Insper

Elementos de Sistemas

Aula 10 – Programação em Assembly

“A carta que escrevi hoje é mais longa que o usual pois não tive tempo de fazer ela mais curta."
“Je vous écris une longue lettre parce que je n'ai pas le temps d'en écrire une courte."
***Blaise Pascal (1623–1662)* matemático francês**

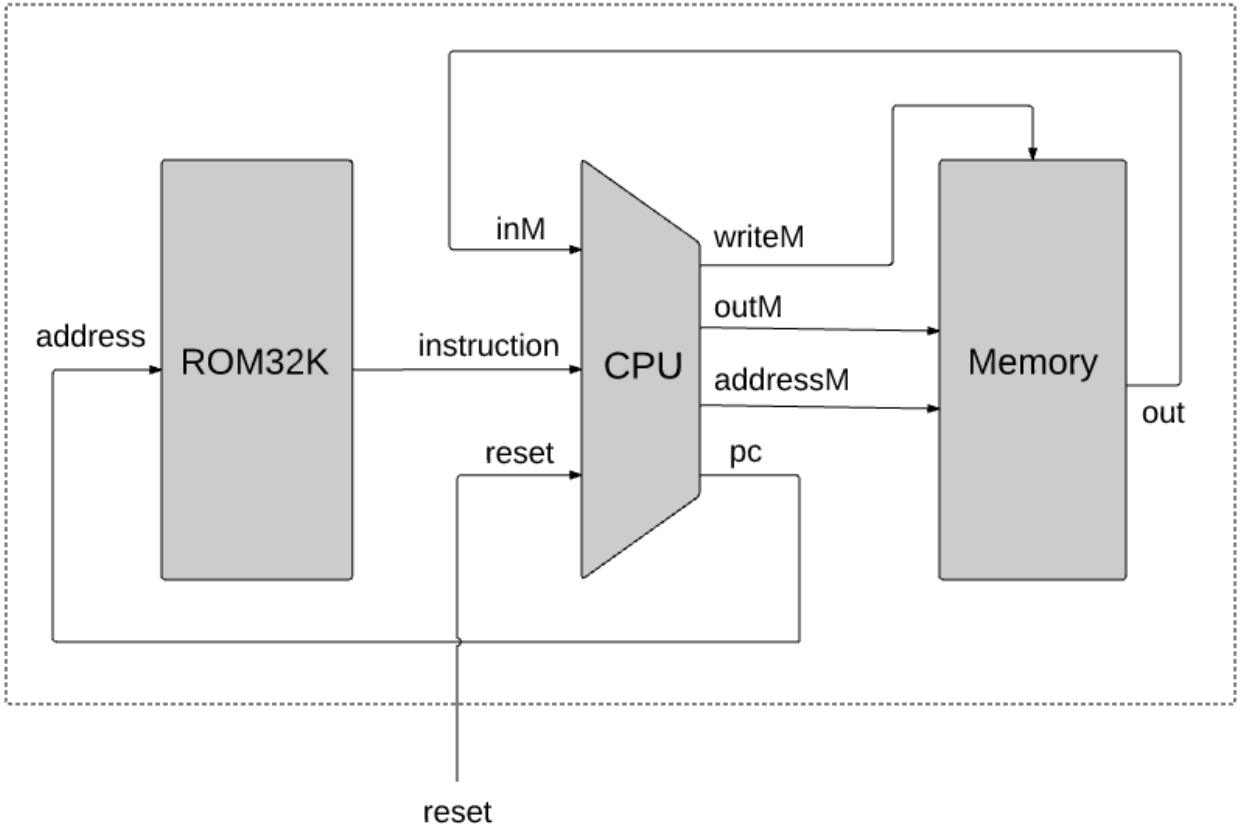
Engenharia - Luciano Soares <lpsoares@insper.edu.br>

Objetivo da Aula

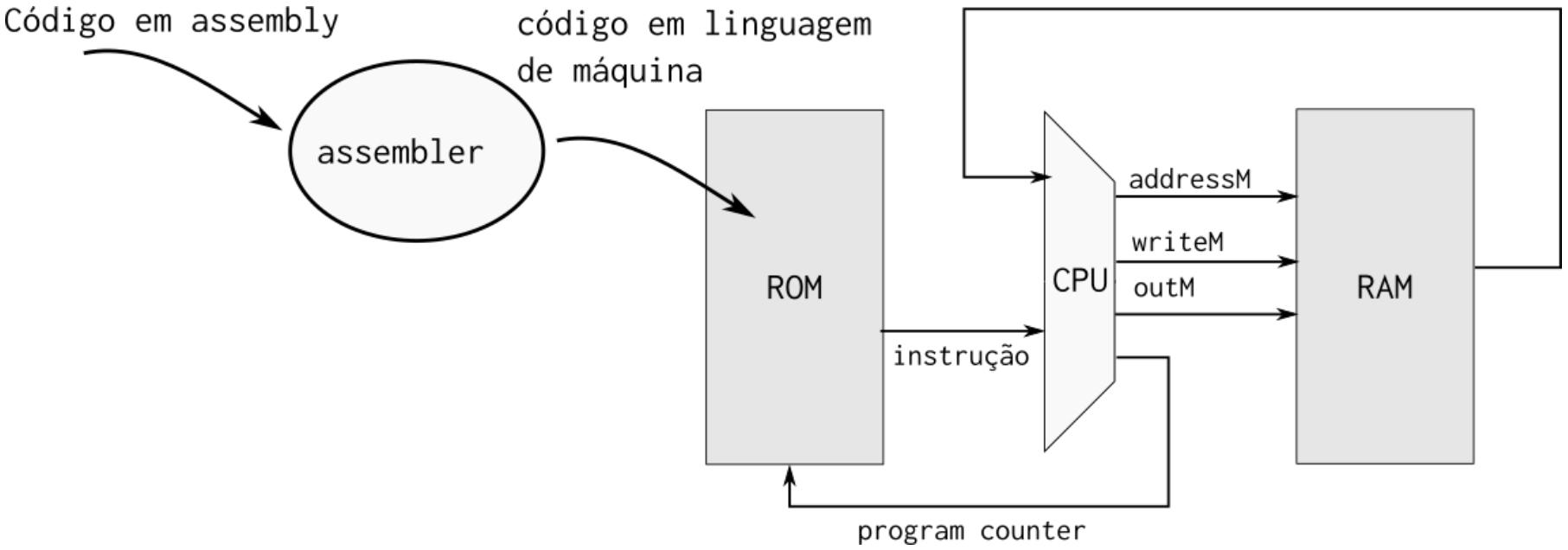
- Mapear Dispositivos em Memória;
- Programar em Assembly.
- Jump

Conteúdos: Interfaces e Periféricos, Hierarquia de Memória;

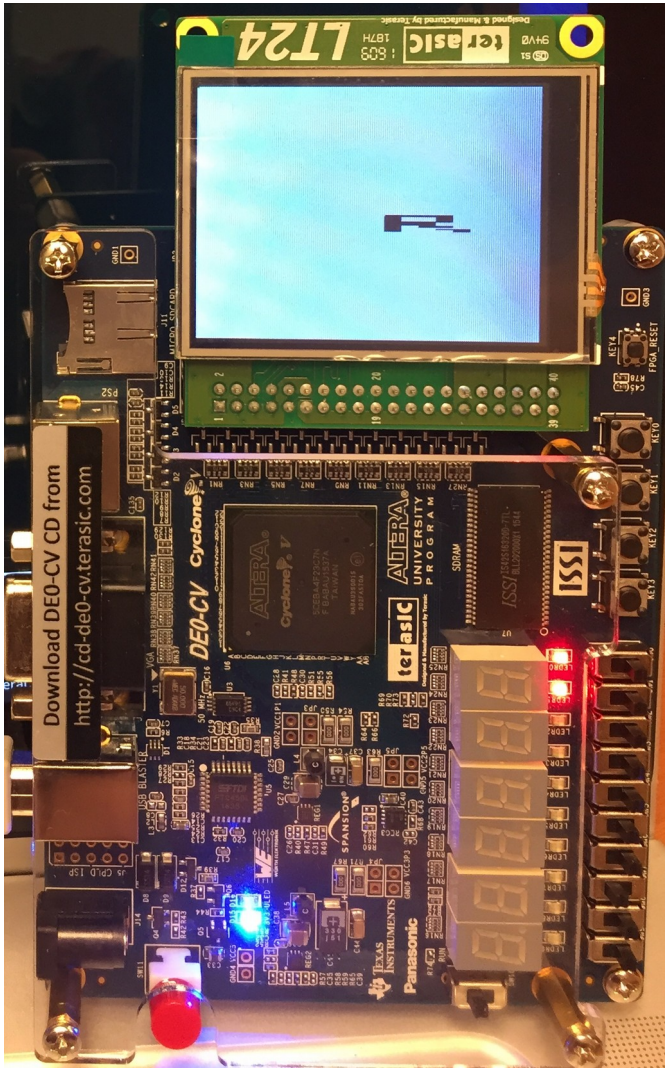
Z01 - Computador



Insper


Código em assembly
assembler
código em linguagem
de máquina
ROM
instrução
CPU
addressM
writeM
outM
RAM
program counter

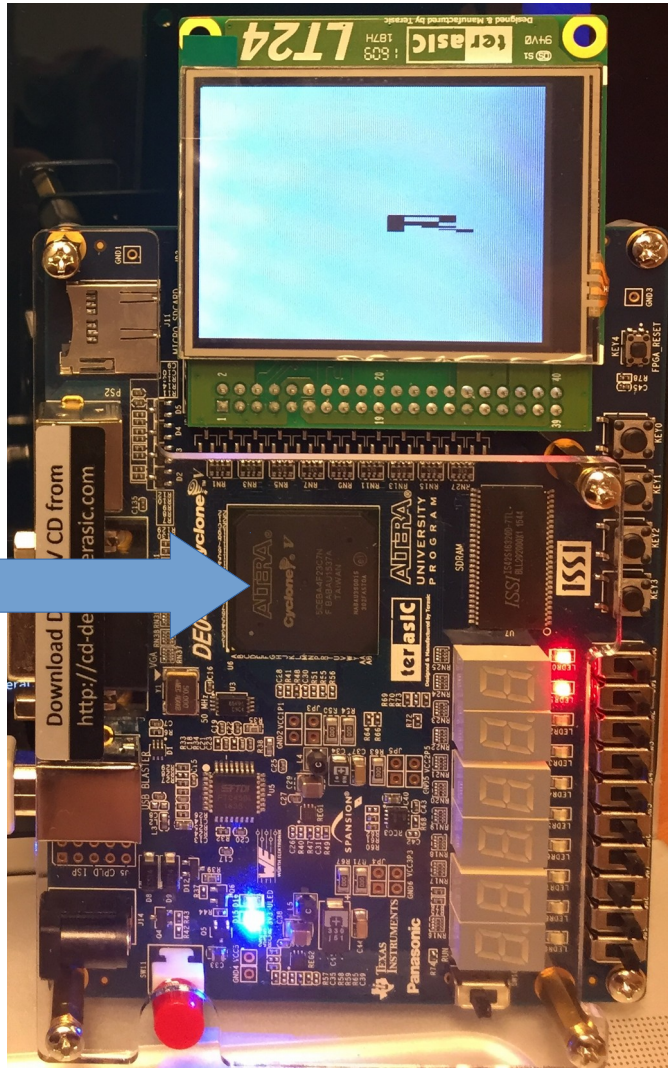
Z01 - Computador



Insper

Z01 – Computador - Periféricos

Z01



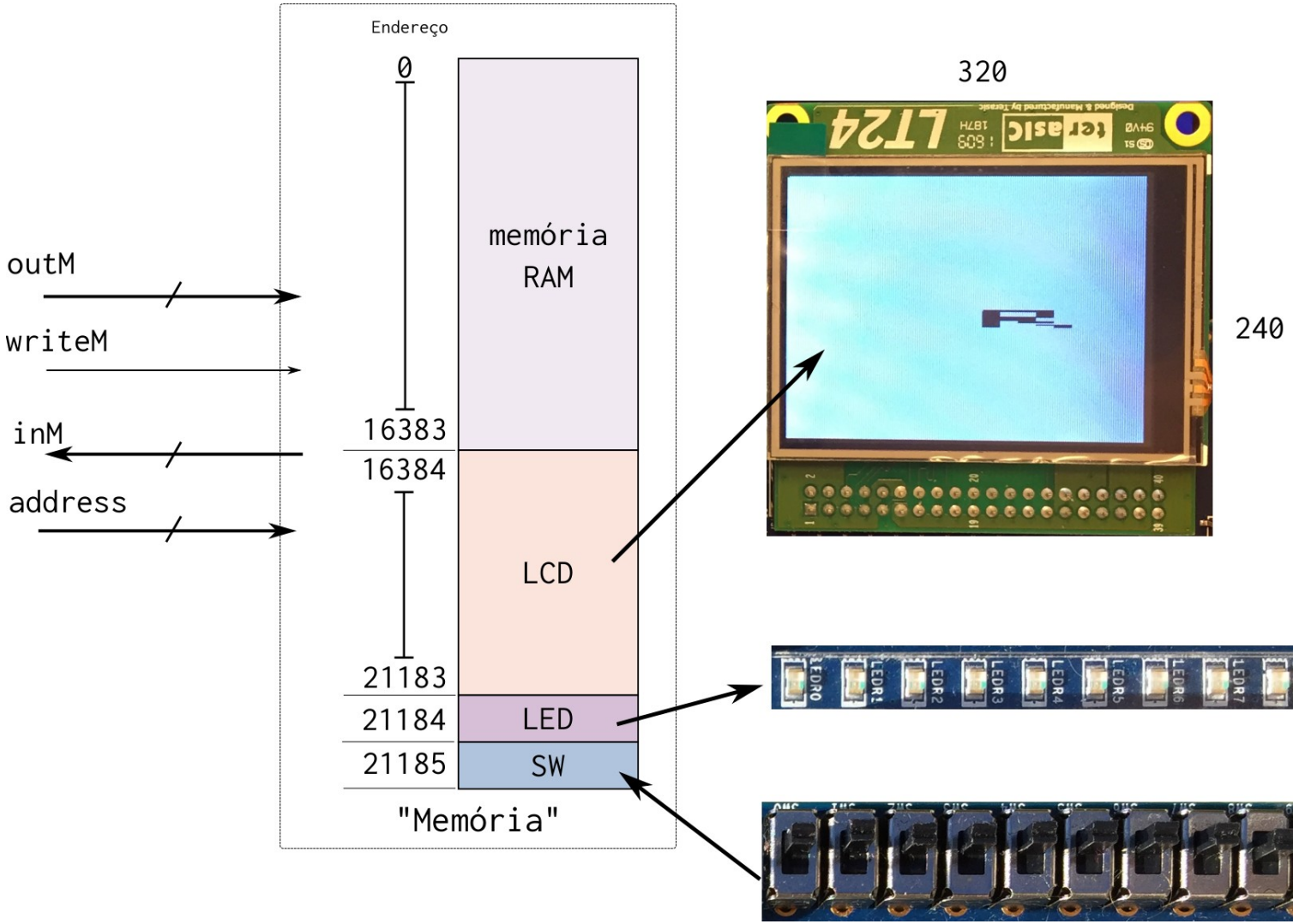
LCD – Output

LEDs – Output
Chaves – Input

Insper

I/O Mapeado em Memória

Z01 – I/O Mapeado em memória



Insper

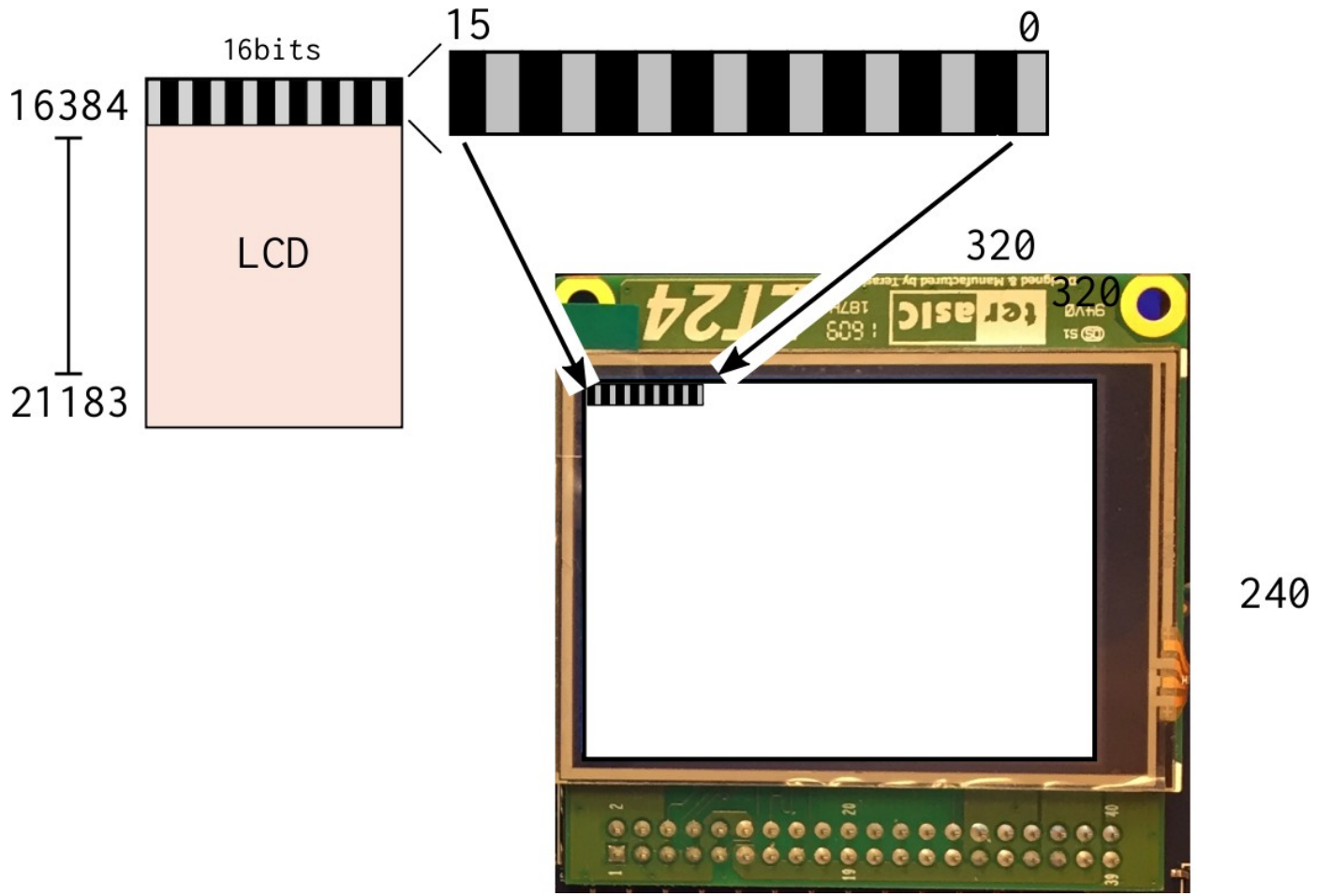
Z01 – RAM

Read/Write

- 16384 endereços
- Alguns endereços possuem nomes (labels) :

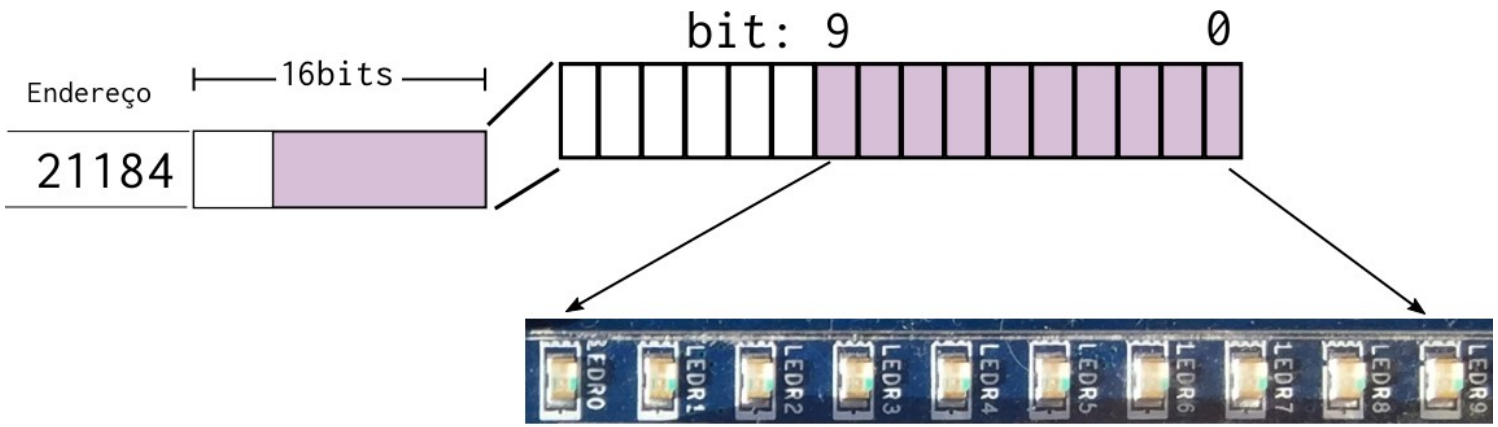
Label	Endereço
SP	0
LCL	1
ARG	2
THIS	3
THAT	4
R0-R15	0-15

Z01 – LCD - 320x240 pxs



Insper

Z01 – LEDs (10 leds)

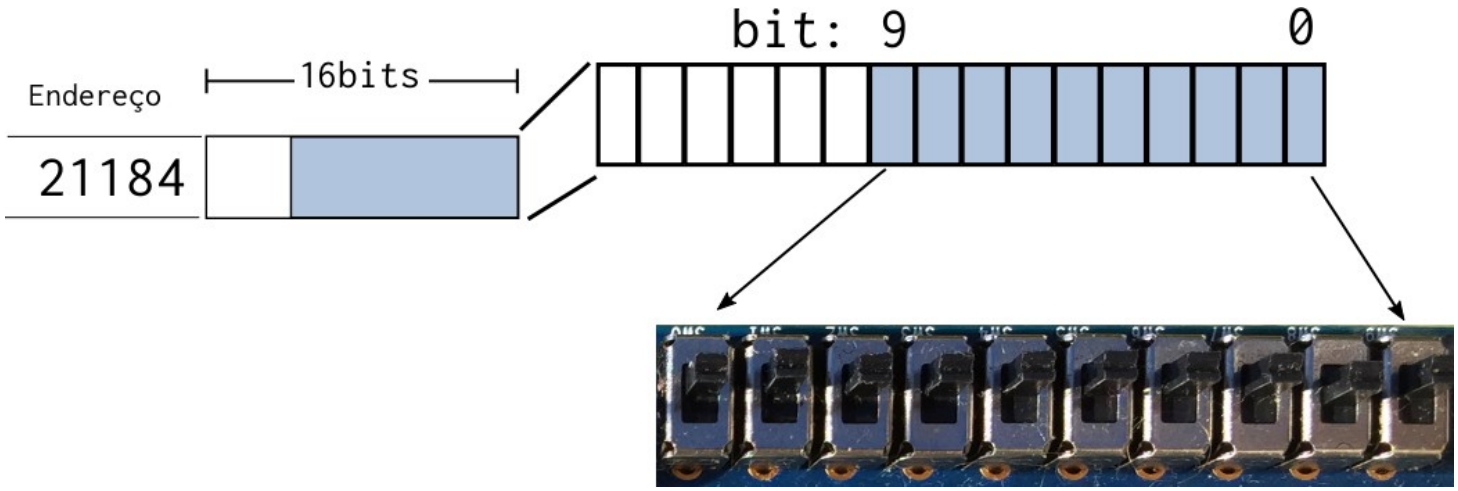


Acende

Apaga

Insper

Z01 – SWs (10 chaves)



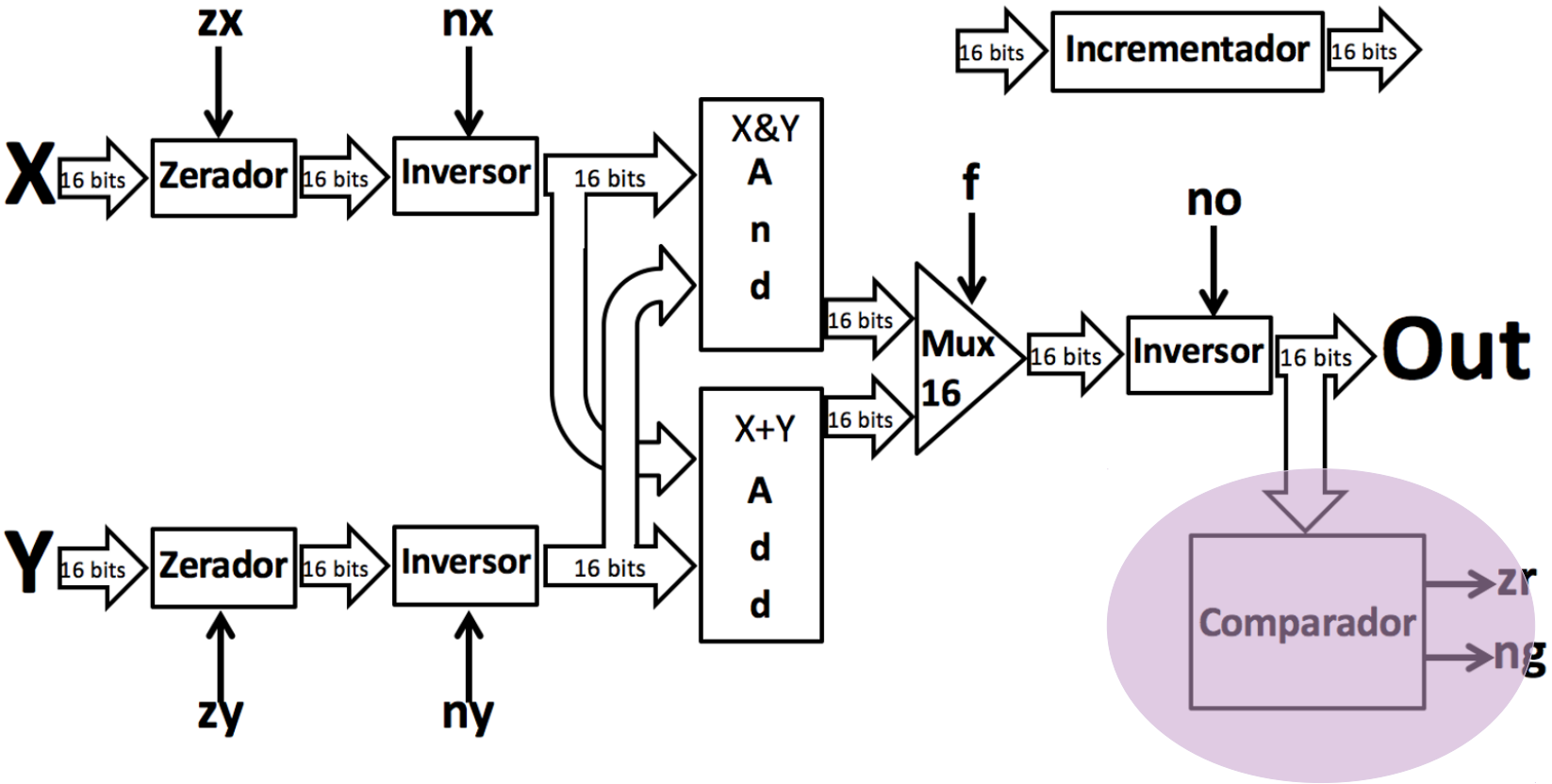
Insper

JUMPs

Z01 – JUMPS

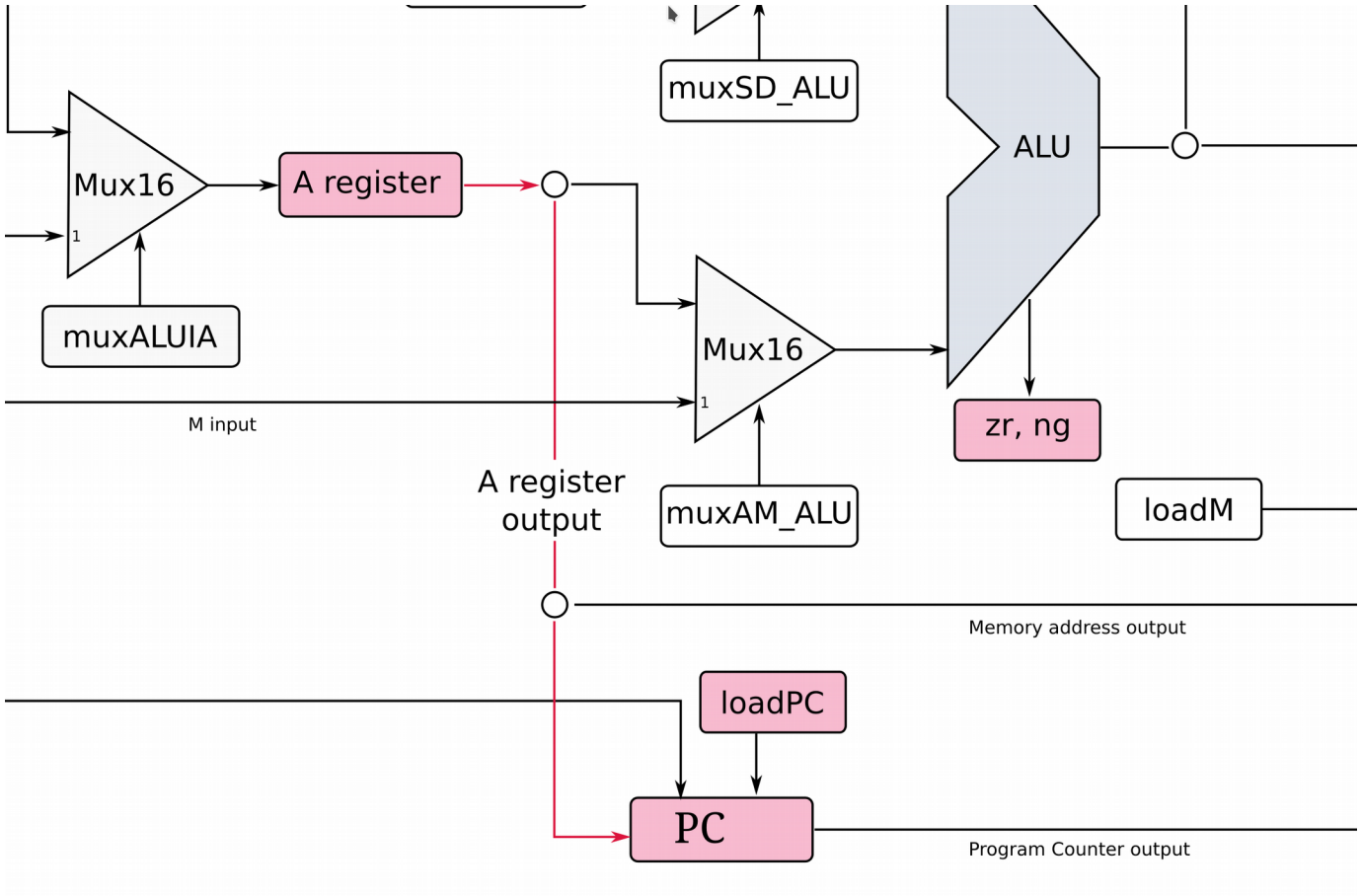
- Forma de executar condição
- Possibilita executarmos um programa de forma não linear
- Verificar o valor de um registrador e dependendo da condição faz o salto.

Z01 – JUMPS – Como funciona ?



Insper

Z01 – JUMPS – Como funciona ?



Insper

Z01 – JUMPS – exemplo 1 Incondicional (jmp)

```
S = 0
while(1):
    S = S + 1
```

Z01 – JUMPS – exemplo 1 - Incondicional (jmp)

```
1. leaw $0, %A          ; Carrega 0 em A
2. movw %A, %S          ; Move A para S

ADD:                    ; Label para saltar
3. incw %S              ; Incrementa S
4. leaw $ADD, %A        ; Carrega endereço do label ADD
                        ; (3 no caso)
5. jmp                  ; Salto incondicional
6. nop                  ; No-Operation
                        ; (necessário após jump)
```

Z01 – JUMPS – exemplo 1 - Incondicional (jmp)

```
1. leaw $0, %A          ; Carrega 0 em A
2. movw %A, %S          ; Move A para S

3. incw %S              ; Incrementa S
4. leaw $3, %A          ; Carrega endereço do

5. jmp                  ; Salto incondicional
6. nop                  ; No-Operation
```

Z01 – JUMPS – exemplo 2 condicional

```
S = 0
While(D < 2):
    S = S + 1
```

Z01 – JUMPS – exemplo 2 - condicional

```
 leaw $0, %A
 movw %A, %S                    ; Carrega 0 em S
 leaw $2, %A
 movw %A, %D                    ; Carrega 2 em D

WHILE:
  leaw $END, %A
  je %D                          ; Salta se D == 0
  nop

  incw %S                        ; Incrementa S
  decw %D                        ; Decrementa D
  leaw $WHILE, %A
  jmp                            ; Salto incondicional
  nop                            ; No-Operation
                                 ; (necessário após jump)

END:                             ; FIM
```

Mudança assembly jumps

JG – Desvia Execução se Maior que Zero

```
jg reg
```

Descrição: A instrução *jg* desvia, o fluxo de execução, para o endereço armazenado em %A, somente se o valor do reg. for maior que zero.

Assembly: jg %S

JGE – Desvia Execução se Maior Igual a Zero

```
jge reg
```

Descrição: A instrução *jge* faz um desvio, no fluxo de execução, para o endereço armazenado em %A, somente se o valor do reg. for maior ou igual a zero.

Assembly: jge %S

JL – Desvia Execução se Menor que Zero

```
jl reg
```

Descrição: A instrução *jl* faz desvio, no fluxo de execução, para o endereço armazenado em %A, somente se o valor do reg. for menor que zero.

Assembly: jl %S

Insper

www.insper.edu.br