memória de código (activar sinal de RD de CBUS);
  - Guardar o código da instrução, devolvido pela memória de programa, no IR (activar sinal de controlo LIR);
  - Endereçar posição seguinte (activar sinal IPC).
- **Descodificar a instrução e executá-la (depende da instrução) - CTU**
  - Endereçar memória código (ABUS=PC);
  - Ler da memória de programa o valor a carregar em Acc (RD de CBUS);
  - Guardar valor devolvido pela memória em Acc (activar sinal LAcc);
  - Endereçar posição seguinte (activar sinal IPC).

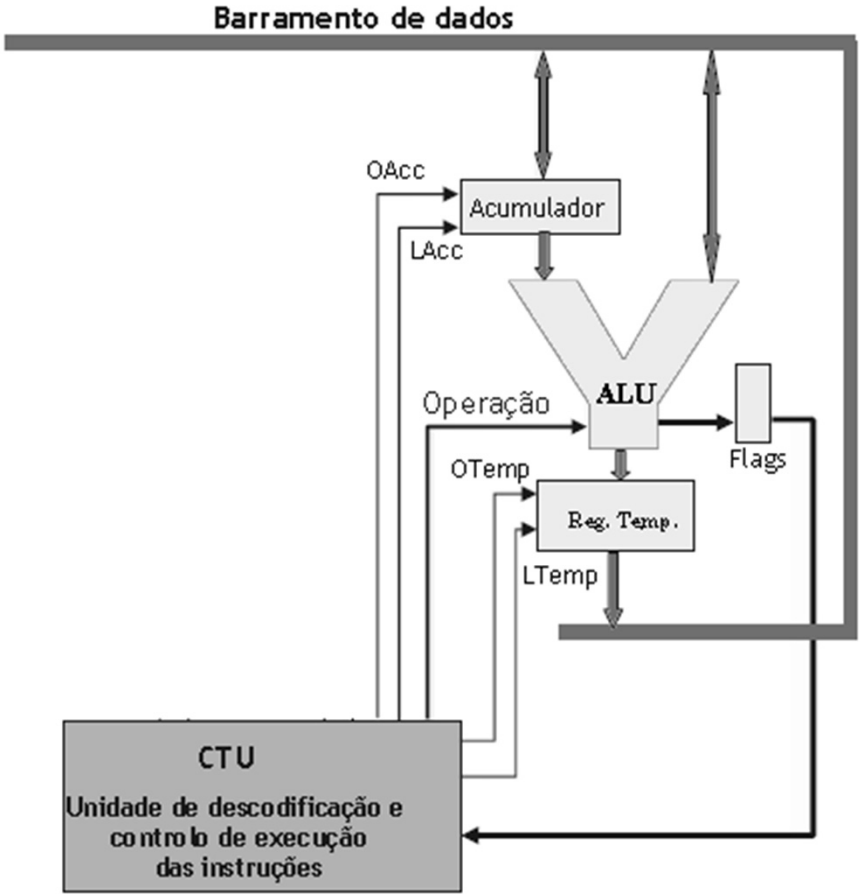


ALU

O registo de trabalho da ALU (Acumulador) é necessário para permitir que a ALU realize operações sobre dois operandos em simultâneo: dados do barramento e registo A;

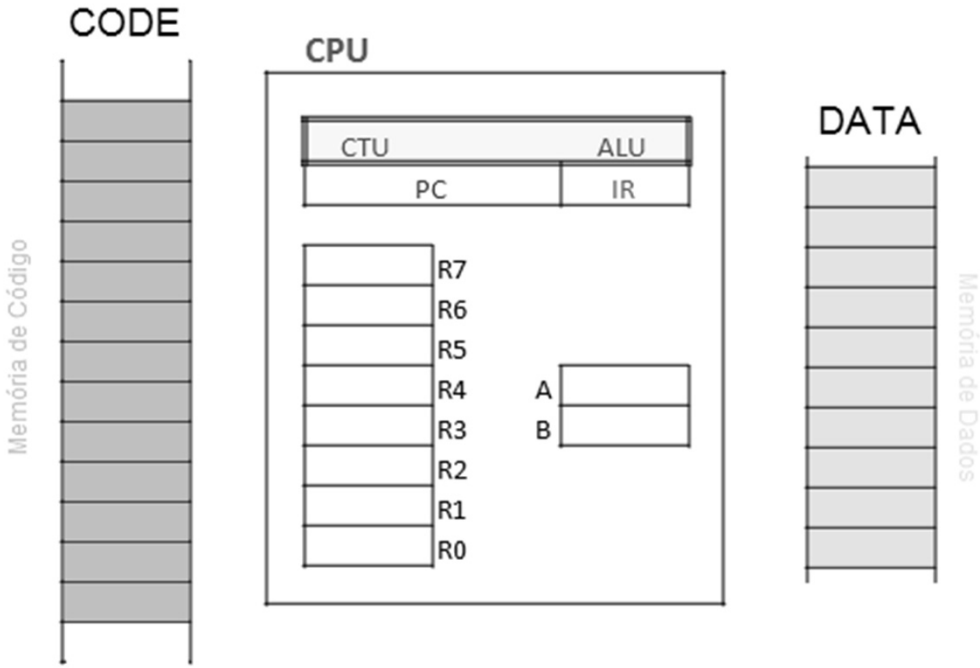
Pelo mesmo motivo é necessário um registo para armazenar temporariamente o resultado da ALU.

Flags: Carry, Zero e Paridade

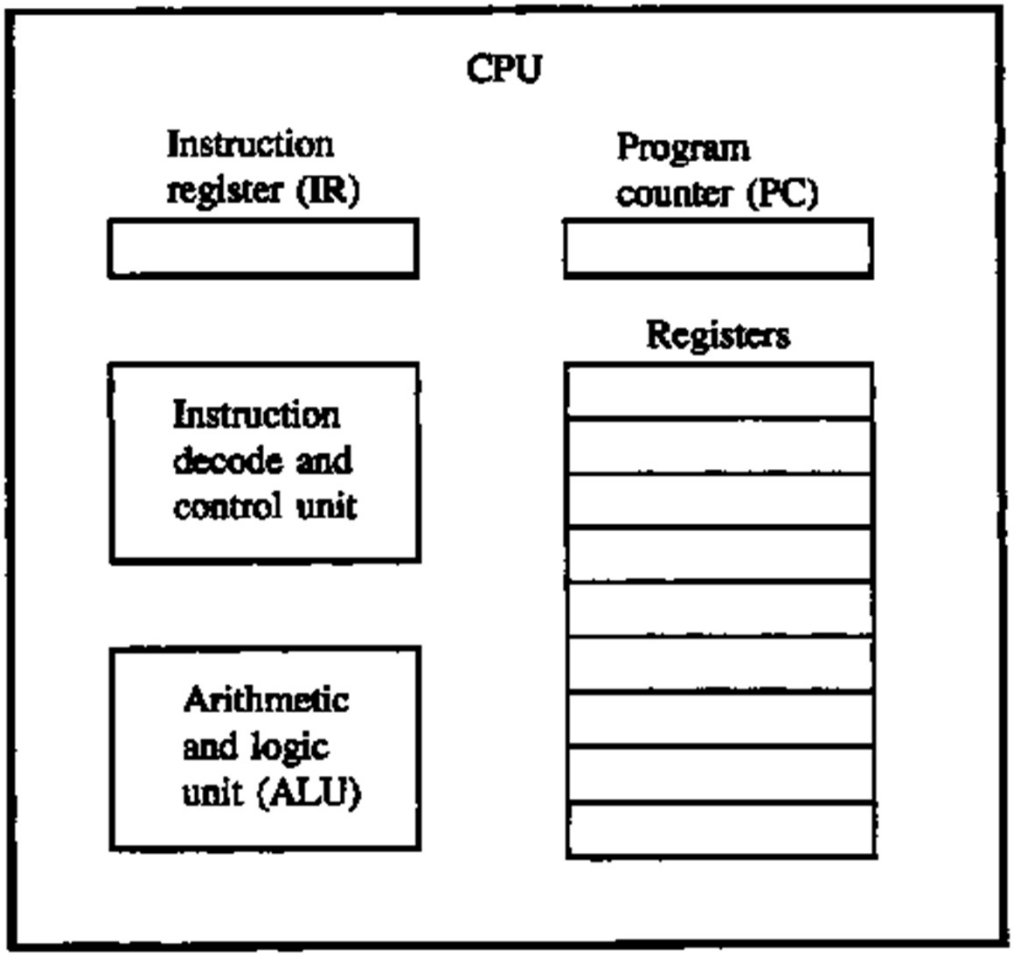


CPU ...

- CODE: memória onde são guardadas as instruções/programa. Não volátil;
- Qual o tamanho máximo da CODE?
- Memória de dados para armazenar as variáveis, o seu tamanho é fixo e igual a 256 bytes.
- Registo Acumulador – A – registo de trabalho da ALU, único endereçável ao bit;
- Registo B – registo extra da ALU;
- 8 registos de propósito geral R0 a R7;
- Flags: CY – carry e Z – zero
- Endereçar memória de dados só se podem utilizar os registos A, R0 e R1.



CPU

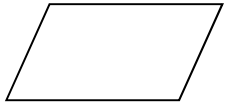


Exemplo de uma unidade de processamento central

Conceitos de algoritmia

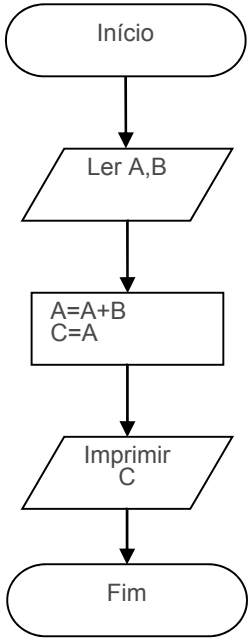
- Representações
  - Fluxogramas
    - Simbologia
      - Início/fim de um programa

      - Entrada/saída de dados

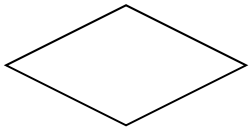


      - Processamento

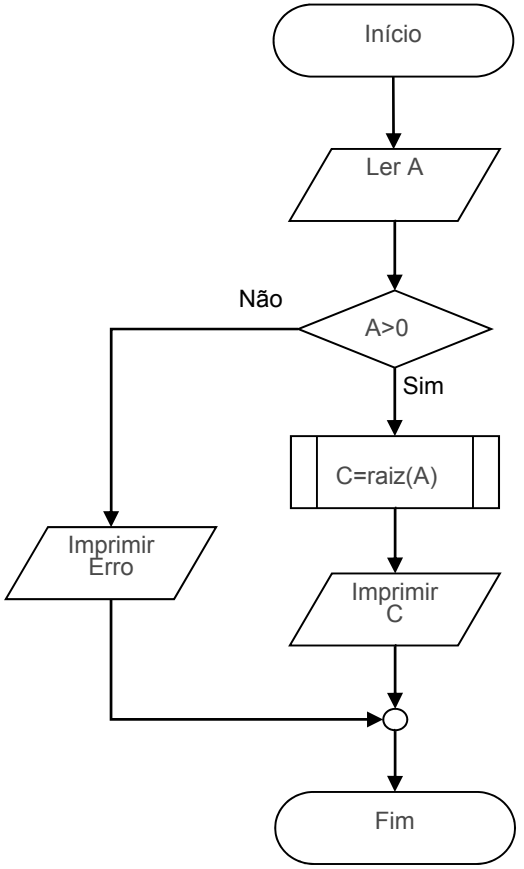
Exemplo



Conceitos de algoritmia

- Representações
  - Fluxogramas
    - Simbologia
      - Linha de fluxo
      - Decisão 
      - Subrotina 
      - Conector de fluxos 

Exemplo



Conceitos de algoritmia

- Representações
  - Pseudo-código
    - Notação
      - Início/fim de um programa

        `Início` `Fim`

      - Entrada/saída de dados

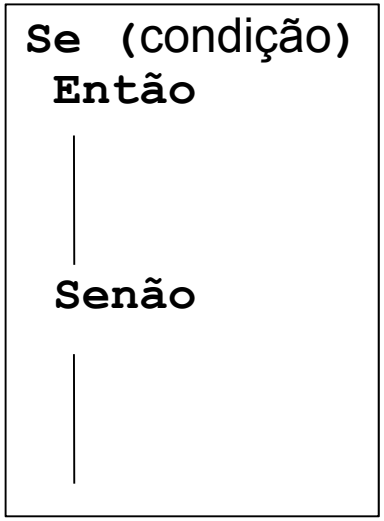
        `Ler()` `Escrever()`

      - Processamento (para detalhar depois)

        `[ ]`

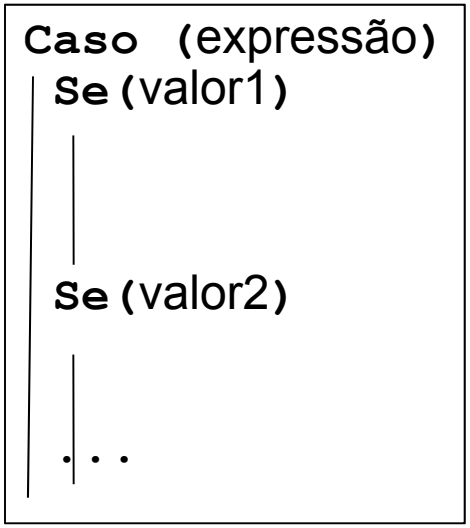
Conceitos de algoritmia

- Representações
  - Pseudo-código
    - Notação
      - Decisão



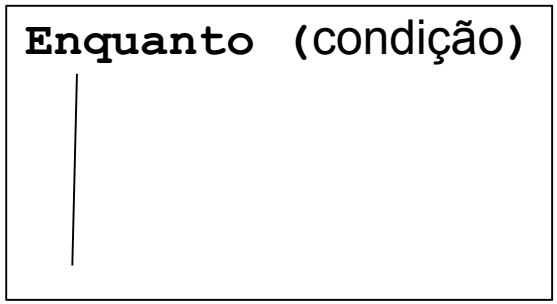
Conceitos de algoritmia

- Representações
  - Pseudo-código
    - Notação
      - Decisão estruturada



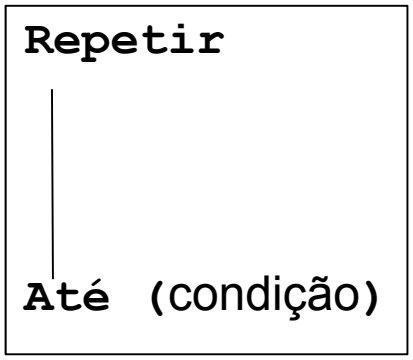
Conceitos de algoritmia

- Representações
  - Pseudo-código
    - Notação
      - Ciclos
        - `enquanto`



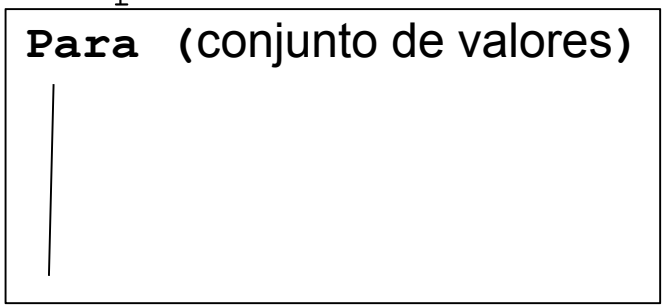
Conceitos de algoritmia

- Representações
  - Pseudo-código
    - Notação
      - Ciclos
        - `repetir até`



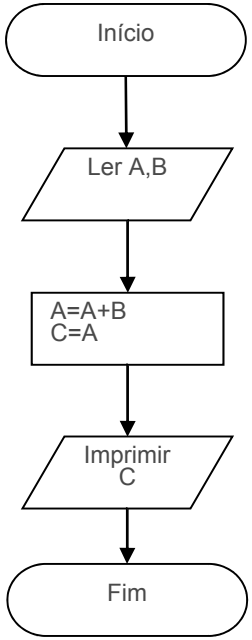
Conceitos de algoritmia

- Representações
  - Pseudo-código
    - Notação
      - Ciclos
        - `para`



Conceitos de algoritmia

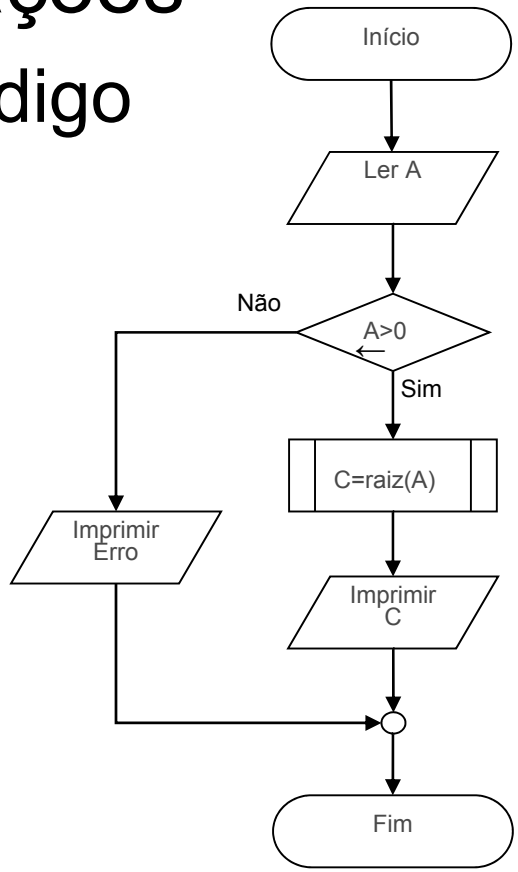
- Representações
  - Pseudo-código
    - Exemplo




```
Início
  Ler(A,B)
  A←A+B
  C←A
  Escrever(C)
Fim
```

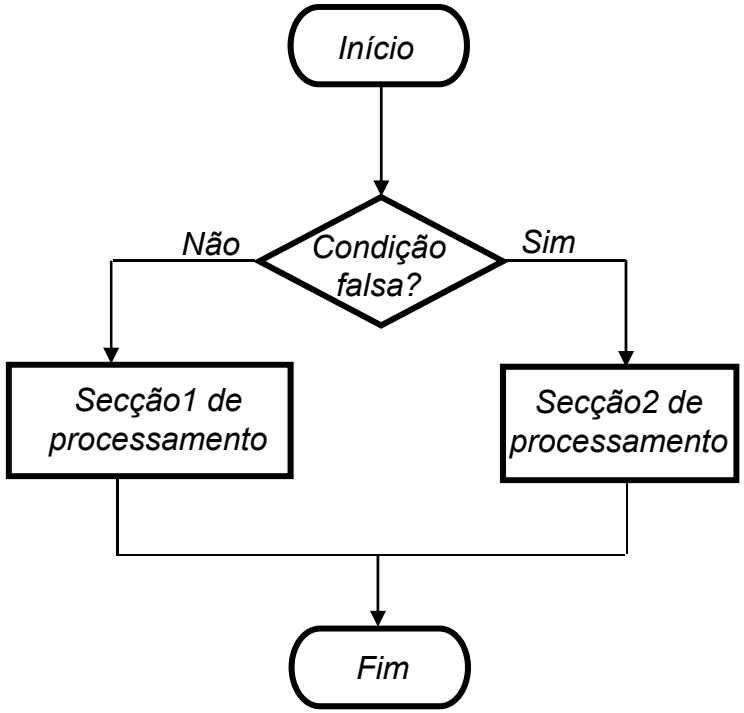
Conceitos de algoritmia

- Representações
  - Pseudo-código
    - Exemplo




```
Início
 Ler(A)
 Se (A>0)
  Então
   C=raiz(A)
   Escrever (A)
  Senão
   Escrever('ERRO')
Fim
```

Estruturas do IF/THEN/ELSE



<u>Exercício:</u>Converta um dígito armazenado no endereço apontado por R0 para um caracter ASCII que representa o seu valor hexadecimal. O valor apontado por R0 contém apenas um dígito hexadecimal (o MSnibble é 0). Guarde o resultado no registo B.

<u>Exemplo:</u>

Entrada: (R0) = 0Ch

Saída: B = 43h = 'C'

Entrada: (R0) = 06h

Saída: B = 36h = '6'