UNIVERSIDADE FEDERAL DE ITAJUBÁ

SIN120 – Organização e Arquitetura de Computadores

5

Rafael Frinhani

frinhani@unifei.edu.br

2017

OBJETIVOS

Fornecer uma visão geral da programação em baixo nível com linguagem Assembly, fazer uma relação com conceitos dessa linguagem com lógica digital, ISA.

AGENDA

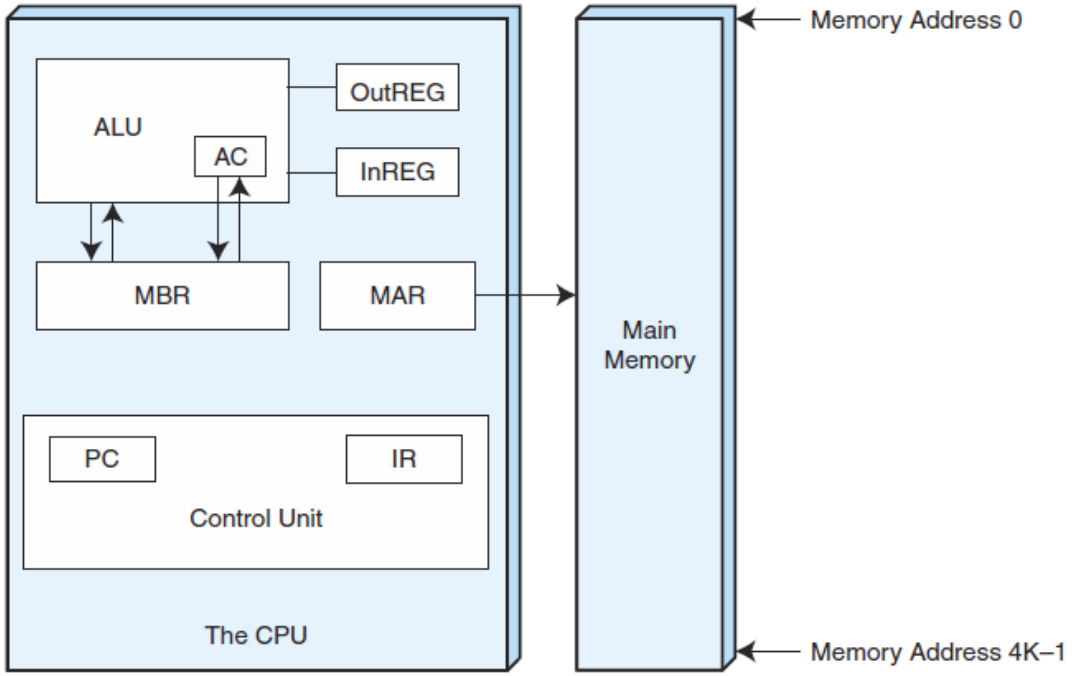
MARIE

- Características e Arquitetura
- Registradores
- Ambiente
- *Datapath e Fetch-Decode-Execute Cycle*
- ISA *(Instruction Set Architecture)*

MARIE – Características e Arquitetura

MARIE ***(Machine Architecture that is Really Intuitive and Easy)***

Arquitetura que consiste de memória (para armazenar programas e dados) e uma CPU (consistindo de uma ULA e diversos registradores).



MARIE – Características e Arquitetura

- Binária, complemento de dois
- *Stored Program*, palavras de tamanho fixo
- Palavra (mas não byte) endereçável
- 4K palavras de memória principal (12 bits por endereço)
- 16-bit para dados (palavras possuem 16 bits)
- 16-bit para instruções, 4 para opcode e 12 para endereços
- *Accumulator* (AC) de 16-bit
- *Instruction Register* (IR) de 16-bit
- *Memory Buffer Register* (MBR) de 16-bit
- *Program Counter* (PC) de 12-bit
- *Memory Address Register* (MAR) de 12-bit
- *Input Register* de 8-bit
- *Output Register* de 8-bit

MARIE – Registradores

AC: *Accumulator*, armazena valores de dados. Registrador de propósito geral que contém o dado que a CPU ira processar.

MAR: *Memory Address Register*, armazena o endereço de memória do dado que será referenciado.

MBR: *Memory Buffer Register*, armazena tanto o dado que foi lido da memória, quanto o dado que está pronto para ser escrito na memória.

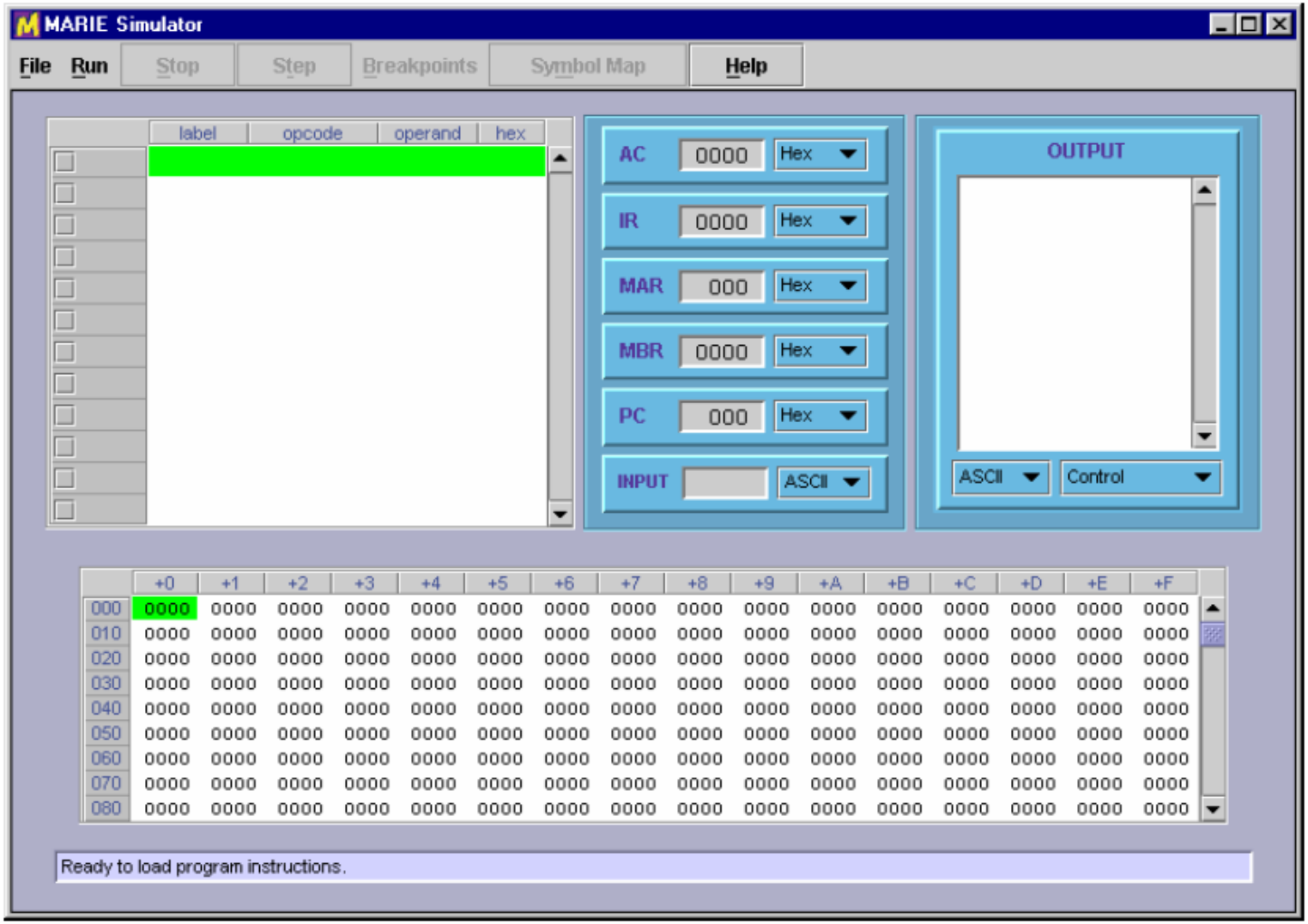
PC: *Program Counter*, armazena o endereço da próxima instrução a ser executada no programa.

IR: *Instruction Register*, armazena a próxima instrução a ser executada.

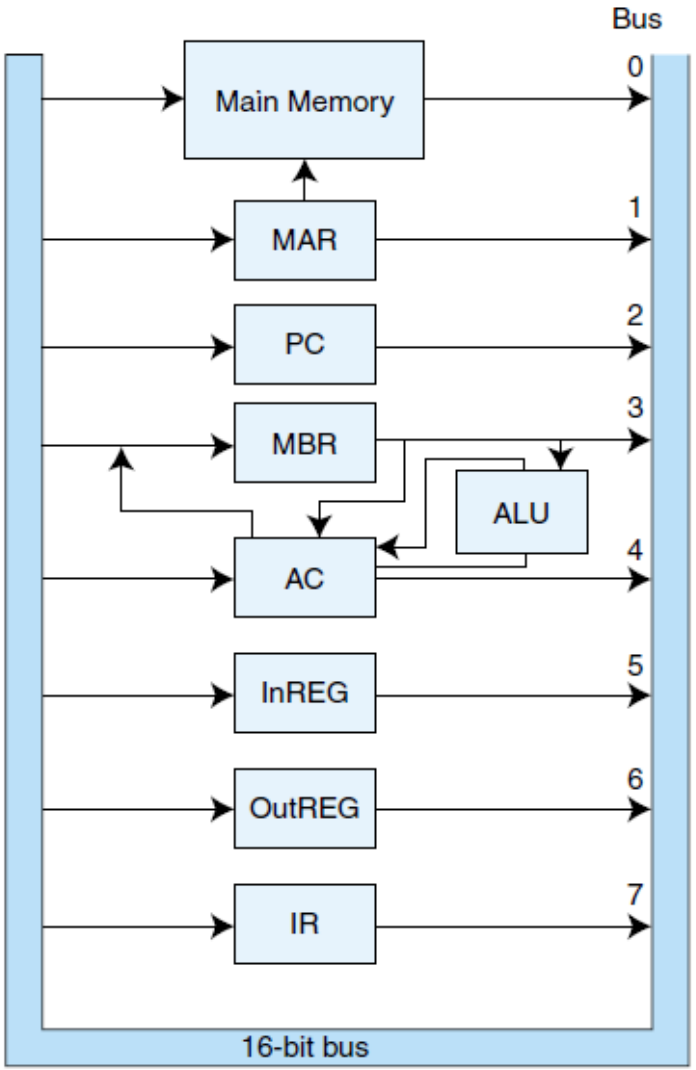
InREG: *Input Register*, armazena dados do dispositivo de entrada.

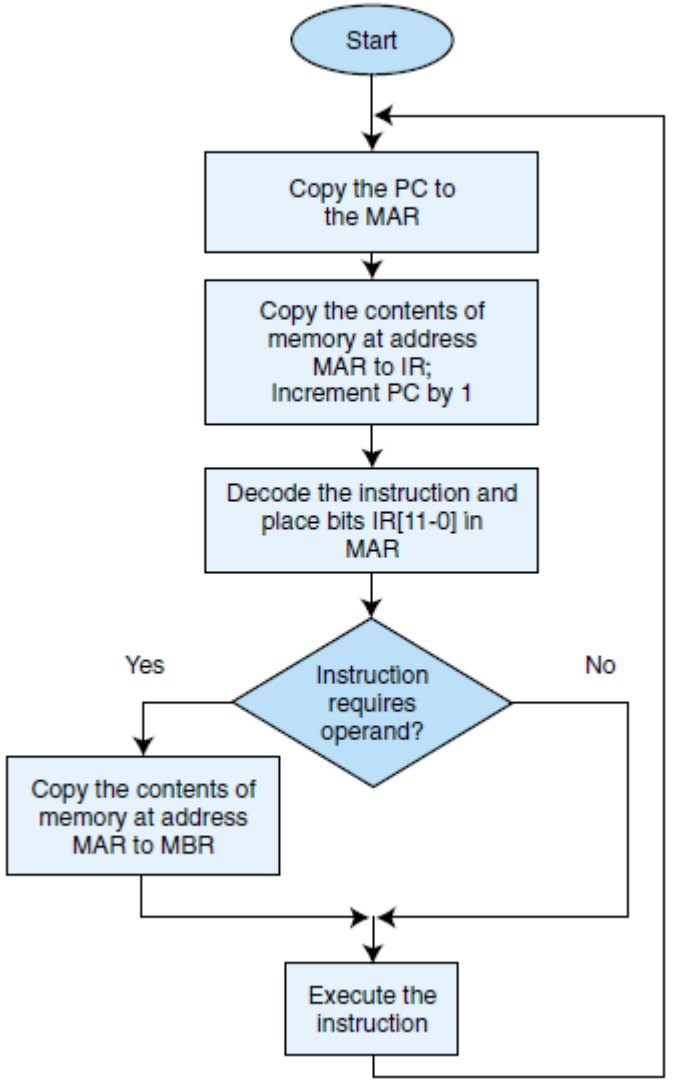
OutREG: *Output Register*, armazena dados do dispositivo de saída.

MARIE - Ambiente

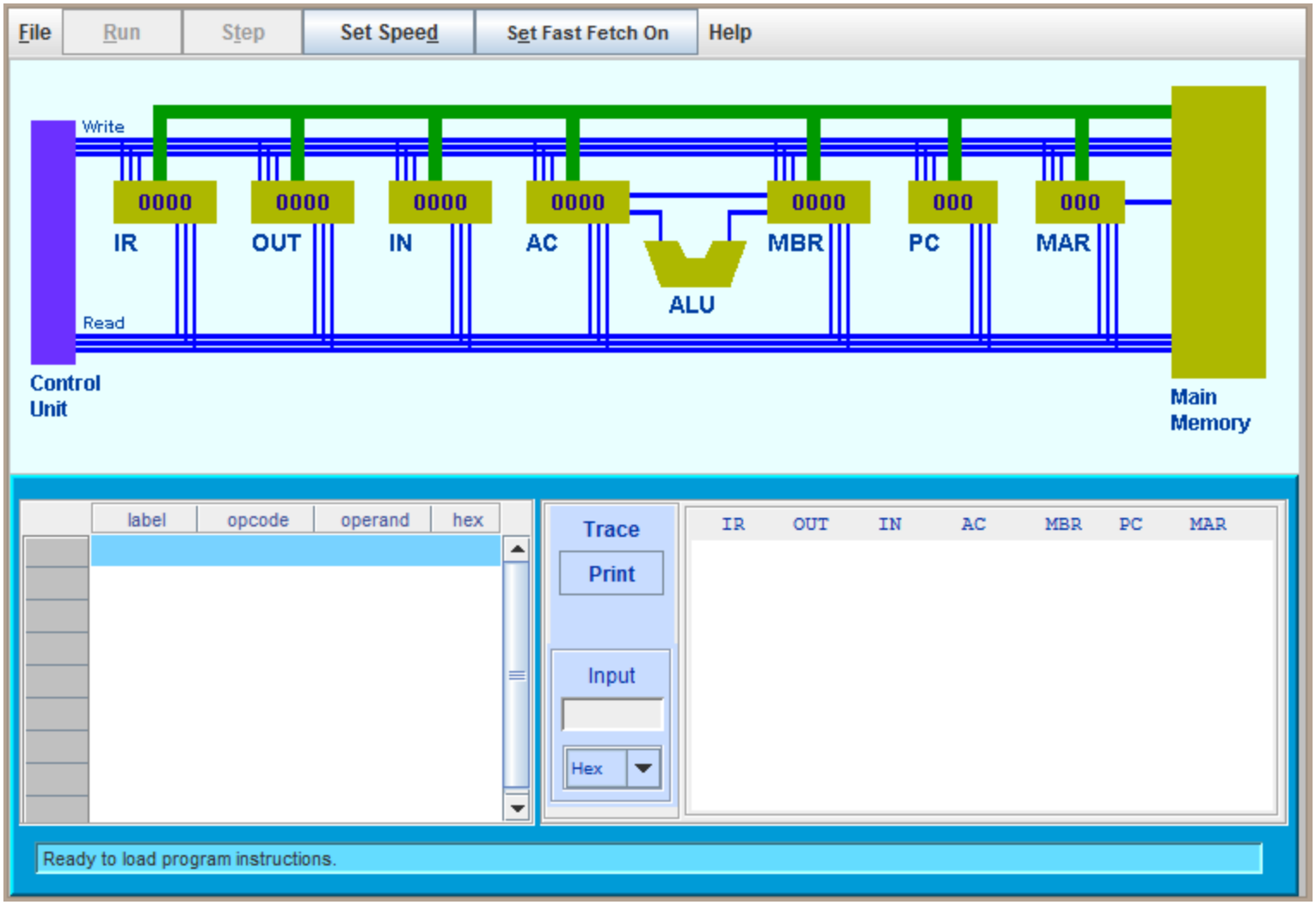


MARIE – *Data Path e Fetch-Decode-Execute-Cycle*





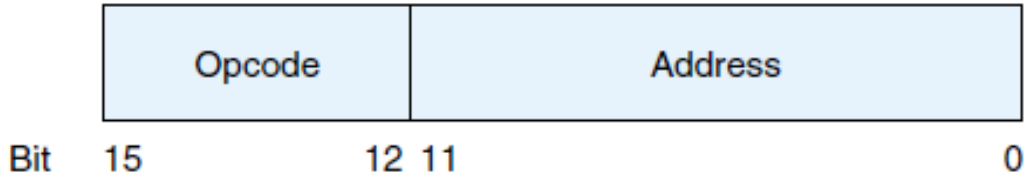
MARIE – *Data Path e Fetch-Decode-Execute-Cycle*



MARIE – *Instruction Set Architecture*

Instruction Number		Instruction	Meaning
Bin	Hex		
0001	1	Load *X*	Load the contents of address *X* into AC.
0010	2	Store *X*	Store the contents of AC at address *X*.
0011	3	Add *X*	Add the contents of address *X* to AC and store the result in AC.
0100	4	Subt *X*	Subtract the contents of address *X* from AC and store the result in AC.
0101	5	Input	Input a value from the keyboard into AC.
0110	6	Output	Output the value in AC to the display.
0111	7	Halt	Terminate the program.
1000	8	Skipcond	Skip the next instruction on condition.
1001	9	Jump *X*	Load the value of *X* into PC.

Formato da Instrução

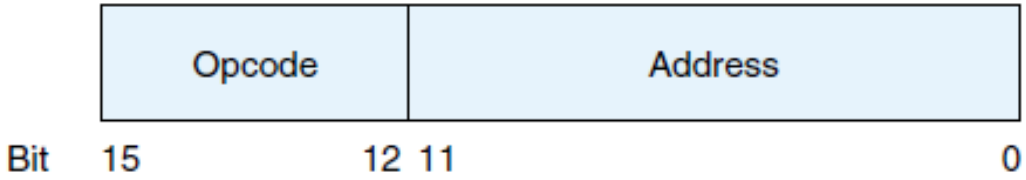


MARIE – *Instruction Set Architecture*

Instruction Number		Instruction	Meaning
Bin	Hex		
0001	1	Load *X*	Load the contents of address *X* into AC.
0010	2	Store *X*	Store the contents of AC at address *X*.
0011	3	Add *X*	Add the contents of address *X* to AC and store the result in AC.
0100	4	Subt *X*	Subtract the contents of address *X* from AC and store the result in AC.
0101	5	Input	Input a value from the keyboard into AC.
0110	6	Output	Output the value in AC to the display.
0111	7	Halt	Terminate the program.
1000	8	Skipcond	Skip the next instruction on condition.
1001	9	Jump *X*	Load the value of *X* into PC.

Skipcond 800: Skip if AC > 0
Skipcond 400: Skip if AC = 0
Skipcond 000: Skip if AC < 0

Formato da Instrução



MARIE – *Instruction Set Architecture*

Instruction Number		Instruction	Meaning
Bin	Hex		
0001	1	Load *X*	Load the contents of address *X* into AC.
0010	2	Store *X*	Store the contents of AC at address *X*.
0011	3	Add *X*	Add the contents of address *X* to AC and store the result in AC.
0100	4	Subt *X*	Subtract the contents of address *X* from AC and store the result in AC.
0101	5	Input	Input a value from the keyboard into AC.
0110	6	Output	Output the value in AC to the display.
0111	7	Halt	Terminate the program.
1000	8	Skipcond	Skip the next instruction on condition.
1001	9	Jump *X*	Load the value of *X* into PC.

Instruction Number (hex)	Instruction	Meaning
0	JnS *X*	Store the PC at address *X* and jump to *X* + 1.
A	Clear	Put all zeros in AC.
B	AddI *X*	Add indirect: Go to address *X*. Use the value at *X* as the actual address of the data operand to add to AC.
C	JumpI *X*	Jump indirect: Go to address *X*. Use the value at *X* as the actual address of the location to jump to.

MARIE – Exercícios

1) Implemente um programa que some dois números armazenados na memória.

MARIE – Exercícios

Como implementar comandos de repetição (ex. for, while) se a ISA do MARIE não prevê tais instruções?

MARIE – Exercícios

1) Implemente um programa que some dois números armazenados na memória.

2) Implemente um programa que utilize um loop para somar 5 números. Usar OUTPUT.

MARIE – Exercícios

Como implementar comandos condicionais (ex. if-then-else) se a ISA do MARIE não prevê tais instruções?

MARIE – Exercícios

1) Implemente um programa que some dois números armazenados na memória.

2) Implemente um programa que utilize um loop para somar 5 números. Usar OUTPUT.

3) Implemente uma rotina if-then-else do código abaixo:

```
if X = Y then
   X := X × 2
else
   Y := Y − X;
```

MARIE – Exercícios

4) Faça um programa que lê um número de 1 a 9 e imprime os resultados da tabuada de multiplicação (ex. para o número 4, a impressão será: 4, 8, 12, ... 36, 40). Usar OUTPUT.

5) Faça um programa que lê um dividendo e um divisor e faz a divisão do dividendo pelo divisor. Usar OUTPUT.

6) Faça um programa que calcule o fatorial de um número.

7) Faça um programa que realize a ordenação pelo método bolha de uma sequência de 7 números.