Linguagem de Montagem

Introdução
Aula 01

Edmar André Bellorini

Ano letivo 2016

Introdução

- Linguagem de Máquina (Baixo Nível)
  - É a linguagem que o computador realmente entende e executa
  - Composta por *strings* binárias
- Linguagem de Montagem (Código Intermediário)
  - É semelhante à Linguagem de Máquina,
  - Usa-se textos e números para compreensão humana
- Linguagem de Alto Nível
  - Linguagem com alto nível de abstração da Linguagem de Montagem

Linguagens (Alto Nível vs Montagem vs Máquina)

L. de Alto Nível (C)

```c
int main(){
   puts("Ola");
   return 0;
}
```

L. de Montagem (as)

```
section .data
   strOla :  db "Ola", 10
   strOlaL:  equ $ - strOla

section .text
        global _start

_start:
   mov eax, 4
   mov ebx, 1
   mov ecx, strOla
   mov edx, strOlaL
   int 0x80

   mov eax, 1
   mov ebx, 0
   int 0x80
```

L. de Máquina (parcial)

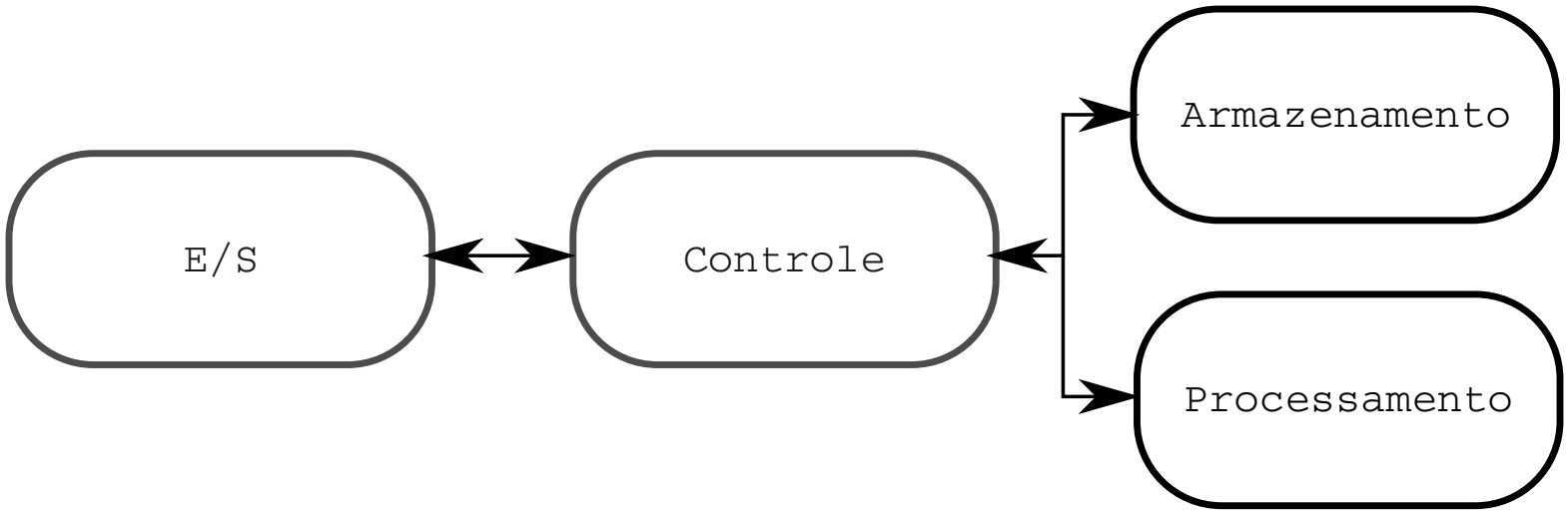
```
7f454c46020101000000
00000000000002003e00
01000000b00040000000
00040000000000000000
00010000000000000000
...
000000bb00000000cd80
4f6cc3a10a0000002e73
796d746162002e737472
746162002e7368737472
746162002e7465787400
2e646174610000000000
00000000000000000000
...
2e6f007374724f6c6100
7374724f6c614c005f73
74617274005f5f627373
5f7374617274005f6564
617461005f656e6400
```

Linguagem de Montagem

- Por que estudar LM?
  - Como as instruções são executadas?
  - Como os dados estão representados em memória?
  - Como um programa interage com o S.O. e outros programas?
  - Como as instruções de Alto Nível são traduzidas para Baixo Nível?
    - Deseja tornar-se um melhor programador de Linguagens de Alto Nível?
  - Como é a arquitetura de um computador x86 ou x64?

Arquitetura

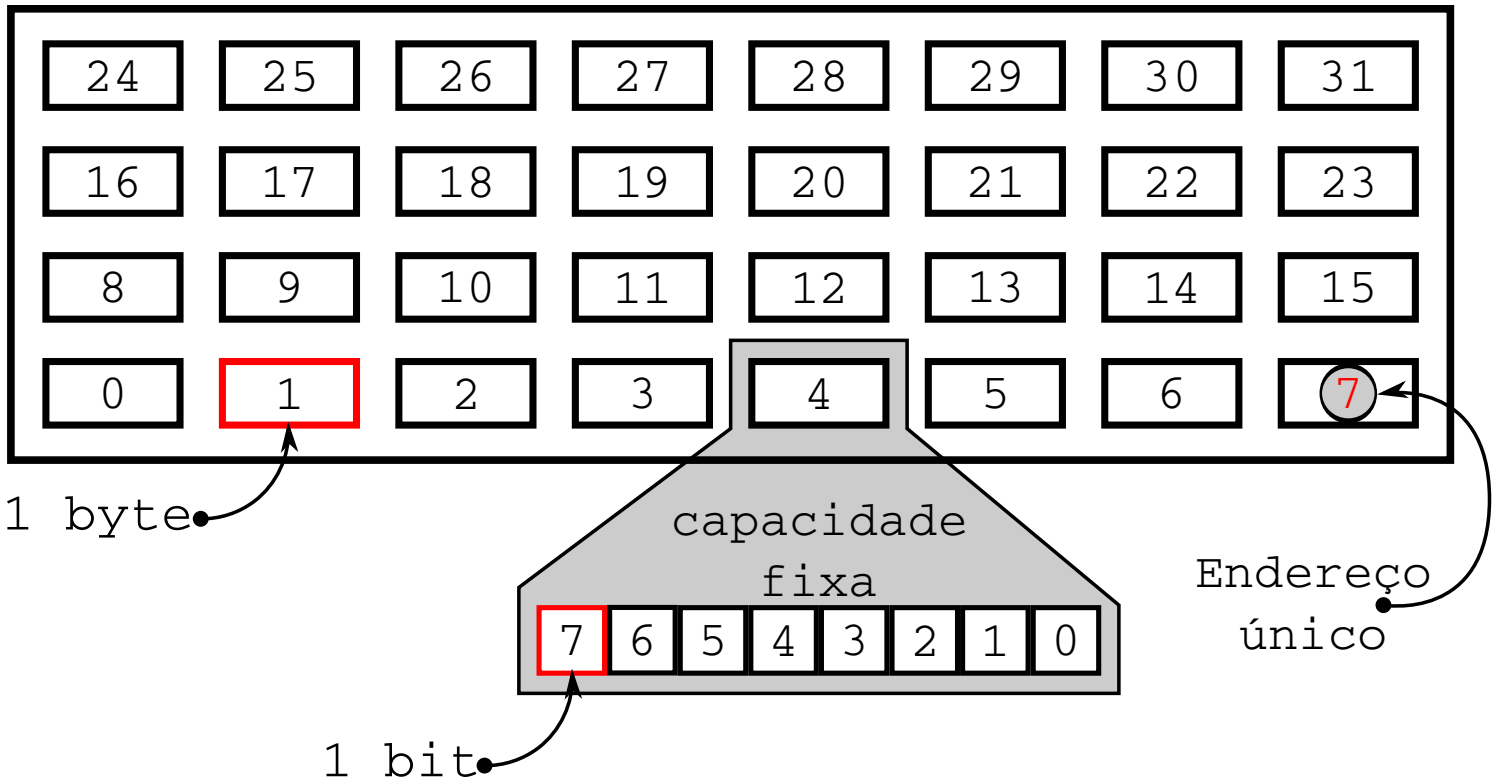
- Von Neumann
  - Armazenamento
  - Unidade de Processamento
  - Movimentação de Dados (E/S)
  - Controle do Fluxo de Dados (Conexões)



Armazenamento (Memória)

- Função
  - Local onde residem as informações
    - Sistema Operacional
    - Programas e Dados do usuário
- Estrutura
  - Local constituído de “espaços” para armazenamento
    - Cada espaço tem uma capacidade fixa (8 bits)
    - Cada espaço tem um endereço único

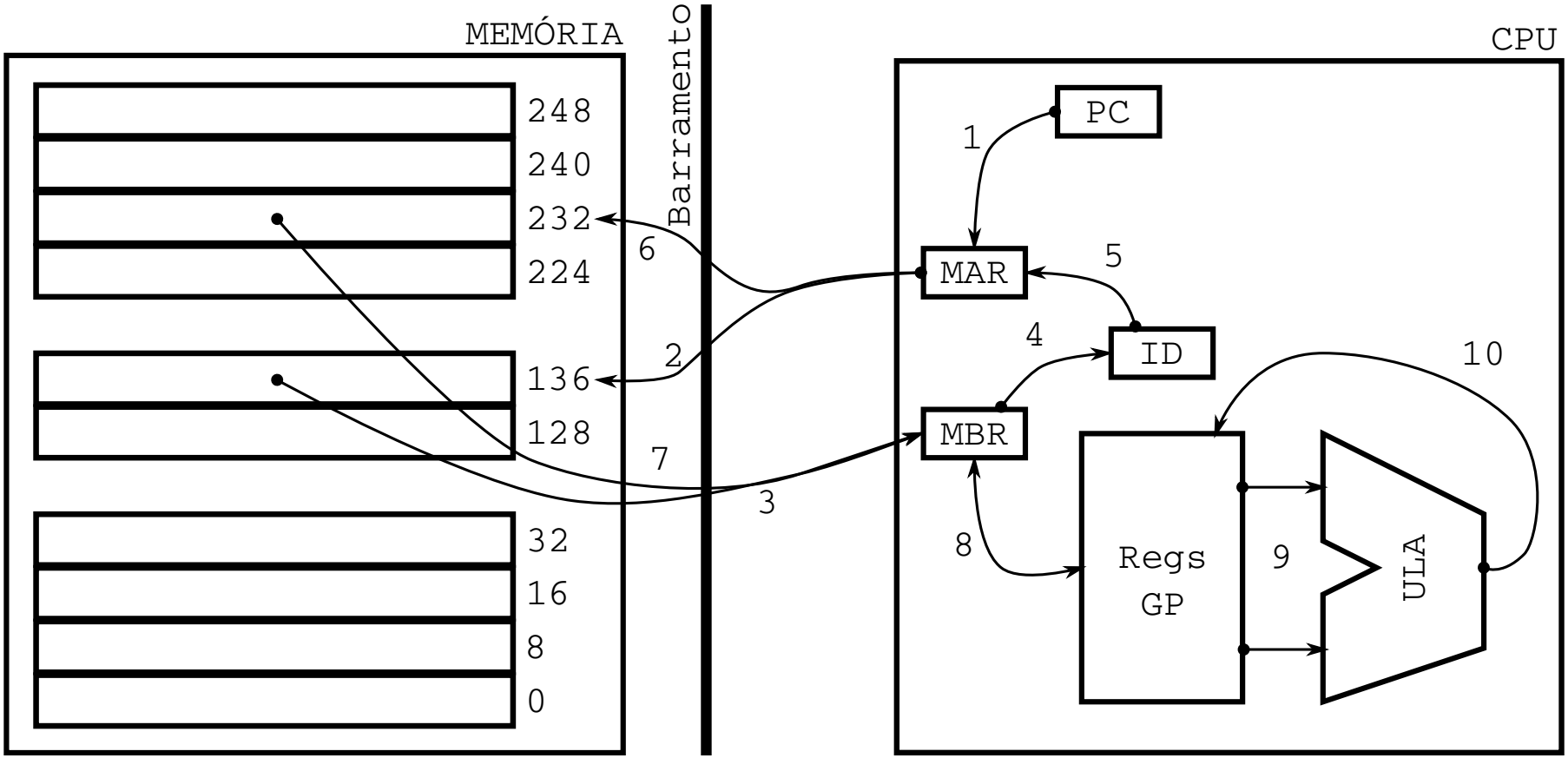
Armazenamento (Memória)



Unidade de Processamento

- Função
  - Ler instruções da memória (uma-a-uma)
  - Executar instruções
- Estrutura
  - Contador de Programa (PC - `Program-Counter`)
  - Decodificador de Instruções (ID - `Instruction Decoder`)
  - Registradores de Propósito Geral (GP - `General-Purpose`)
  - Registradores de Propósito Específico (MAR e MBR)
  - Unidade Lógica e Aritmética

Unidade de Processamento



Montador NASM

- **N**etwide **Ass**e**m**bler
  - Sintaxe simplificada
    Sintaxe Intel / alta legibilidade
  - Compatível com Windows e **Linux**
  - Programas x86 e x64
    no terminal execute: `uname -m`
    i686 ou i386 = máquina 32 bits (x86)
    x86_64 = máquina 64 bits (x64)
  - Licensa `BSD - 2 Clause`
  - Constante desenvolvimento
    Versão 2.12 lançada em 26/02/2016 21h06m
    www.nasm.us

Sintaxe Intel vs AT&T

- Sintaxe Intel
  - Instrução:
    mnemônico destino, fonte
  - Exemplo:
    mov eax, 4
    copia para o local chamado eax o valor 4
  - Montadores: NASM, TASM (Turbo), MASM (Microsoft)
- Sintaxe AT&T
  - Instrução:
    mnemônico fonte, destino
  - Exemplo:
    mov $4, %eax
    copia o valor 4 para o local chamado eax
  - Montadores: GAS(GNU), nativo em Linux (as)

Sintaxe

Intel

```
1  section .data
2     strOla :  db "Ola", 10
3     strOlaL:  equ $ - strOla
4
5  section .text
6           global _start
7
8  _start:
9     mov eax, 4
10    mov ebx, 1
11    mov ecx, strOla
12    mov edx, strOlaL
13    int 0x80
14
15    mov eax, 1
16    mov ebx, 0
17    int 0x80
```

AT&T

```
1  .section .data
2     strOla:  .string "Ola\n\0"
3     .equ strOlaL, .-strOla
4
5  .section .text
6  .globl _start
7
8  _start:
9     movl $4, %eax
10    movl $1, %ebx
11    movl $strOla, %ecx
12    movl $strOlaL, %edx
13    int $0x80
14
15    movl $1, %eax
16    movl $0, %ebx
17    int $0x80
```

HelloWorld

hello.c

```c
int main(){
    puts("Ola");
    return 0;
}
```

```asm
section .data
    strOla :  db "Ola", 10
    strOlaL:  equ $ - strOla

section .text
        global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, strOla
    mov edx, strOlaL
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Compilar? Montar? Linkar?

- Compilar o código `helloworld.c` com compilador `gcc`

  ```
  gcc hello.c -o helloC.x
  ```

- Montar e `linkar` o código `helloworld.s` com:
  - montador `nasm` para máquinas 32 bits

    ```
    nasm -f elf32 hello.asm
    ```
  - montador `nasm` para máquinas 64 bits

    ```
    nasm -f elf64 hello.asm
    ```
  - linkador `ld`

    ```
    ld hello.o -o helloS.x
    ```

- Qual é o tamanho dos arquivos executáveis (linguagem de máquina) resultantes?
  - *Fast, Powerful and Small*

Executar

- Para executar arquivo (em Linguagem de Máquina):
  - O arquivo `helloworldC.x` compilado:

    `./helloC.x`
  - O arquivo `helloworldS.x` montado e linkado:

    `./helloS.x`
- Observações
  - O ./ indica para o S.O. Linux que o programa a ser executado encontra-se no diretório atual.
  - O retorno da execução do programa é avaliado com:

    `echo $?`

Exercícios de Fixação

- EF0101: Elaborar um executável que imprima seu nome no terminal.
  - Utilize o código exemplo com sintaxe Intel para auxiliá-lo
  - É necessário montar, linkar e executar o código
  - Se algum erro for apresentado durante a produção do executável: anote, avalie e corrija.

Exercício de Fixação

- EF0102: Com o código EF0101 completo, efetue as seguintes alterações, de forma independente, gere o executável, execute e documente os resultados:
  - Comente a linha 6 (linha que contém global _start)
    É possível comentar uma linha utilizando ";" (ponto-e-vírgula)
  - Comente a linha 12 (mov edx, strOlaL)
  - Altere a linha 12
    de mov edx, strOlaL para mov edx, 2
  - Comente a linha 13 (int 0x80)
  - Altere a linha 16
    de mov ebx, 0 para mov ebx, 5
    Após executar digite "echo $?" no terminal
    Teste outros valores

Relatório

- O modelo de relatório para a disciplina de LM está disponível em anexo desta aula
  Arquivo **modeloRelatorioLM.odt**

- Este modelo será usado para todas as práticas que contiverem exercícios de fixação a serem entregues

- Somente serão aceitos os relatórios em formato .pdf com nome do arquivo seguindo o padrão
  PXX.nome.sobrenome.pdf
  Onde PXX é o número da prática, neste caso: P01
  - Para futuras aulas práticas que requerem arquivos fontes junto com relatório será necessário empacotar todos os arquivos no formato .zip com o mesmo padrão de nome
    PXX.nome.sobrenome.zip