

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.”

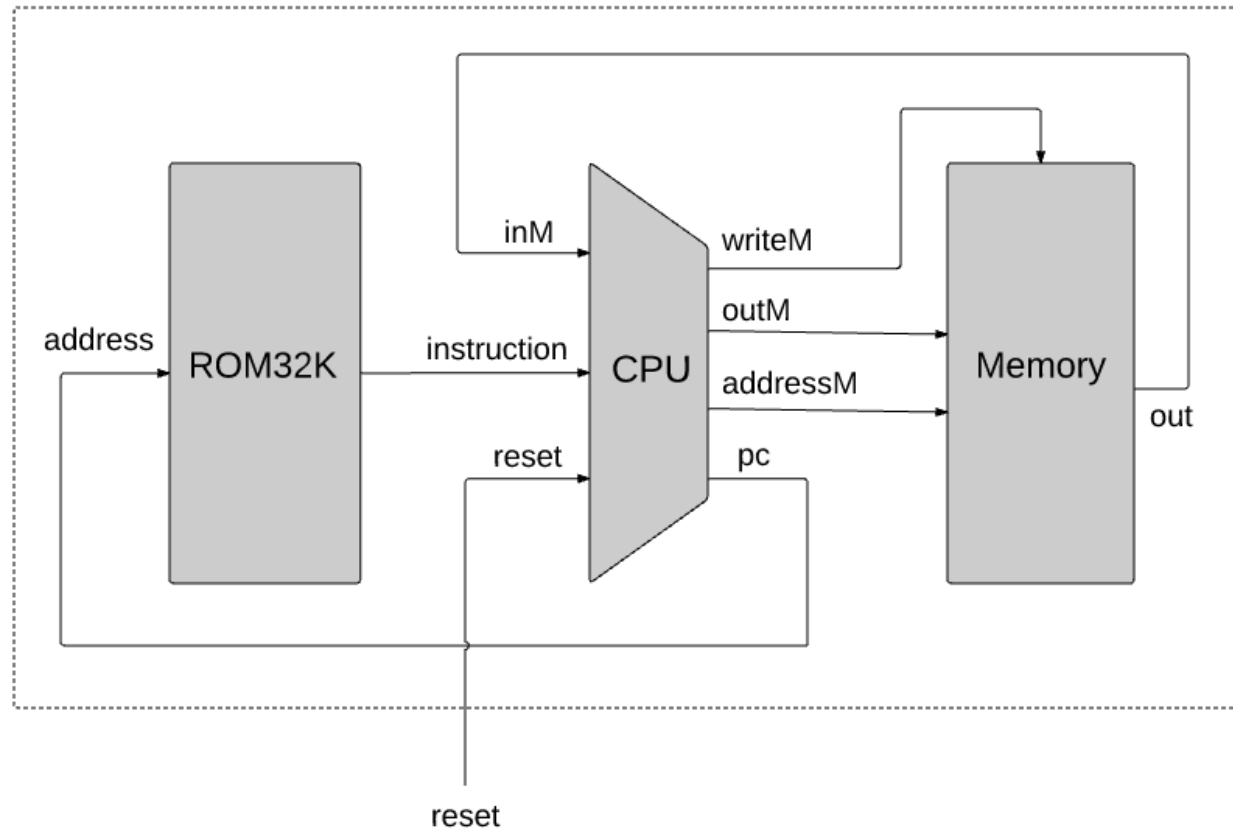
Blaise Pascal (1623–1662) matemático francês

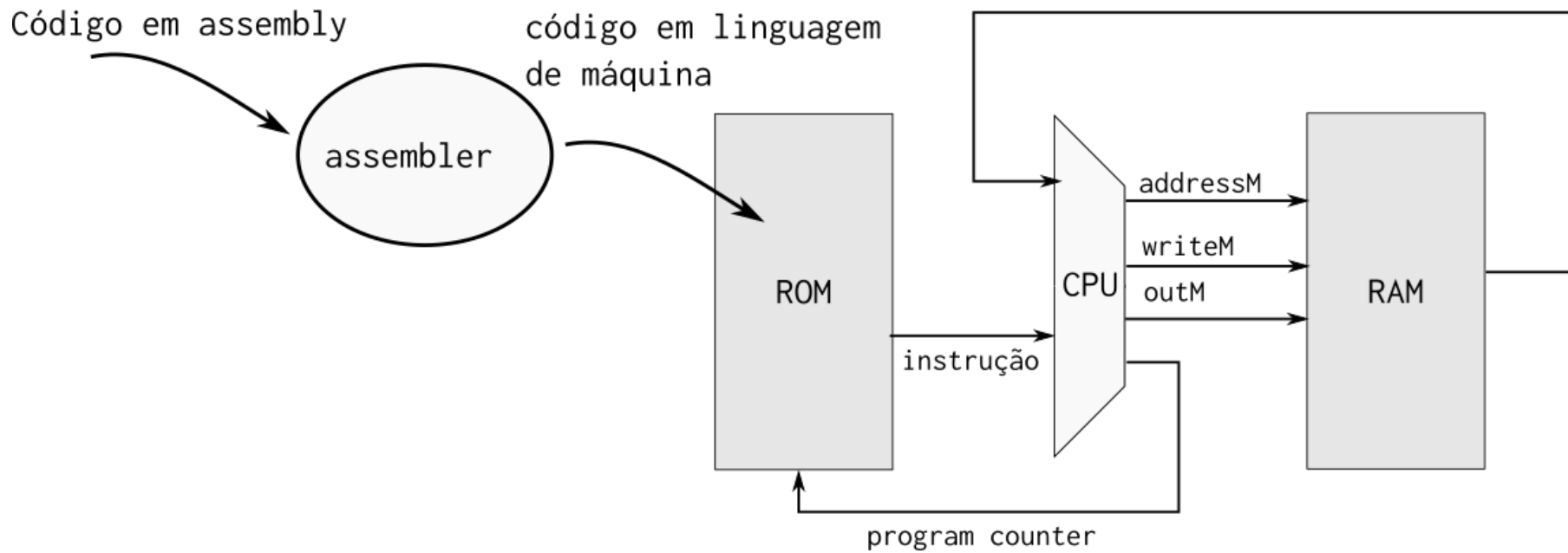
Objetivo da Aula

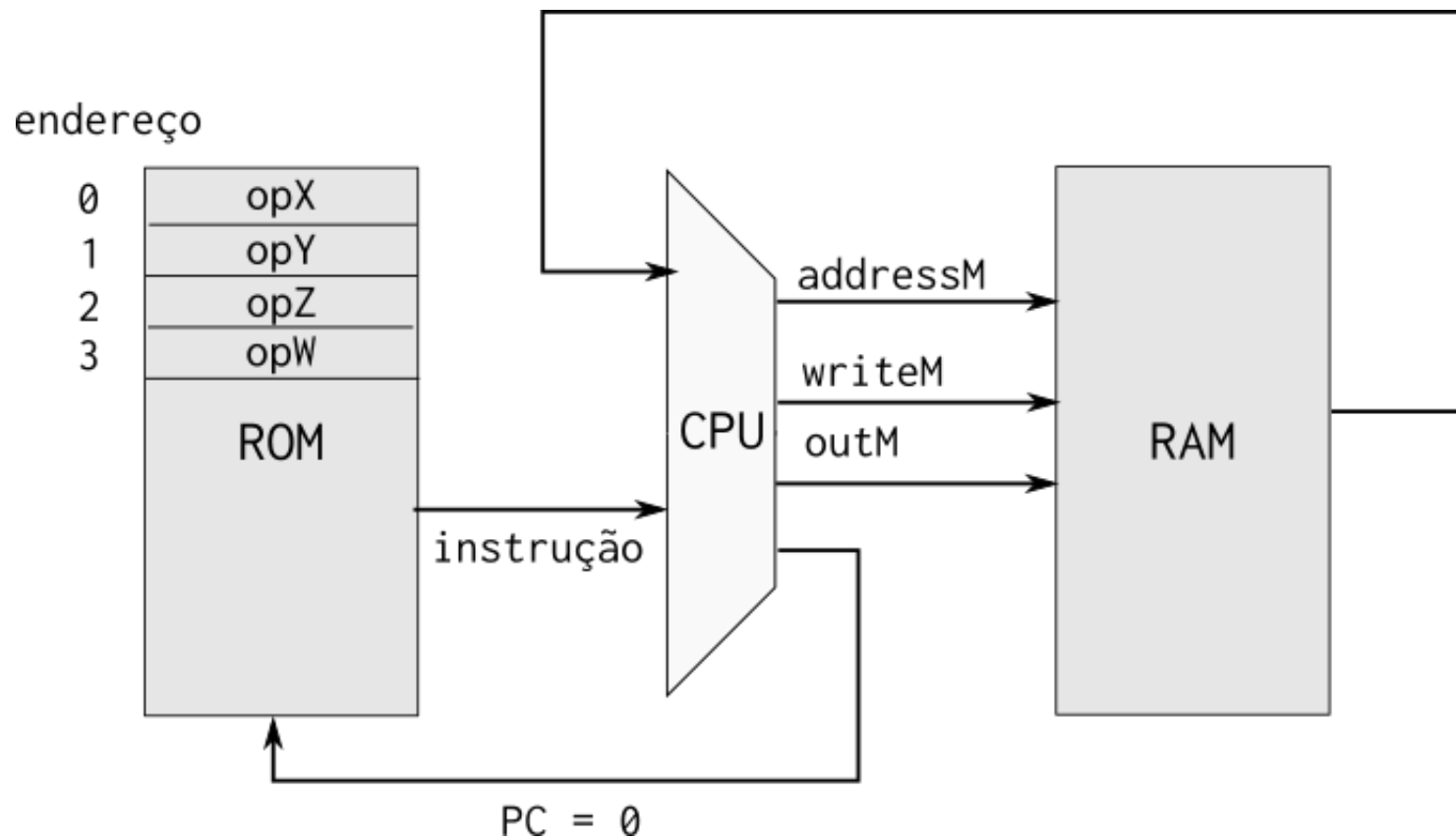
- Mapear Dispositivos em Memória;
- Programar em Assembly.

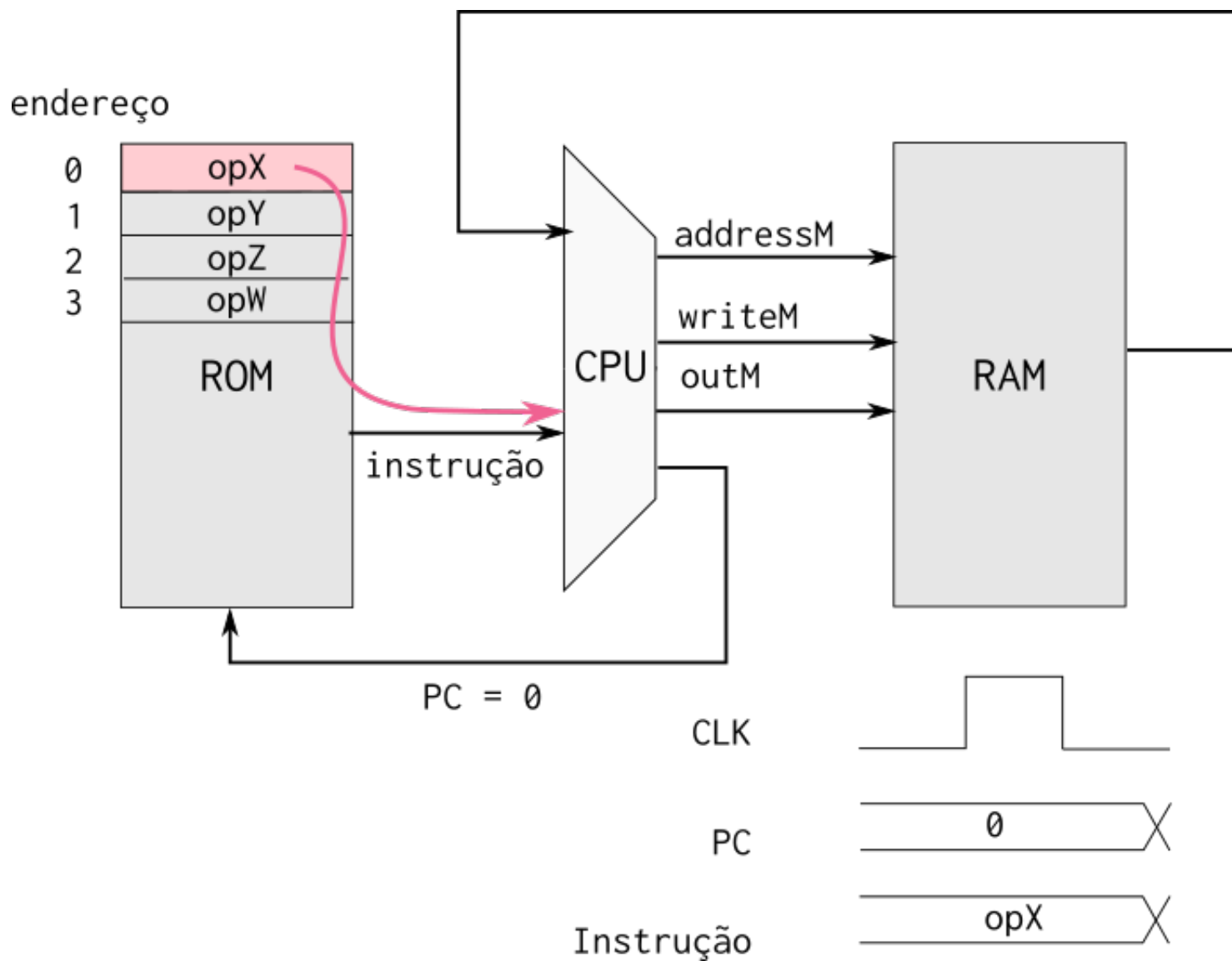
Conteúdos: Interfaces e Periféricos, Hierarquia de Memória;

Z01 - Computador

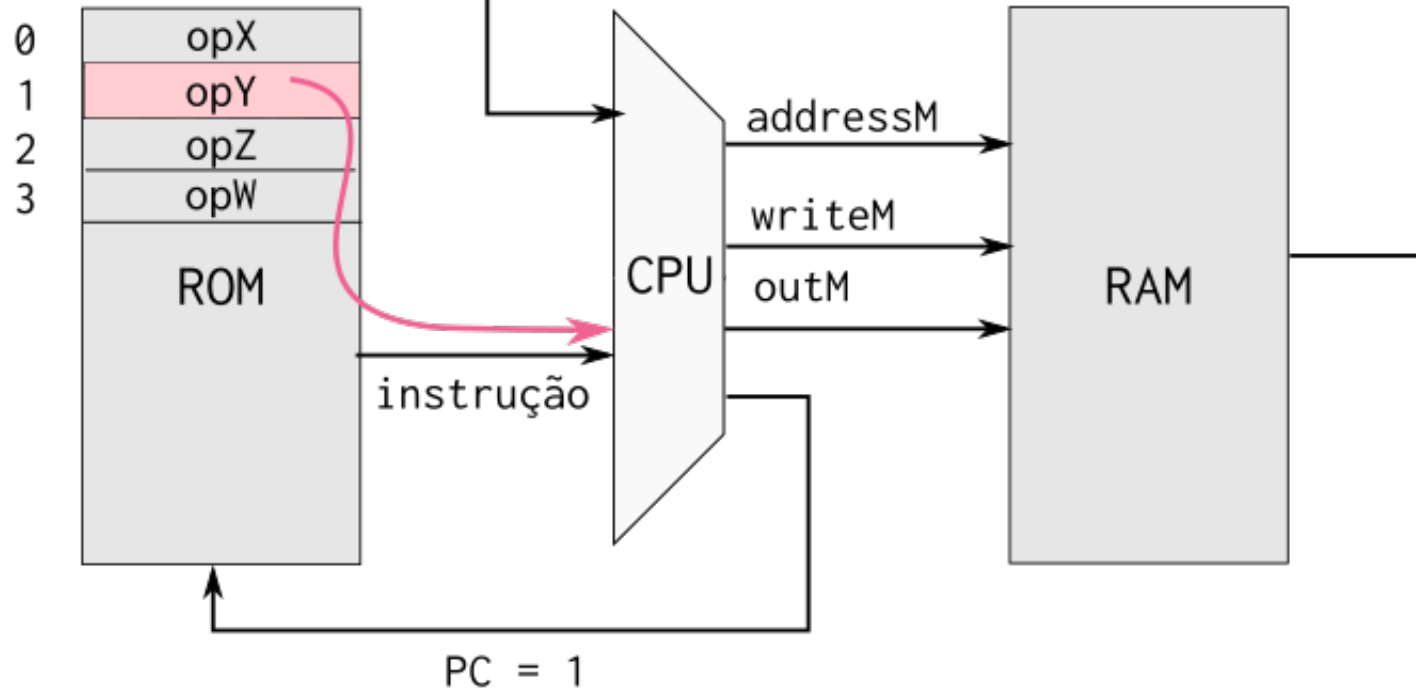




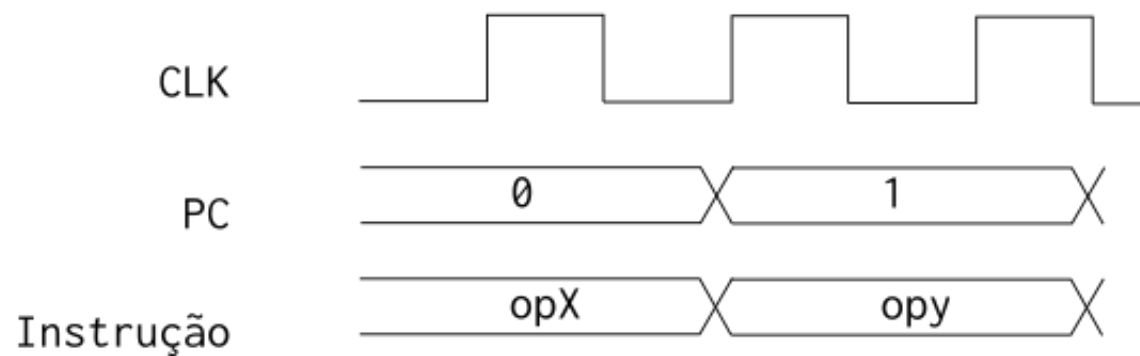




endereço



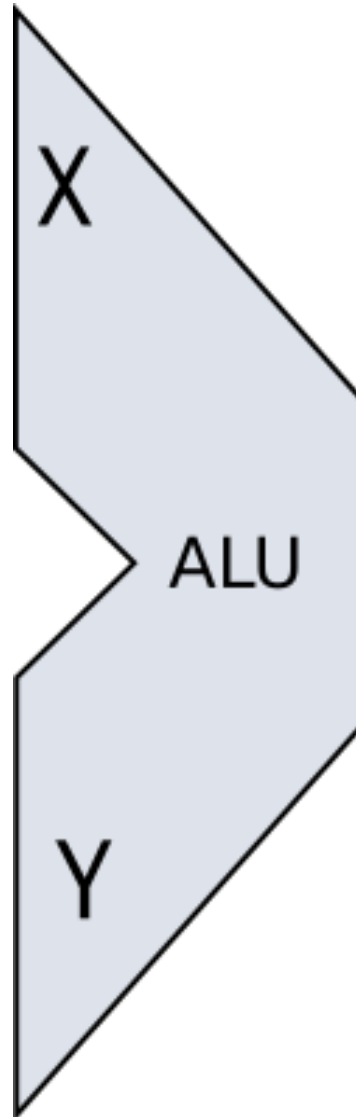
PC = 1



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



Z01 – ULA

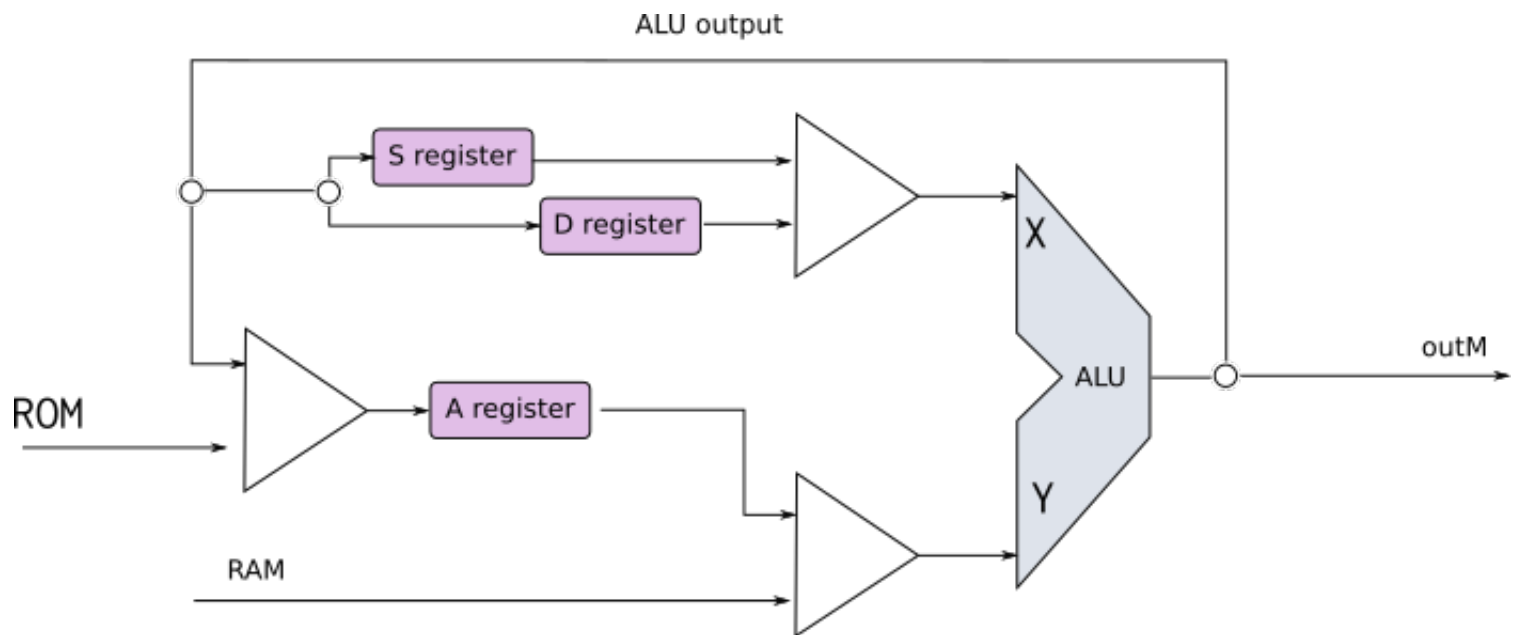
Entradas e saídas da ULA

- S register
- D register
- A register
- RAM

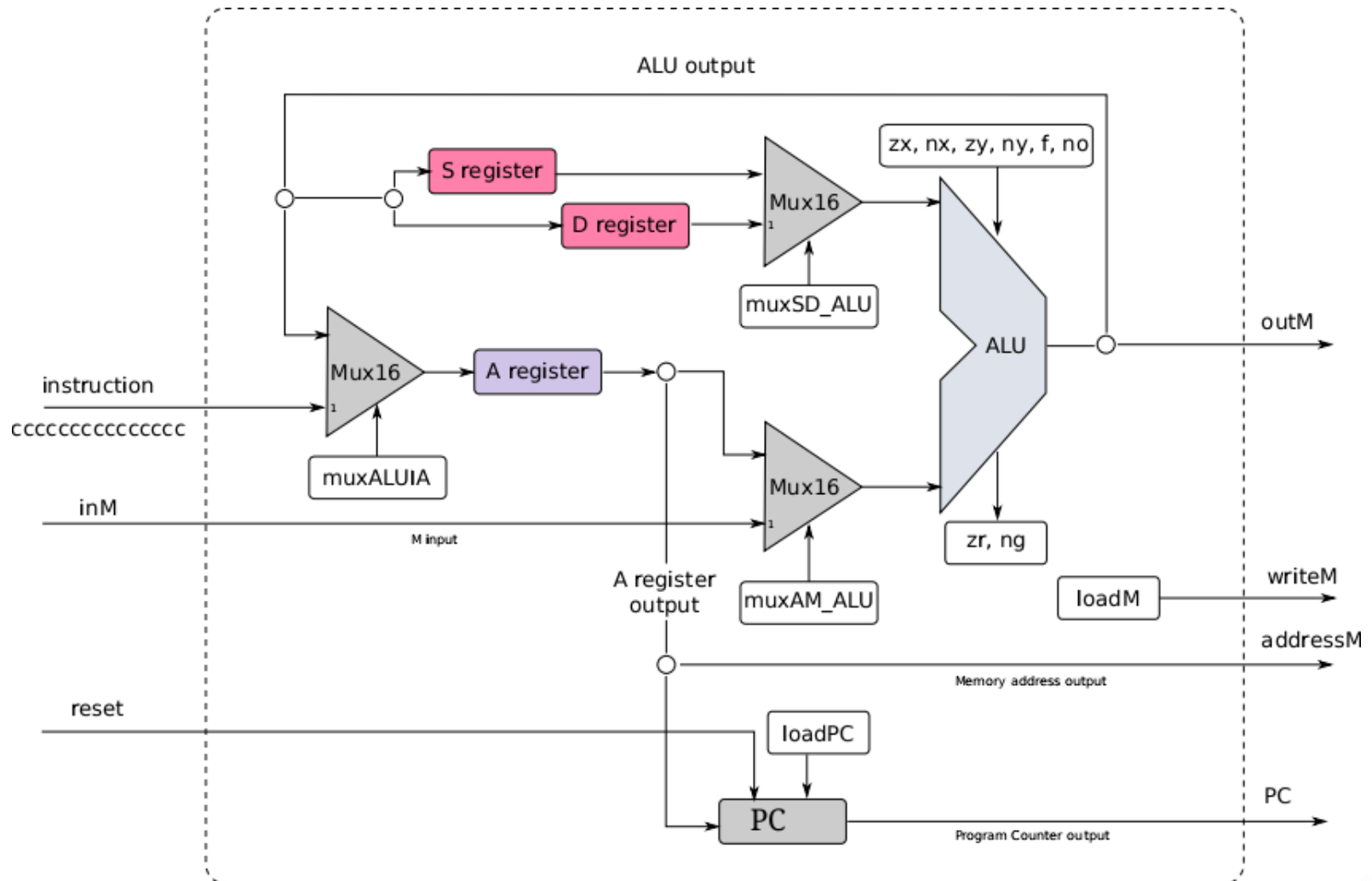


- S register
- D register
- A register
- RAM

Z01 – Registradores ?



Z01 - CPU



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

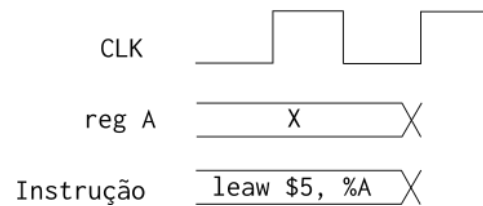
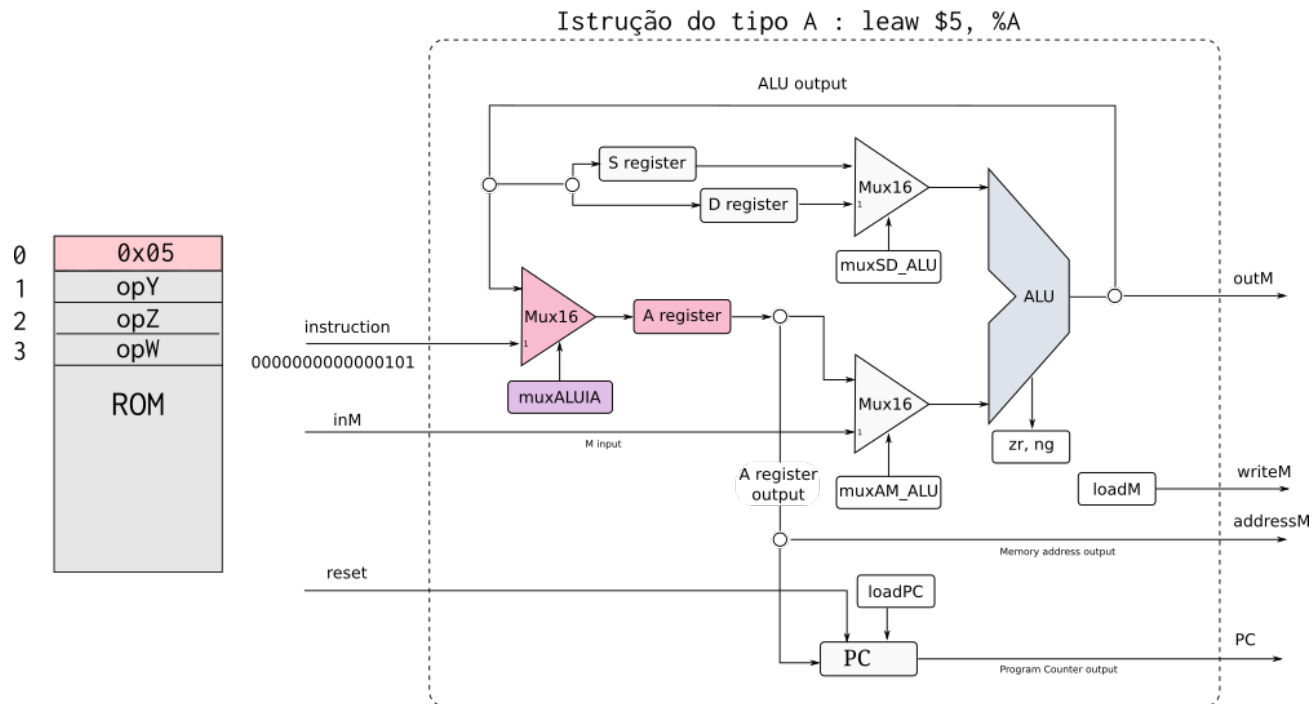
- Movw %A, %S

Exemplo – Instrução tipo A

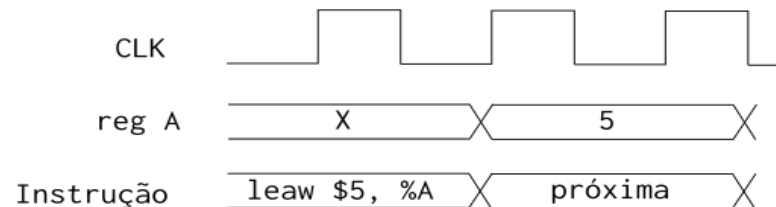
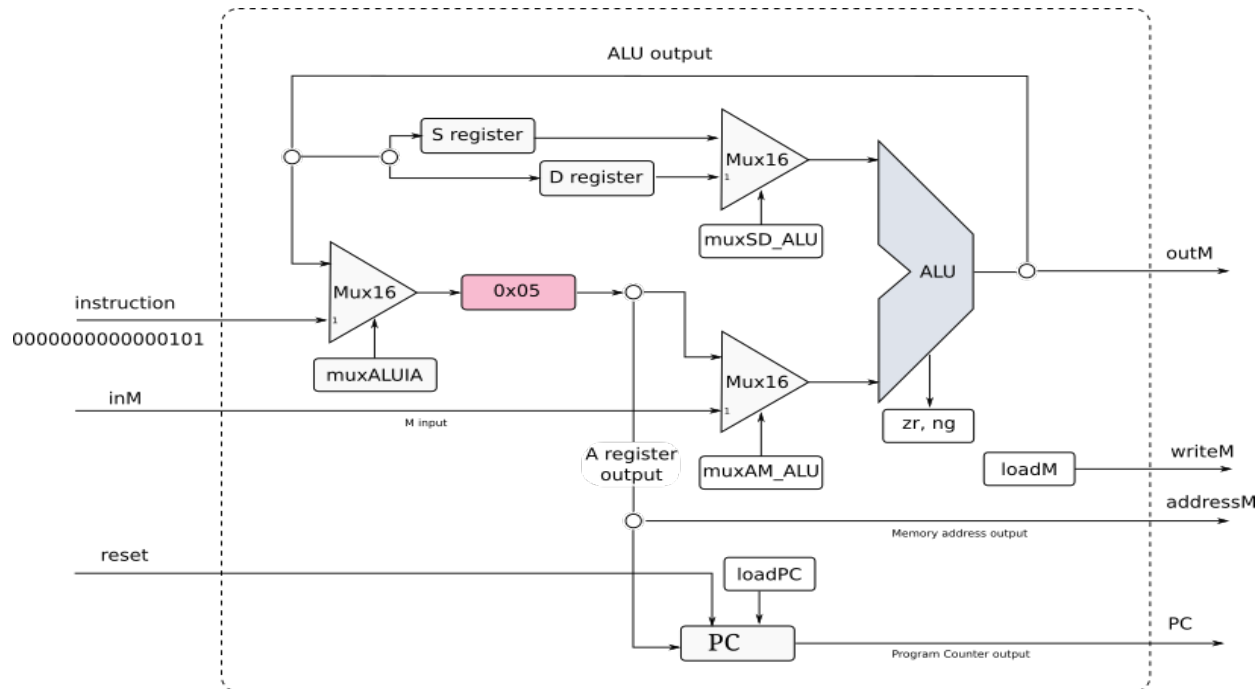
`leaw $5, %A`

Carrega no registrador A a constante 5

Exemplo – leaw \$5, %A



Exemplo – leaw \$5, %A

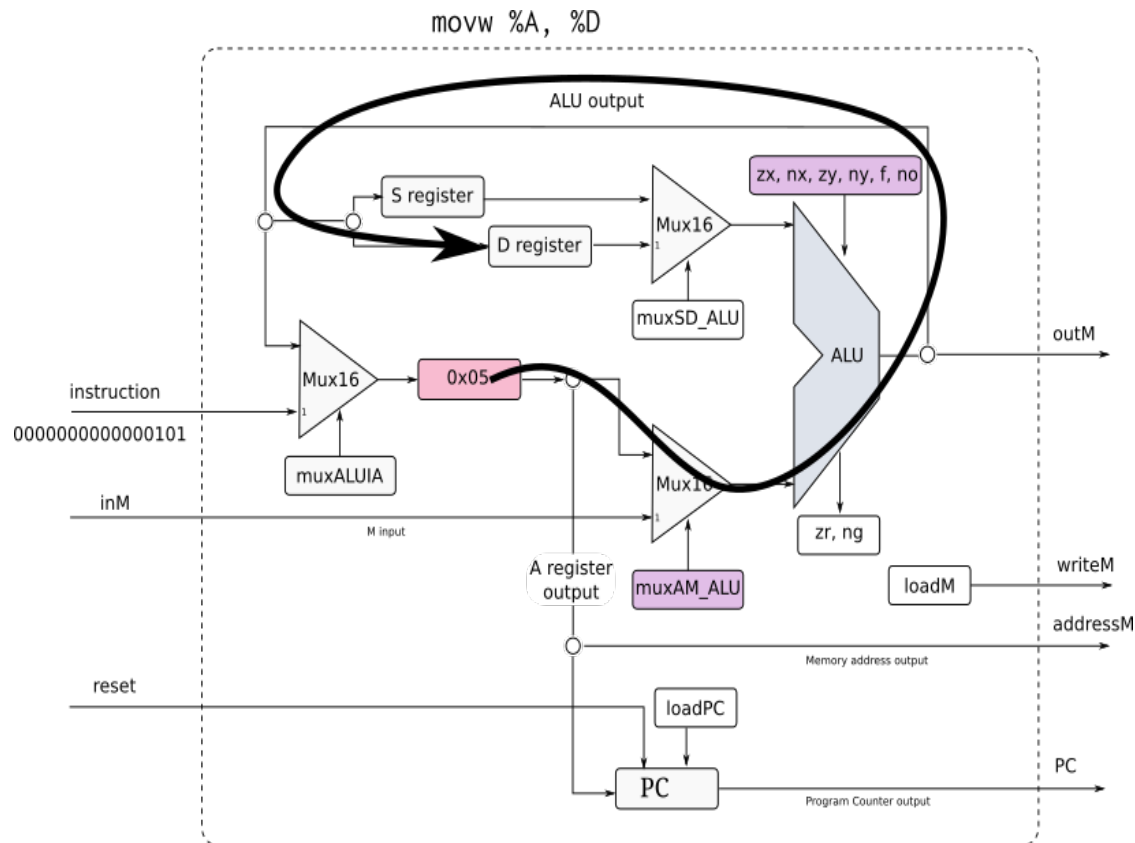


Exemplo – Instrução tipo D

`movw %A, %D`

Carrega no registrador D o valor do
registrador A

Exemplo – movw %A, %D

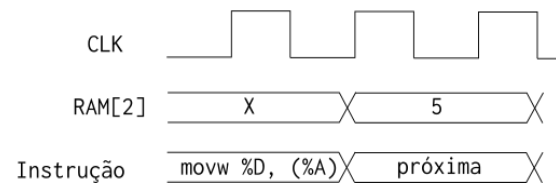
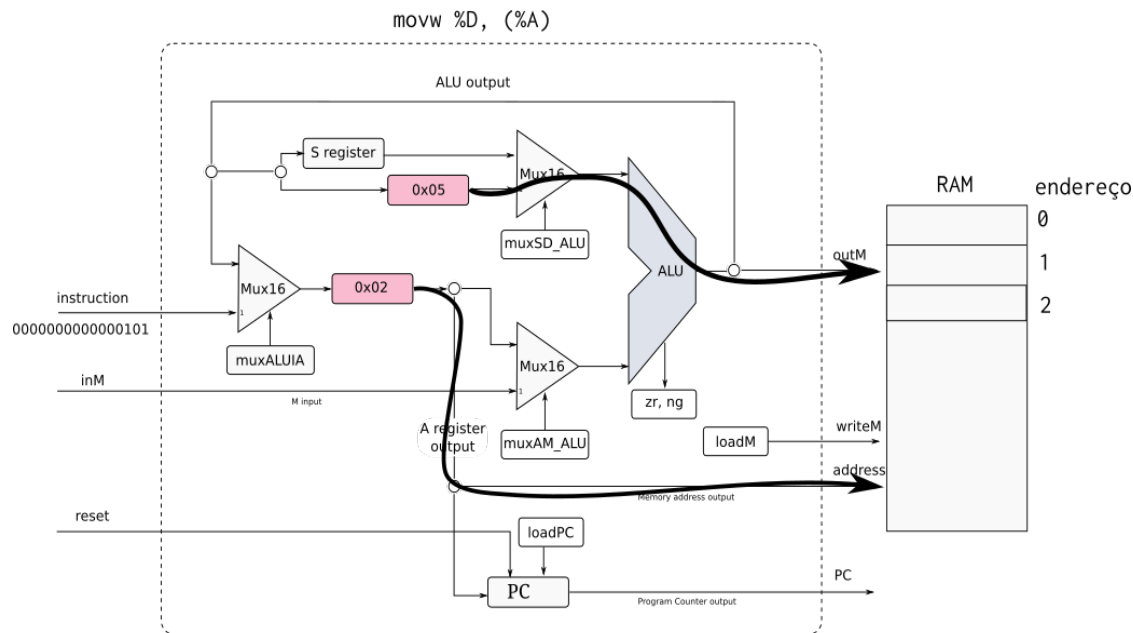


Exemplo – Instrução tipo D

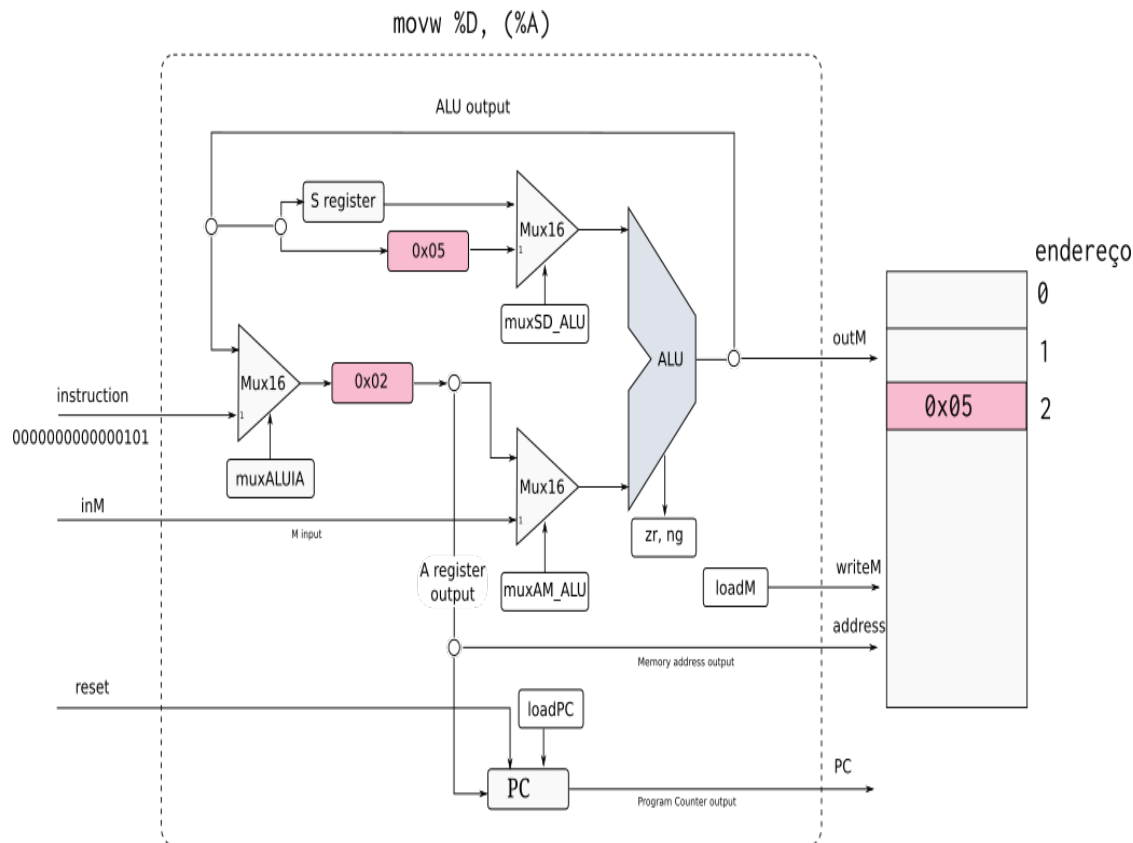
`movw %D, (%A)`

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

Exemplo – movw %D, (%A)



Exemplo – movw %D, (%A)



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 :

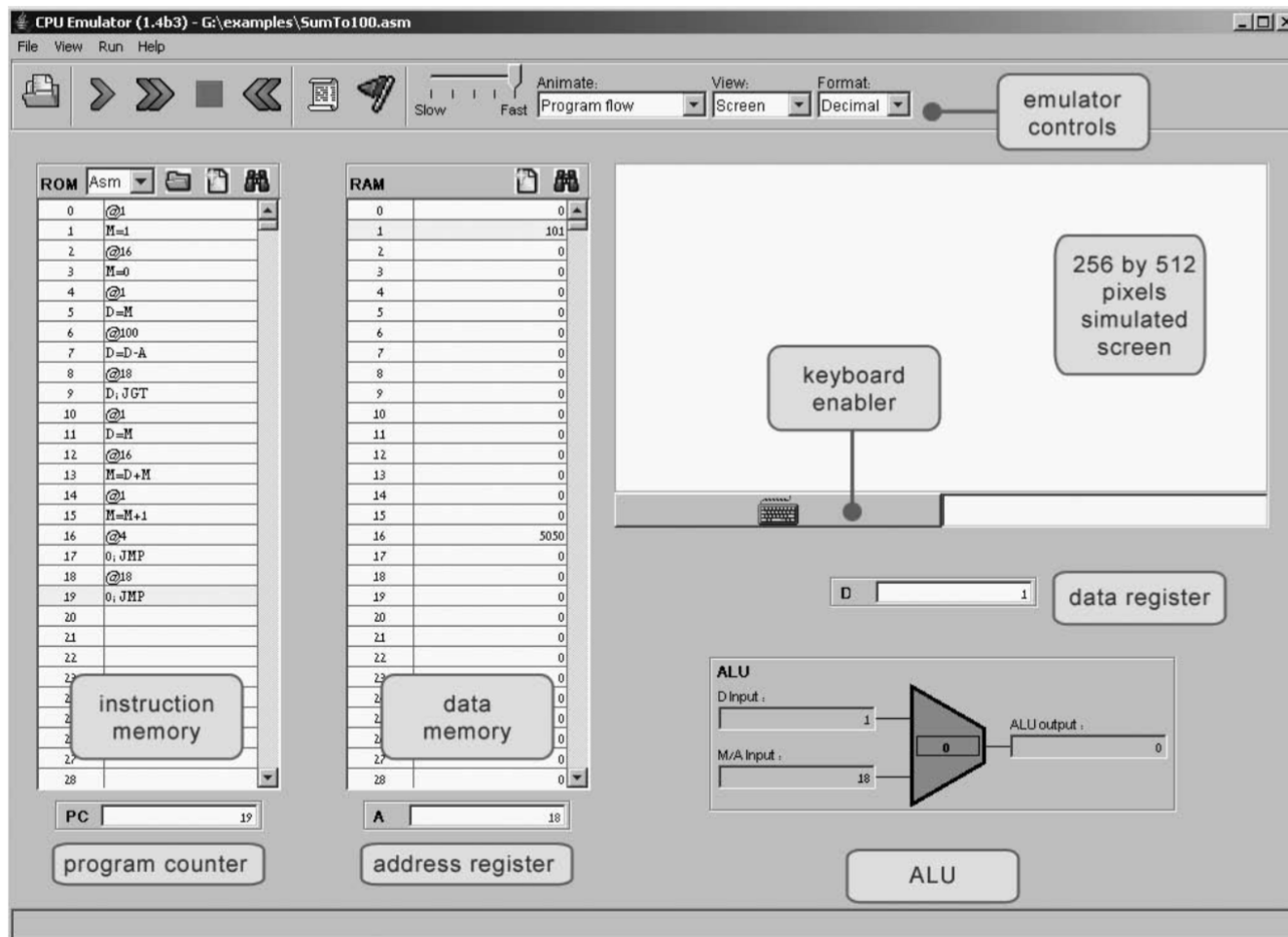
Os símbolos SCREEN e KBD são automaticamente predefinidos para se referir aos endereços de RAM 16384 e 24576, respectivamente

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.



Z01 Simulator

RESimulatorGUI

Arquivo Ação Visualizar Opções Ajuda

ROM

0	leaw \$R1,%A
1	movw (%A),%D
2	leaw \$NEGA,%A
3	jl
4	nop
5	leaw \$R0,%A
6	movw %D,(%A)
7	leaw \$END,%A
8	jmp
9	nop
10	NEGA:
11	negw %D
12	leaw \$R0,%A
13	movw %D,(%A)
14	END:
15	leaw \$END,%A
16	jmp

RAM

0	0000000000000000
1	0000000000000000
2	0000000000000000
3	0000000000000000
4	0000000000000000
5	0000000000000000
6	0000000000000000
7	0000000000000000
8	0000000000000000
9	0000000000000000
10	0000000000000000
11	0000000000000000
12	0000000000000000
13	0000000000000000
14	0000000000000000
15	0000000000000000
16	0000000000000000

Instruções 100

Registradores

A 0000000000000000

D 0000000000000000

S 0000000000000000

inM 0000000000000000

outM 0000000000000000

```
graph TD; A --> V; B --> V; F --> V; V --> D; V --> R;
```

Insper

www.insper.edu.br