

INSTITUTO MAUÁ DE TECNOLOGIA



Engenharia de Computação

# Arquitetura e Organização de Computadores

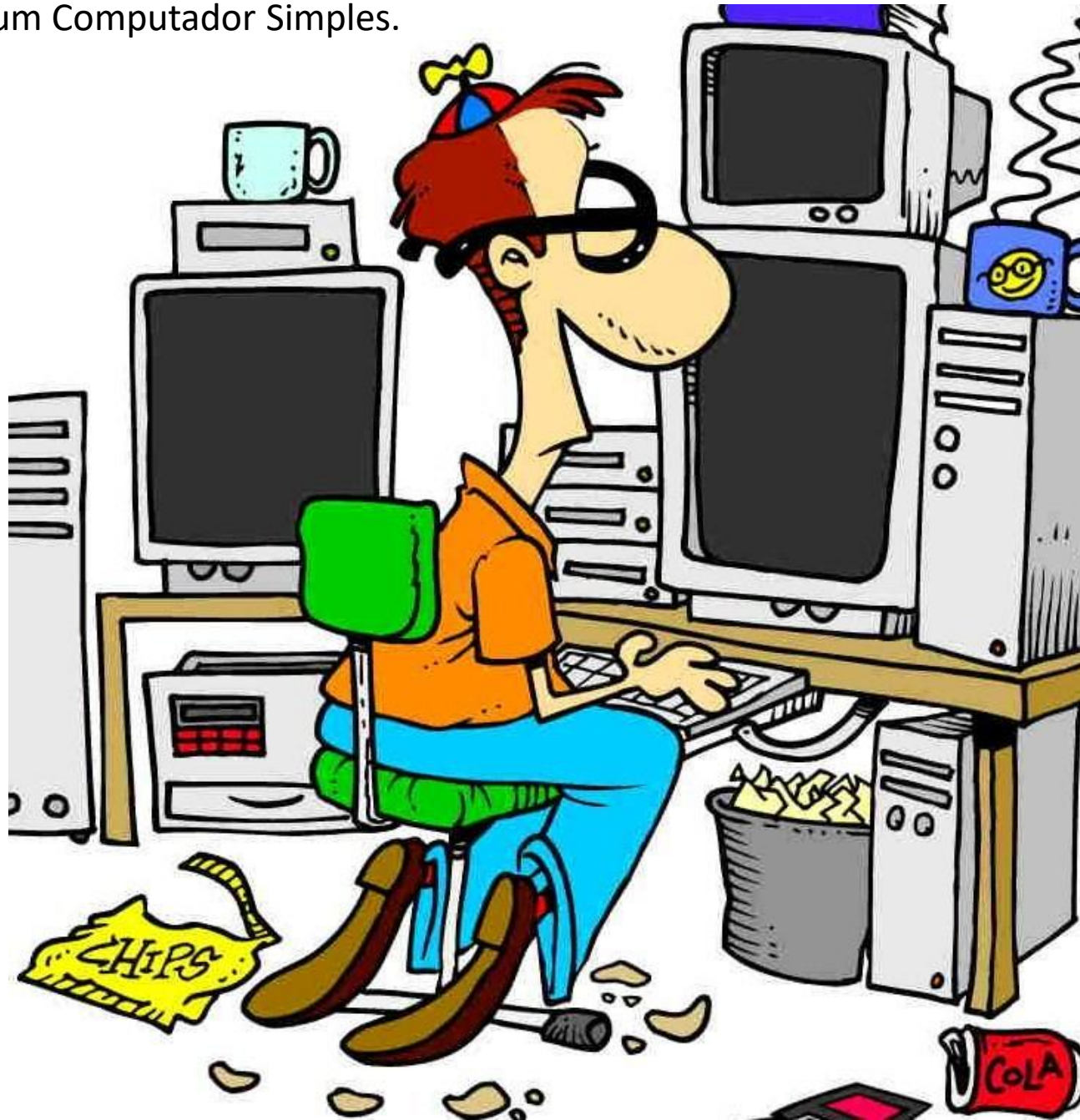
## ECM245

## 2017

Professor Dr. João Carlos Lopes Fernandes

E-mail: [jlopesf@maua.br](mailto:jlopesf@maua.br)

MARIE: Uma Introdução a um Computador Simples.



# ARQUITETURA MARIE

- Marie (Machine Architecture that is Really Intuitive and Easy), é uma arquitetura simples consistindo de memória (para armazenar programas e dados) e de uma CPU (contendo ULA e diversos registradores)

# ARQUITETURA MARIE

## ■ Características:

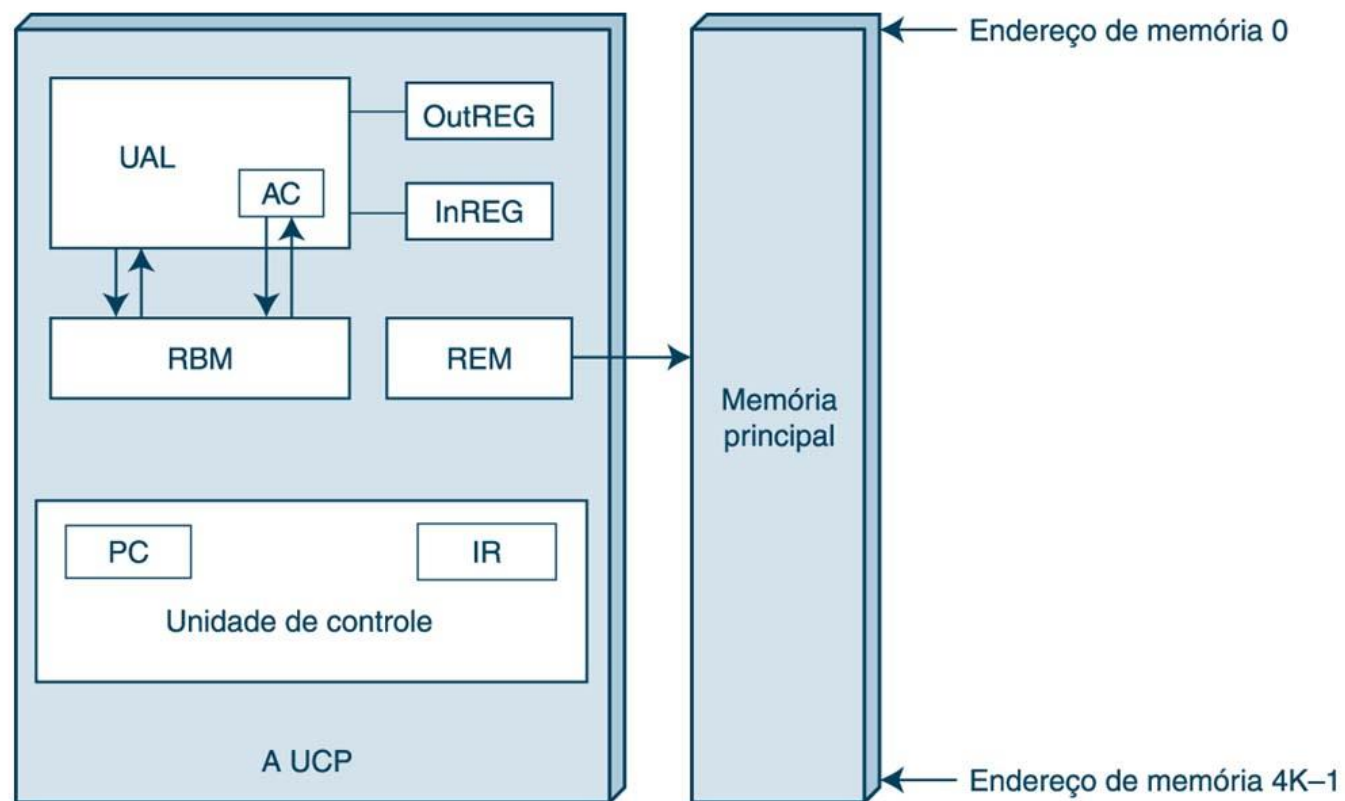
- Binária, complemento de dois;
- Endereçável por palavra (palavra de tamanho fixo);
- 4K de palavras de memória principal (12 bits por endereço);
- Dados de 16 bits (palavras com 16 bits);
- Instruções de 16 bits, 4 para código de operação e 12 para endereço;
- Um acumulador de 16 bits (AC);
- Um registrador de instrução de 16 bits (IR);
- Um registrador de buffer de memória de 16 bits (RBM);
- Um contador de programa de 12 bits (PC);

# ARQUITETURA MARIE

- Características:
  - Um registrador de endereço de memória de 12 bits (REM);
  - Um registrador de entrada de 8 bits;
  - Um registrador de saída de 8 bits.

# ARQUITETURA MARIE

- Arquitetura Marie



# ARQUITETURA MARIE

- Registradores e Barramentos
  - Unidade Lógica Aritmética (ALU): A **ULA** executa as principais operações lógicas e aritméticas do computador. Ela soma, subtrai, divide, determina se um número é positivo ou negativo ou se é zero. Além de executar funções aritméticas, uma ULA deve ser capaz de determinar se uma quantidade é menor ou maior que outra e quando quantidades são iguais. A ULA pode executar funções lógicas com letras e com números.
  - Memória ou **memória principal** (MEM): responsável pelo armazenamento temporário das instruções, dados e seu processamento.

# ARQUITETURA MARIE

- Registradores e Barramentos
  - Registrador de instruções (**IR**): detém a próxima instrução a ser executada no programa.
  - Contador de Programa (**PC**): detém o próximo endereço de instrução a ser executado no programa.
  - Registrador de endereço de memória (**REM**): especifica um endereço de memória para a próxima leitura ou escrita.
  - Registrador de Buffer de Memória (**RBM**): contém dados a serem escritos na memória ou recebe dados lidos da memória.
  - Acumulador (**AC**): Responsável por guardar registros de dados. Este é um registro de uso geral e mantém os dados que a CPU precisa processar. A maioria dos computadores atualmente possuem múltiplos desses registros de uso geral.

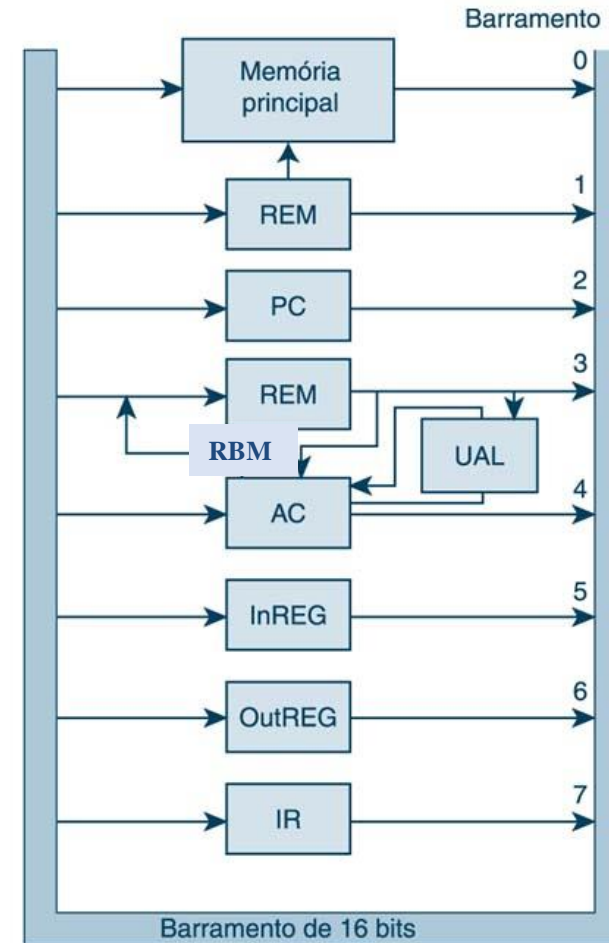


# ARQUITETURA MARIE

- Registradores e Barramentos
  - Registrador de entrada (**InREG**): Armazena os dados inseridos pelos componentes de entrada. (ex. teclado)
  - Registrador de saída (**OutREG**): Armazena os dados que serão enviados aos componentes de saída. (ex. monitor)

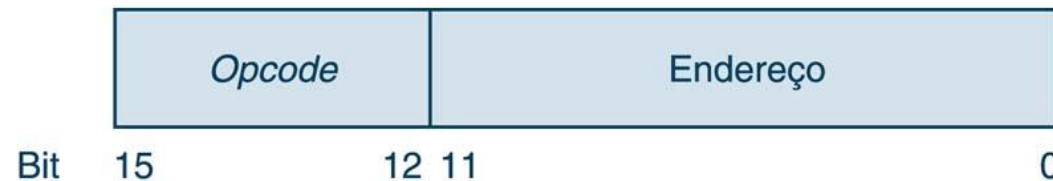
# ARQUITETURA MARIE

- Caminho de dados:
  - Caminho entre o **REM** e a memória (permite o endereçamento da memória pela CPU);
  - Caminho separado do **RBM** para **AC** e **UAL** (operações aritméticas).



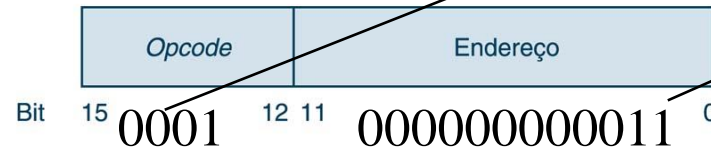
# ARQUITETURA MARIE

- Arquitetura do conjunto de instruções
  - A arquitetura do conjunto de instruções (ISA) especifica as instruções que o computador pode executar o formato de cada instrução (é uma interface entre o HW e SW);
  - Cada instrução de Marie consiste de 16 bits. Os 4 bits mais significativos (de 12 à 15), compõe o opcode (código de operação) que especifica a instrução a ser executada (com um total de 16 instruções). Os 12 bits menos significativos (0 à 11), formam um endereço que é o tamanho máximo da memória ( $2^{12} - 1$ ).



# ARQUITETURA MARIE

## ■ Conjunto de instruções Marie



Número da instrução		Instrução	Significado
Bin	Hex		
0001	1	Load X	Carrega o conteúdo do endereço X em AC.
0010	2	Store X	Armazena o conteúdo de AC no endereço X.
0011	3	Add X	Adiciona o conteúdo do endereço X a AC e armazena o resultado em AC.
0100	4	Subt X	Subtrai o conteúdo do endereço X de AC e armazena o resultado em AC.
0101	5	Input	Entra com um valor obtido via teclado em AC.
0110	6	Output	Exibe o valor de AC na tela.
0111	7	Halt	Termina o programa.
1000	8	Skipcond	Pula a próxima instrução sob condições.
1001	9	Jump X	Carrega o valor de X no PC.

- Os 4 bits mais à esquerda representa o Opcode, que representa a instrução **Load**. O 12 bits restantes representam o endereço do valor que está carregando, que é o endereço 3 na memória principal.

# ARQUITETURA MARIE

- Notação de transferência entre registradores
  - A notação simbólica usada para descrever o comportamento das micro-operações é denominada notação de transferência entre registradores (RTN – *register transfer notation*), que representa o dado real em uma posição X na memória e  $\leftarrow$  para indicar uma transferência de informação.
  - Notação de transferência entre registradores de algumas instruções do ISA do Marie.
  - Load X
    - $REM \leftarrow X$
    - $RBM \leftarrow M[REM]$
    - $AC \leftarrow RBM$
  - Store X
    - $REM \leftarrow X$
    - $RBM \leftarrow AC$
    - $M[REM] \leftarrow RBM$

**M = Memória Principal**



# ARQUITETURA MARIE

- Notação de transferência entre registradores
  - Add X
    - $REM \leftarrow X$
    - $RBM \leftarrow M[REM]$ 
      - $AC \leftarrow AC + RBM$
  - Subt X
    - $REM \leftarrow X$
    - $RBM \leftarrow M[REM]$
    - $AC \leftarrow AC - RBM$
  - Input
    - $AC \leftarrow InREG$
  - Output
    - $OutREG \leftarrow AC$
  - Halt
    - Nenhuma operação é realizada nos registradores, a máquina simplesmente cessa a execução do programa.

# ARQUITETURA MARIE

- Notação de transferência entre registradores
  - Jump X
    - Essa instrução causa um desvio incondicional para o endereço dado, X. Portanto, para executar esta instrução, X deve ser carregado no PC.
    - $PC \leftarrow X$

# SIMULADOR MARIE

- Conjunto de estendido de instruções

Número da instrução (hex)	Instrução	Significado
0	JnS X	Armazena o PC no endereço X e desvia para $X + 1$ .
A	Clear	Coloca zeros no AC.
B	AddI X	Soma indireta: Lê do endereço X. Usa o valor em X como o endereço real do operando de dados a ser somado ao AC.
C	JumpI X	Desvio indireto: Lê do endereço X. Usa o valor em X como o endereço real da posição para a qual desviar.



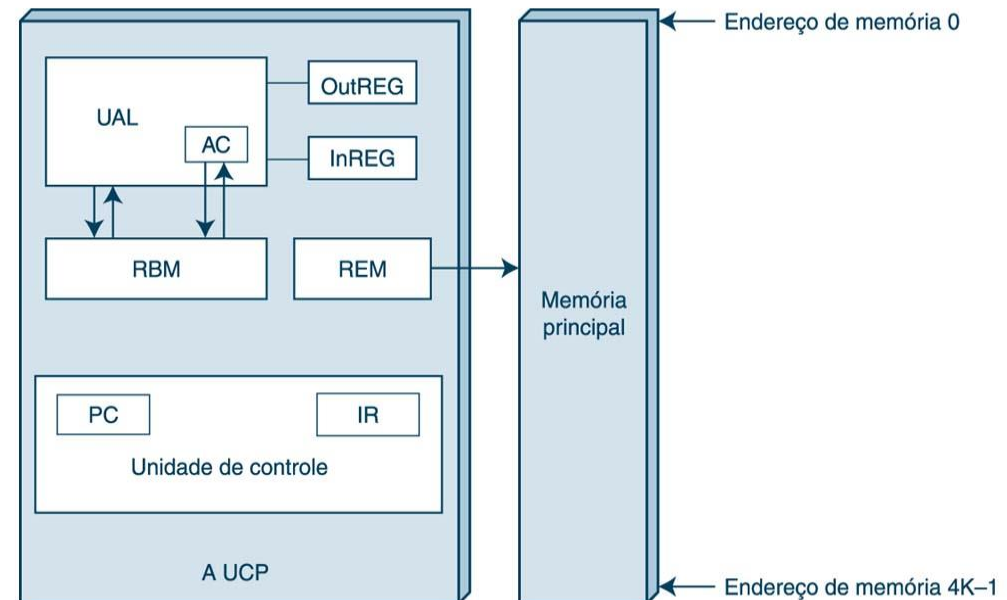
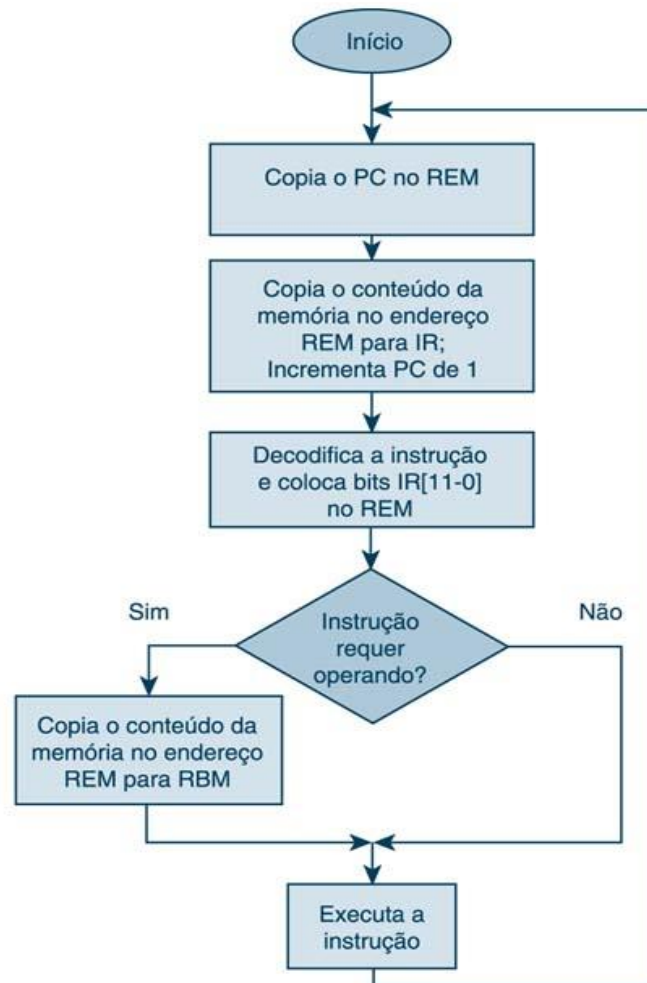
# SIMULADOR MARIE

- Conjunto completo de instruções

Opcode	Instrução	RTN
0000	JnS <i>X</i>	RBM ← PC REM ← <i>X</i> M[REM] ← RBM RBM ← <i>X</i> AC ← 1 AC ← AC + RBM PC ← AC
0001	Load <i>X</i>	REM ← <i>X</i> RBM ← M[REM] AC ← RBM
0010	Store <i>X</i>	REM ← <i>X</i> , RBM ← AC M[REM] ← RBM
0011	Add <i>X</i>	REM ← <i>X</i> RBM ← M[REM] AC ← AC + RBM
0100	Subt <i>X</i>	REM ← <i>X</i> RBM ← M[REM] AC ← AC - RBM
0101	Input	AC ← InREG
0110	Output	OutREG ← AC
0111	Halt	
1000	Skipcond	Se IR[11-10] = 00 então Se AC < 0 então PC ← PC + 1 senão Se IR[11-10] = 01 então Se AC = 0 então PC ← PC + 1 senão Se IR[11-10] = 10 então Se AC > 0 então PC ← PC + 1
1001	Jump <i>X</i>	PC ← IR[11-0]
1010	Clear <i>X</i>	AC ← 0
1011	AddI <i>X</i>	REM ← <i>X</i> RBM ← M[REM] REM ← RBM RBM ← M[REM] AC ← AC + RBM
1100	JumpI <i>X</i>	REM ← <i>X</i> RBM ← M[REM] PC ← RBM

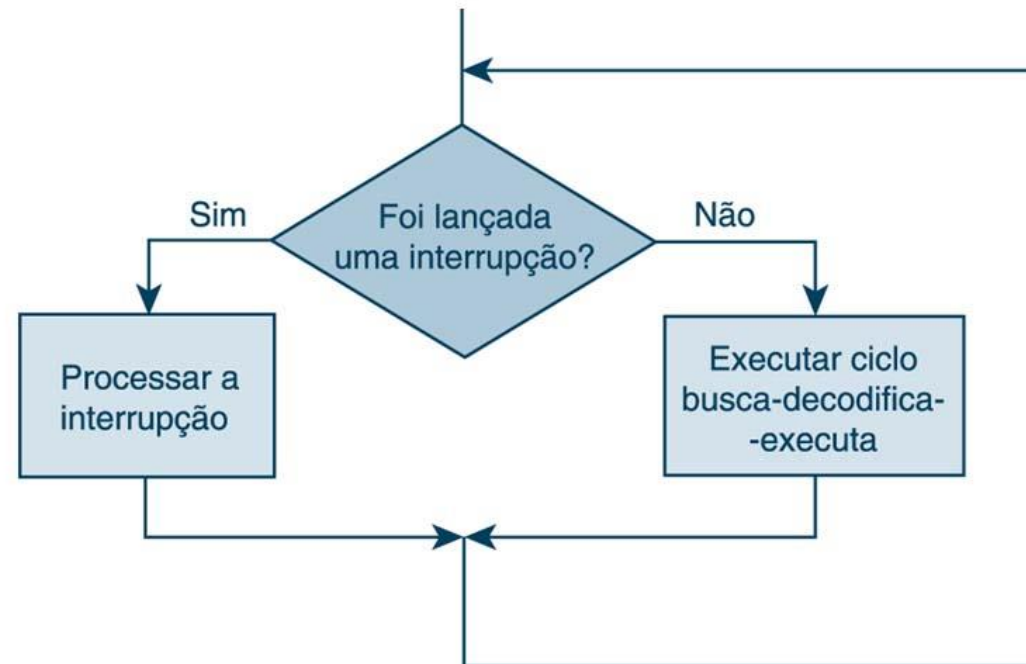
# ARQUITETURA MARIE

- Ciclo de busca-decodifica-executa



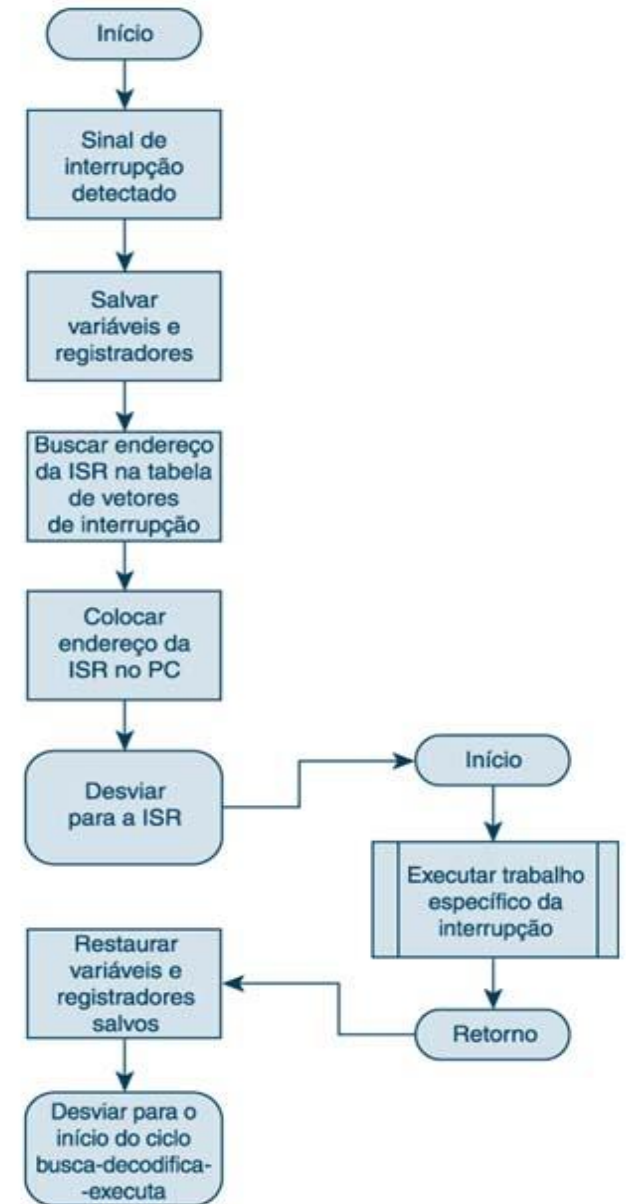
# ARQUITETURA MARIE

- Ciclo de busca-decodifica-executa com verificação de interrupção



# ARQUITETURA MARIE

- Processar uma interrupção



# ARQUITETURA MARIE

- Um programa simples: um programa para **somar 2 números**:
  - Inicialmente os dois números encontram-se na memória principal, serão carregados pela CPU, somados e armazenados na memória.
  - Na coluna de instrução, tem-se o programa em linguagem simbólica e na coluna conteúdo o seu correspondente em linguagem de máquina.
  - O ciclo busca-decodifica-executa é iniciado pela primeira instrução do programa encontrado no PC com endereço da primeira instrução onde é carregado para execução (100h).
  - O programa carrega **0023h ou 35d** no AC. Depois soma o valor **FFE9h ou -23d** que se encontra no endereço 105h, resultando no valor de 12d no AC. A instrução Store armazena o valor na posição de memória 106h.
  - Quando o programa finaliza (Halt) e o conteúdo da posição de memória 106 muda para **000Ch ou 12d**.

# ARQUITETURA MARIE

- Um exemplo do programa – Soma 2 números

Endereço		Instrução	
100		Load	X
101		Add	Y
102		Store	Z
103		Halt	
104	X,	DEC	35
105	Y,	DEC	-23
106	Z,	HEX	0000
<hr/>			
104	X,	HEX	<b>0023</b>
105	Y,	HEX	<b>FFE9h</b>
106	Z,	HEX	0000

# ARQUITETURA MARIE

- Lista de operações do programa para somar dois números

(a) Load 104

Passo	RTN	PC	IR	REM	RBM	AC
(valores iniciais)		100	-----	-----	-----	-----
Carrega	REM ← PC	100	-----	100	-----	-----
	IR ← M[REM]	100	1104	100	-----	-----
	PC ← PC+1	101	1104	100	-----	-----
	REM ← IR[11-0]	101	1104	104	-----	-----
Decodifica	(Decodifica IR[15-12])	101	1104	104	-----	-----
Obtém operando	RBM ← M[REM]	101	1104	104	0023	-----
Executa	AC ← RBM	101	1104	104	0023	0023

(b) Add 105

Passo	RTN	PC	IR	REM	RBM	AC
(valores iniciais)		101	1104	104	0023	0023
Carrega	REM ← PC	101	1104	101	0023	0023
	IR ← M[REM]	101	3105	101	0023	0023
	PC ← PC+1	102	3105	101	0023	0023
	REM ← IR[11-0]	102	3105	105	0023	0023
Decodifica	(Decodifica IR[15-12])	102	3105	105	0023	0023
Obtém operando	RBM ← M[REM]	102	3105	105	FFE9	0023
Executa	AC ← AC + RBM	102	3105	105	FFE9	000C

(c) Store 106

Passo	RTN	PC	IR	REM	RBM	AC
(valores iniciais)		102	3105	105	FFE9	000C
Carrega	REM ← PC	102	3105	102	FFE9	000C
	IR ← M[REM]	102	2106	102	FFE9	000C
	PC ← PC+1	103	2106	102	FFE9	000C
	REM ← IR[11-0]	103	2106	106	FFE9	000C
Decodifica	(Decodifica IR[15-12])	103	2106	106	FFE9	000C
Obtém operando	(não necessária)	103	2106	106	FFE9	000C
Executa	RBM ← AC	103	2106	106	000C	000C
	M[REM] ← RBM	103	2106	106	000C	000C

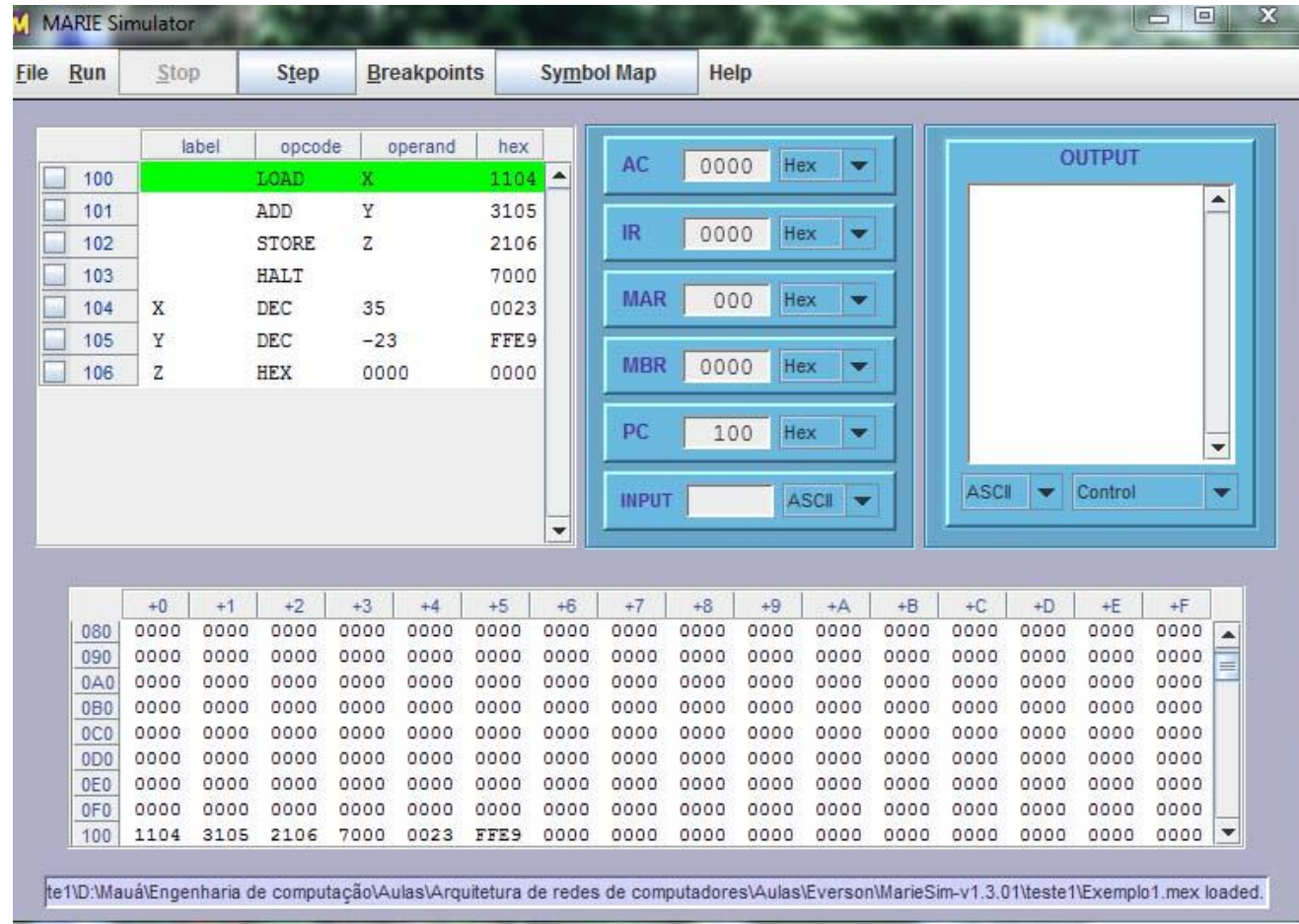
# ARQUITETURA MARIE

- Montador (Assembler)
  - Converte instruções de linguagem simbólica em linguagem de máquina, facilitando o uso palavras e símbolos ao invés de longas sequências de números.
  - O assembler converte a linguagem simbólica (mnemônicos) em linguagem de máquina (valores binários).
  - O assembler lê um arquivo fonte (programa simbólico) e produz o arquivo objeto (código de máquina).
  - Pode-se usar também rótulos (nomes simples) para identificar ou atribuir nomes a endereços de memória.



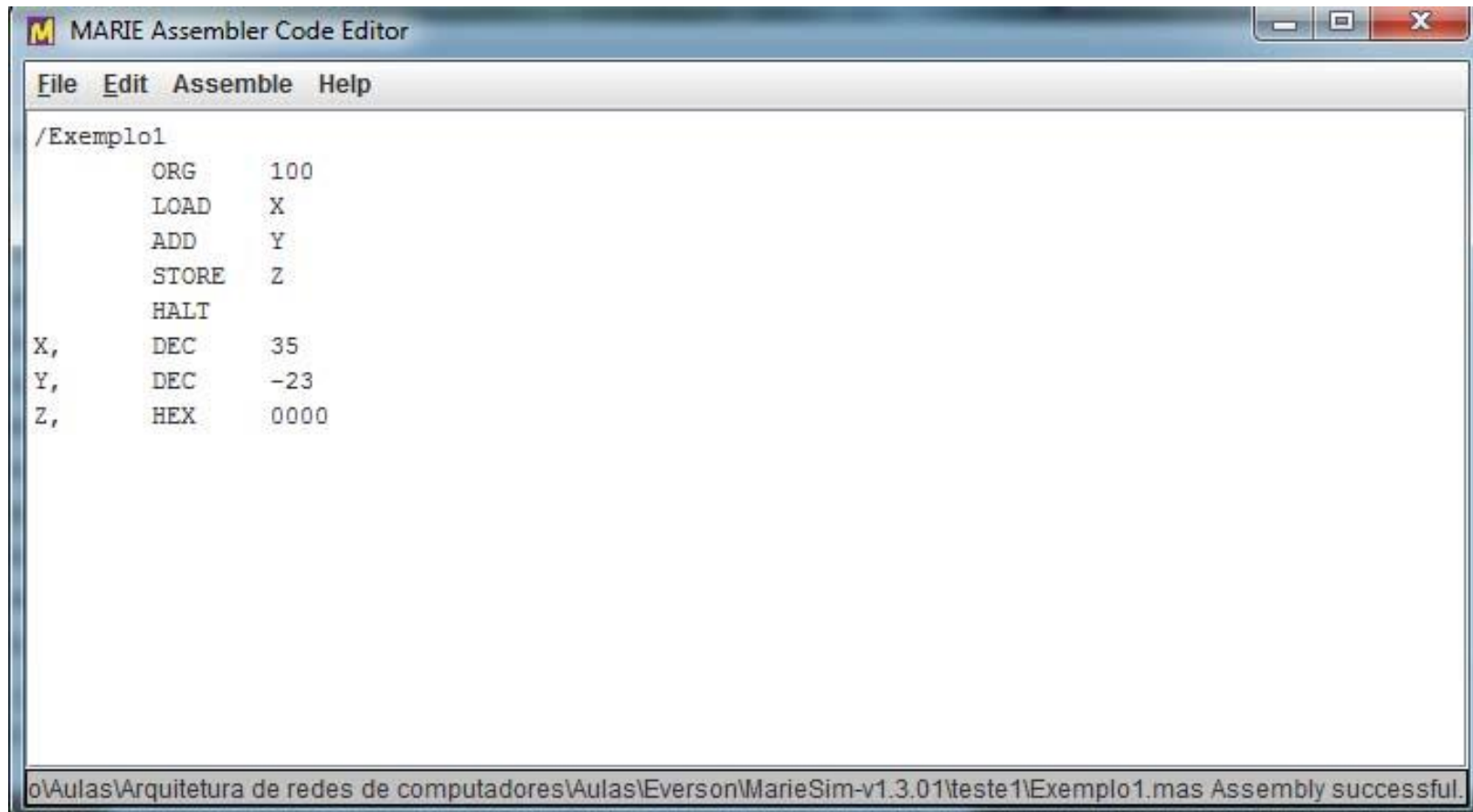
# ARQUITETURA MARIE

- Simulador Marie



# ARQUITETURA MARIE

- Simulador Marie



The screenshot shows a window titled "MARIE Assembler Code Editor". The window has a menu bar with "File", "Edit", "Assemble", and "Help". The main text area contains the following assembly code:

```
/Exemplo1
    ORG    100
    LOAD   X
    ADD    Y
    STORE  Z
    HALT
X,      DEC    35
Y,      DEC   -23
Z,      HEX    0000
```

At the bottom of the window, a status bar displays the message: "o:\Aulas\Arquitetura de redes de computadores\Aulas\Everson\MarieSim-v1.3.01\teste1\Exemplo1.mas Assembly successful."

# ARQUITETURA MARIE

## ■ Simulador Marie

Extensão dos arquivos	Descrição	Tipo	Criado por	Usado por
.mas	Código fonte assembly	Modo texto	Marie Editor	Marie Editor e Assembler
.lst	Arquivo de listagem assembly	Modo texto	Assembler	MarieEditor
.map	Tabela de símbolo	Modo texto	Assembler	MarieSim
.mex	Código executável	Arquivo objeto Java	Assembler	MarieSim
.dmp	Dump	Modo texto	MarieSim	MarieSim

# ARQUITETURA MARIE

- Listagem em assembler

```

|                                     /Exemplo1
|                                     ORG 100
100 1104 | LOAD X
101 3105 | ADD Y
102 2106 | STORE Z
103 7000 | HALT
104 0023 | X DEC 35
105 FFE9 | Y DEC -23
106 0000 | Z HEX 0000
```

Assembly successful.

## SYMBOL TABLE

Symbol	Defined	References
X	104	100
Y	105	101
Z	106	102

# ARQUITETURA MARIE

- Simulador Marie

(a) Load 104

Passo	RTN	PC	IR	REM	RBM	AC
(valores iniciais)		100	-----	-----	-----	-----
Carrega	REM $\leftarrow$ PC	100	-----	100	-----	-----
	IR $\leftarrow$ M[REM]	100	1104	100	-----	-----
	PC $\leftarrow$ PC+1	101	1104	100	-----	-----
Decodifica	REM $\leftarrow$ IR[11-0]	101	1104	104	-----	-----
	(Decodifica IR[15-12])	101	1104	104	-----	-----
Obtém operando	RBM $\leftarrow$ M[REM]	101	1104	104	0023	-----
Executa	AC $\leftarrow$ RBM	101	1104	104	0023	0023

# ARQUITETURA MARIE

- Simulador Marie

