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**

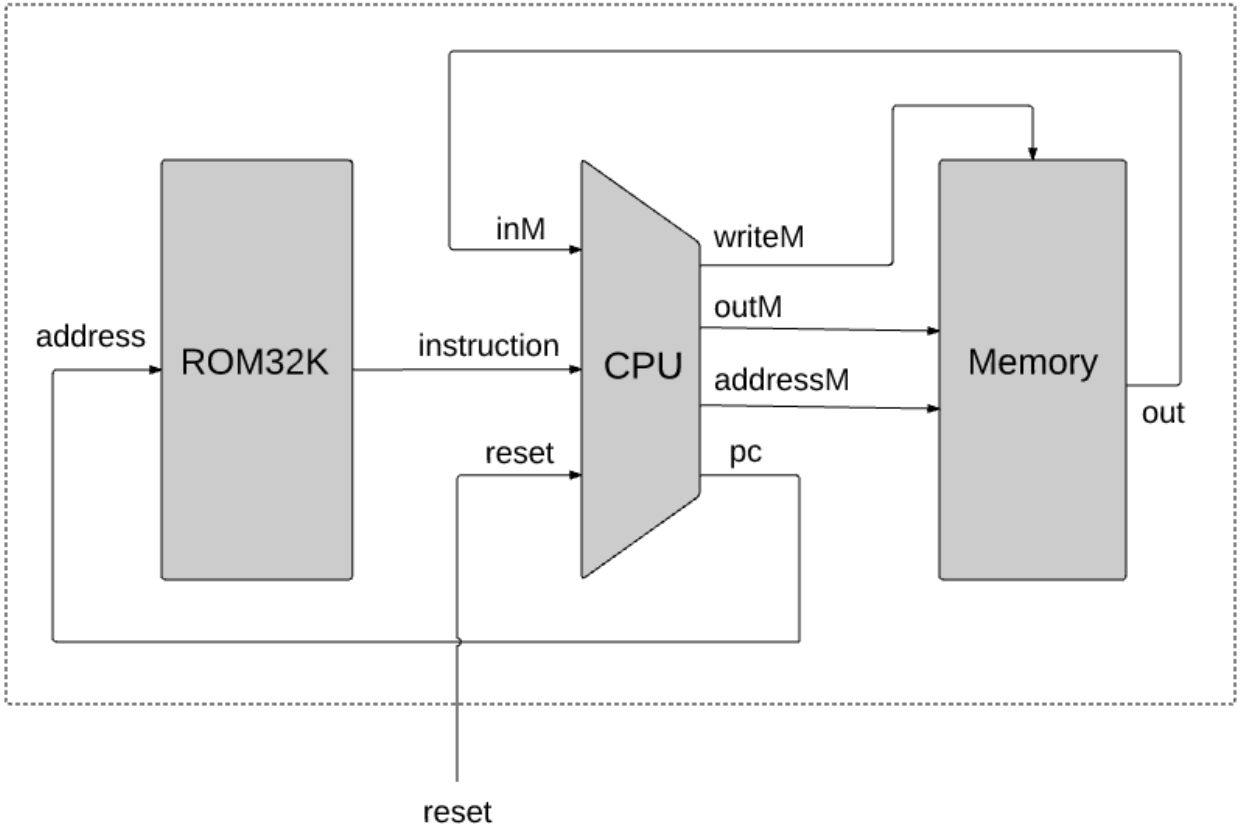
Engenharia - Luciano Soares <lpsoares@insper.edu.br>

Objetivo da Aula

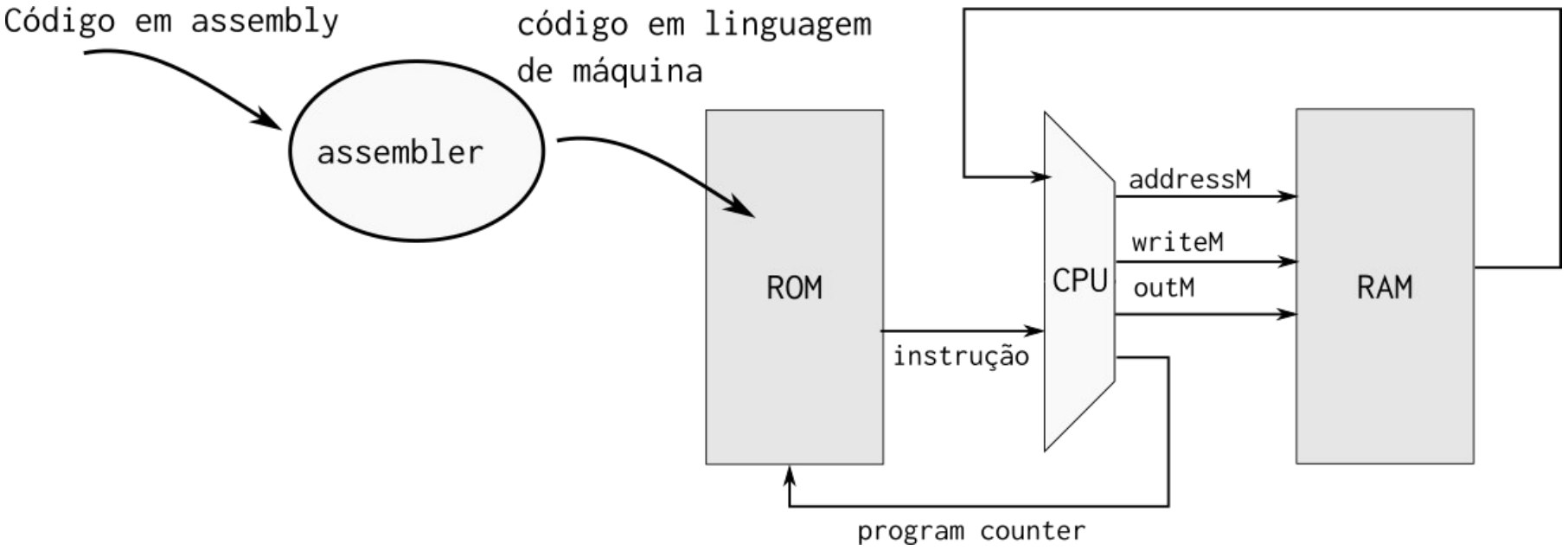
- Mapear Dispositivos em Memória;
- Programar em Assembly.

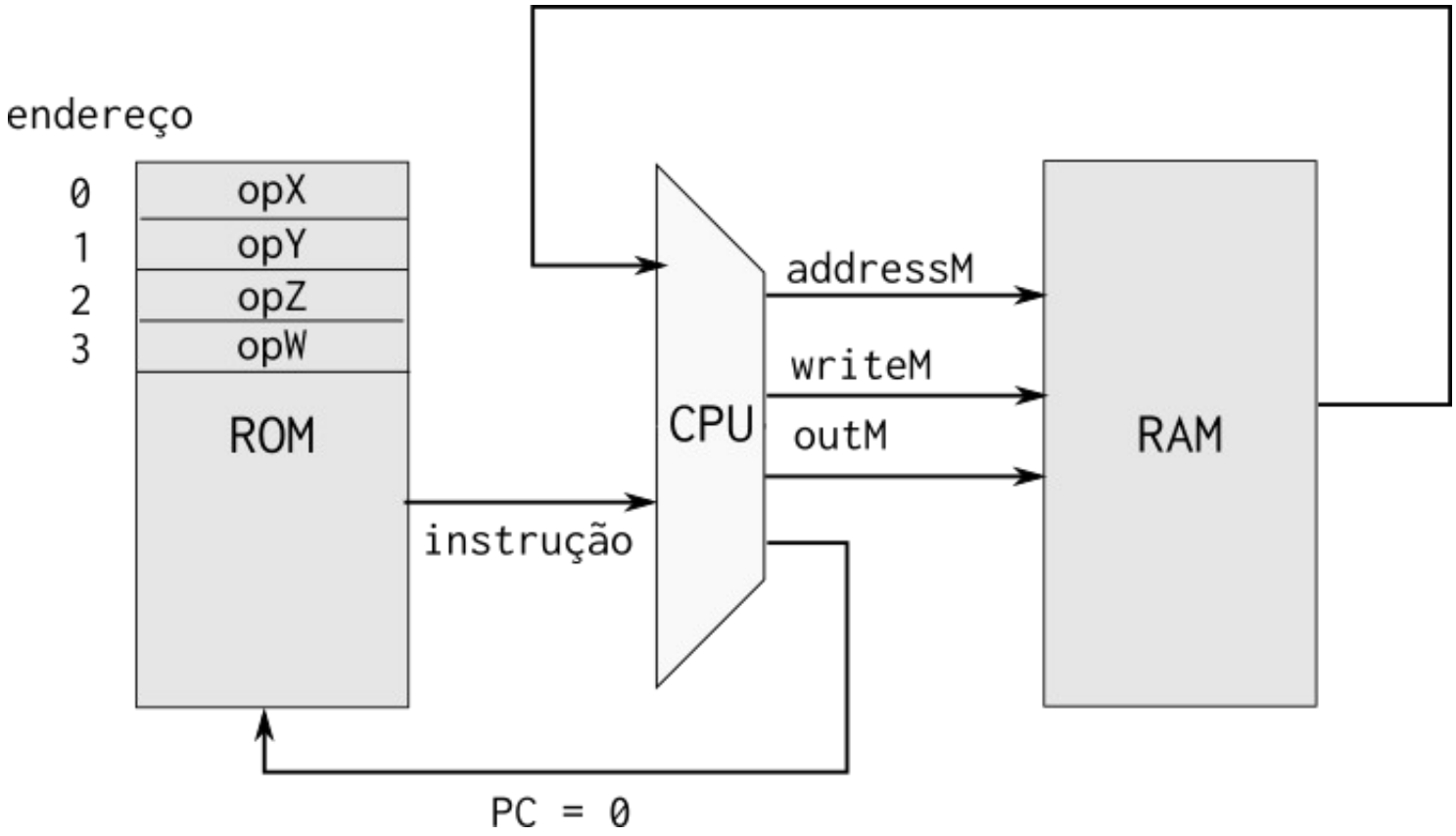
Conteúdos: Interfaces e Periféricos, Hierarquia de Memória;

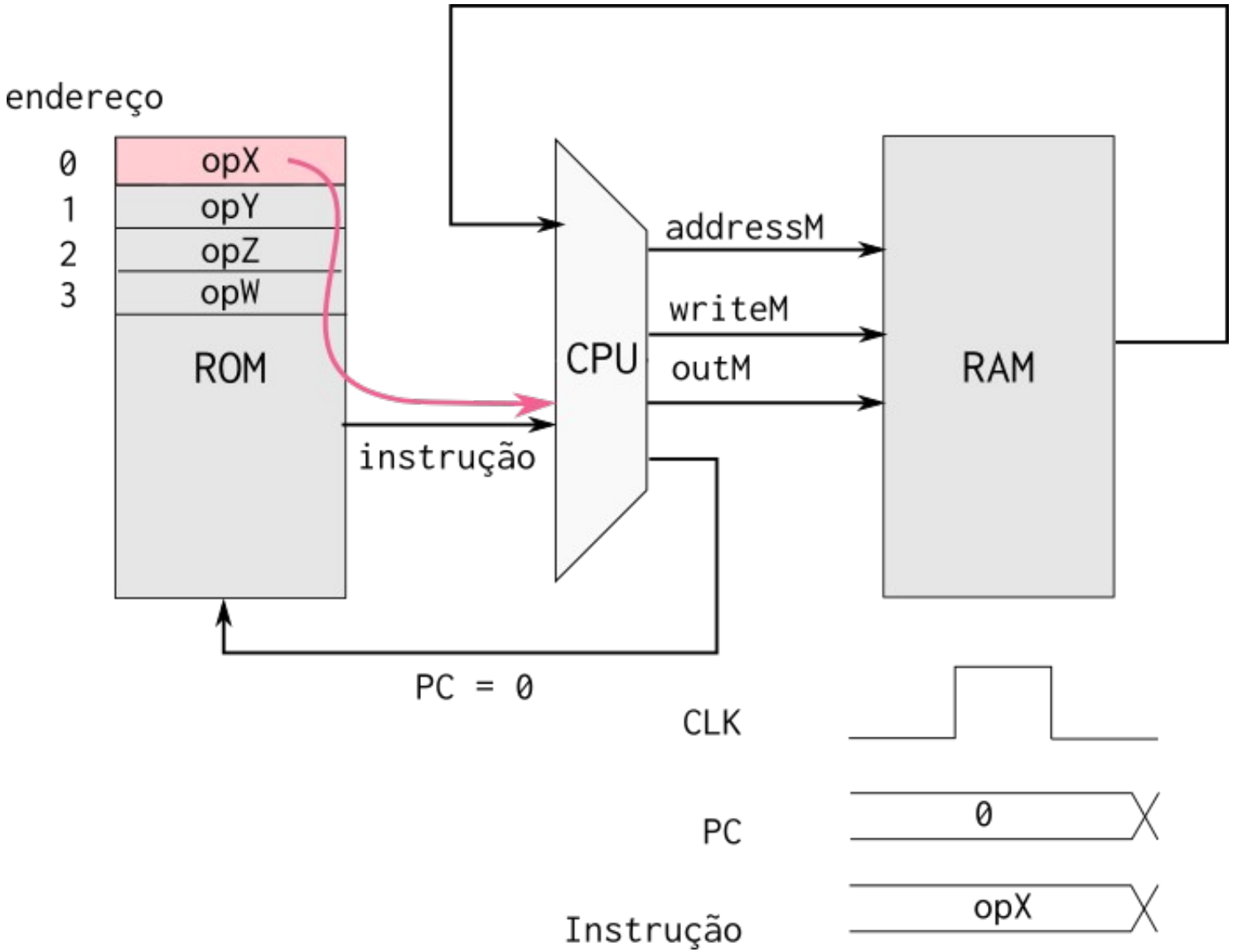
Z01 - Computador

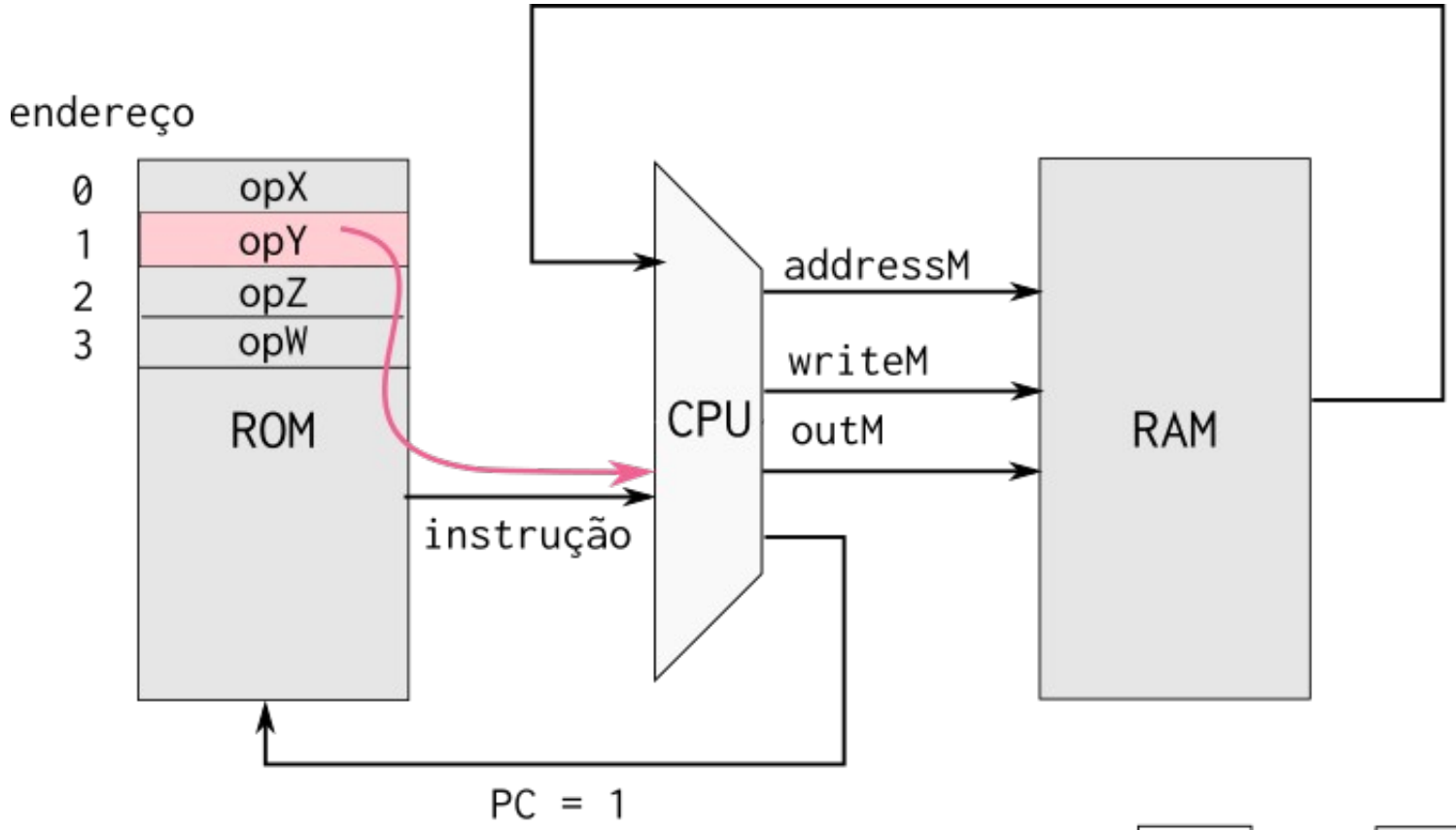


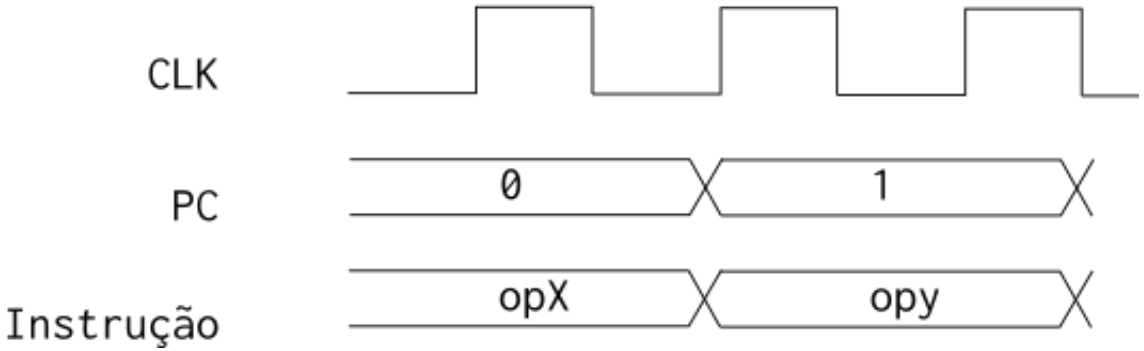
Insper


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


endereço
0
1
2
3
opX
opY
opZ
opW
ROM
instrução
CPU
addressM
writeM
outM
RAM
PC = 0


endereço
0
1
2
3
opX
opY
opZ
opW
ROM
instrução
CPU
addressM
writeM
outM
RAM
PC = 0
CLK
PC
0
Instrução
opX


endereço
0
1
2
3
opX
opY
opZ
opW
ROM
CPU
addressM
writeM
outM
RAM
instrução
PC = 1


CLK
PC
0
1
Instrução
opX
opy

Insper

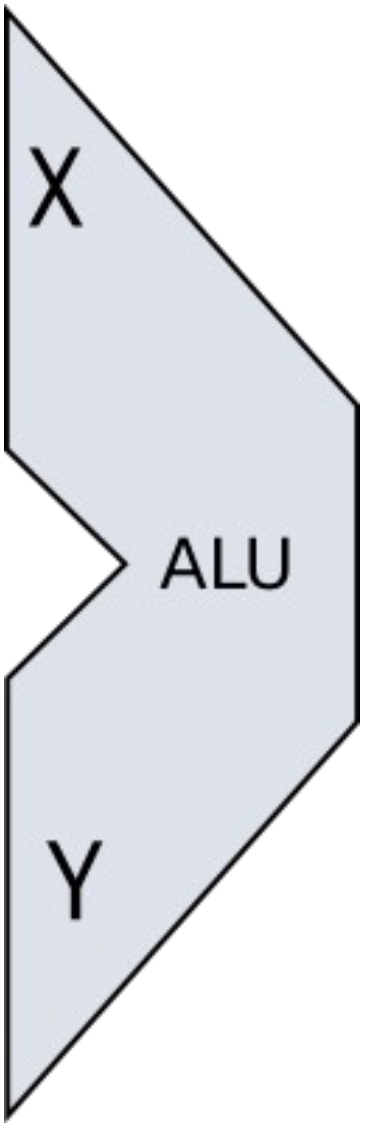
Instruções - Assembly

- Maneira de controlar o Hardware via software

- Cada linha de código (em assembly) é uma instrução para o computador

- Existem alguns tipos de instrução :
    - Aritmética (+, -, &, |, ….)
    - Movimentação de dados (RAM[0] = 1)
    - Jumps (para implementar if, while, …)
    - Carregamento Efetivo do Endereço

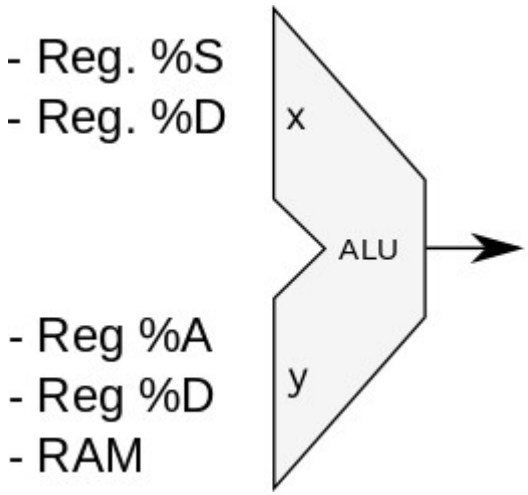
Z01 – ULA



Insper

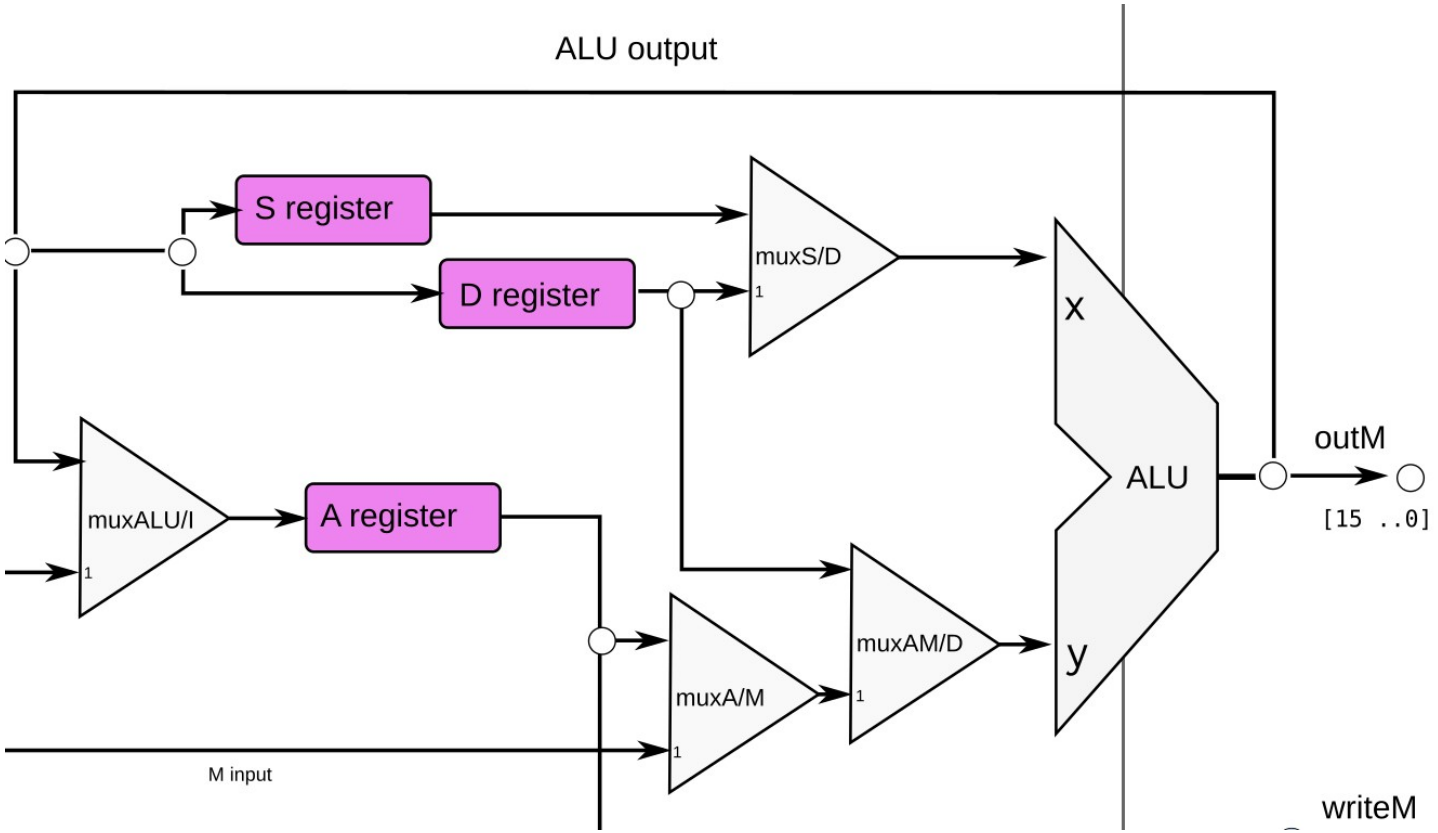
Z01 – ULA

Entradas e saídas da ULA



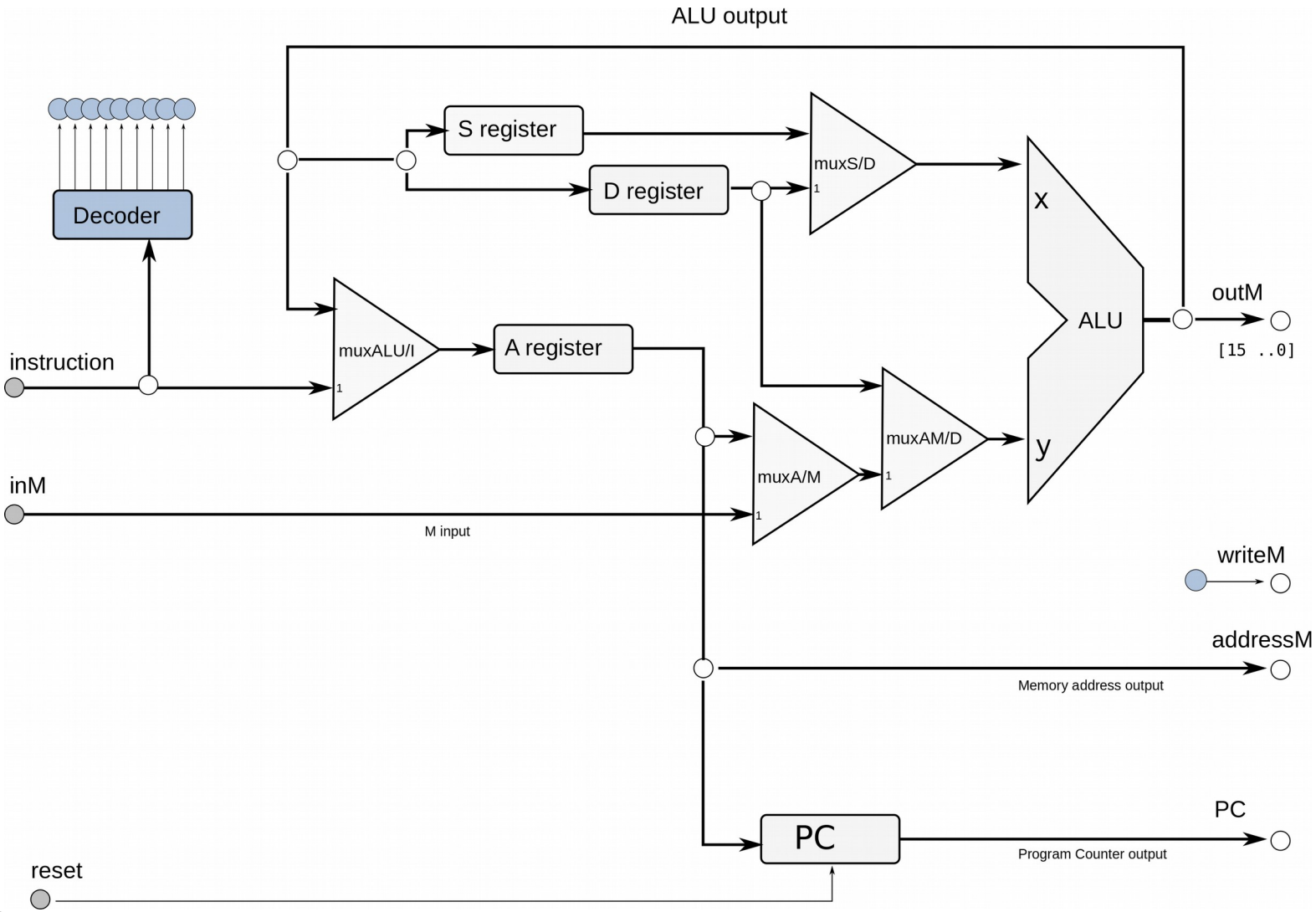
Insper

Z01 – Registradores ?



Insper

Z01 - CPU



Insper

Instruções - Assembly

- Classificamos todas as instruções em dois tipos :

- **A-Instruction**

- Permite que carreguemos uma constante definida no programa para dentro da CPU

leaw $5, %A

- **C-Instruction**

- todas as outras operações que manipulam a CPU

movw %A, %S

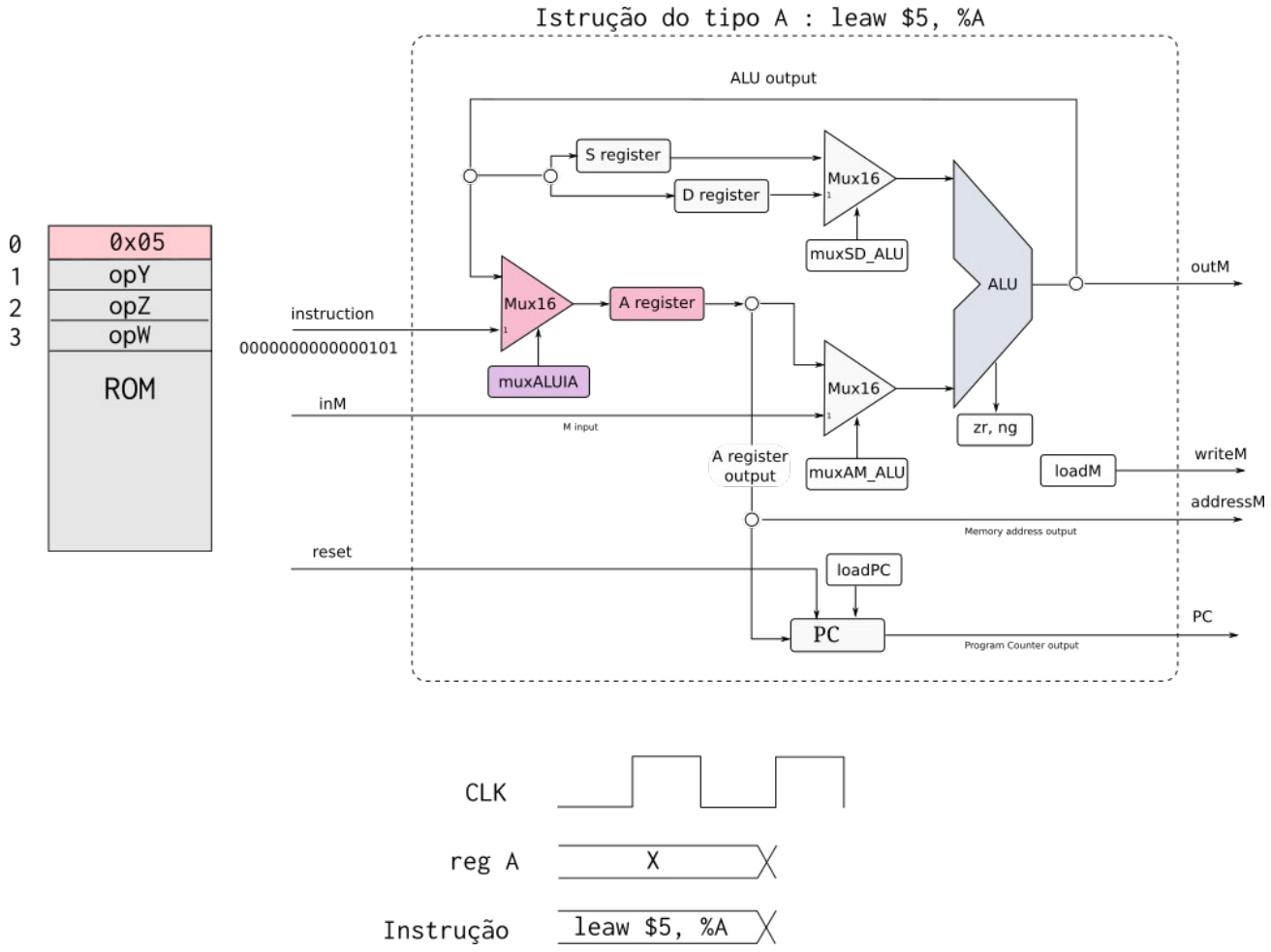
Exemplo – Instrução tipo A

leaw $5, %A

Carrega no registrador A a constante 5

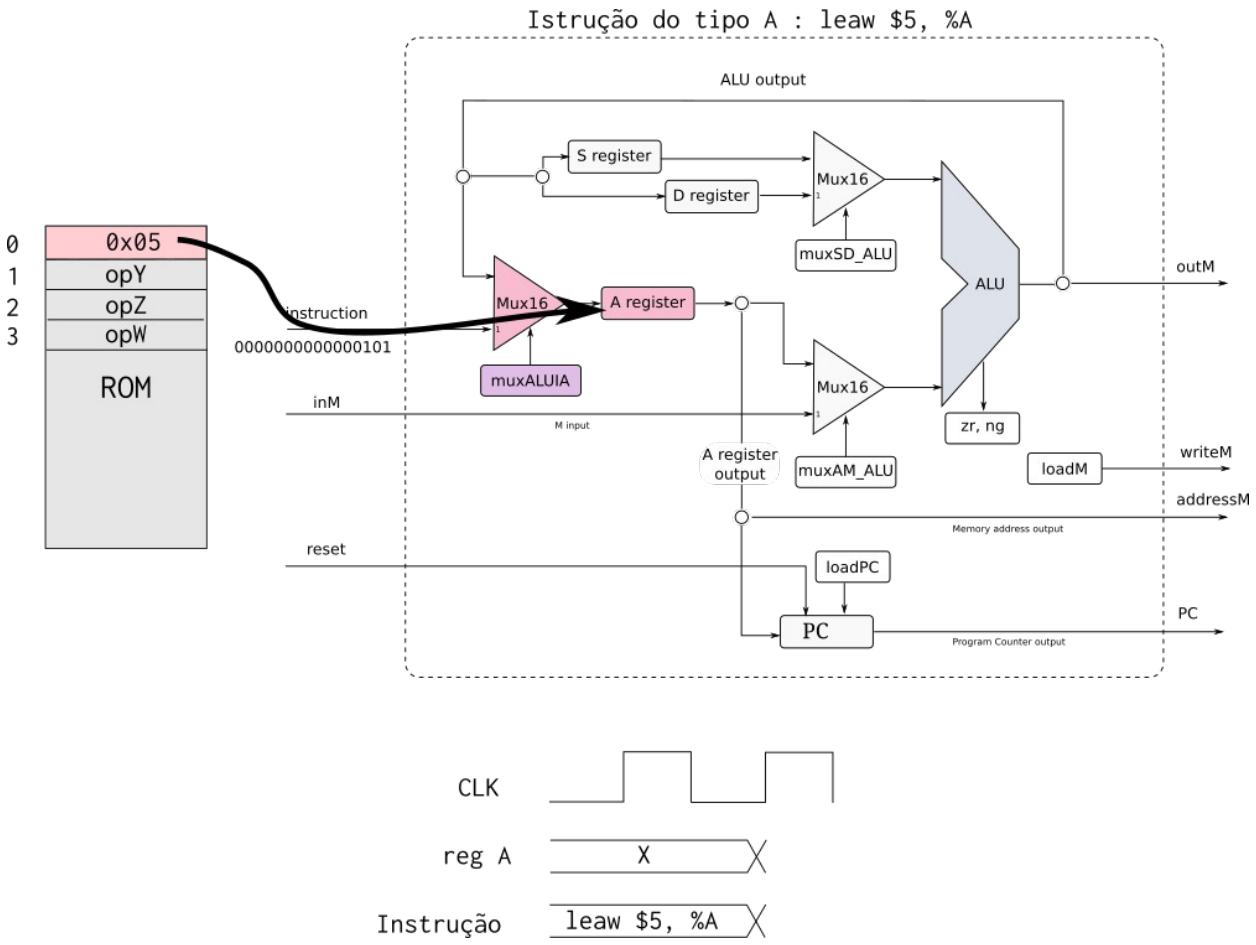
Exemplo – leaw $5, %A

ESSE EXEMPLO É PARA UMA VERSÃO ATIGA DO HW! ONDE NÃO POSSUI O MUXAM_D



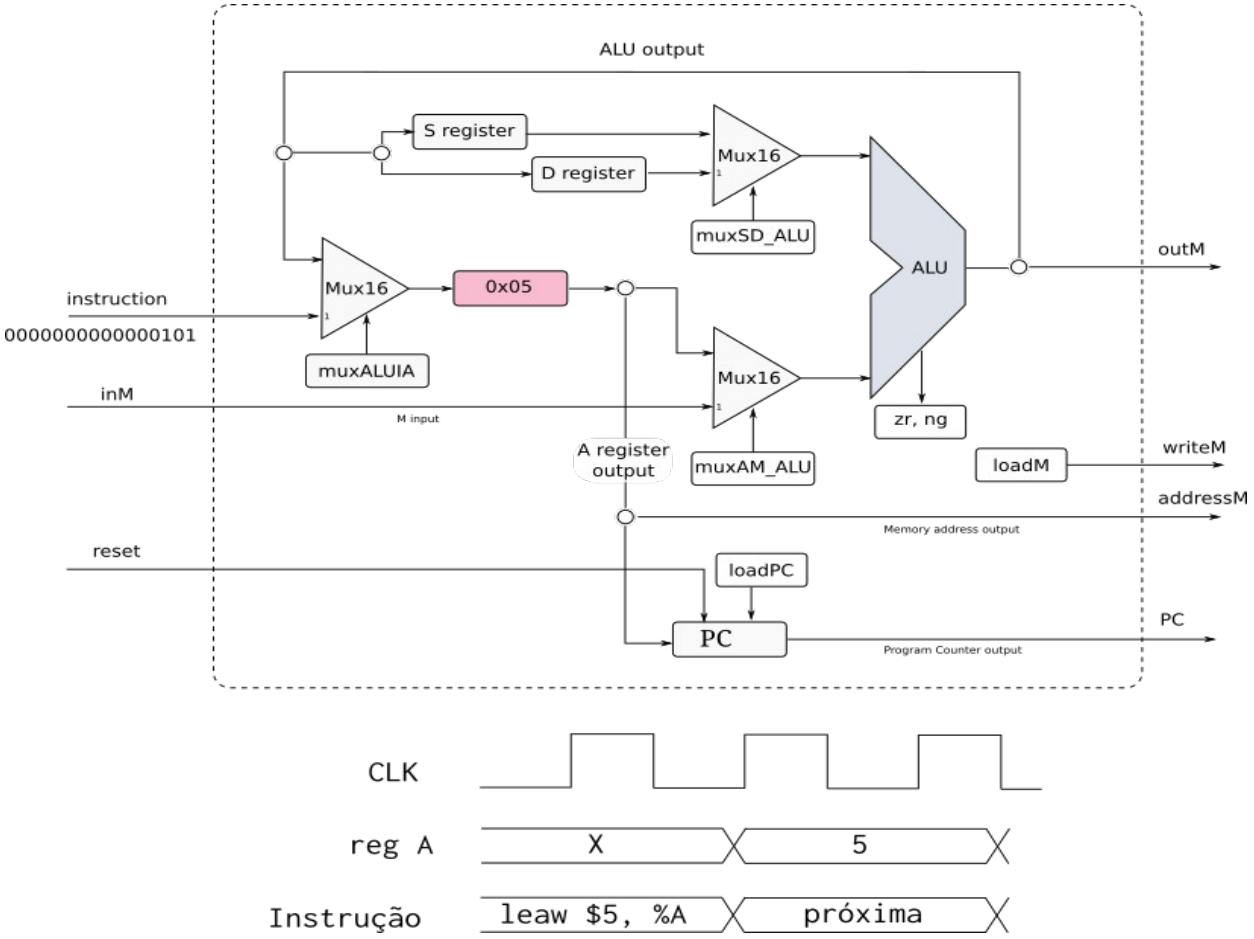
Exemplo – leaw $5, %A

ESSE EXEMPLO É PARA UMA VERSÃO ATIGA DO HW! ONDE NÃO POSSUI O MUXAM_D



Exemplo – leaw $5, %A

ESSE EXEMPLO É PARA UMA VERSÃO ATIGA DO HW! ONDE NÃO POSSUI O MUXAM_D



Insper

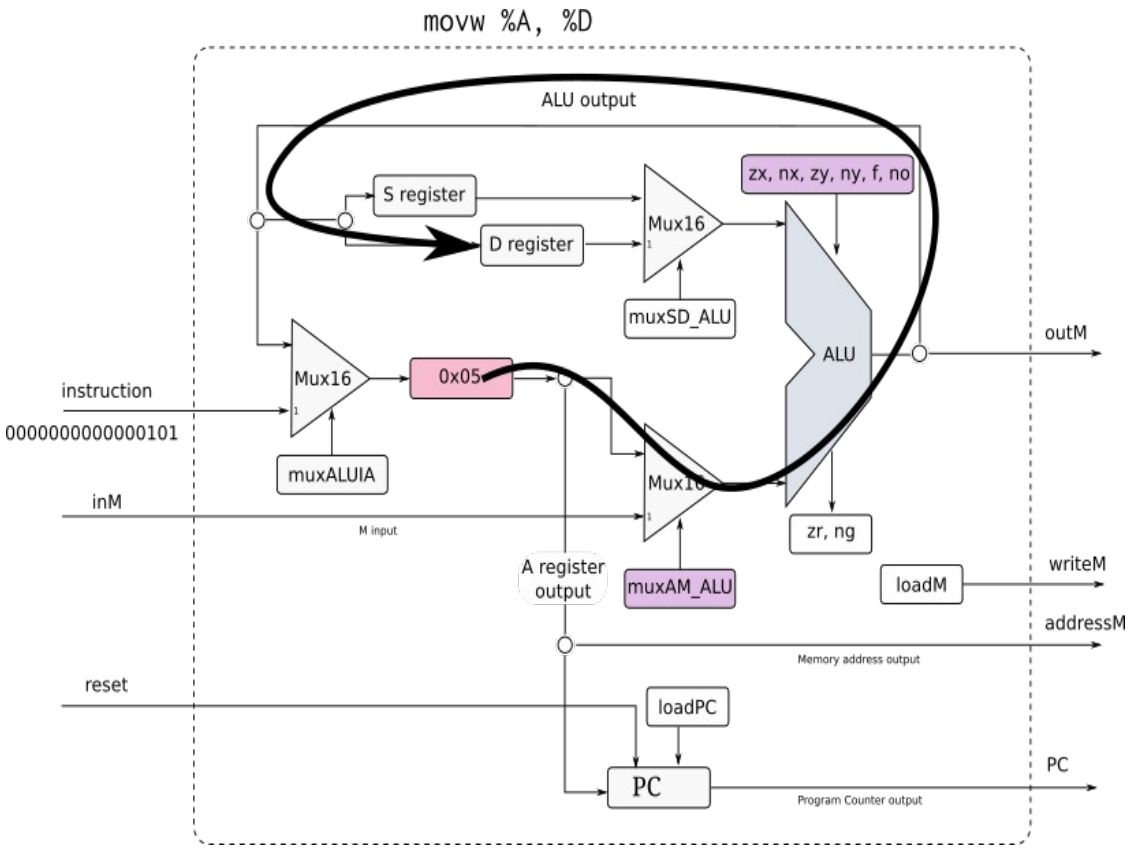
Exemplo – Instrução tipo D

movw %A, %D

Carrega no registrador D o valor do registrador A

Exemplo – movw %A, %D

ESSE EXEMPLO É PARA UMA VERSÃO ATIGA DO HW! ONDE NÃO POSSUI O MUXAM_D



Insper

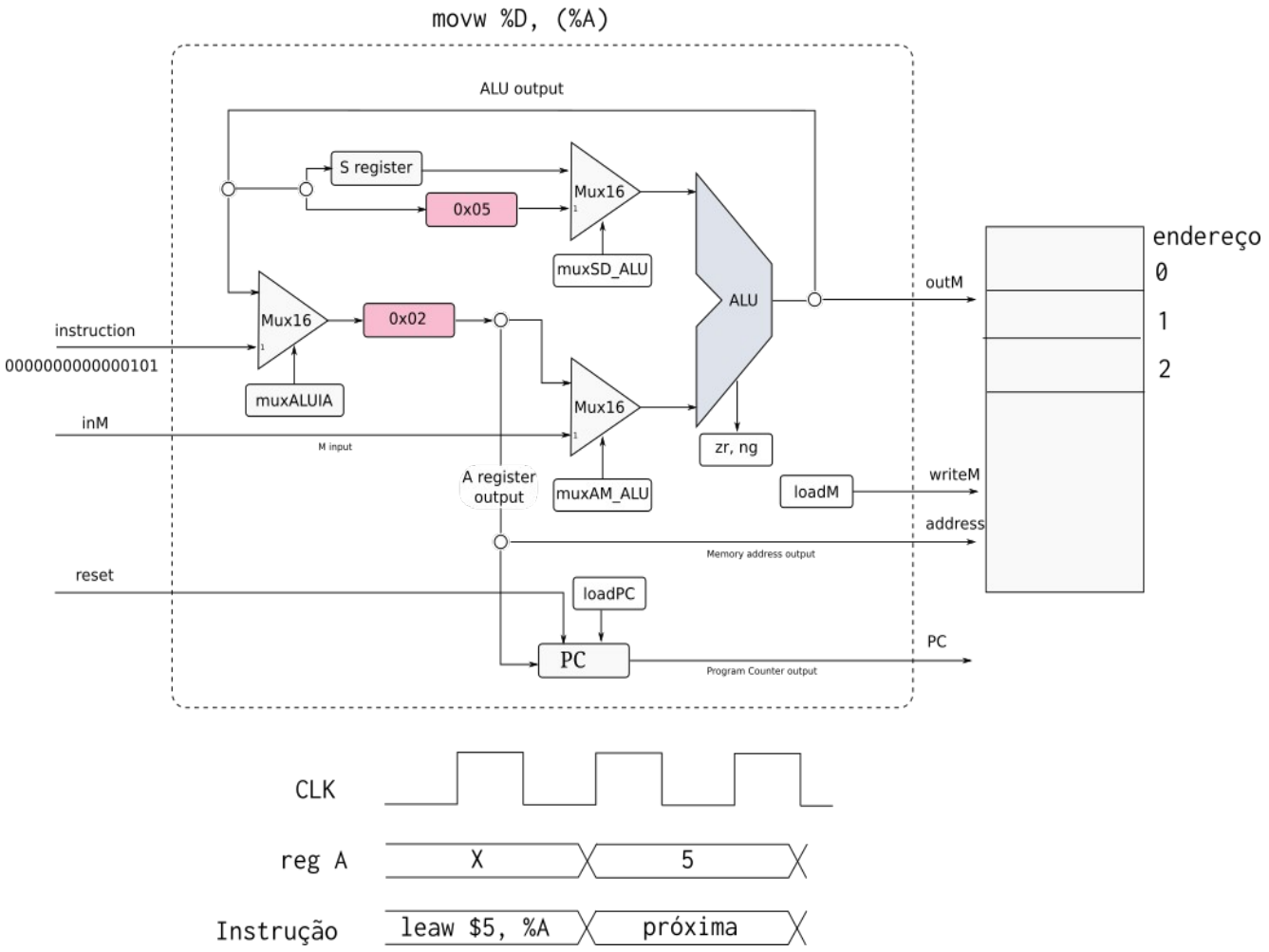
Exemplo – Instrução tipo D

movw %D, (%A)

Carrega na RAM endereço que %A aponta o valor do registrador D

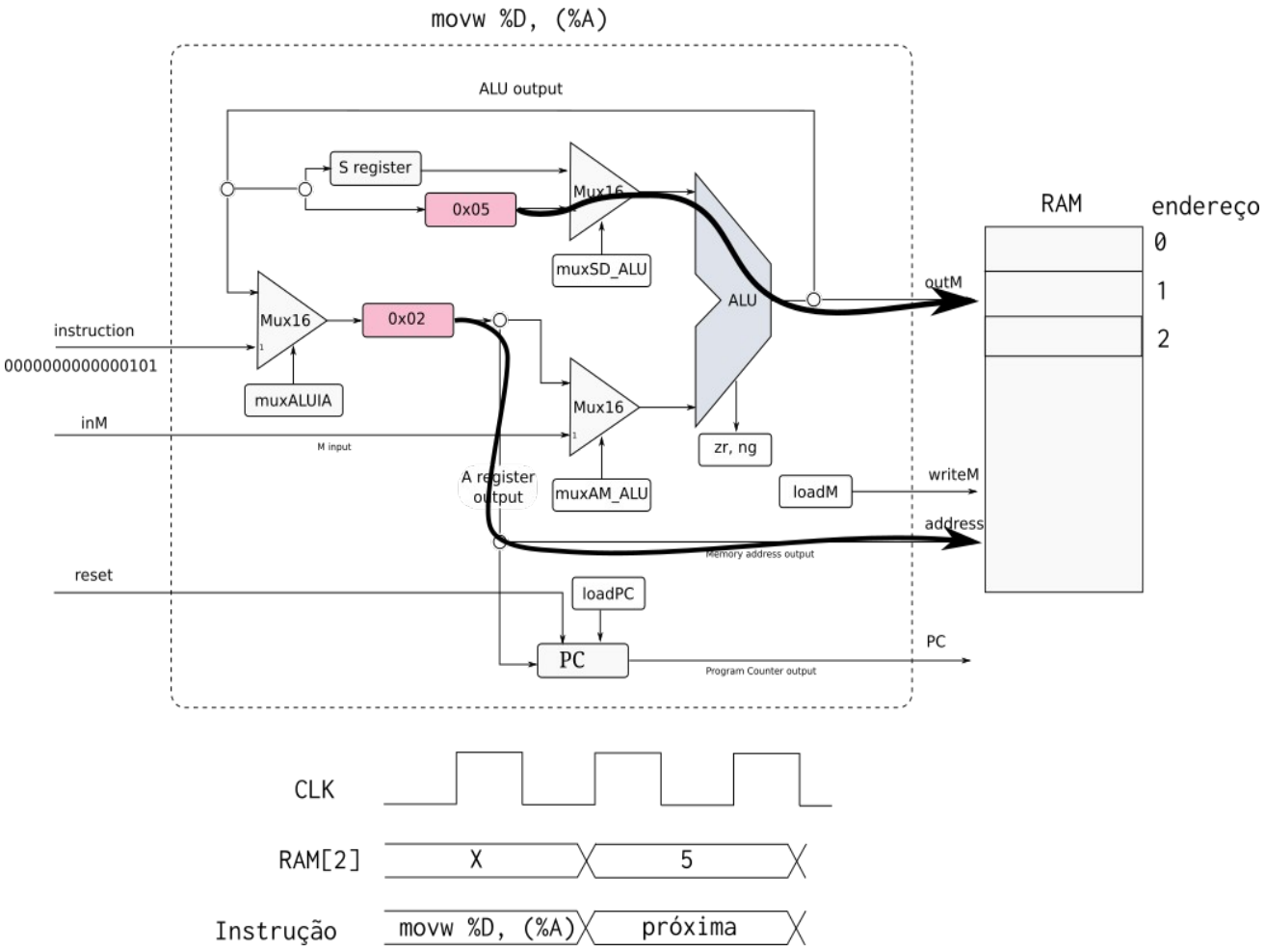
Exemplo – movw %D, (%A)

ESSE EXEMPLO É PARA UMA VERSÃO ATIGA DO HW! ONDE NÃO POSSUI O MUXAM_D



Exemplo – movw %D, (%A)

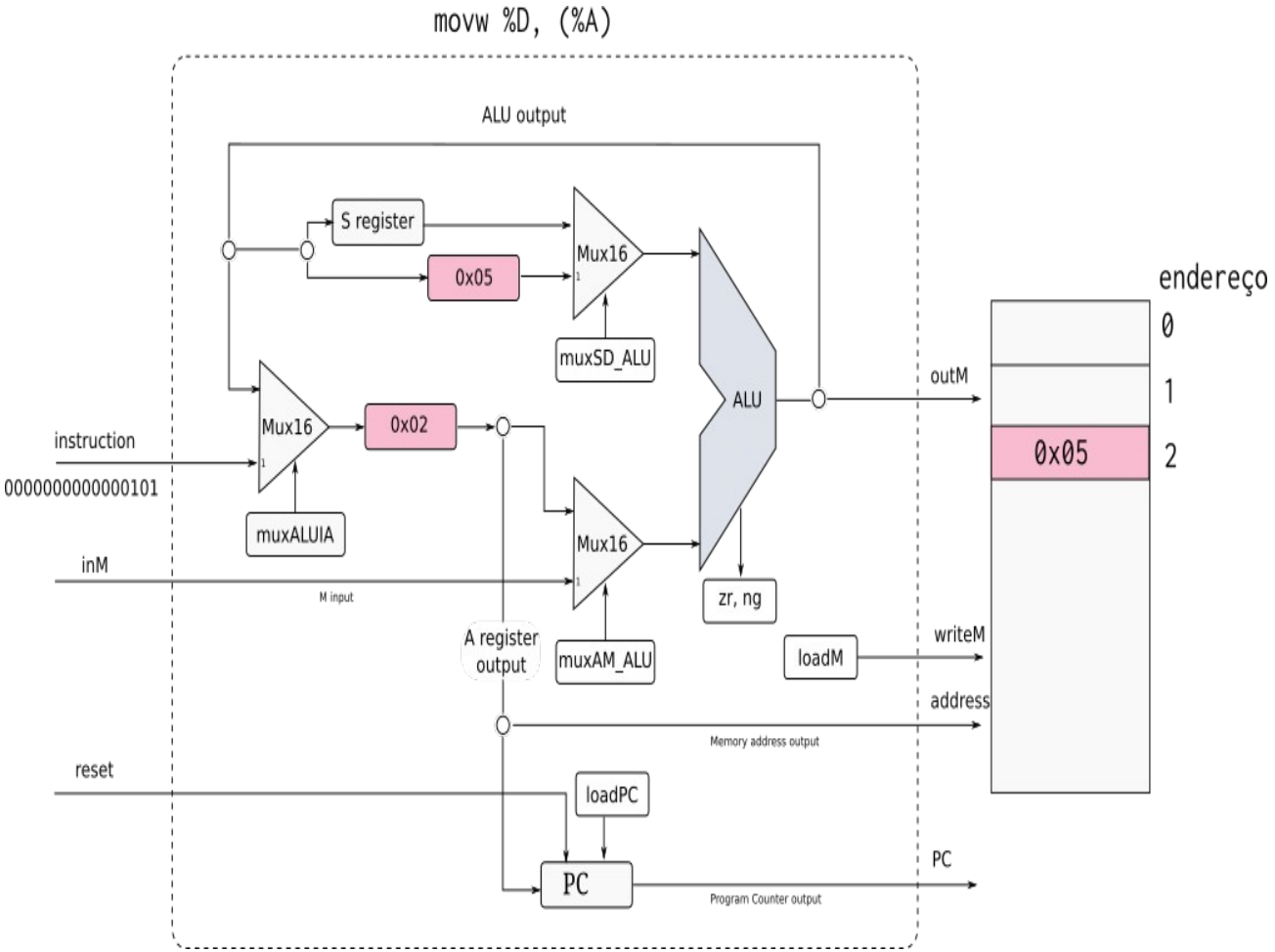
ESSE EXEMPLO É PARA UMA VERSÃO ATIGA DO HW! ONDE NÃO POSSUI O MUXAM_D



Insper

Exemplo – movw %D, (%A)

ESSE EXEMPLO É PARA UMA VERSÃO ATIGA DO HW! ONDE NÃO POSSUI O MUXAM_D



Insper

Arquivos Binários do Hack

Os arquivos binários (em linguagem de máquina) são compostos de textos com 16 bits de 0 e 1. Sendo uma instrução por cada linha, e cada linha representa uma posição de memória. Os arquivos tem uma extensão .hack

```
0000000000010000
1110111111001000
0000000000010001
1110101010001000
0000000000010000
1111110000010000
0000000001100100
1110010011010000
0000000000010010
1110001100000001
0000000000010000
1111110000010000
0000000000010001
1111000010001000
0000000000010000
1111110111001000
0000000000000100
1110101010000111
0000000000010010
1110101010000111
```

Arquivos Assembly do Hack

Arquivos Assembly do Hack tem a extensão .asm

Proposta não usarmos:

Usaremos o Assemble Z0.

```
// Adds 1+...+100.
        @i
        M=1
        @sum
        M=0
(LOOP)
        @i
        D=M
        @100
        D=D-A
        @END
        D;JGT
        @i
        D=M
        @sum
        M=D+M
        @i
        M=M+1
        @LOOP
        0;JMP
(END)
        @END
        0;JMP
```

Arquivos Assembly do Z0

Arquivos Assembly do Z0 tem a extensão .nasm

Vejam o Cheat Sheet.

```
leaw $2,%A
dec (%A)
movw (%A),%D
leaw $LOOP,%A
jg
not
leaw $1,%A
movw (%A),%D
leaw $4,%A
movw %D,(%A)
movw %D,(%A)
leaw $LOOP,%A
jg
END:
leaw $END,%A
jmp ; Loop
```

Assembly AT&T para o Z0

Registradores virtuais:

Os símbolos R0, ..., R15 são automaticamente predefinidos para se referir aos endereços de RAM 0, ..., 15

Ponteiros de I/O :

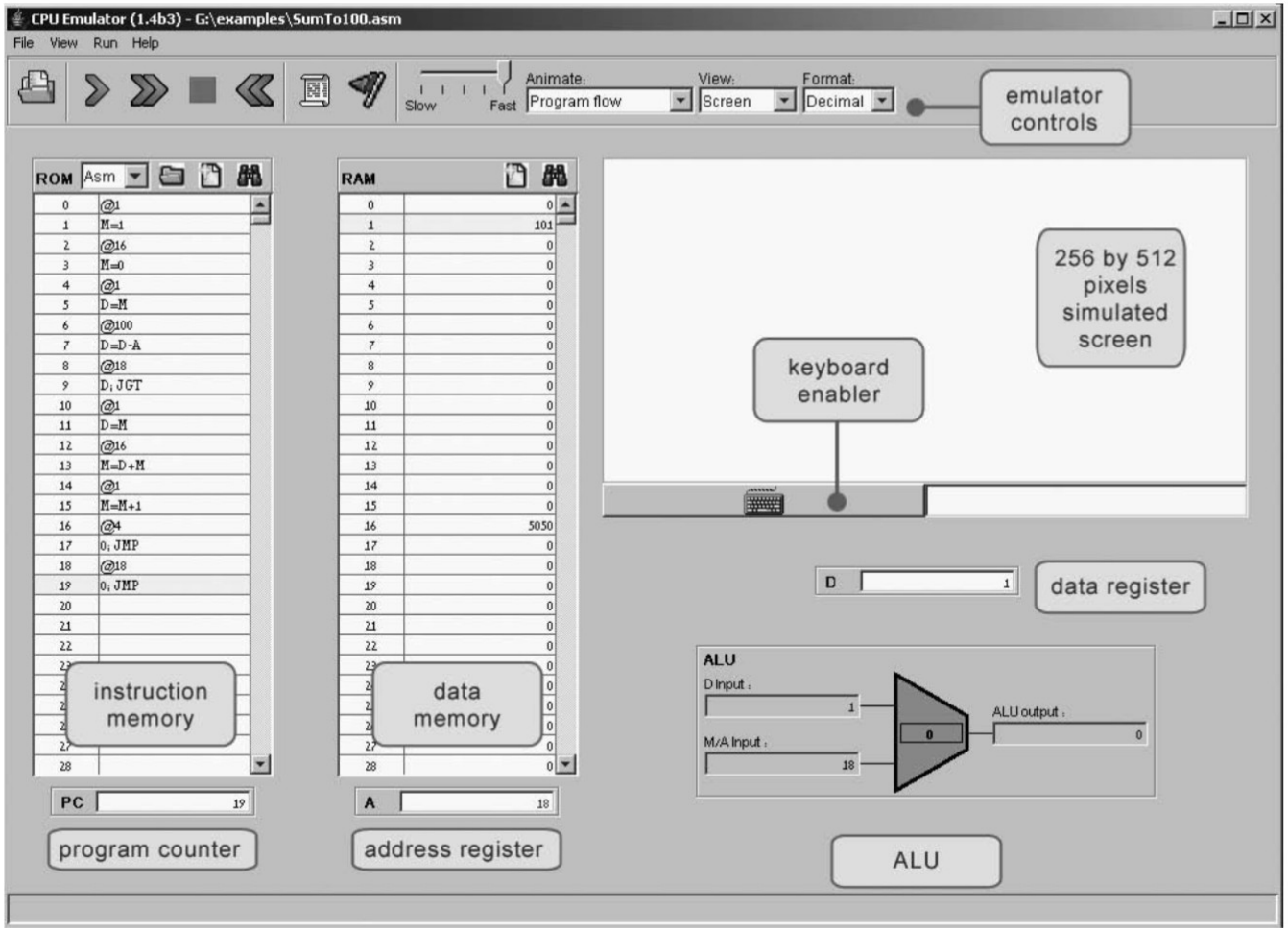
Os símbolos SCREEN e KBD são automaticamente predefinidos para se referir aos endereços de RAM 16384 e 24576, respectivamente

Ponteiros de controle da VM:

os símbolos SP, LCL, ARG, THIS, e THAT são automaticamente predefinidos para se referir ao endereços de RAM 0-4, respectivamente

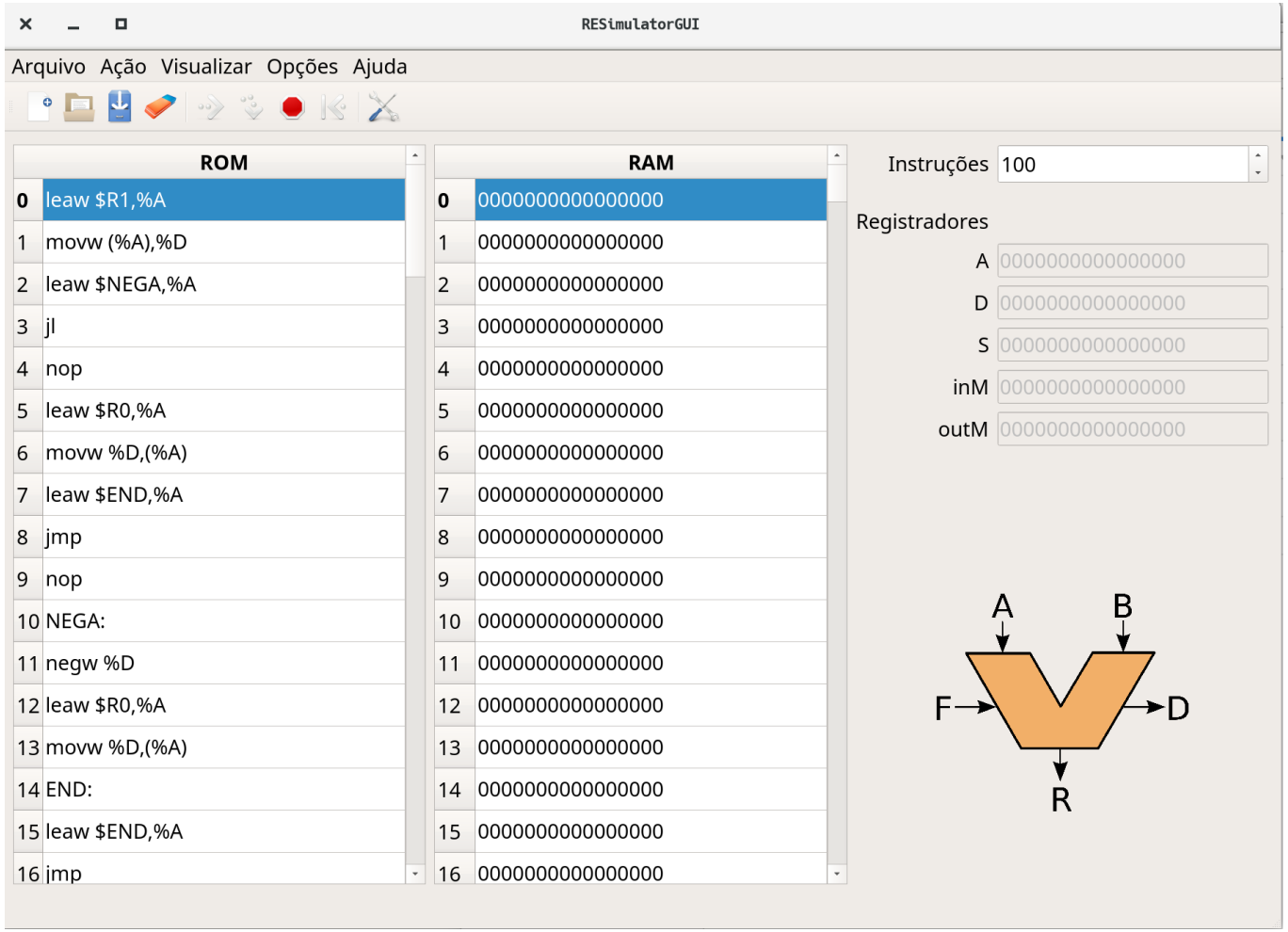
Emulador de CPU (CPU Emulator)

Simula um código de máquina no computador Hack.



Insper

Z01 Simulator



Insper

Insper

www.insper.edu.br