

# Workshop de Assembly:

; João Oliveira  
; joao.ferreira.oliveira@tecnico.ulisboa.pt  
; Jonyleo#8960

# Resumo:

- \* Introdução
- \* Ambiente de Desenvolvimento
- \* PEPE Assembly
- \* Exercício (Live coding)
- \* Periféricos
- \* Exercício (Live coding)
- \* Stack
- \* Exercício (Live coding)
- \* Erros comuns e Debugging
- \* Duvidas

# Introdução

; O que é assembly?

; Cuidados a ter

# Ambiente de Desenvolvimento:

; Simulador



# Ambiente de Desenvolvimento:

; IDE / Editor de texto



; <https://github.com/Jonyleo/PEPEAsm/>

# PEPE Assembly:

; Constantes

1000H	; 0000H -> 0F000H
010101b	; 0b -> 1111111111111111b
19	; -32768 -> 32767
“Ola\n”	; não é terminado em 0
NAME EQU 10	; semelhante a #define em C

# PEPE Assembly:

; Labels - Identificar zonas do código

nextSlide:

; código ou

; dados

# PEPE Assembly:

; Directiva PLACE

PLACE 1000H

; dados

PLACE 0

; codigo



# PEPE Assembly:

; Registos

; 16 bits

Rx ; 0 - 15

RL ; R11

SP ; R12

RE ; R13

BTE ; R14

TEMP ; R15

Main registers			Level: System		
PC	0008	H			
R0	FFFF	H	R1	0000	H
R2	0000	H	R3	0000	H
R4	0000	H	R5	0000	H
R6	0000	H	R7	0000	H
R8	0000	H	R9	0000	H
R10	0000	H	R11	0000	H
SP	0000	H	RE	0006	H
BTE	0000	H	TEMP	0000	H
SSP	0000	H	USP	0000	H

# PEPE Assembly:

; Instruções - Manipular dados, controlo de fluxo e  
; configuração

MOV R0, 10

PUSH R9

ADD R4, 11b

XOR R2, R3

JMP nextSlide

# PEPE Assembly:

; Operandos

MOV R0, 10H

ADD R5, R1

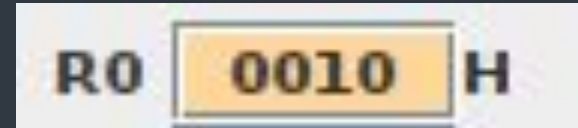
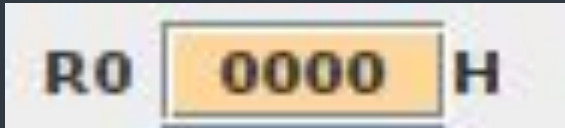
NOT R0

RET

# PEPE Assembly:

; Instrução fundamental

MOV R0, 10H



# PEPE Assembly:

; Instruções aritméticas e lógicas

ADD Rx, Rx	MUL Rx, Rx
SUB Rx, Rx	DIV Rx, Rx
CMP Rx, Rx	MOD Rx, Rx
ADD Rx, C	AND Rx, Rx
SUB Rx, C	OR Rx, Rx
CMP Rx, C	XOR Rx, Rx

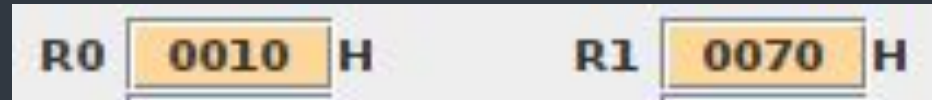
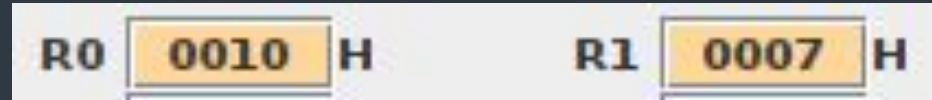
# PEPE Assembly:

; Instruções aritméticas e lógicas

MOV R0, 10H

MOV R1, 7H

MUL R1, R0



# PEPE Assembly:

; Instruções aritméticas e lógicas

INC Rx

DEC Rx

NEG Rx

NOT Rx

SHL Rx, C

SHR Rx, C

ROL Rx, C

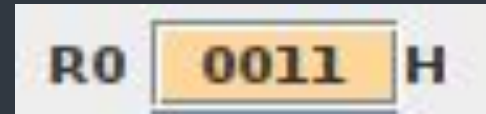
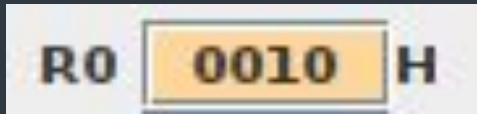
ROR Rx, C

# PEPE Assembly:

; Instruções aritméticas e lógicas

MOV R0, 10H

INC R0





# PEPE Assembly:

; Instruções controlo de fluxo

CALL label

JMP label

RET

# PEPE Assembly:

; Instruções controlo de fluxo

JMP label

PLACE 100H

label:

; código



# PEPE Assembly:

; Saltos condicionais - Dependem do RE

JZ label

JLT label

JEQ label

JNZ label

JLE label

JNE label

JN label

JGT label

JNN label

JGE label

JP label

JC label

JNP label

JNC label

# PEPE Assembly:

; Saltos condicionais - Dependem do RE

MOV R0, 10H

MOV R1, 7H

MUL R1, R0

JZ label

PC 0006 H



PC 0100 H

# PEPE Assembly:

; Saltos condicionais - Dependem do RE

MOV R0, 10H

MOV R1, 0H

MUL R1, R0

JZ label

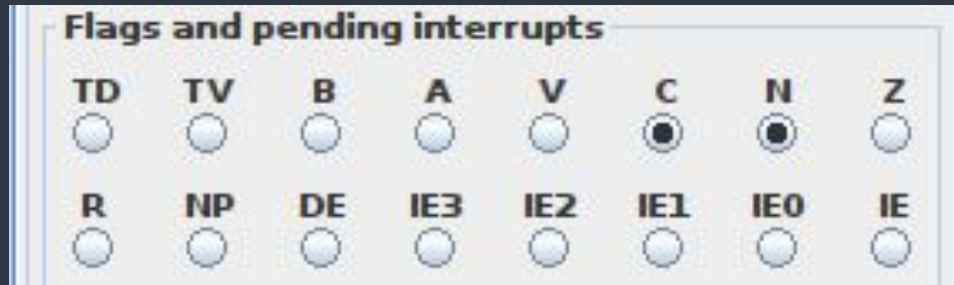
PC 0006 H



PC 0008 H

# PEPE Assembly:

; Registo de estado



# PEPE Assembly:

; Memória

PLACE 1000H

WORD 1234H

STACK 10H

TABLE 10H

STRING "ola", 10b

Memory contents																											
Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		0	1	2	3	4	5	6			
1000 H	12	34	00	00	00	00	00	00	00	00	00	00	00	00	00	00			4								
1010 H	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00											
1020 H	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00											
1030 H	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00											
1040 H	00	00	6F	6C	61	02	00	00	00	00	00	00	00	00	00	00				o	l	a					
1050 H	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00											

# PEPE Assembly:

; Leitura

MOV R0, [1000H]

MOV R0, 1000H

MOV R1, [R0]

MOV R1, [R0 + R2]

MOVB R1, [R0]



# PEPE Assembly:

; Escrita

MOV [1000H], R0

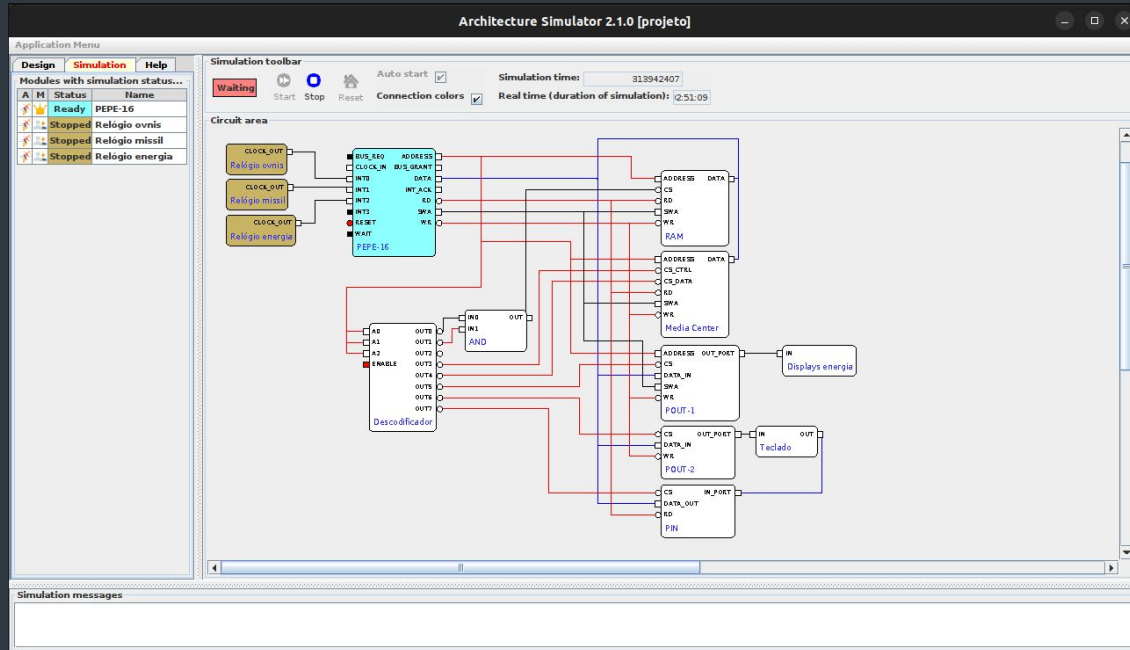
MOV R0, 1000H

MOV [R0], R1

MOV [R0 + R2], R1

MOVB [R0], R1

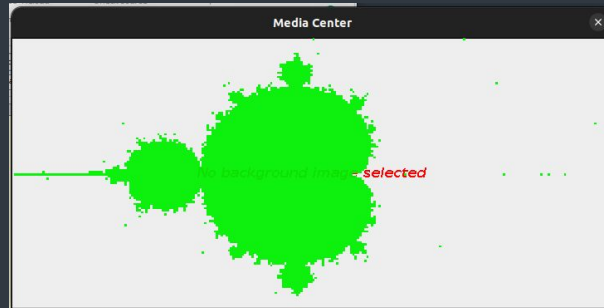
# Exercicio (Live coding):



# Periféricos:

; Pixel Screen / Media Center

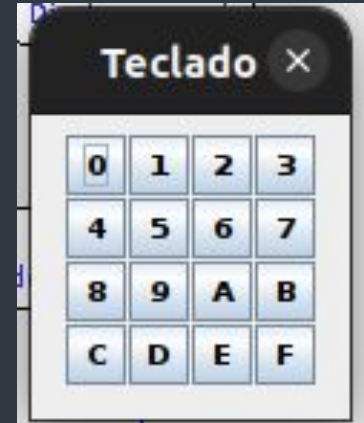
```
PXL_SCREEN_CTRL EQU 6000H  
PXL_SCREEN_DATA EQU 8000H
```



# Periféricos:

; Teclado

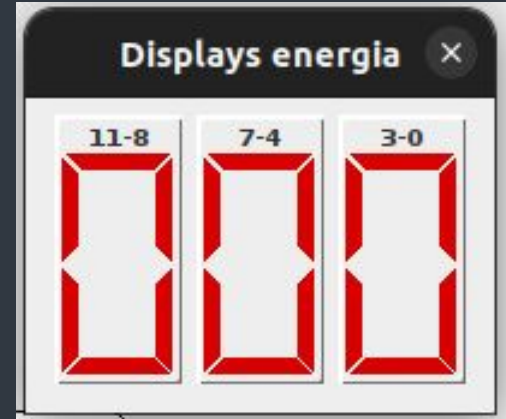
```
TEC_LIN EQU 0C000H  
TEC_COL EQU 0E000H
```



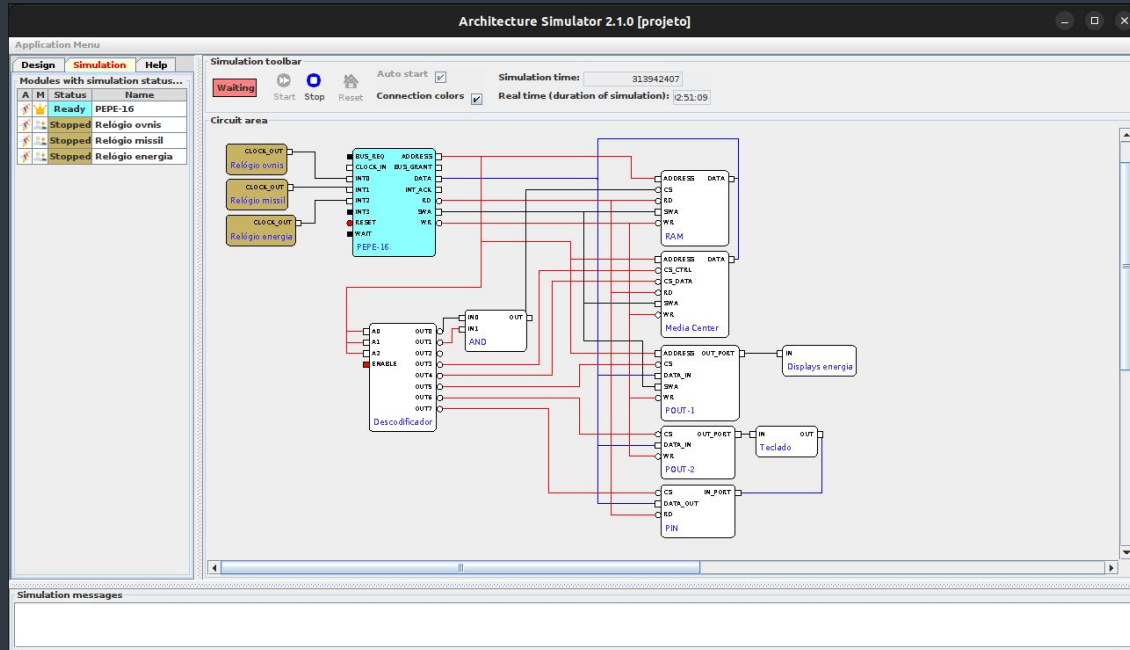
# Periféricos:

; Displays

DISPLAYS EQU 0A000H



# Exercicio (Live coding):



# Stack:

; Declaração e Stack pointer

PLACE 1000H

STACK 100H

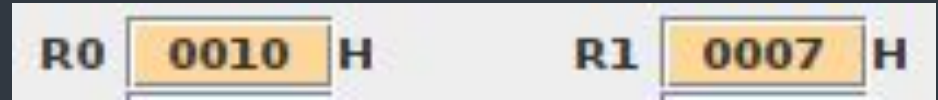
SP\_init

PLACE 1000H

MOV SP, SP\_init

# Stack:

; Utilização



10H      CALL rotina  
12H      MOV R0, 10H

...      ...

rotina:

100H      PUSH R0  
100H      PUSH R1



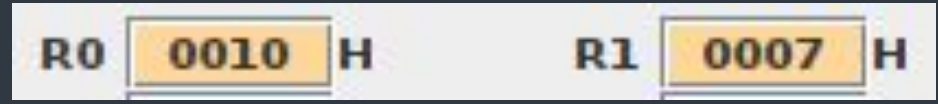
# Stack:

; Utilização

10H CALL rotina  
12H MOV R0, 10H  
...

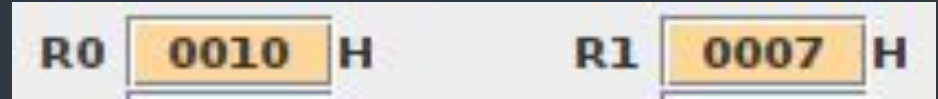
rotina:

100H PUSH R0  
100H PUSH R1



# Stack:

; Utilização



10H    CALL rotina  
12H    MOV R0, 10H  
...



rotina:

100H    PUSH R0  
100H    PUSH R1



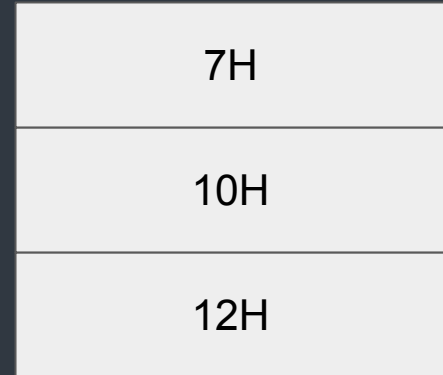
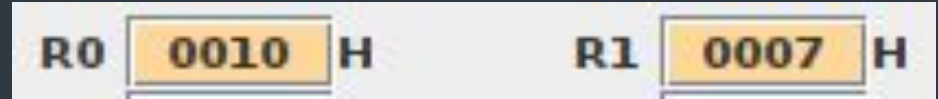
# Stack:

; Utilização

10H      CALL rotina  
12H      MOV R0, 10H  
...

rotina:

100H      PUSH R0  
100H      PUSH R1

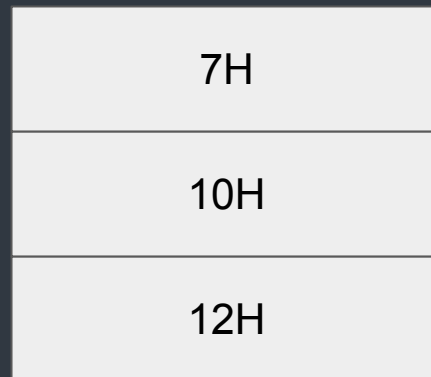
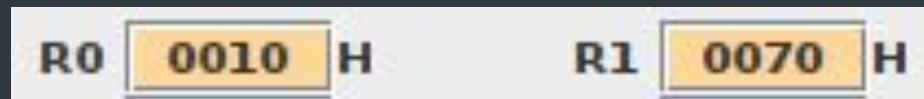


# Stack:

; Utilização

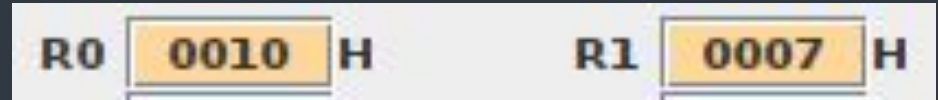
rotina:

```
120H    POP R1
122H    POP R0
124H    RET
```



# Stack:

; Utilização



rotina:

120H  
122H  
124H

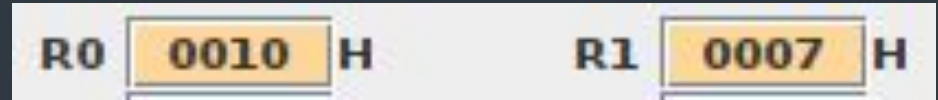
...

POP R1  
POP R0  
RET



# Stack:

; Utilização



rotina:

...  
120H POP R1  
122H POP R0  
124H RET

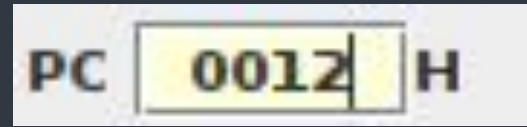
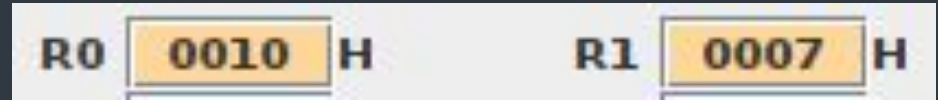


# Stack:

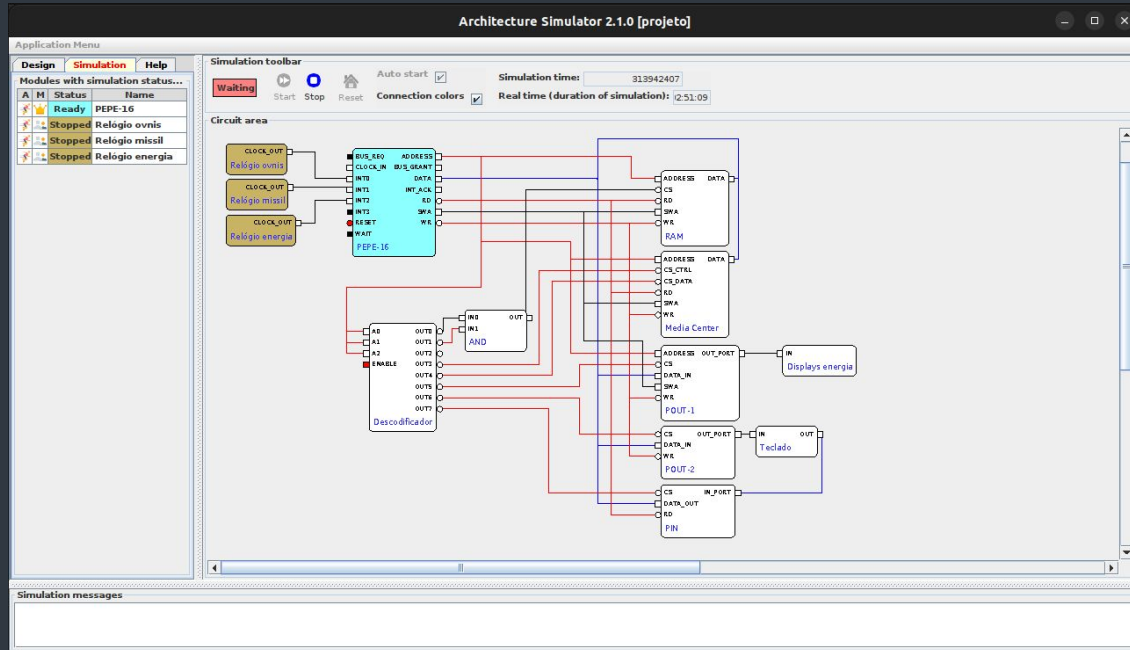
; Utilização

rotina:

...  
120H POP R1  
122H POP R0  
124H RET

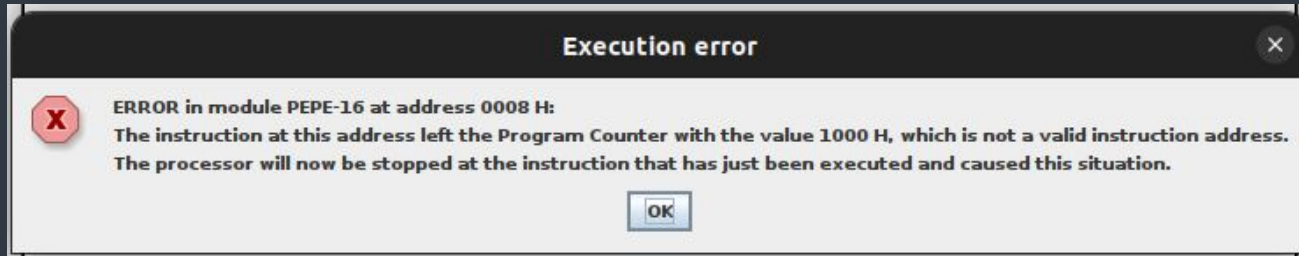
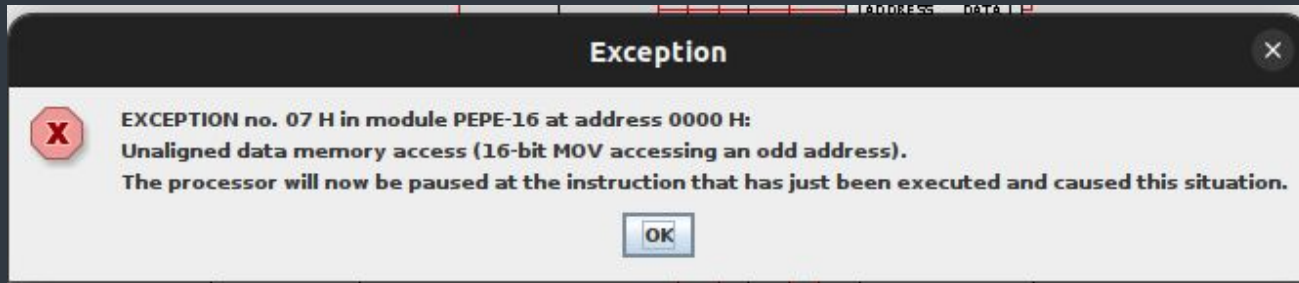


# Exercicio (Live coding):

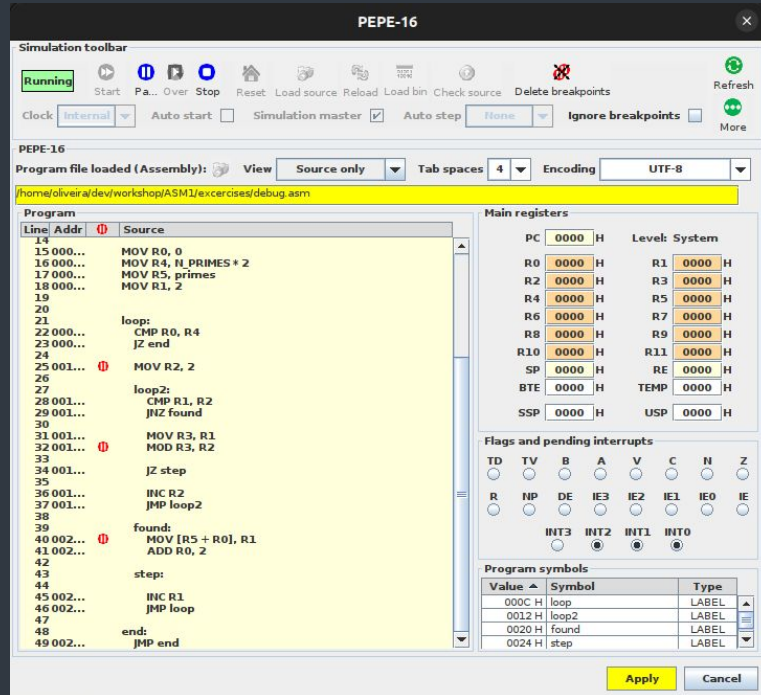




# Erros Comuns e Debugging:



# Exercicio (Live debugging):



# Duvidas: