

**Instituto de
Computação**

UNIVERSIDADE ESTADUAL DE CAMPINAS



Organização Básica de computadores e linguagem de montagem

MC404 - 2o semestre de 2020

Prof. Edson Borin

<https://www.ic.unicamp.br/~edson>

Institute of Computing - UNICAMP

MC404 no 2º semestre de 2020

Regras em:

- <https://moodle.ggte.unicamp.br/course/view.php?id=8084>

Familiarize-se com os critérios de avaliação!

Por que aprender linguagem de montagem?

Permite compreender o funcionamento de uma CPU

Utilizado na:

- Programação de máquinas baseadas em microcontroladores
- Programação de sistemas embarcados (*embedded systems*)
- Programação de trechos críticos (tempo e/ou memória)
- Acesso a recursos não disponíveis em alto nível

OBS.: A linguagem de montagem é absolutamente ligada ao *hardware*, depende de cada máquina específica (diferentemente das linguagens de alto nível, como C, C++ e Java)

Por que aprender linguagem de montagem?

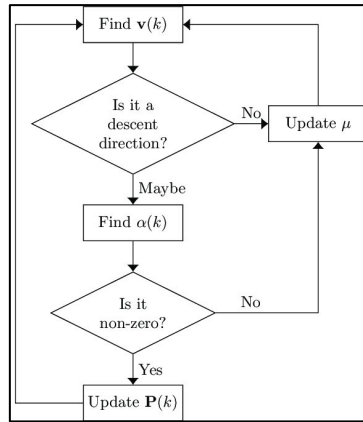
Permite entender como programas escritos em linguagens de alto nível, como C ou Java, são traduzidos para a linguagem de máquina

Conceitos Básicos

Resolução de problemas com Computadores

Problema

Resolução de problemas com Computadores



Problema

Algoritmo

Modelo computacional

Algorithm 1: Algorithm to compute the sequence of digits $d_b^{m-1}d_b^{m-2}\dots d_b^1d_b^0$ that represent value V on base b .

input : Value V and base b .

output: Sequence of digits $d_b^{m-1}d_b^{m-2}\dots d_b^1d_b^0$.

$i = 0$;

$tmp = V$;

while $tmp \neq 0$ **do**

$rem = tmp \bmod b$;

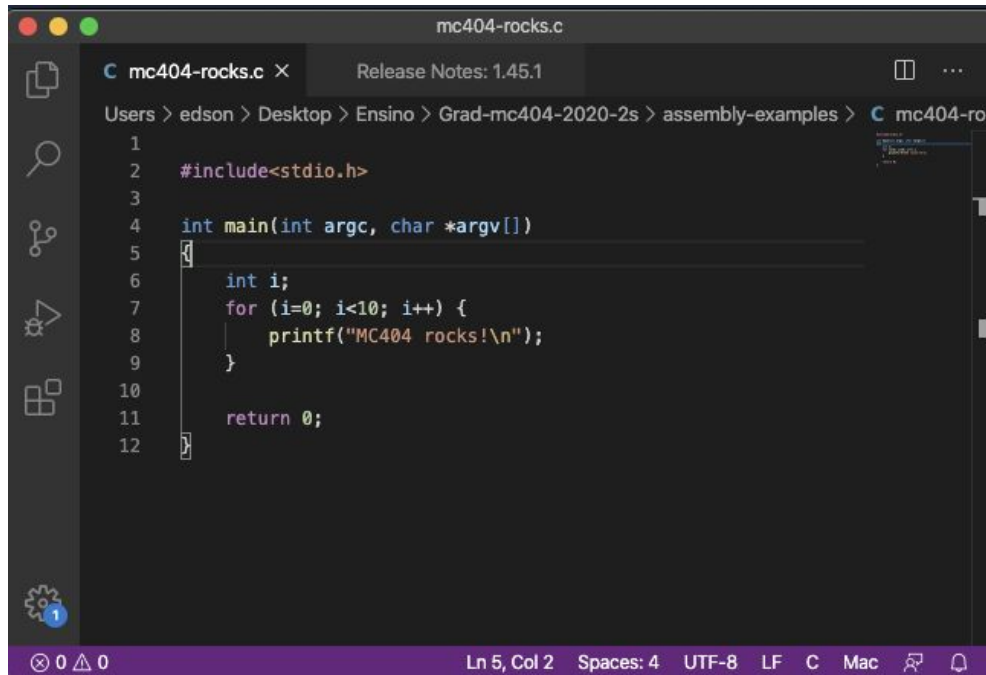
$d_b^i = symbol_from_value(rem, b)$;

$tmp = tmp / b$;

$i = i + 1$;

end

Resolução de problemas com Computadores



```
1
2 #include<stdio.h>
3
4 int main(int argc, char *argv[])
5 {
6     int i;
7     for (i=0; i<10; i++) {
8         printf("MC404 rocks!\n");
9     }
10
11     return 0;
12 }
```

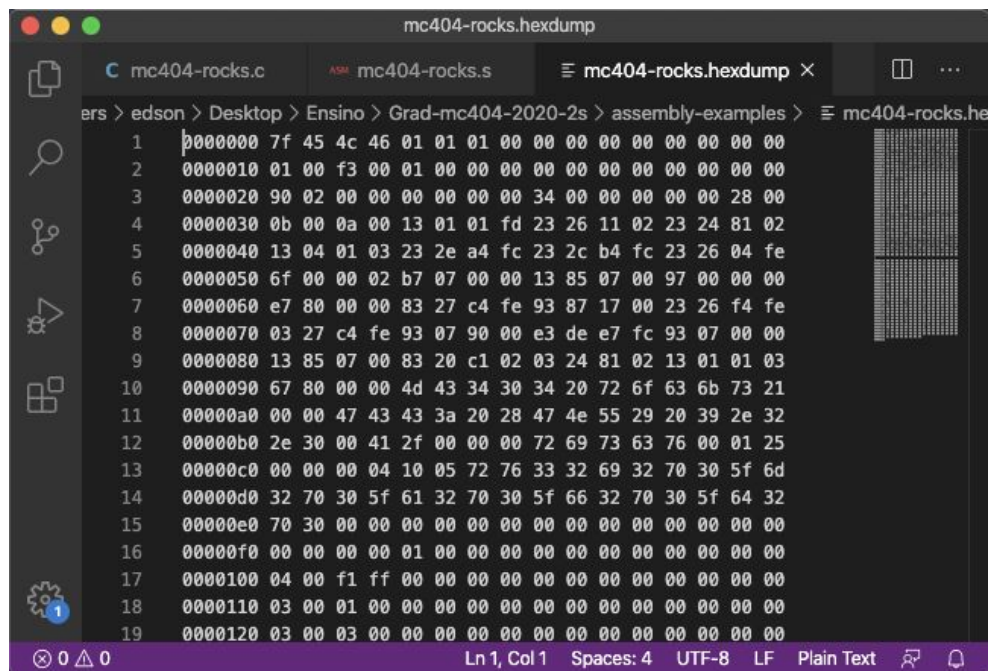
Problema

Algoritmo

Programa em linguagem
de alto nível (p.ex.: C)

Sintaxe e Semântica da Linguagem

Resolução de problemas com Computadores



The screenshot shows a hex dump editor window titled 'mc404-rocks.hexdump'. The editor displays a list of memory addresses and their corresponding hexadecimal values. The addresses range from 00000001 to 00000120. The values are displayed in two columns: the first column shows the address and the second column shows the hexadecimal value. The values are mostly zeros, with some non-zero values interspersed. The editor has a dark theme and a sidebar on the left with various icons. The status bar at the bottom shows 'Ln 1, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and 'Plain Text'.

```
1 00000001 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
2 00000010 01 00 f3 00 01 00 00 00 00 00 00 00 00 00 00 00
3 00000020 90 02 00 00 00 00 00 00 34 00 00 00 00 00 28 00
4 00000030 0b 00 0a 00 13 01 01 fd 23 26 11 02 23 24 81 02
5 00000040 13 04 01 03 23 2e a4 fc 23 2c b4 fc 23 26 04 fe
6 00000050 6f 00 00 02 b7 07 00 00 13 85 07 00 97 00 00 00
7 00000060 e7 80 00 00 83 27 c4 fe 93 87 17 00 23 26 f4 fe
8 00000070 03 27 c4 fe 93 07 90 00 e3 de e7 fc 93 07 00 00
9 00000080 13 85 07 00 83 20 c1 02 03 24 81 02 13 01 01 03
10 00000090 67 80 00 00 4d 43 34 30 34 20 72 6f 63 6b 73 21
11 000000a0 00 00 47 43 43 3a 20 28 47 4e 55 29 20 39 2e 32
12 000000b0 2e 30 00 41 2f 00 00 00 72 69 73 63 76 00 01 25
13 000000c0 00 00 00 04 10 05 72 76 33 32 69 32 70 30 5f 6d
14 000000d0 32 70 30 5f 61 32 70 30 5f 66 32 70 30 5f 64 32
15 000000e0 70 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00
16 000000f0 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00
17 00001000 04 00 f1 ff 00 00 00 00 00 00 00 00 00 00 00 00
18 00001100 03 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00
19 00001200 03 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Problema

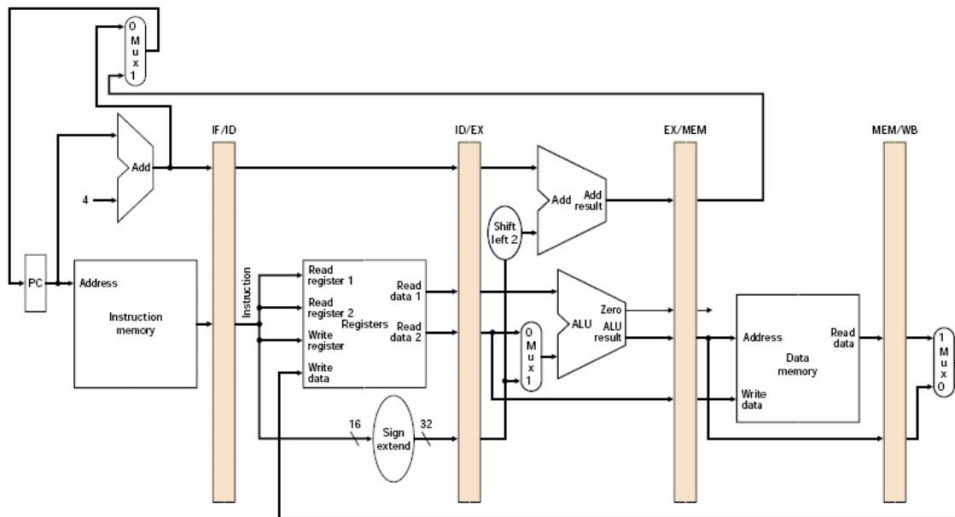
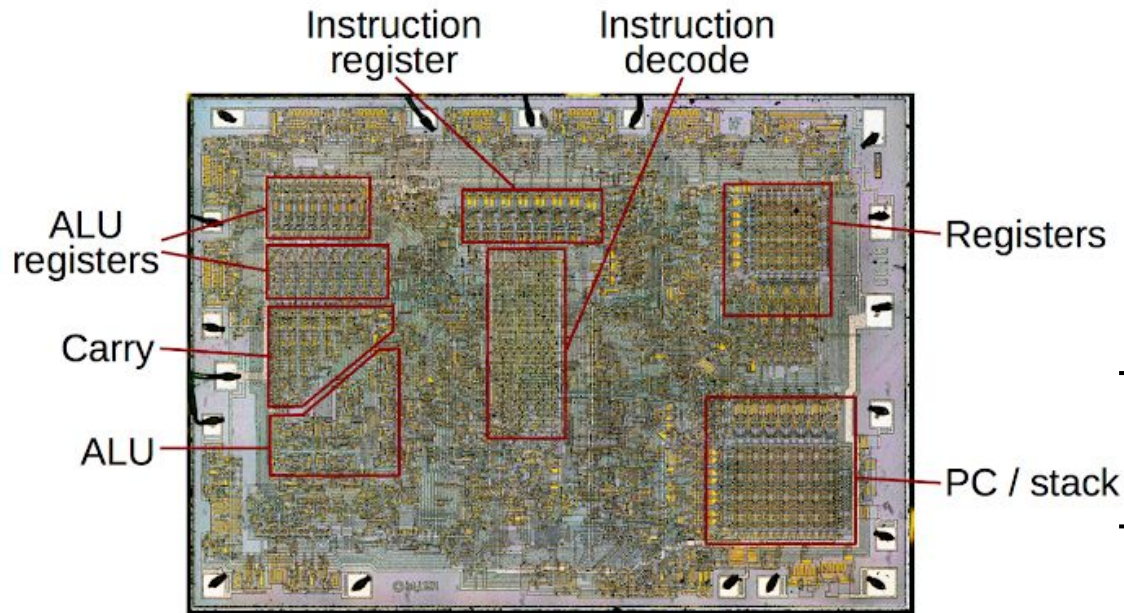
Algoritmo

Programa em linguagem
de alto nível (p.ex.: C)

Programa em linguagem
de máquina

Organização e arquitetura do computador

Resolução de problemas com Computadores



Problema

Algoritmo

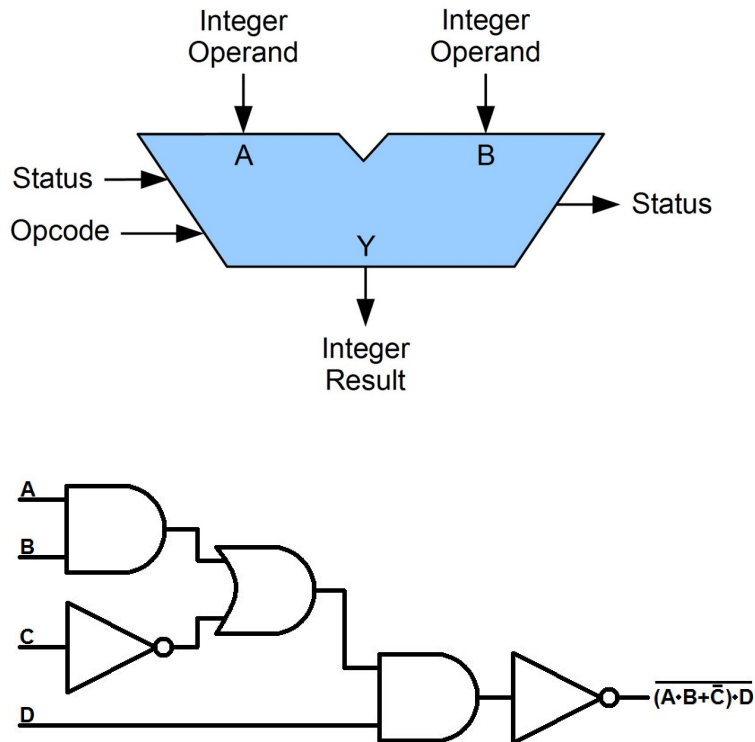
Programa em linguagem
de alto nível (p.ex.: C)

Programa em linguagem
de máquina

Microarquitetura do
processador

Dispositivos lógicos
(registradores, ALU, ...)

Resolução de problemas com Computadores



Problema

Algoritmo

Programa em linguagem
de alto nível (p.ex.: C)

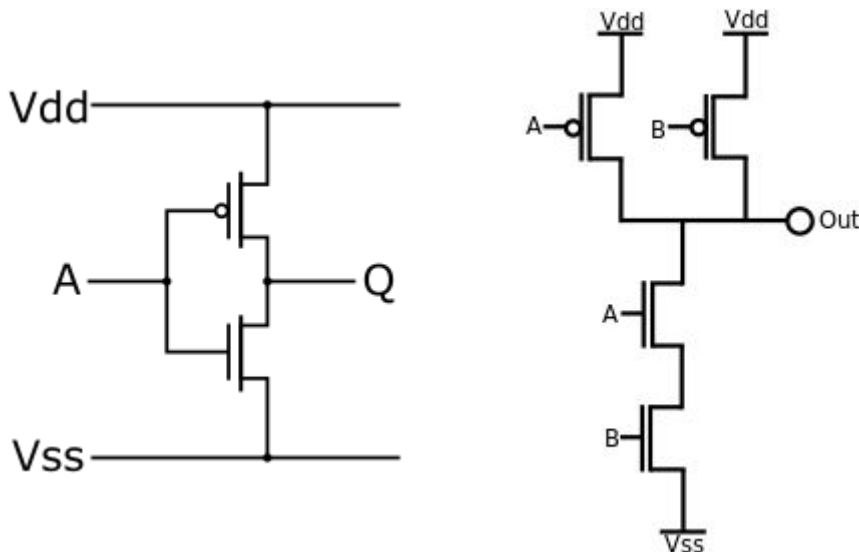
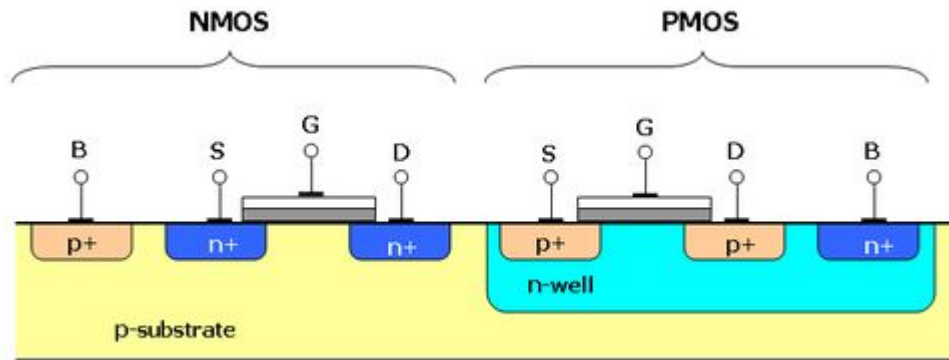
Programa em linguagem
de máquina

Microarquitetura do
processador

Circuitos lógicos

Portas lógicas

Resolução de problemas com Computadores



Problema

Algoritmo

Programa em linguagem
de alto nível (p.ex.: C)

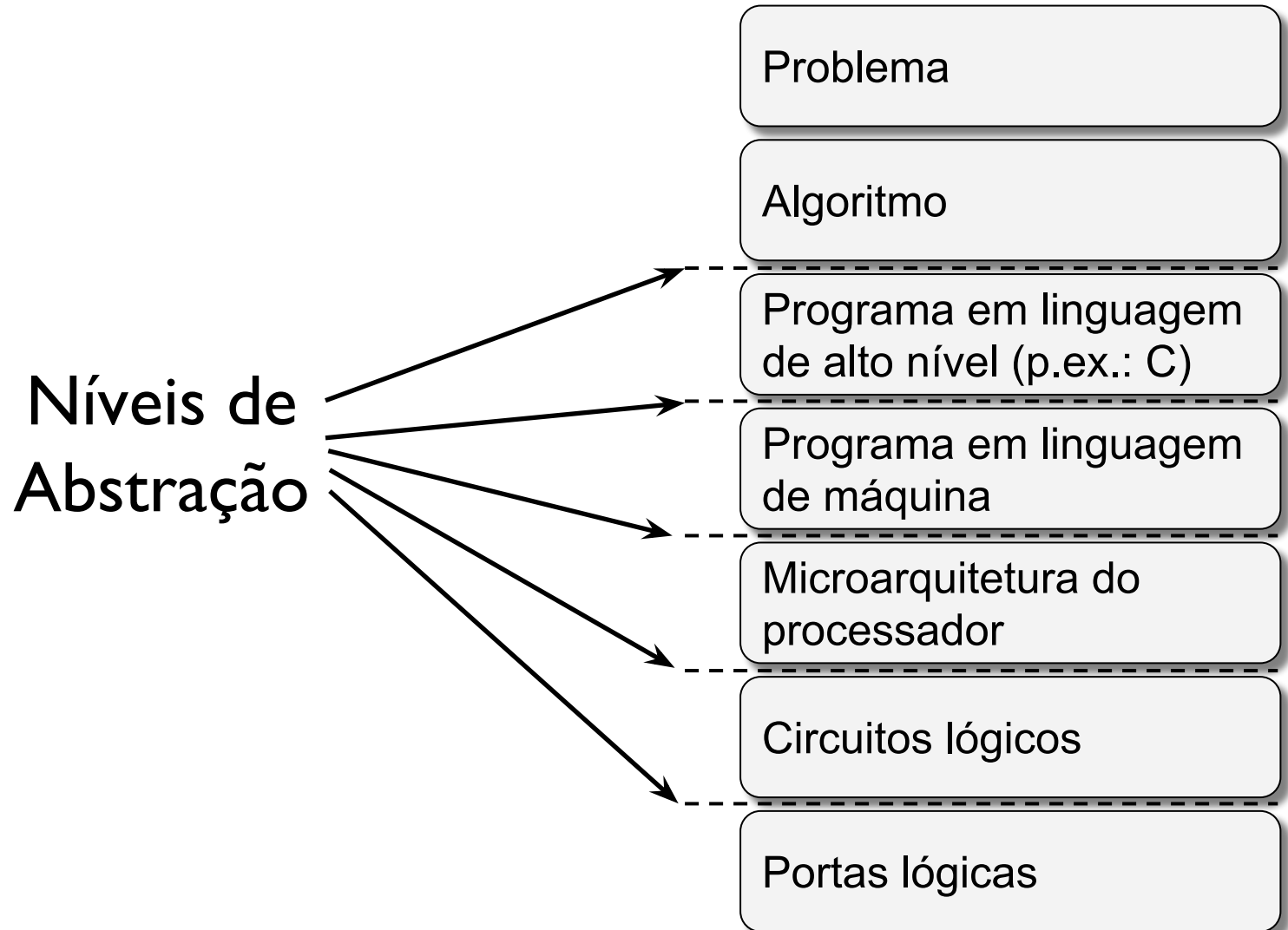
Programa em linguagem
de máquina

Microarquitetura do
processador

Circuitos lógicos

Portas lógicas

Resolução de problemas com Computadores



Resolução de problemas com Computadores

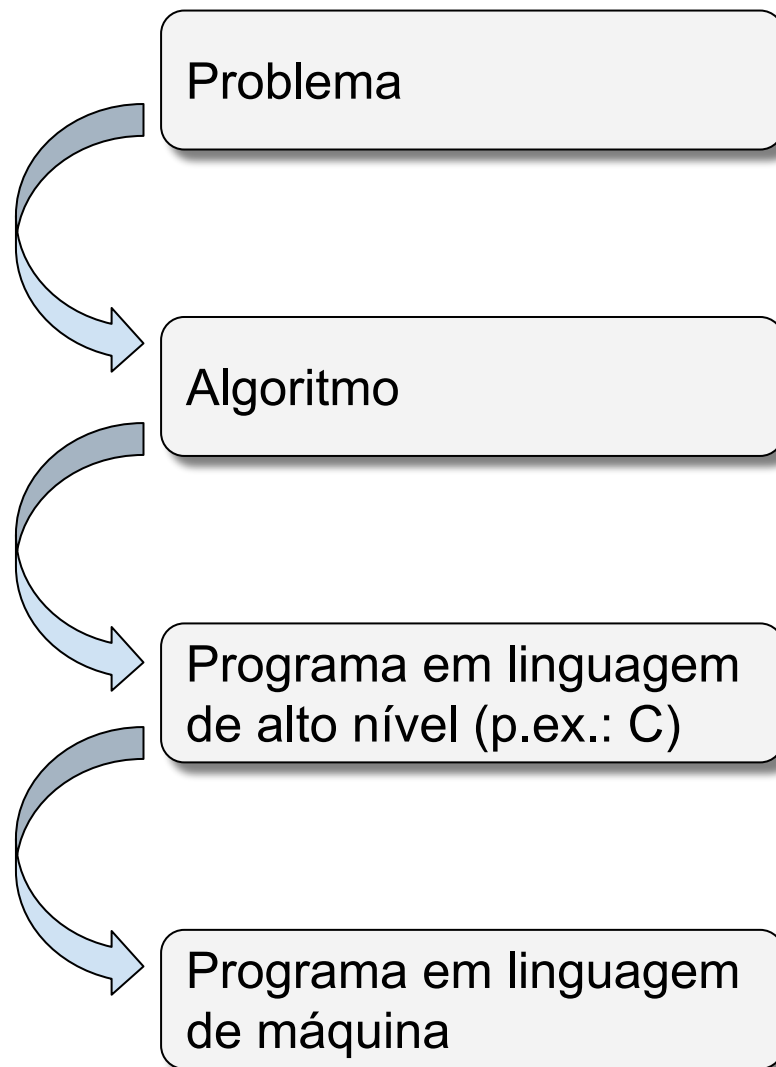
MC102 e MC202

Projeto de Software:
escolher algoritmos e estrutura de dados

Programação:
implementar o projeto com uma linguagem

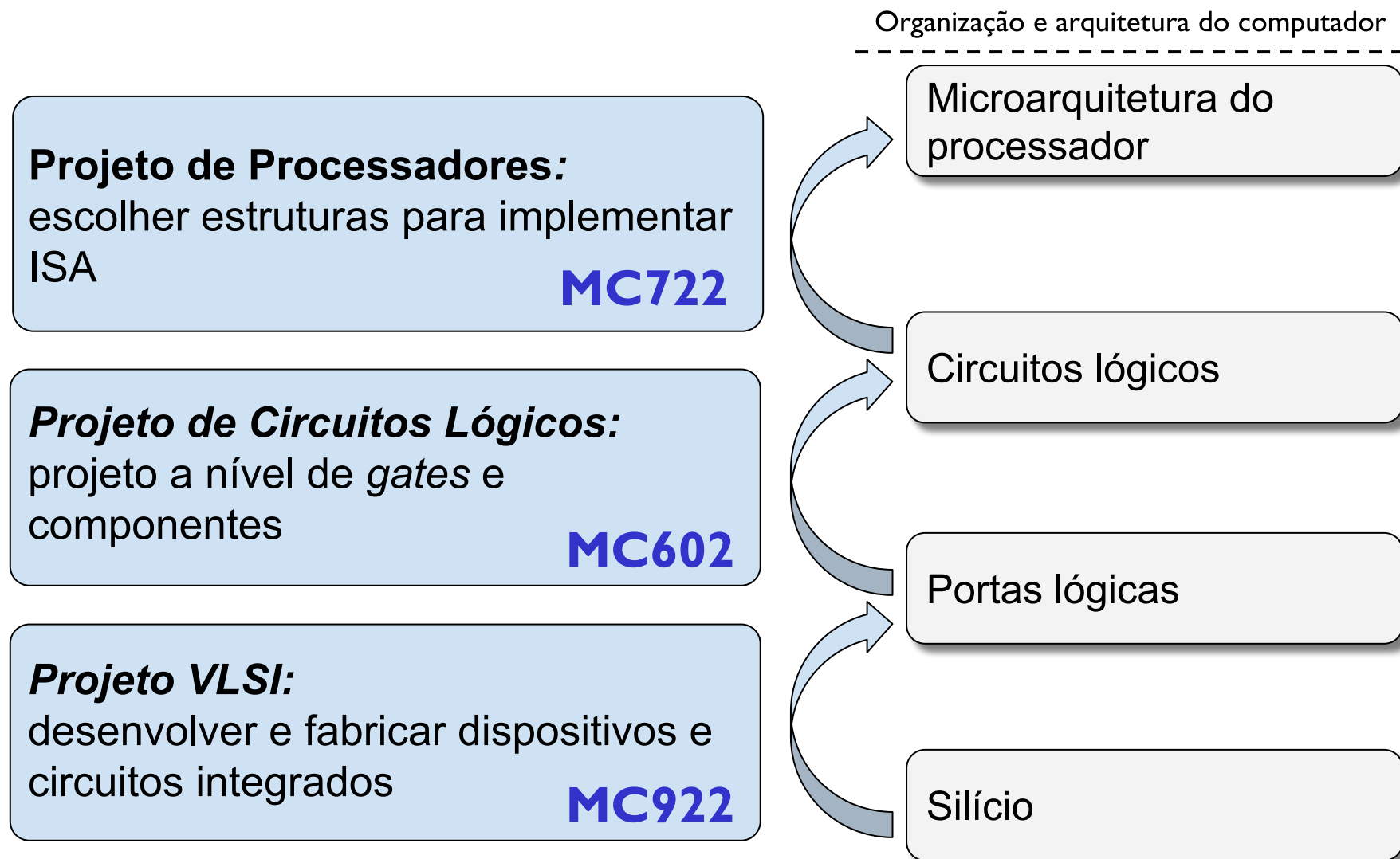
Compilação/Interpretação:
converter linguagem para instruções de máquina

MC404 e MC910



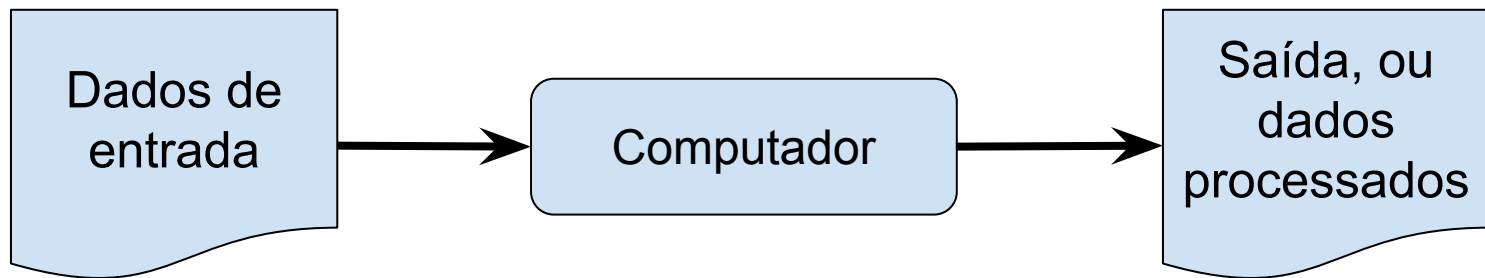
Organização e arquitetura do computador

Resolução de problemas com Computadores



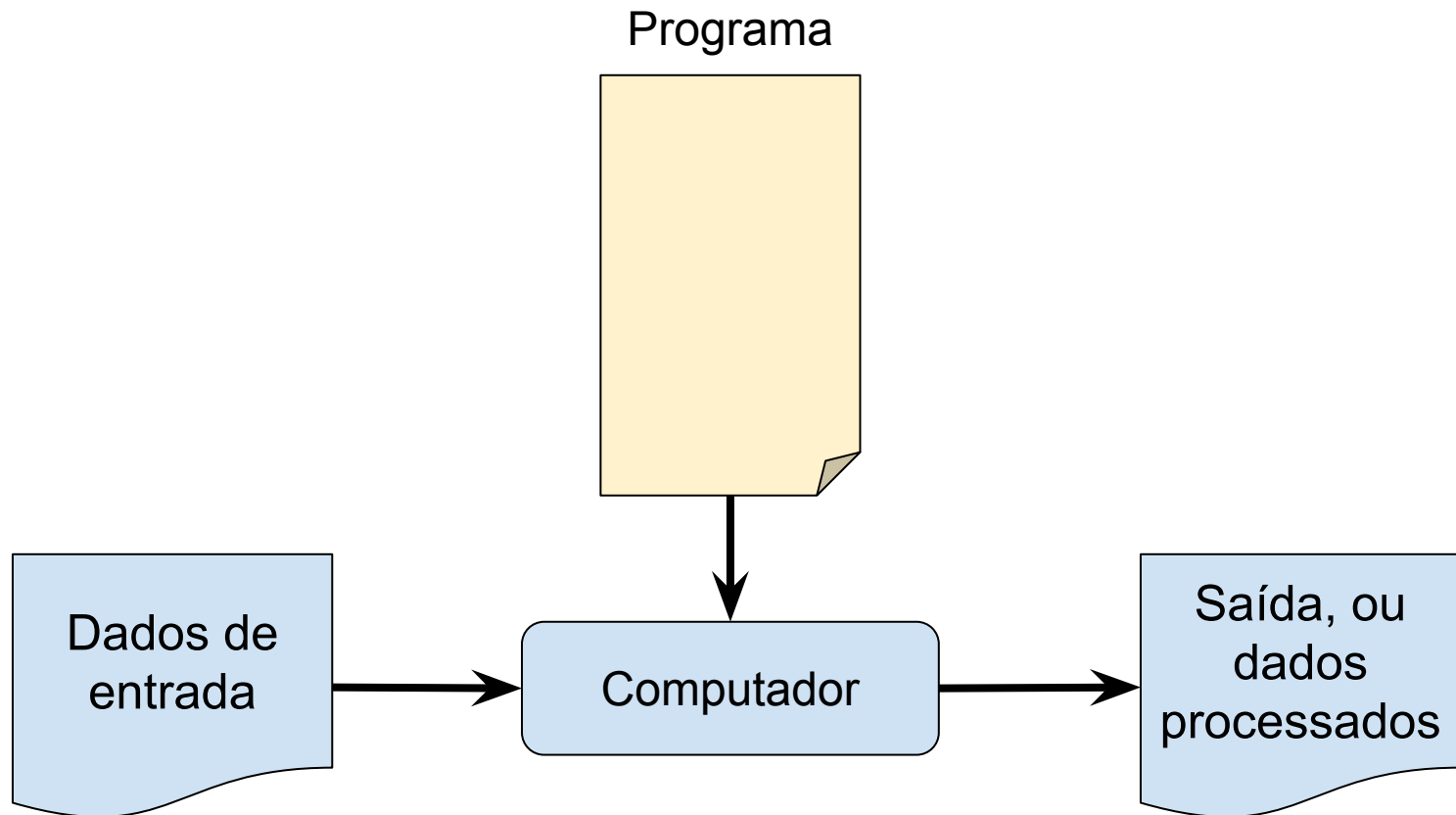
Conceitos Básicos: Computadores

Máquinas para manipular informações (ou dados)



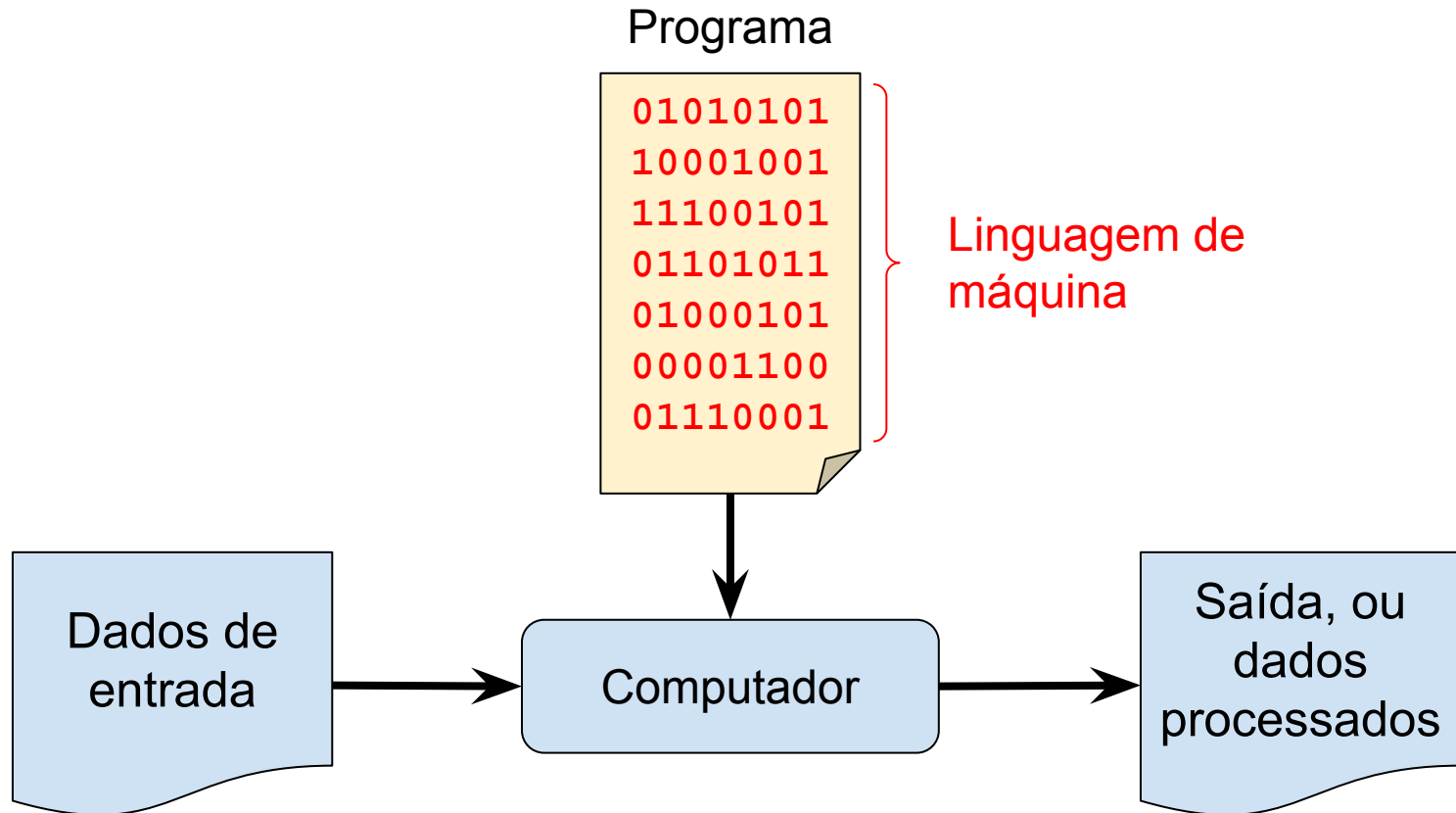
Conceitos Básicos: Computadores

Máquinas "**programáveis**" para manipular informações (ou dados)



Conceitos Básicos: Computadores

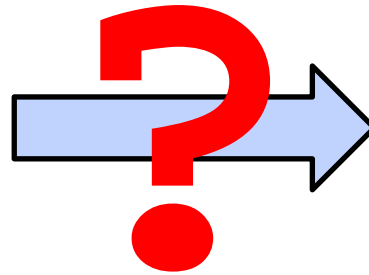
Máquinas "**programáveis**" para manipular informações (ou dados)



Conceitos Básicos: Linguagens de Programação

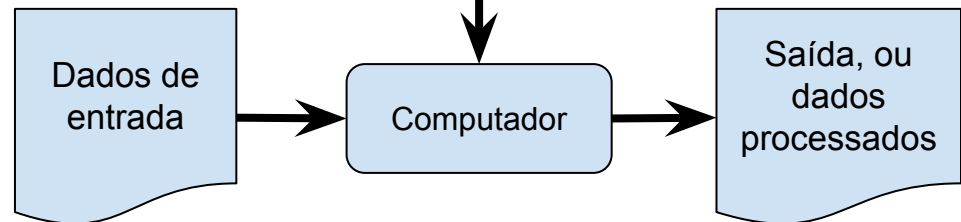
Programa em
linguagem de alto
nível (E.g., C)

```
int func(int a)
{
    return a*113;
}
```

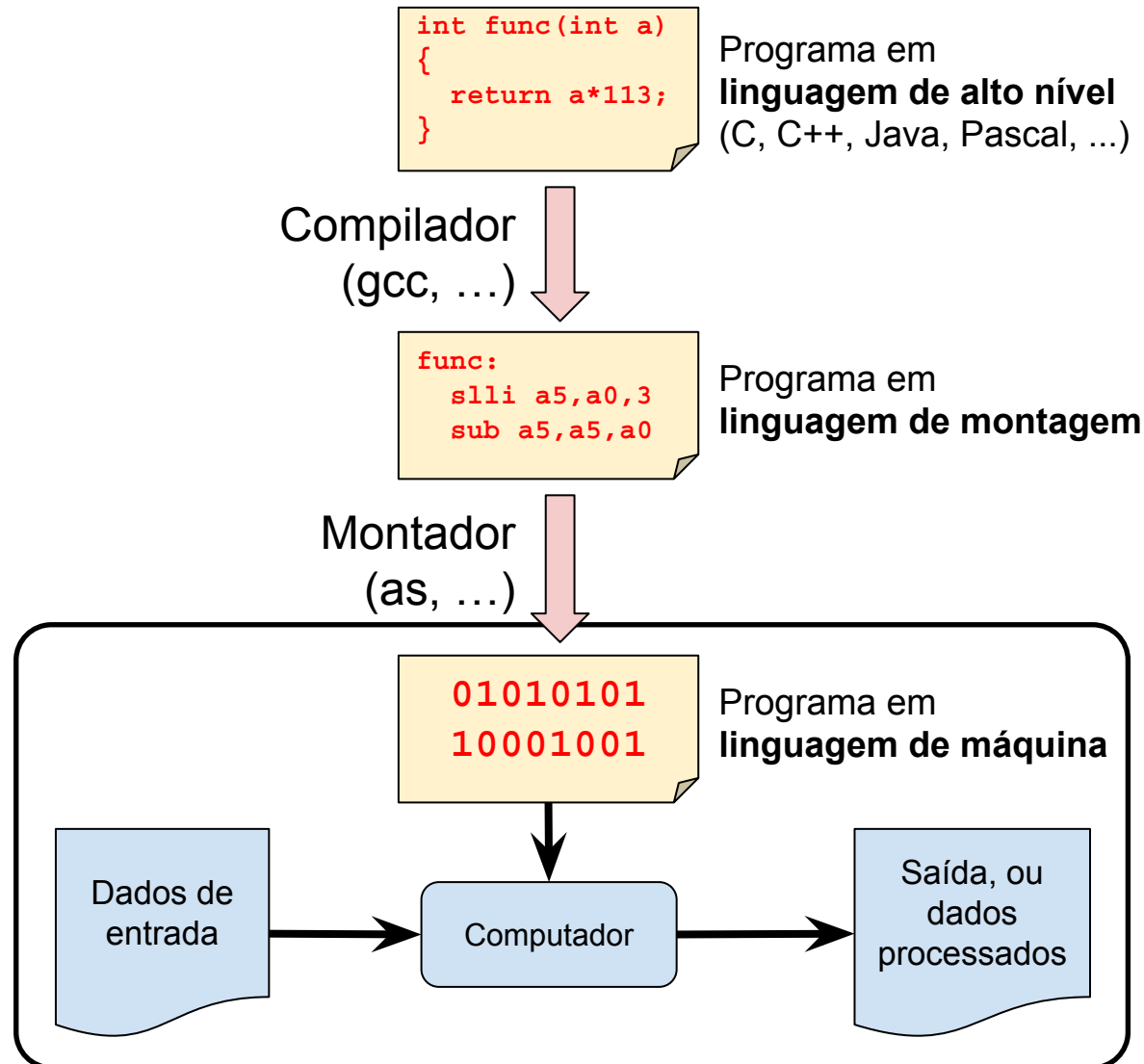


Programa em
linguagem de
máquina

```
01010101
10001001
11100101
01101011
01000101
00001100
01110001
```



Conceitos Básicos: Linguagens de Programação



Conceitos Básicos: Linguagens de Programação

- Laços, variáveis, objetos, ...
- Independente de máquina

```
int func(int a)
{
    return a*113;
}
```

Programa em
linguagem de alto nível
(C, C++, Java, Pascal, ...)

Compilador
(gcc, ...)

- Ling. de baixo nível
- Sequência de instruções, registradores, posições de memória, ...
- Dependente de máquina

```
func:
    slli a5,a0,3
    sub a5,a5,a0
```

Programa em
linguagem de montagem

Montador
(as, ...)

```
01010101
10001001
```

Programa em
linguagem de máquina

- Codificada de forma binária (0s e 1s)
- Dependente de máquina

Conceitos Básicos: Linguagens de Programação

Programa fonte na linguagem C

```
int func_1(int a, int b, int c)
{
    return (a + (113 * b)) * c;
}
```

Compilador
(gcc, ...)

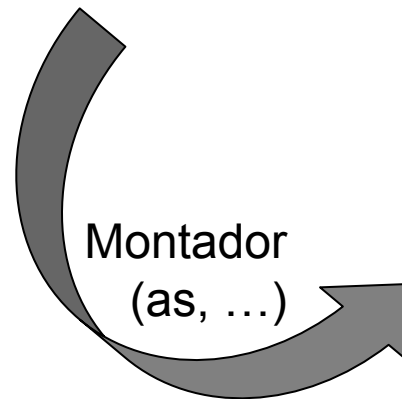
Linguagem de Montagem do x86

```
func_1:
    push    %ebp
    mov     %esp, %ebp
    imul    $113, 12(%ebp), %eax
    add     8(%ebp), %eax
    imul    16(%ebp), %eax
    pop     %ebp
    ret
```

Conceitos Básicos: Linguagens de Programação

Linguagem de Montagem do x86

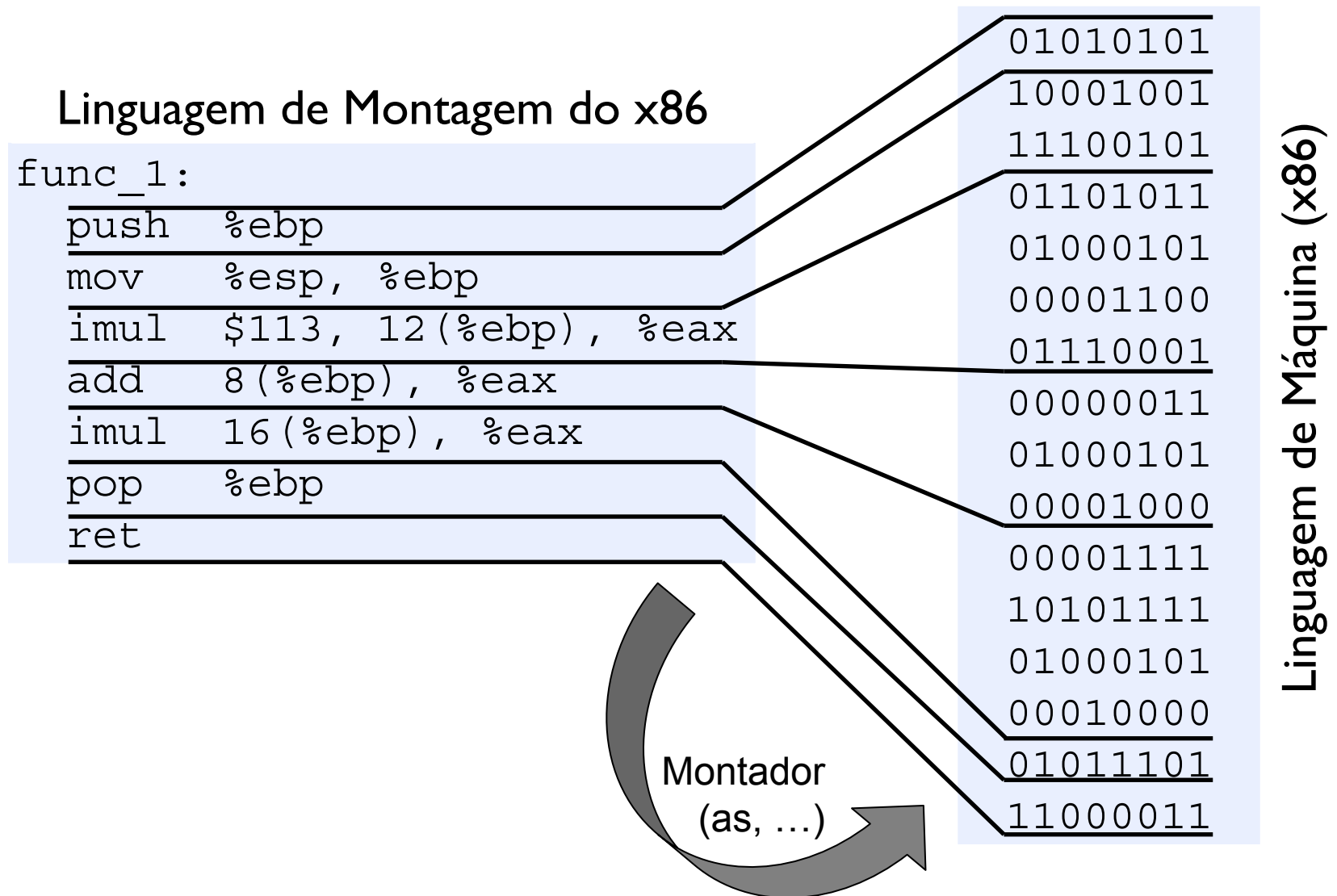
```
func_1:  
  push    %ebp  
  mov     %esp, %ebp  
  imul    $113, 12(%ebp), %eax  
  add     8(%ebp), %eax  
  imul    16(%ebp), %eax  
  pop     %ebp  
  ret
```



```
01010101  
10001001  
11100101  
01101011  
01000101  
00001100  
01110001  
00000011  
01000101  
00001000  
00001111  
10101111  
01000101  
00010000  
01011101  
11000011
```

Linguagem de Máquina (x86)

Conceitos Básicos: Linguagens de Programação



Conceitos Básicos: Linguagens de Programação

Linguagem de Máquina (x86)

```
01010101
10001001
11100101
01101011
01000101
00001100
01110001
00000011
01000101
00001000
00001111
10101111
01000101
00010000
01011101
11000011
```

Código desmontado

```
00000000 <_func_1>:
  0: 55                push    %ebp
  1: 89 e5             mov     %esp,%ebp
  3: 6b 45 0c 71       imul    $0x71,0xc(%ebp),%eax
  7: 03 45 08           add     0x8(%ebp),%eax
  a: 0f af 45 10       imul    0x10(%ebp),%eax
  e: 5d                pop     %ebp
  f: c3                ret
```

desmontador
(objdump, ...)



Conceitos Básicos: Linguagens de Programação

Programa fonte na linguagem C

```
int func_1(int a, int b, int c)
{
    return (a + (113 * b)) * c;
}
```

Linguagem de Montagem do **ARM**

```
_func_1:
    rsb     r3, r1, r1, asl #3
    add     r1, r1, r3, asl #4
    add     r1, r1, r0
    mul     r0, r2, r1
    bx      lr
```

Linguagem de Montagem do **RISC-V**

```
func_1:
    slli    a5, a1, 3
    sub     a5, a5, a1
    slli    a5, a5, 4
    add     a5, a5, a1
    add     a0, a5, a0
    mul     a0, a0, a2
    ret
```

Linguagem de Montagem do **x86**

```
func_1:
    push    %ebp
    mov     %esp, %ebp
    imul    $113, 12(%ebp), %eax
    add     8(%ebp), %eax
    imul    16(%ebp), %eax
    pop     %ebp
    ret
```

Leitura complementar

Capítulo 1.2 do livro do Patterson e Hennessy
(Computer Organization and Design)