

PROGRAMA 1:

Código:

```
addi s2, zero, 4  
addi s3, zero, 3  
addi s4, zero, 7  
addi s5, zero, 5  
addi s6, zero, 6  
add s7, s2, s3
```

Parte 1:

Architecture: RV32IM

Forwarding: DEACTIVAT

Branch Hazard Handling: EXECUTE DELAY SLOT

Inserindo código 1 no webriscv

A1) Início da execução:

Instruction Memory:

O código contém as seguintes instruções:

Endereço 0 (0x0): Instrução: addi s2, x0, 4

Adiciona o valor imediato 4 ao registrador x0 (zero) e armazena o resultado no registrador s2.

Endereço 4 (0x4): Instrução: addi s3, x0, 3

Adiciona o valor imediato 3 ao registrador x0 (zero) e armazena o resultado no registrador s3.

Endereço 8 (0x8): Instrução: addi s4, x0, 7

Essa instrução adiciona o valor imediato 7 ao registrador x0 (zero) e armazena o resultado no registrador s4.

Endereço 12 (0xc): Instrução: addi s5, x0, 5

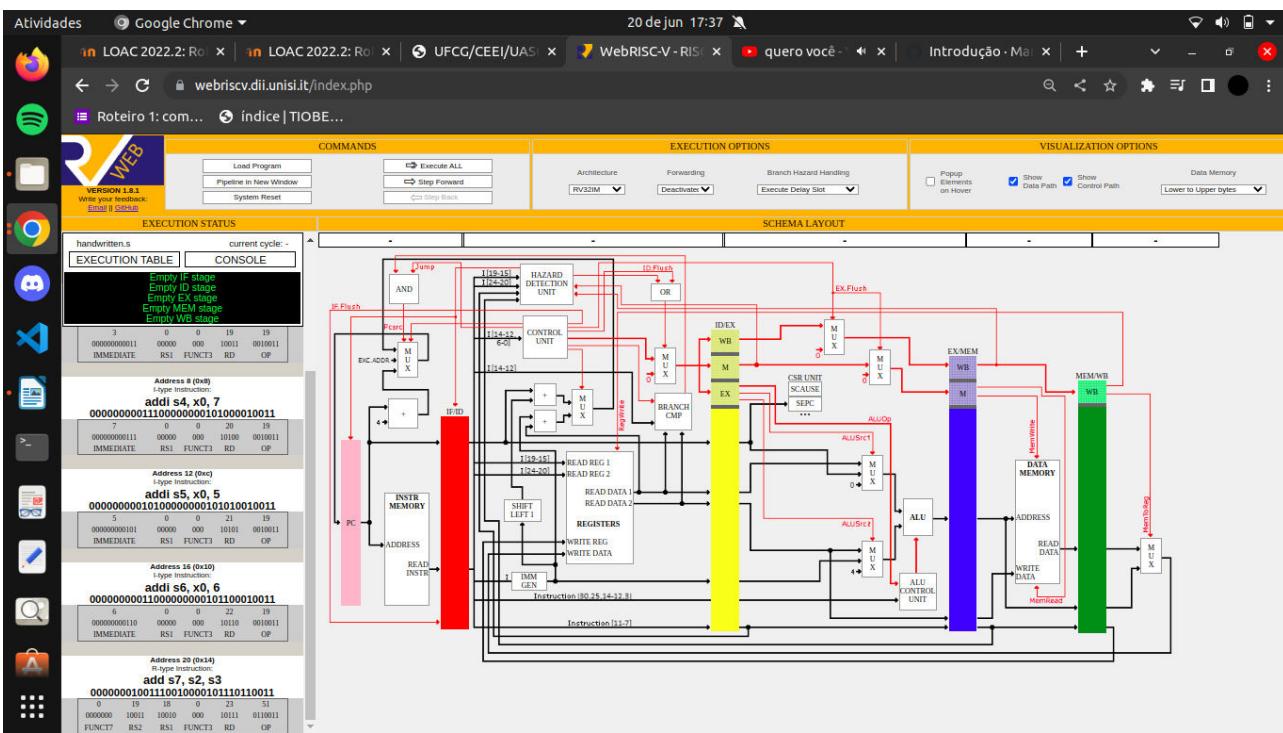
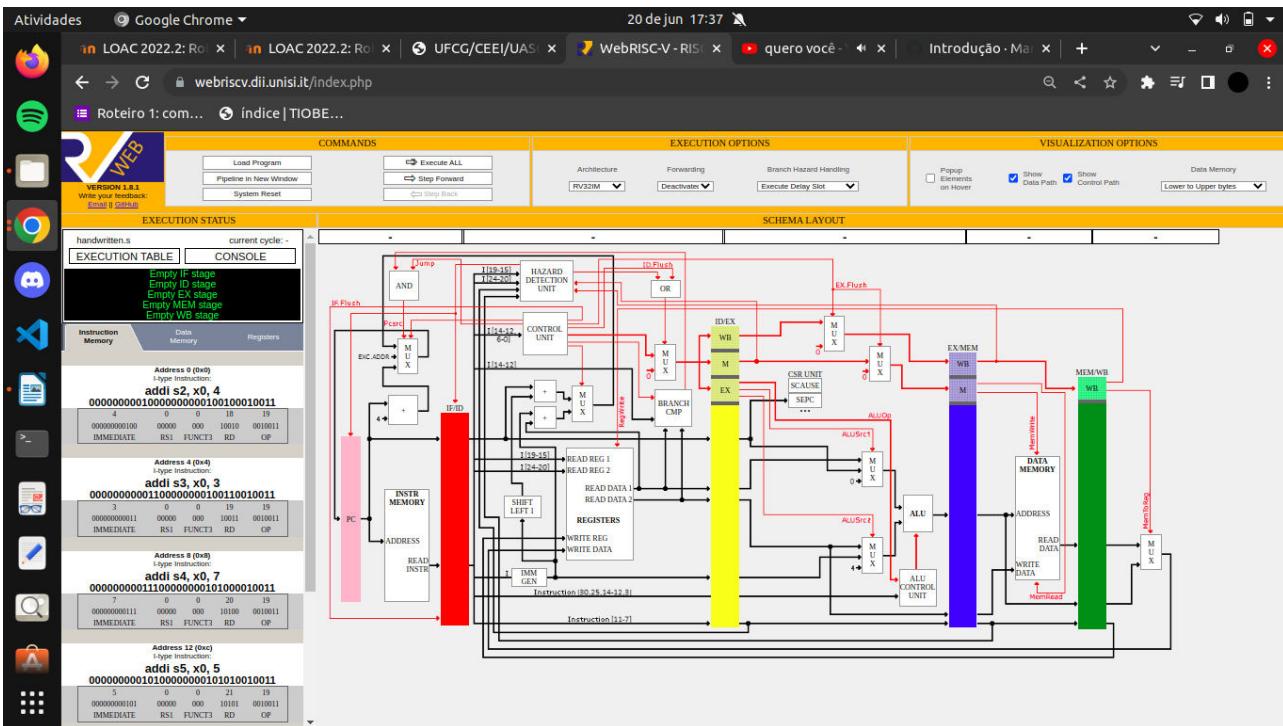
Adiciona o valor imediato 5 ao registrador x0 (zero) e armazena o resultado no registrador s5.

Endereço 16 (0x10): Instrução: addi s6, x0, 6

Adiciona o valor imediato 6 ao registrador x0 (zero) e armazena o resultado no registrador s6.

Endereço 20 (0x14) add s7, s2, s3

Por fim, Adiciona os valores dos registradores s2 e s3 e armazena o resultado no registrador s7.



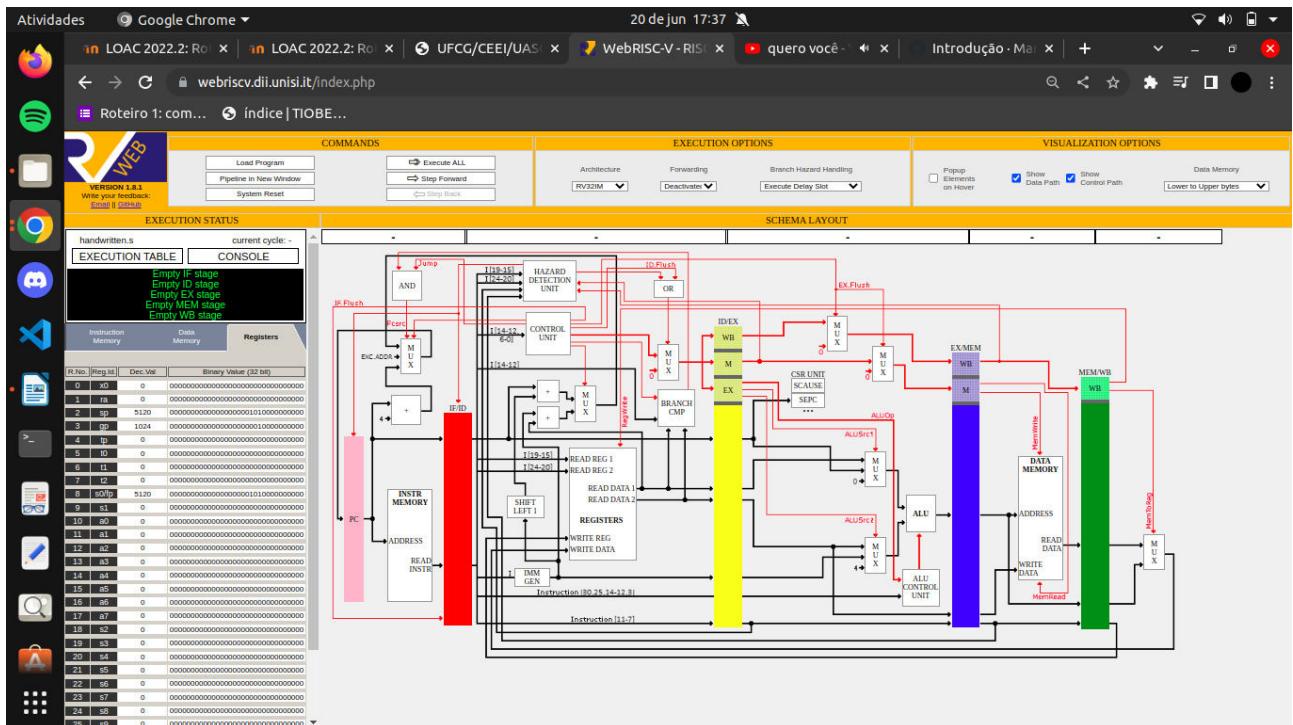
Registers:

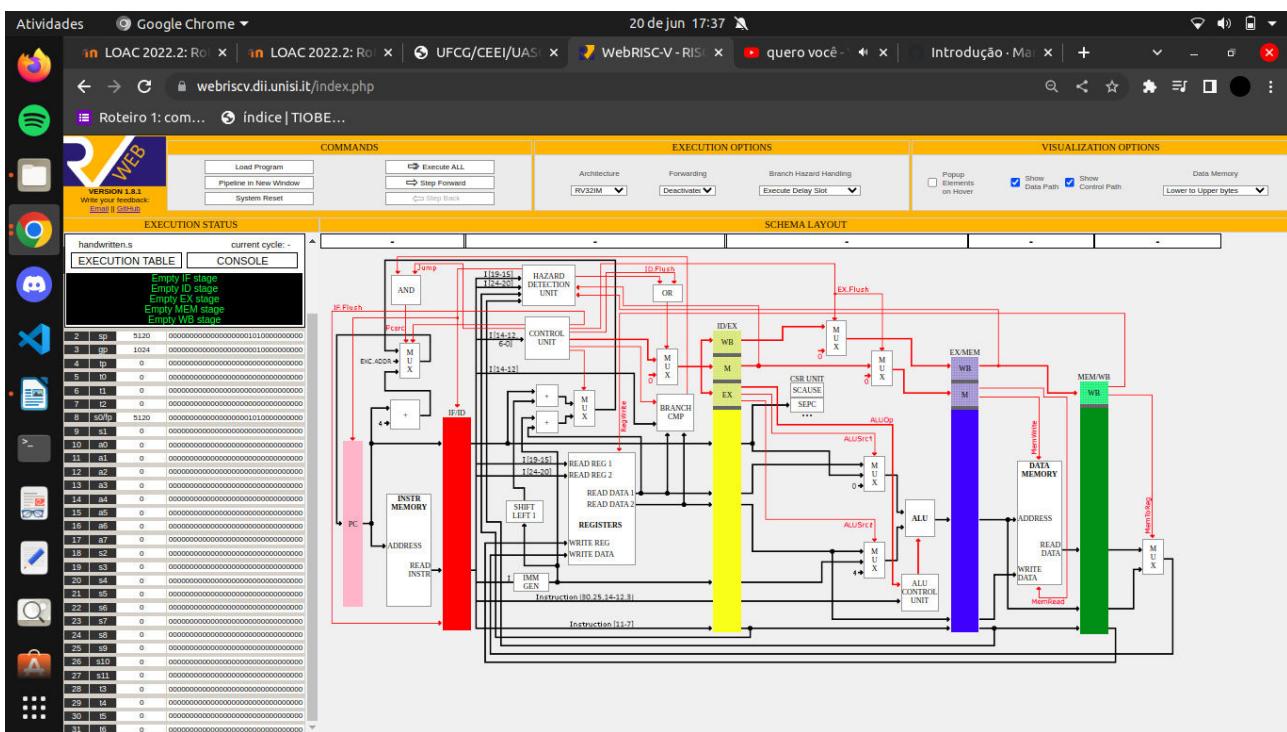
- SP (Stack Pointer): É usado para rastrear o topo da pilha. A pilha é uma área de memória utilizada para armazenar dados temporários, registros salvos e informações relacionadas a chamadas de função.

- GP (Global Pointer): É usado para acessar dados globais. Ele geralmente aponta para uma área de memória que contém variáveis globais ou outros dados compartilhados entre diferentes partes do programa.

- TP (Thread Pointer): É usado para acesso a dados específicos da thread. Em um ambiente multithread, cada thread pode ter seu próprio TP para acessar seus dados exclusivos.

- FP (Frame Pointer) ou S0 (Saved Register 0): É usado como um ponto de referência para acessar variáveis locais e registros salvos no quadro (frame) atual. Ele ajuda a navegar na pilha e acessar os valores corretos durante a execução das chamadas de função.

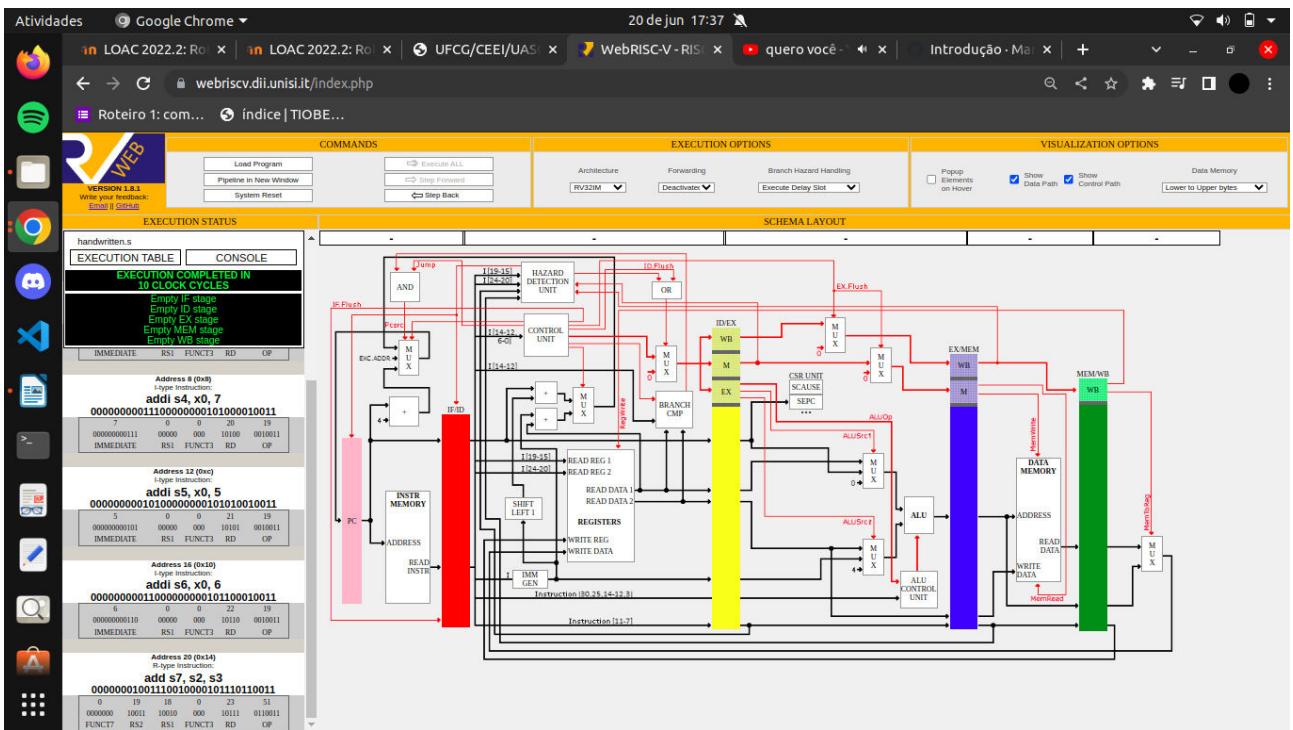
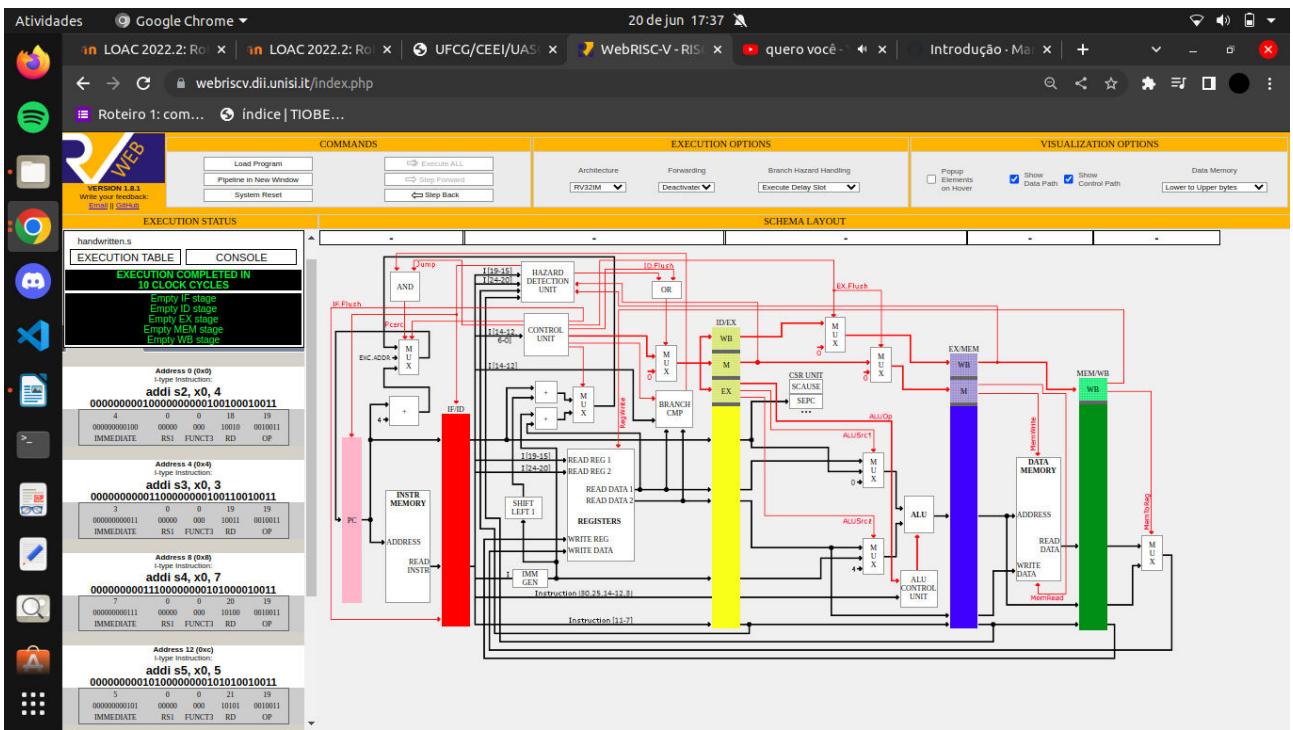




Final da execução:

Instruction Memory:

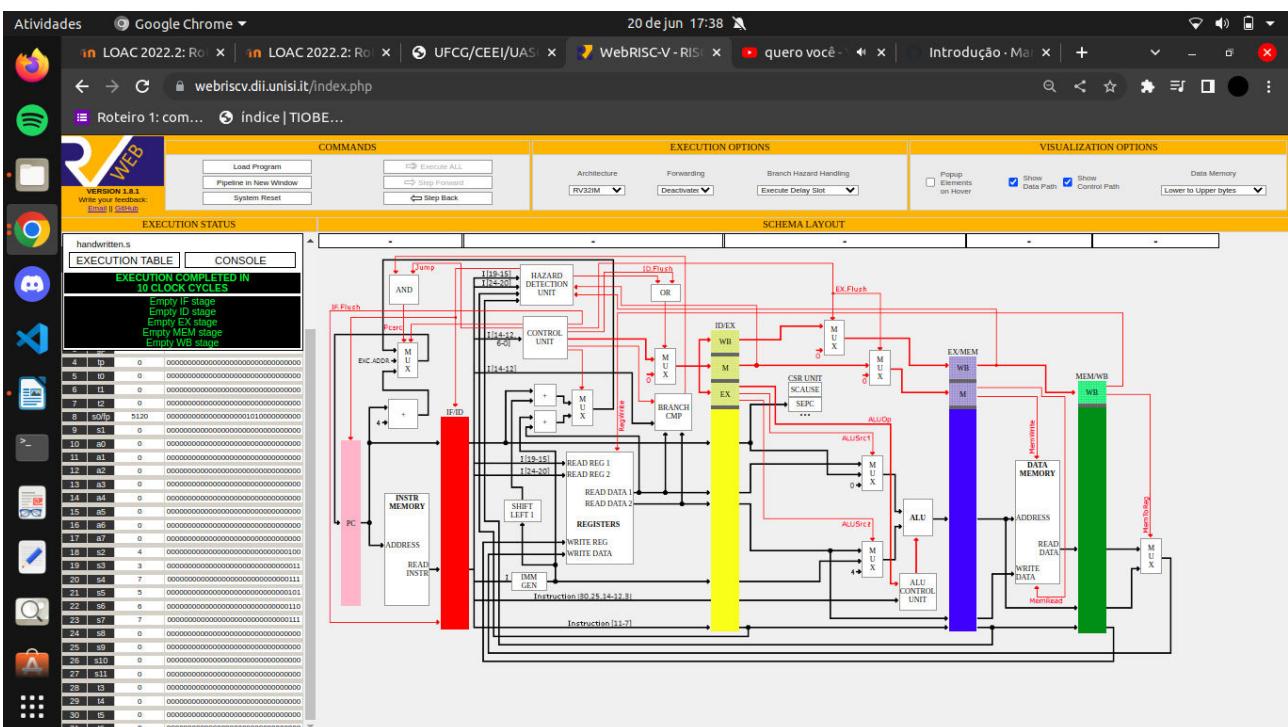
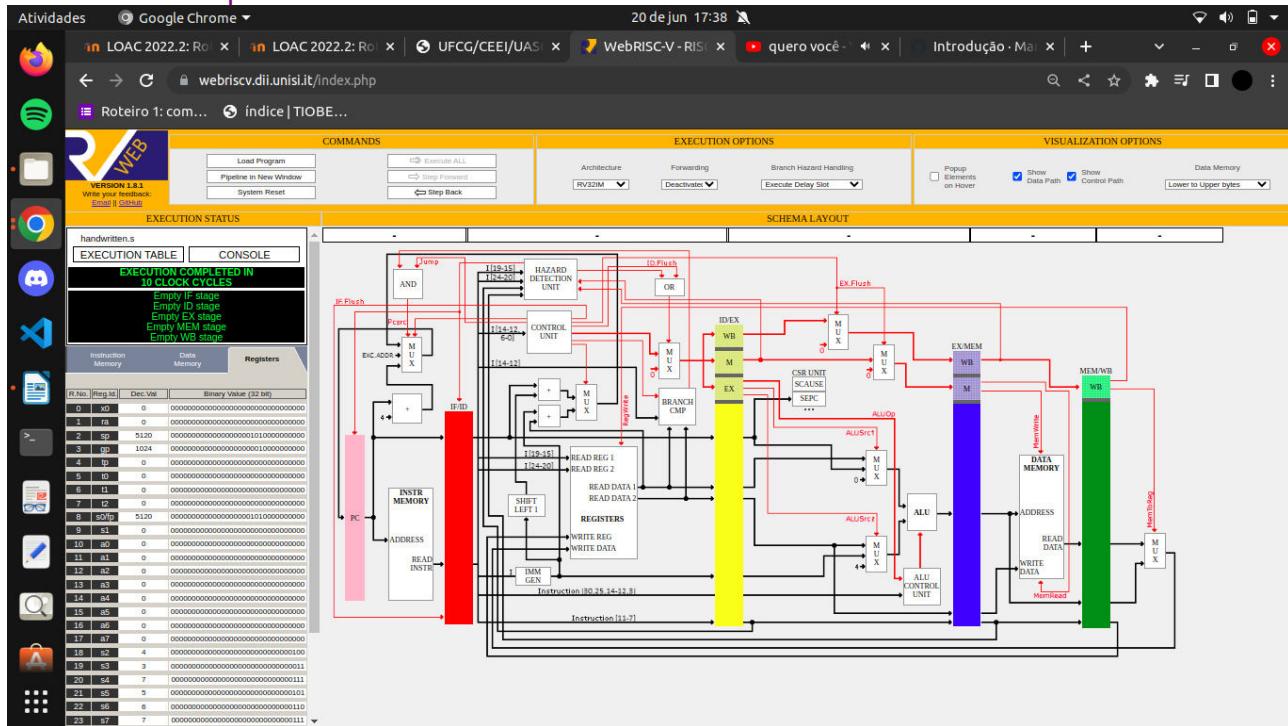
A "Instruction Memory" é lida sequencialmente pelo processador durante a etapa de busca de instruções, onde cada instrução é buscada em seu endereço correspondente na memória e enviada para as etapas subsequentes do pipeline para serem executadas. Após a leitura da instrução, seu conteúdo não é alterado na "Instruction Memory". Essa é uma característica importante do modelo de execução de um processador com pipeline, onde as instruções são buscadas e executadas em paralelo para melhorar o desempenho.



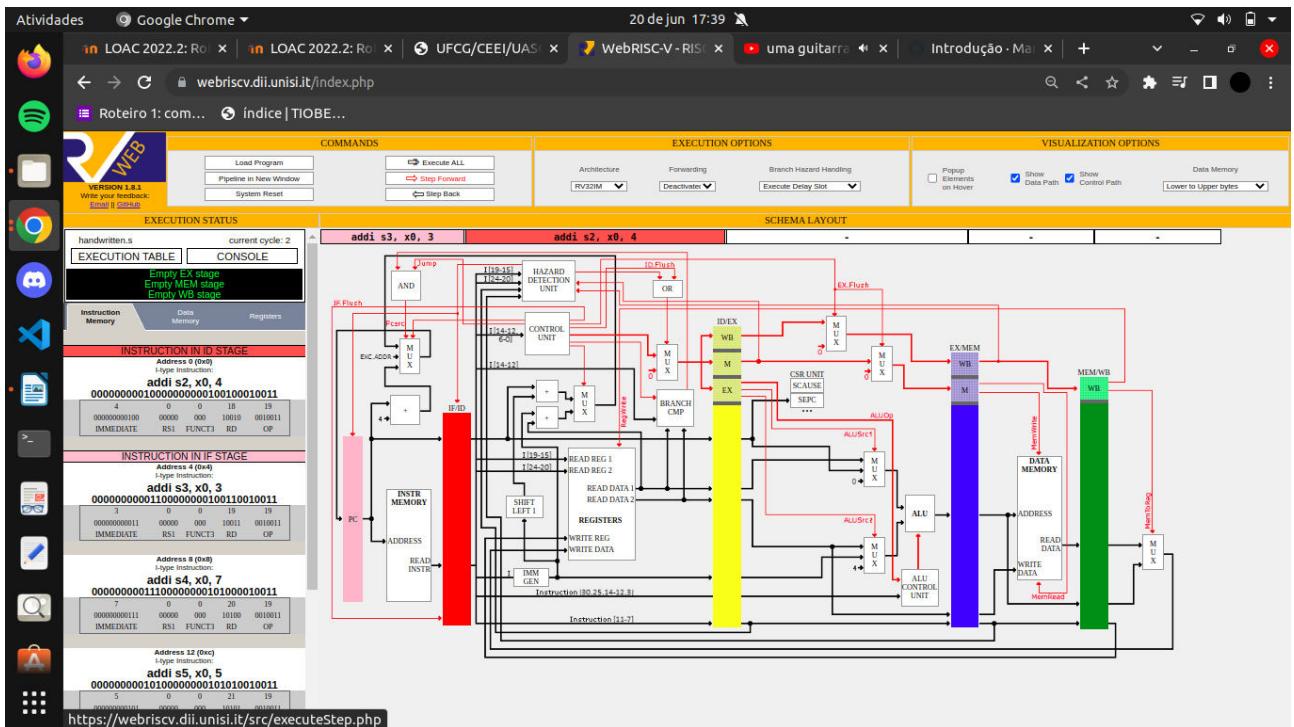
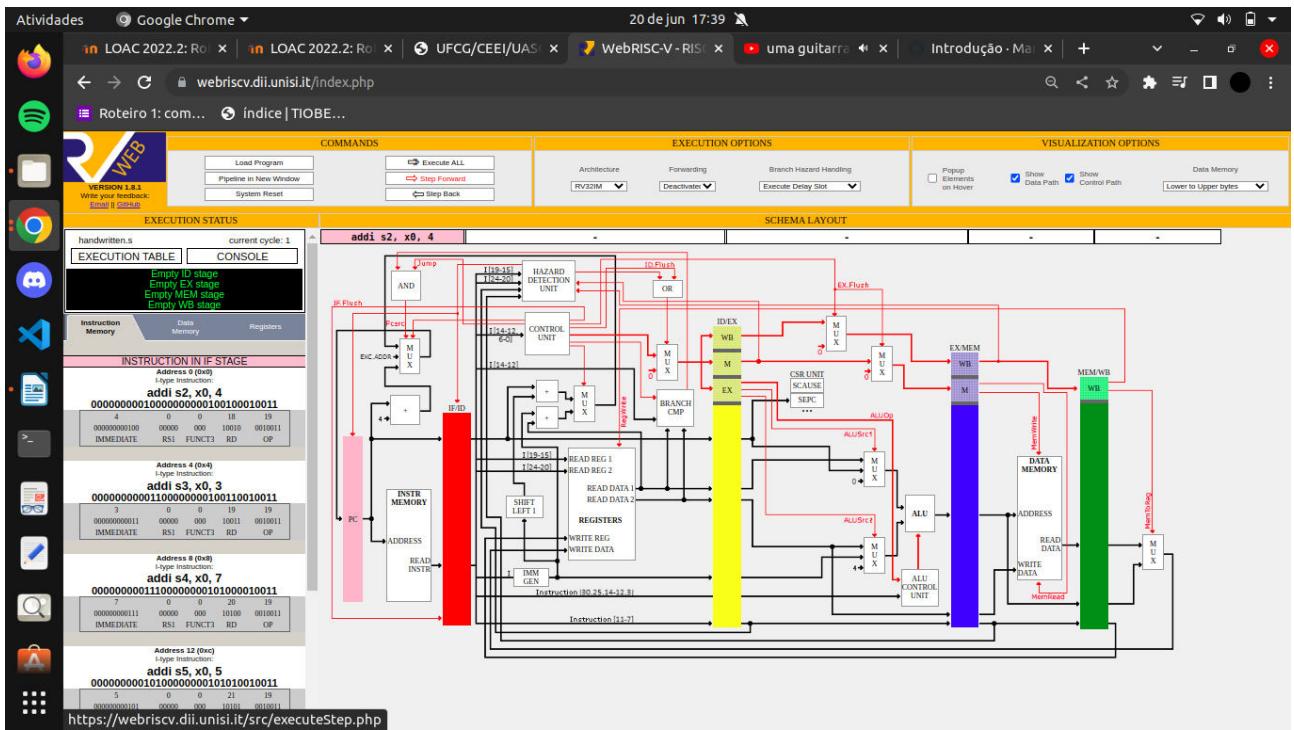
Registers:

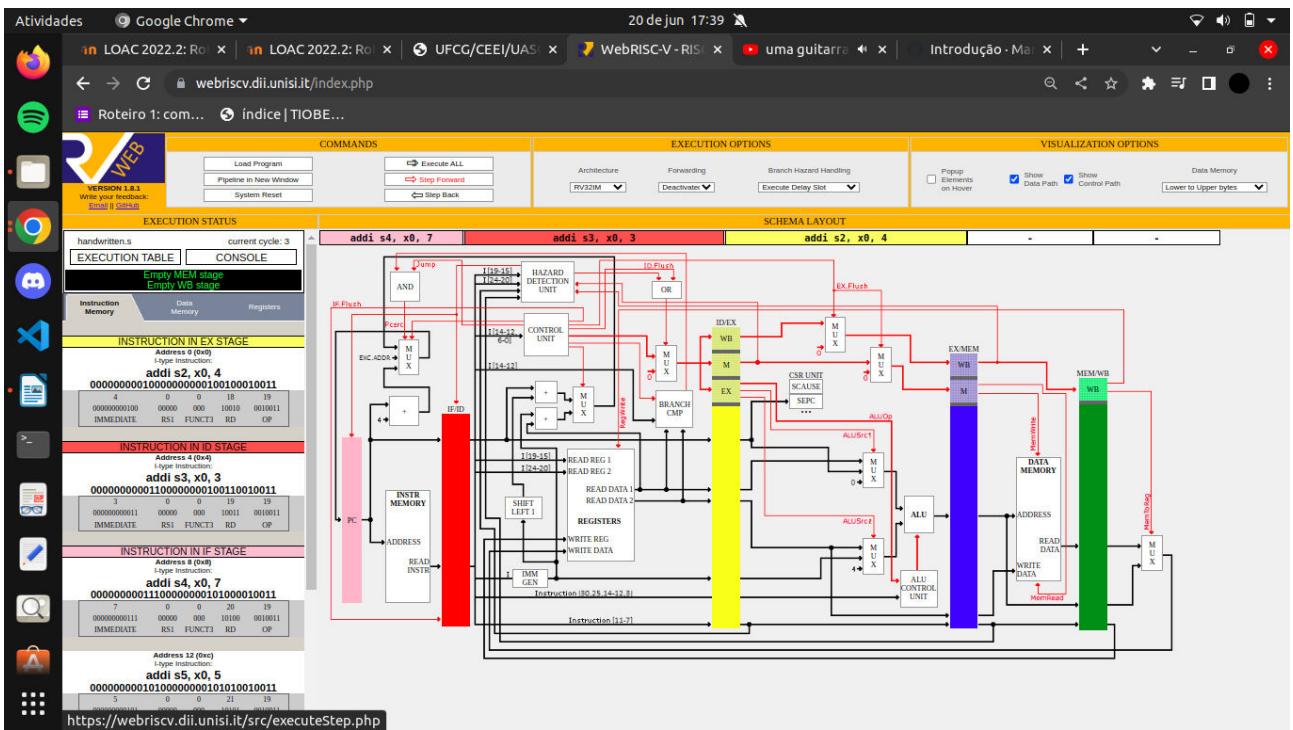
s2 é atualizado para o valor 4
 s3 é atualizado para o valor 3
 s4 é atualizado para o valor 7
 s5 é atualizado para o valor 5
 s6 é atualizado para o valor 6

s7 é atualizado para o valor 7



B1) Passagem em três estágios representativos do Pipeline (“SCHEMA LAYOUT”):



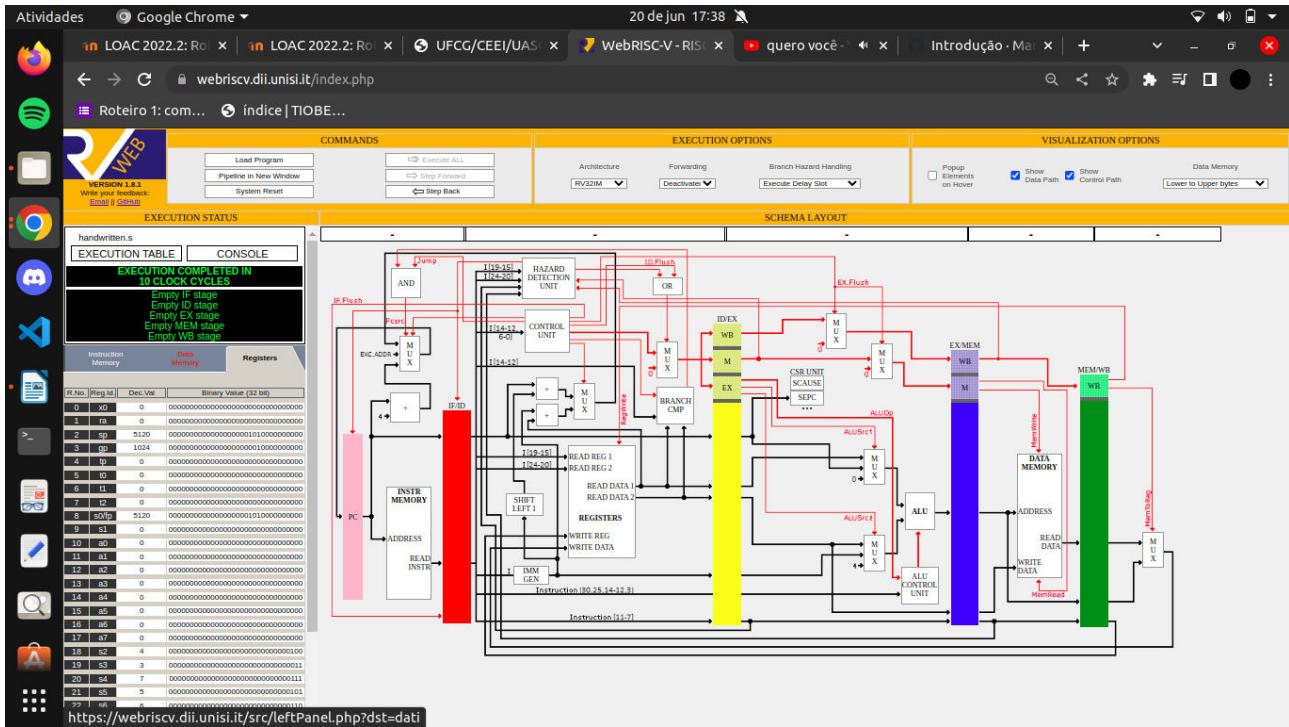


D1) Resultado final da execução em Pipeline, por meio da Tabela da Execução do Programa (“EXECUTION TABLE”):

CPU Cycles										
Instruction	1	2	3	4	5	6	7	8	9	10
addi \$2, x0, 4	F	D	X	M	W					
addi \$3, x0, 3	F	D	X	M	W					
addi \$4, x0, 7	F	D	X	M	W					
addi \$5, x0, 5		F	D	X	M	W				
addi \$6, x0, 6		F	D	X	M	W				
addi \$7, x0, 1			F	D	X	M	W			

E1) Ciclos de CPU necessários para executar esse programa:

Resposta: 10 Ciclos. 6 intruções e 5 etapas: 1 * 5 + 5 = 10



Parte 2:

Architecture: RV32IM

Forwarding: ACTIVATED

Branch Hazard Handling: EXECUTE DELAY SLOT

The screenshot shows the WebRISC-V simulator interface. At the top, there are tabs for "Atividades" and "Google Chrome". The main window displays the "EXECUTION STATUS" section for the file "handwritten.s" with a current cycle of 3. It shows four stages: EXECUTION TABLE, CONSOLE, INSTRUCTION IN EX STAGE, INSTRUCTION IN ID STAGE, INSTRUCTION IN IF STAGE, and INSTRUCTION IN I STAGE. Each stage has an assembly instruction and its binary representation. The INSTRUCTION IN EX STAGE shows the instruction: addi \$2, x0, 4. The INSTRUCTION IN ID STAGE shows addi \$3, x0, 3. The INSTRUCTION IN IF STAGE shows addi \$4, x0, 7. The INSTRUCTION IN I STAGE shows addi \$5, x0, 5. The right side of the interface contains the "ASSEMBLY EDITOR" which lists the assembly code and its corresponding opcodes. Below the assembly editor are sections for "List of Text Instructions", "List of Data Directives", and "Supported System Calls". The "List of Text Instructions" table includes entries like addi \$2, zero, 4, addi \$3, zero, 3, addi \$4, zero, 7, addi \$5, zero, 5, addi \$6, zero, 6, and add \$7, \$2, \$3.

A2) Início da execução:

Endereço 0 (0x0): addi s2, x0, 4

Essa instrução adiciona o valor imediato 4 ao registrador x0 e armazena o resultado no registrador s2.

Endereço 4 (0x4): addi s3, x0, 3

Essa instrução adiciona o valor imediato 3 ao registrador x0 e armazena o resultado no registrador s3.

Endereço 8 (0x8): addi s4, x0, 7

Essa instrução adiciona o valor imediato 7 ao registrador x0 e armazena o resultado no registrador s4.

Endereço 12 (0xc): addi s5, x0, 5

Essa instrução adiciona o valor imediato 5 ao registrador x0 e armazena o resultado no registrador s5.

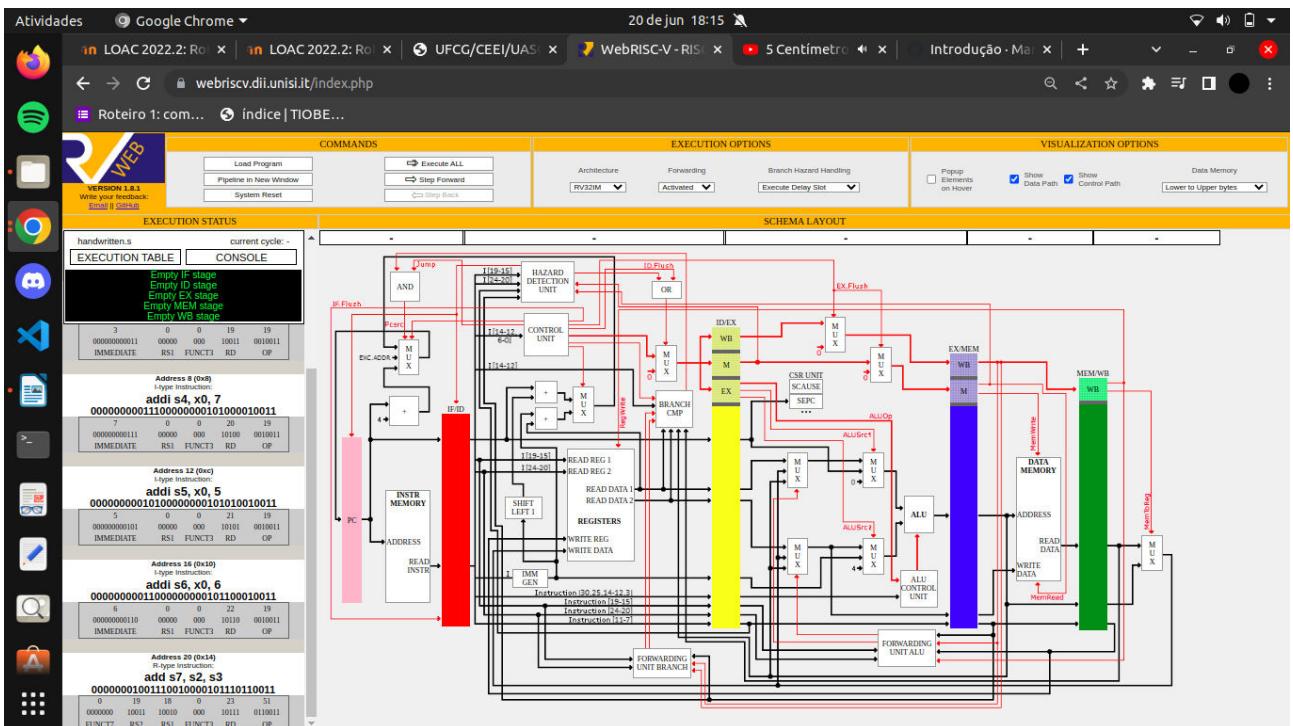
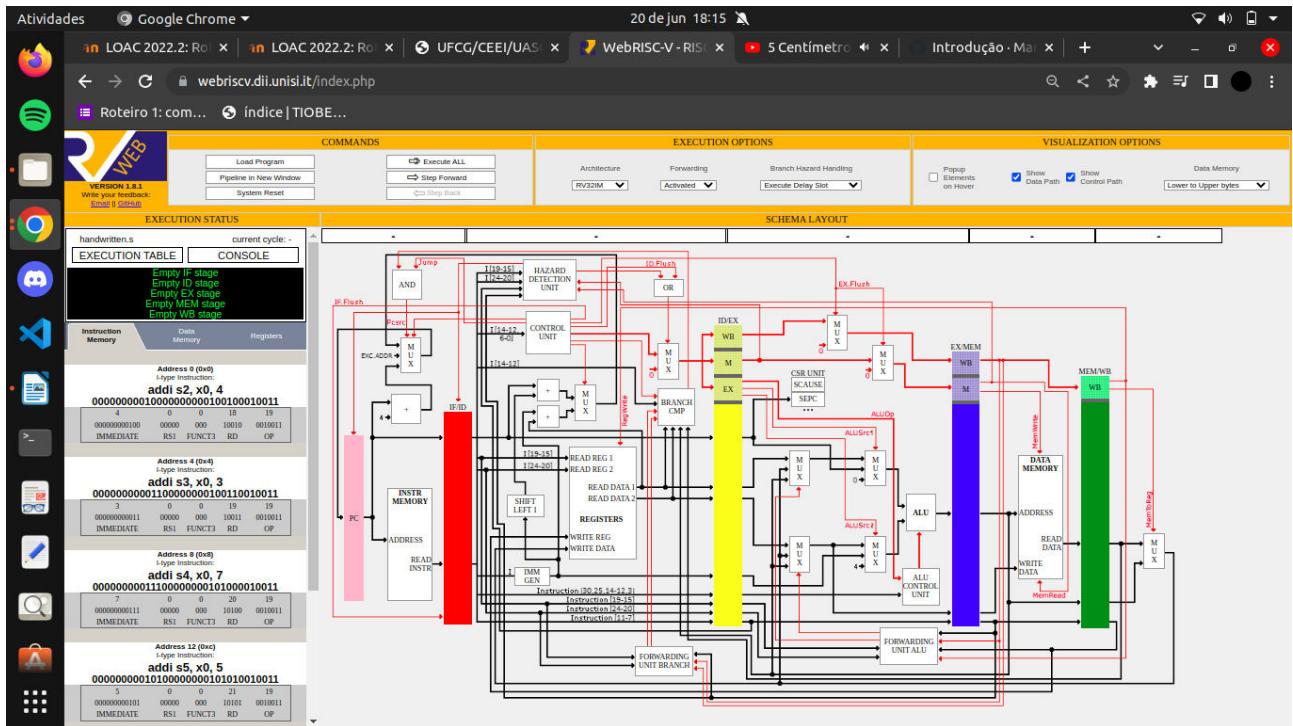
Endereço 16 (0xc): addi s6, x0, 6

Essa instrução adiciona o valor imediato 6 ao registrador x0 e armazena o resultado no registrador s6.

Endereço 20 (0x10): add s7, s2, s3

Essa instrução adiciona o valor do registrador s2 ao valor do registrador s3 e armazena o resultado no

registraror s7



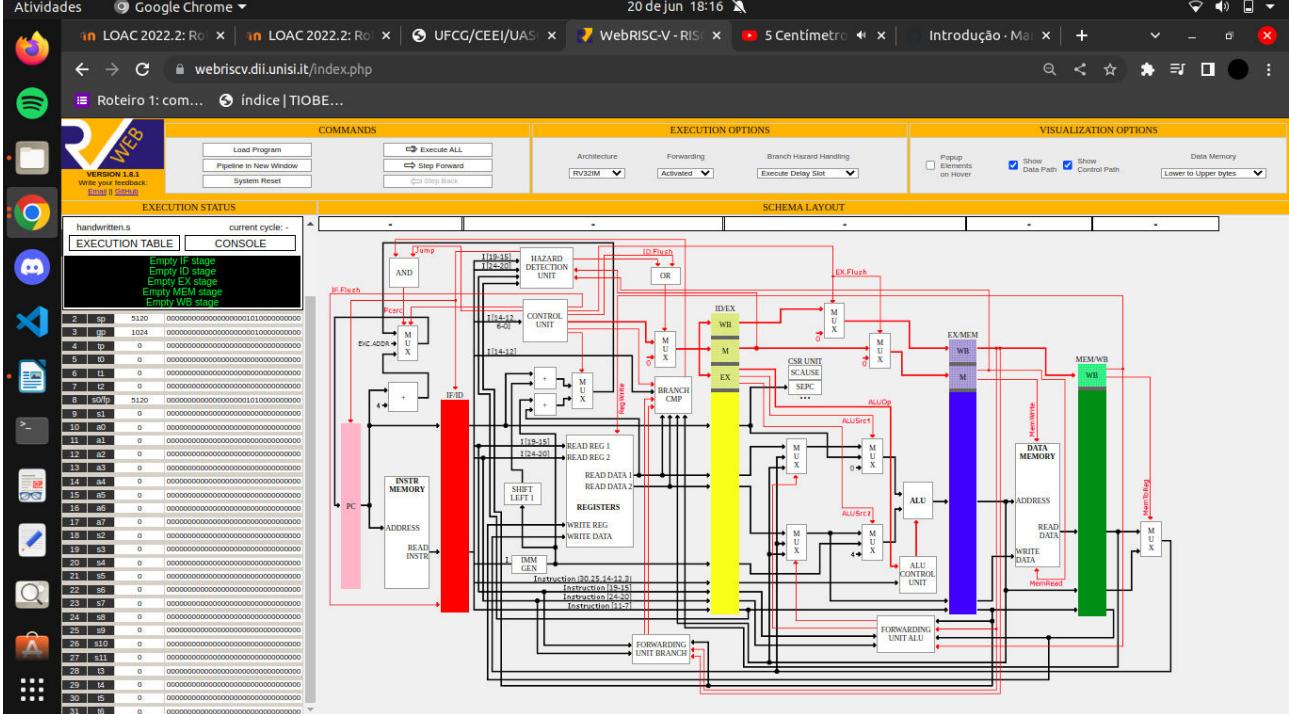
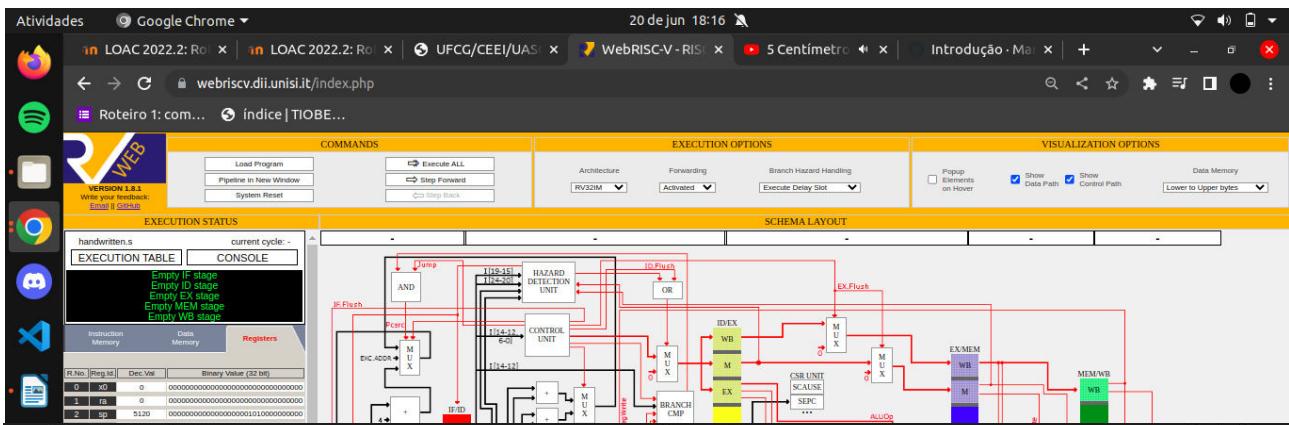
Registers:

- SP (Stack Pointer): É usado para rastrear o topo da pilha. A pilha é uma área de memória utilizada para armazenar dados temporários, registros salvos e informações relacionadas a chamadas de função.

- GP (Global Pointer): É usado para acessar dados globais. Ele geralmente aponta para uma área de memória que contém variáveis globais ou outros dados compartilhados entre diferentes partes do programa.

- TP (Thread Pointer): É usado para acesso a dados específicos da thread. Em um ambiente multithread, cada thread pode ter seu próprio TP para acessar seus dados exclusivos.

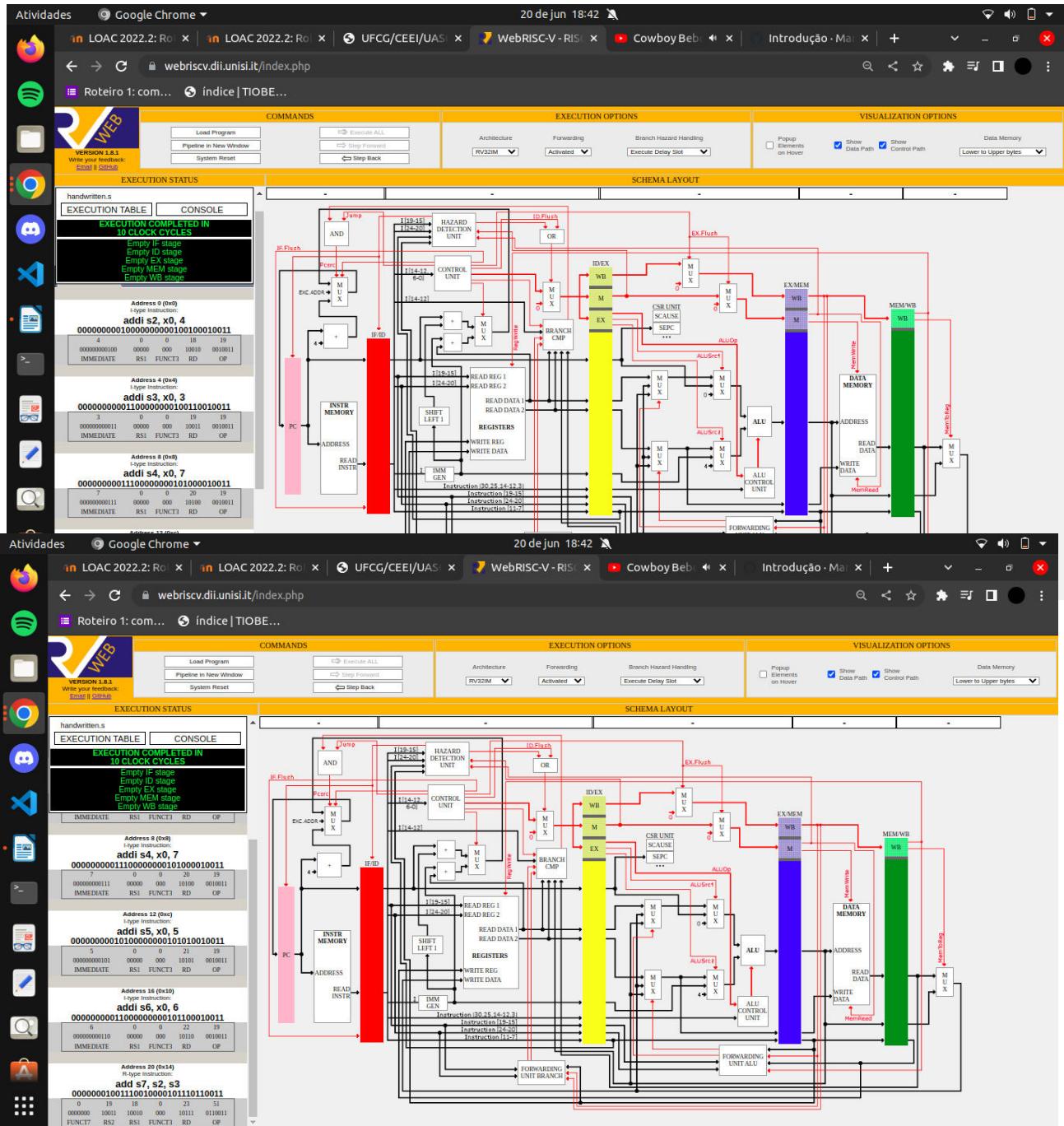
- FP (Frame Pointer) ou S0 (Saved Register 0): É usado como um ponto de referência para acessar variáveis locais e registros salvos no quadro (frame) atual. Ele ajuda a navegar na pilha e acessar os valores corretos durante a execução das chamadas de função.



Final da execução:

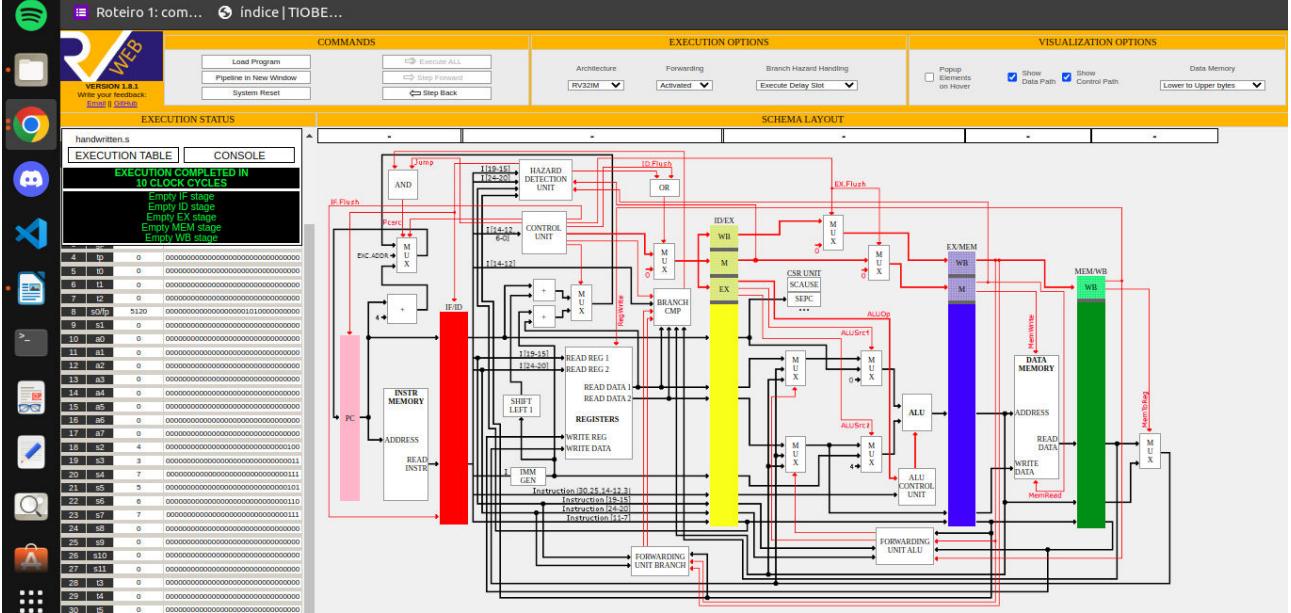
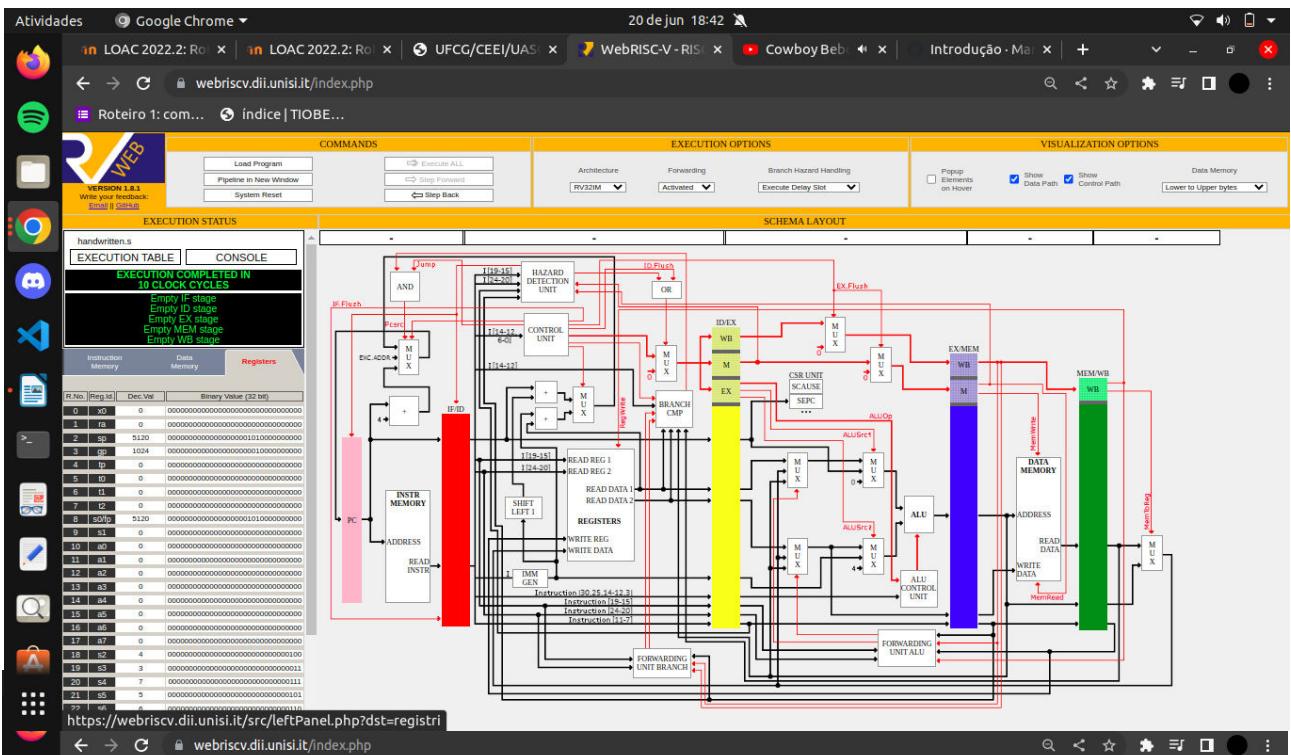
Instruction Memory:

A "Instruction Memory" é uma área de memória dedicada ao armazenamento das instruções do programa. Uma vez que as instruções são carregadas na memória, elas não são alteradas durante a execução do programa. Portanto, o conteúdo da "Instruction Memory" permanece inalterado após a execução do código.

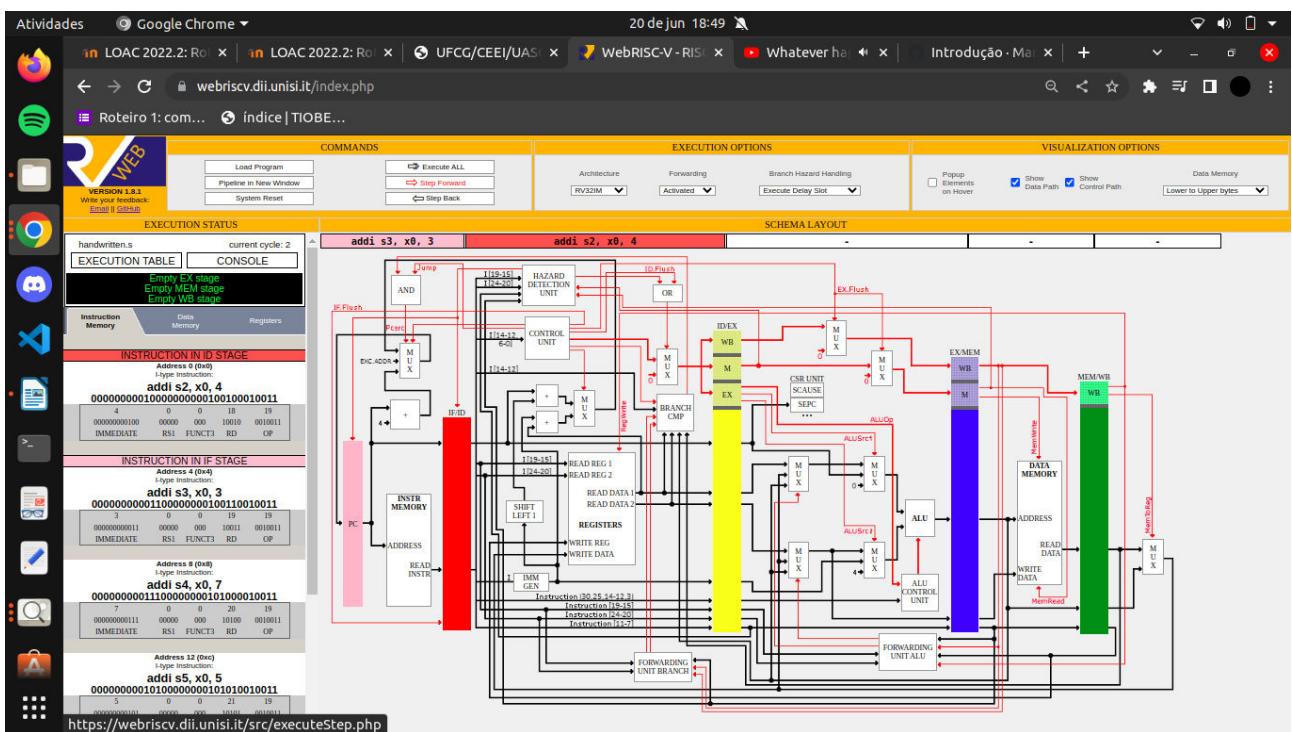
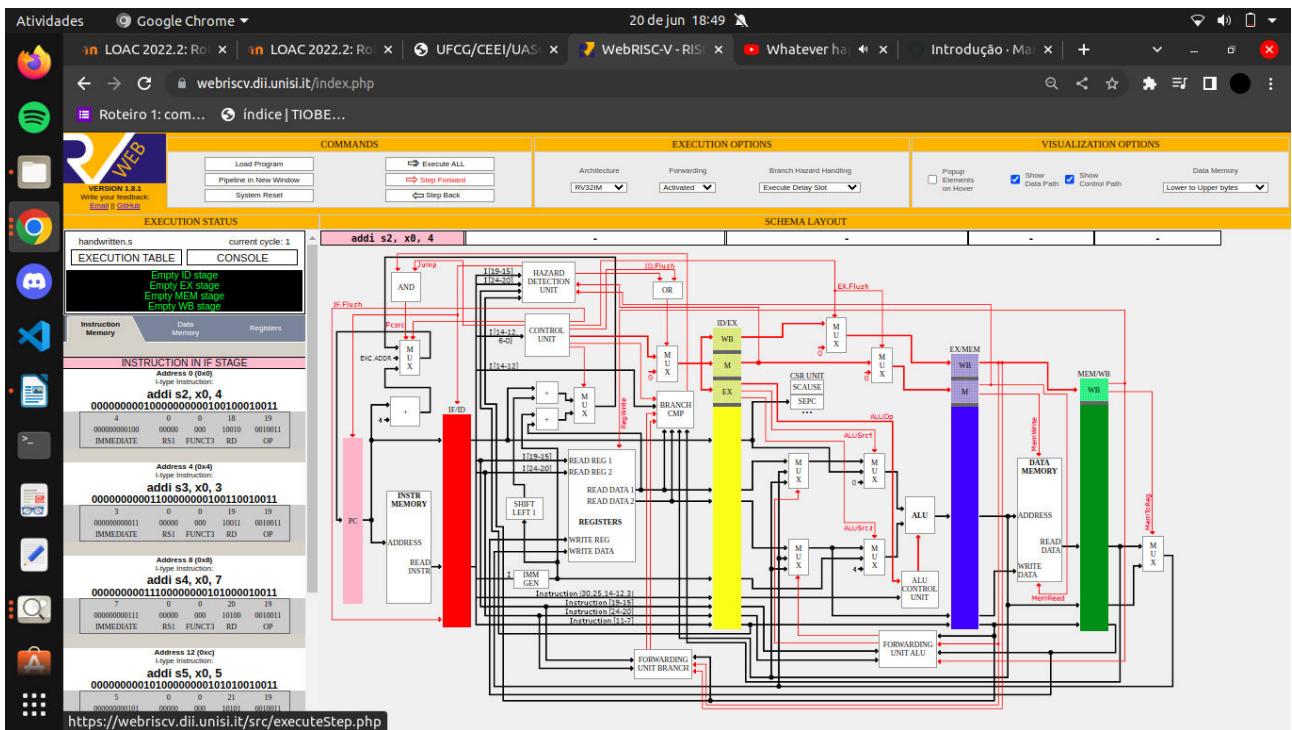


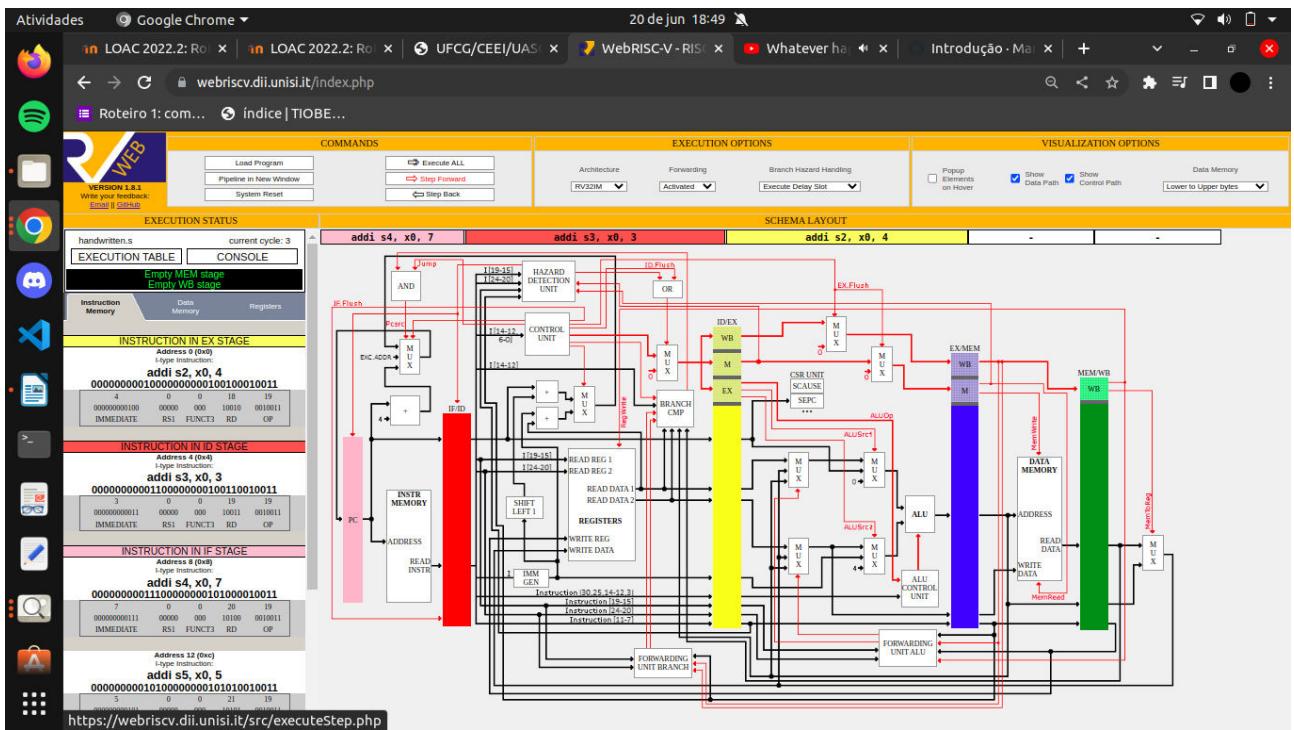
Registers:

- s2 atualizado para o valor 4
- s3 atualizado para o valor 3
- s4 atualizado para o valor 7
- s5 atualizado para o valor 5
- s6 atualizado para o valor 6
- s7 atualizado para o valor 7



B2) Passagem em três estágios representativos do Pipeline (“SCHEMA LAYOUT”):





D2) Resultado final da execução em Pipeline, por meio da Tabela da Execução do Programa (“EXECUTION TABLE”):

Atividades Google Chrome 20 de jun 18:42

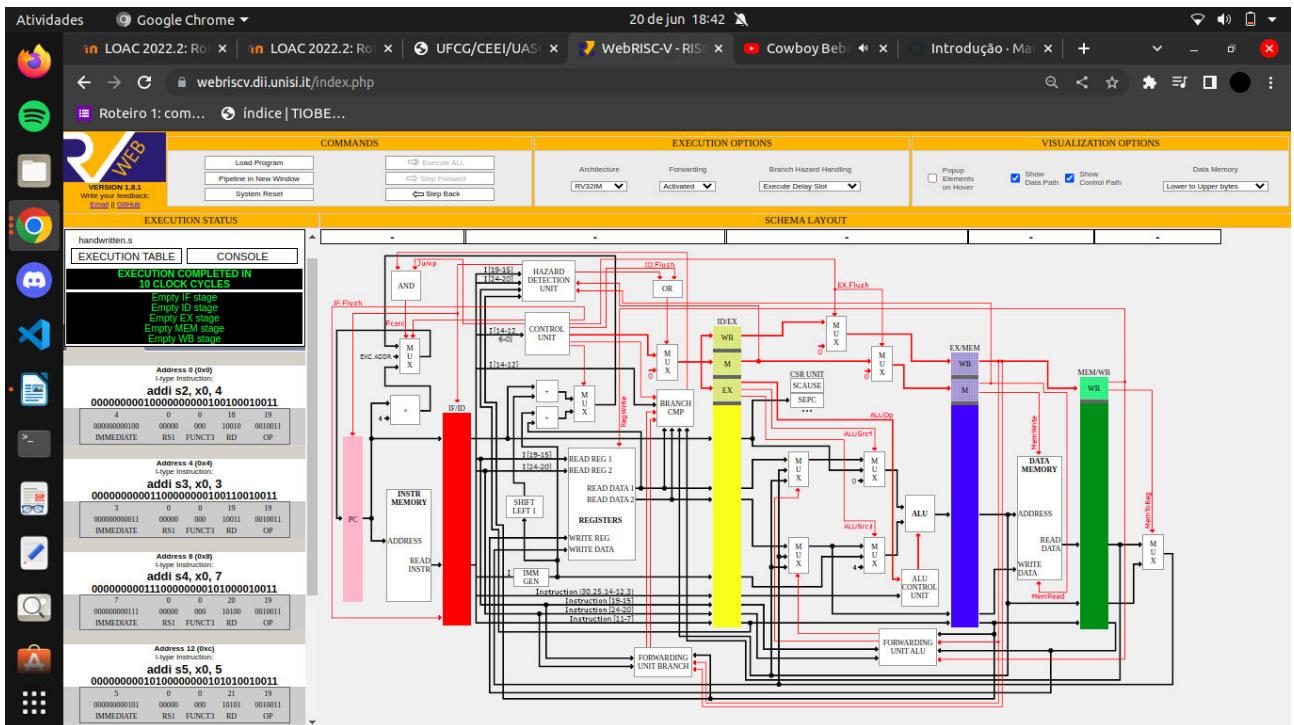
WebRISC-V - RISC-V PIPELINED DATAPATH SIMULATION ONLINE - Google Chrome

EXECUTION TABLE

(FULL LOOPS)		CPU Cycles									
Instruction	1	2	3	4	5	6	7	8	9	10	
addi \$2, x0, 4	F	D	X	M	W						
addi \$3, x0, 3	F	D	X	M	W						
addi \$4, x0, 7	F	D	X	M	W						
addi \$5, x0, 5		F	D	X	M	W					
addi \$6, x0, 6		F	D	X	M	W					
addi \$7, x0, 3			F	D	X	M	W				

E2) Ciclos de CPU necessários para executar esse programa:

Resposta: 10 Ciclos. $1 * 5 + 5 = 10$



PROGRAMA 2:

Código:

```

addi $2, zero, 4
add $3, zero, $2
addi $4, zero, 7
addi $5, zero, 5
addi $6, zero, 6
add $7, $6, $1

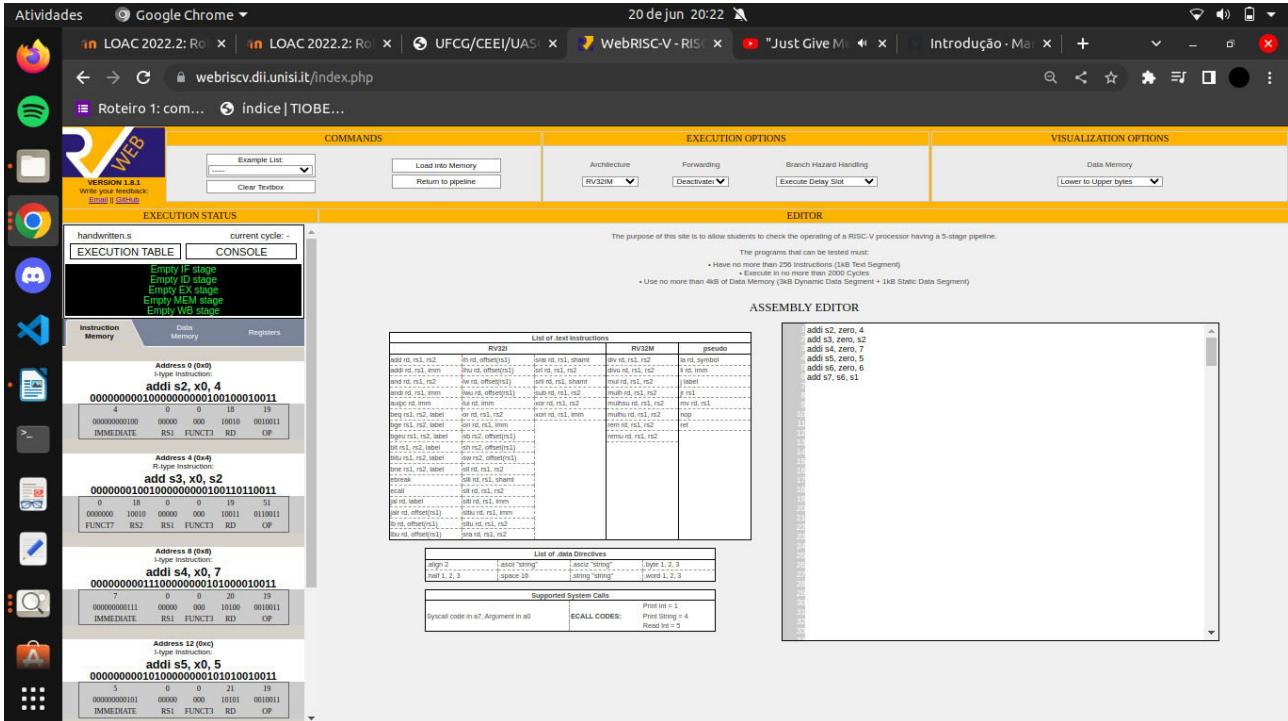
```

Parte 1:

Architecture: RV32IM

Forwarding: DEACTIVAT

Branch Hazard Handling: EXECUTE DELAY SLOT



A1) Início da execução:

Instruction Memory:

Endereço 0 (0x0): addi s2, x0, 4

Adiciona o valor imediato 4 ao registrador s2.

Endereço 4 (0x4): add s3, x0, s2

Adiciona o valor do registrador s2 ao registrador s3.

Endereço 8 (0x8): addi s4, x0, 7

Adiciona o valor imediato 7 ao registrador s4.

Endereço 12 (0xc): addi s5, x0, 5

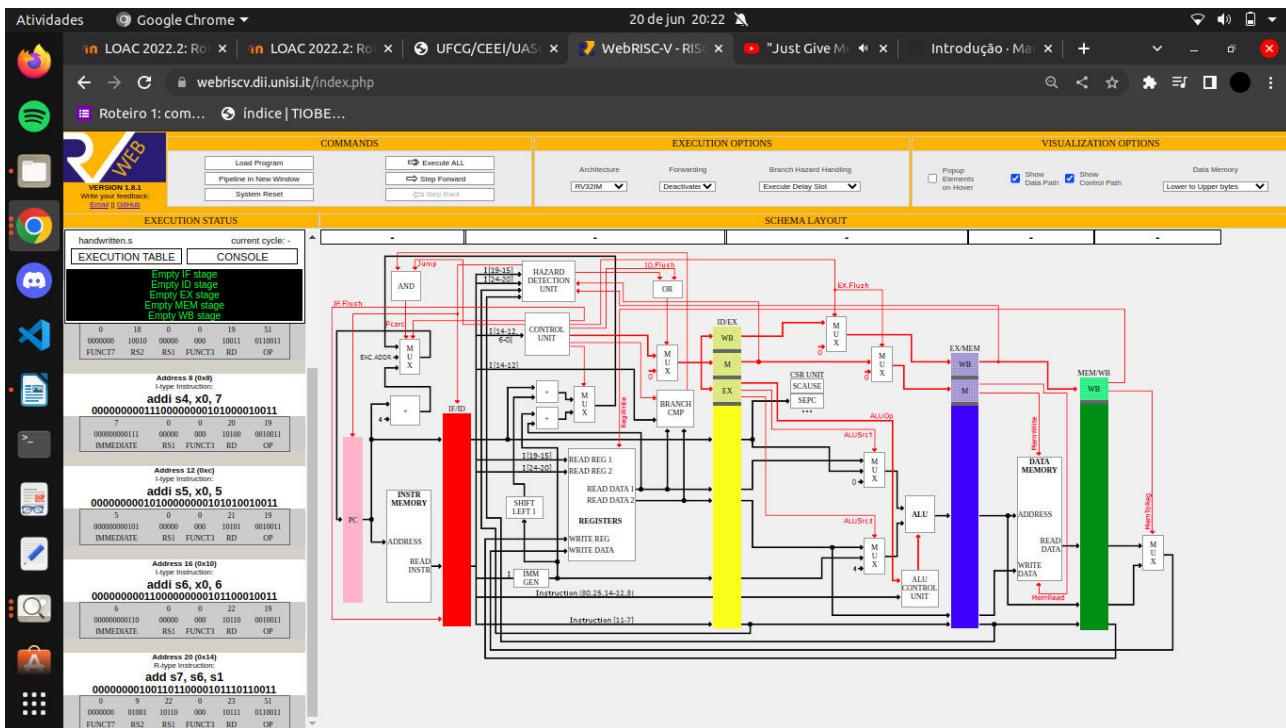
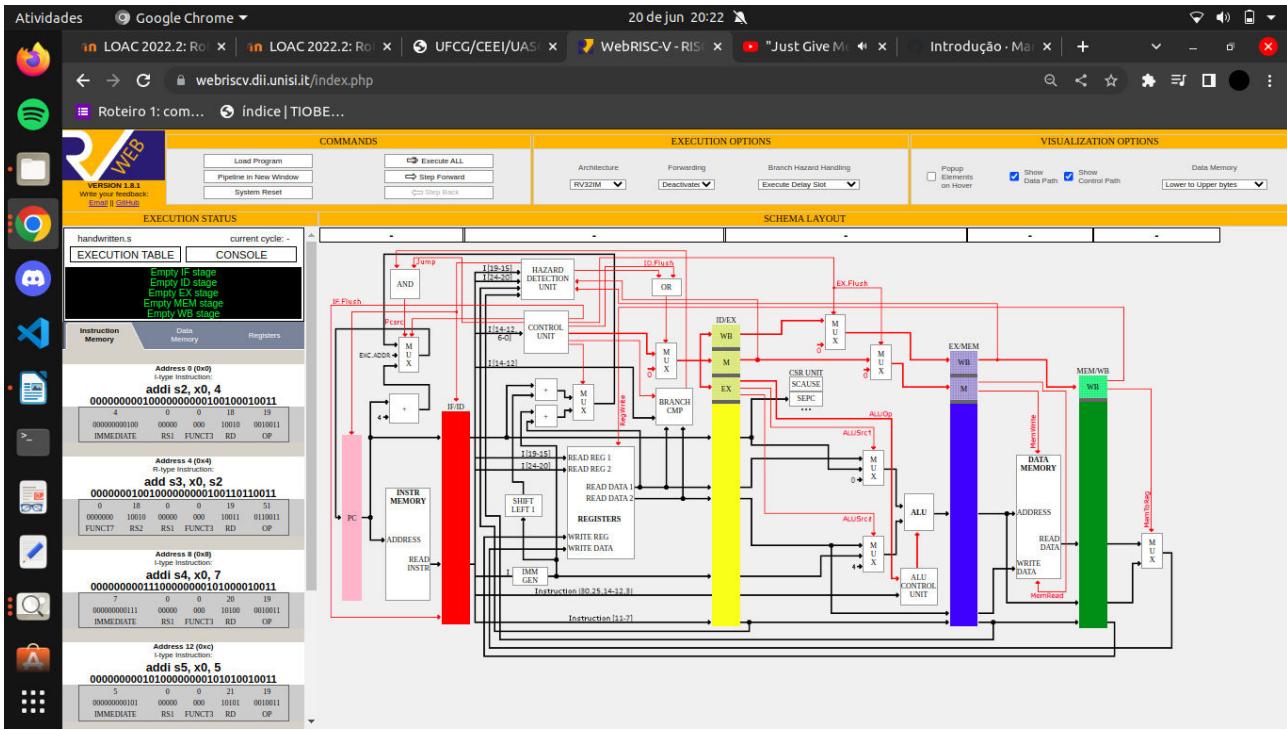
Adiciona o valor imediato 5 ao registrador s5.

Endereço 16 (0x10): addi s6, x0, 6

Adiciona o valor imediato 6 ao registrador s6.

Endereço 20 (0x14): add s7, s6, s1

Adiciona o valor do registrador s6 ao registrador s1 e armazena o resultado em s7.



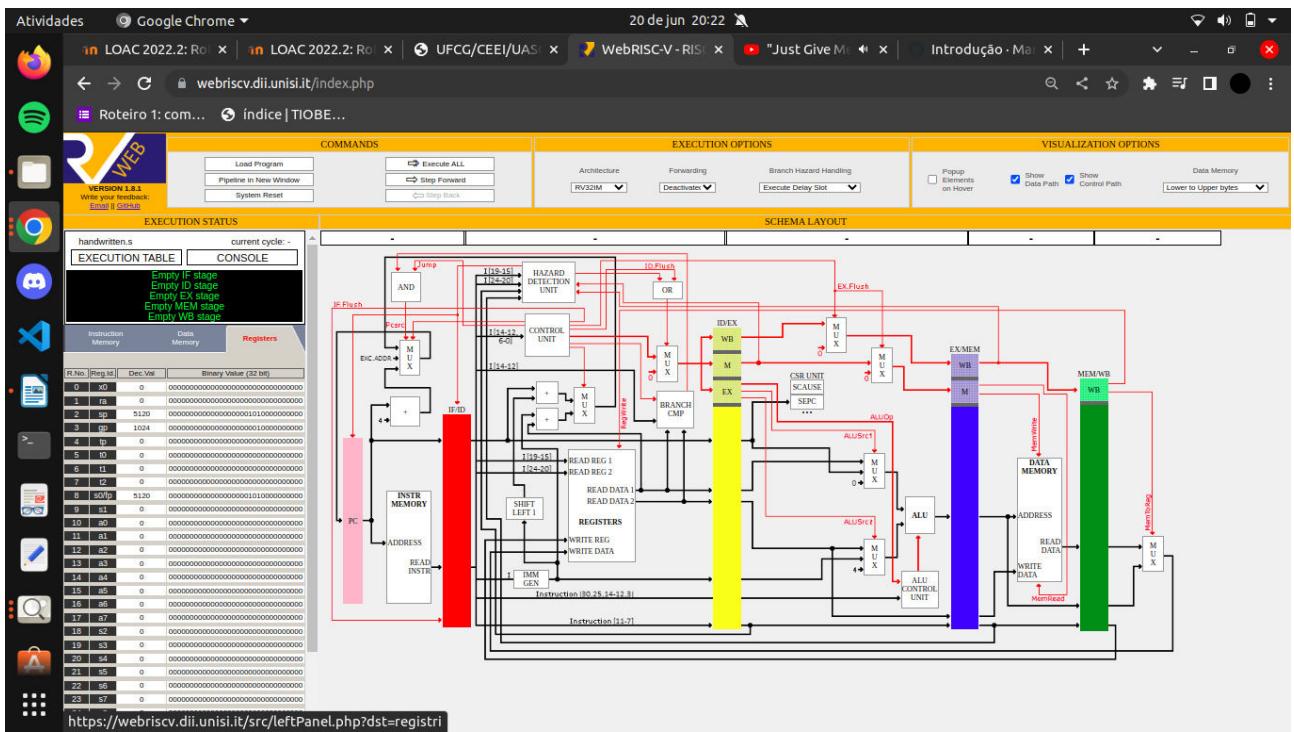
Registers:

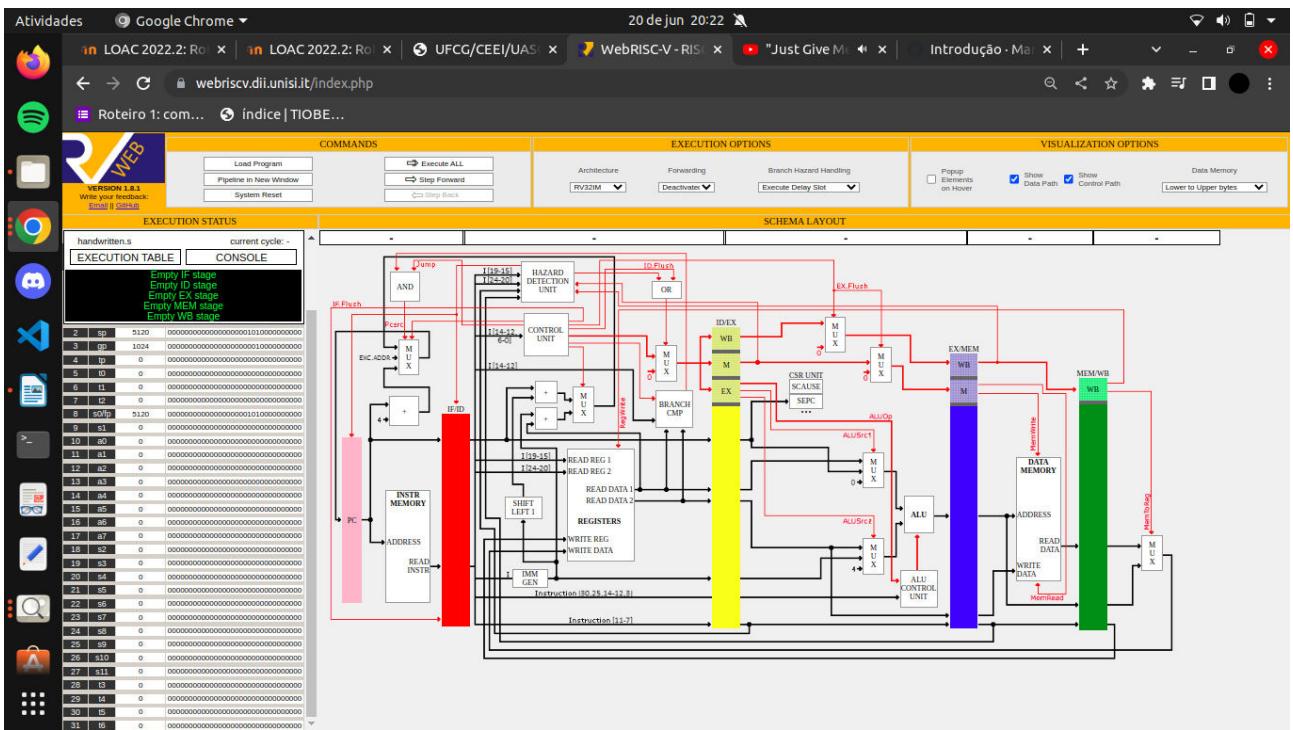
SP (Stack Pointer): É usado para rastrear o topo da pilha. A pilha é uma região de memória usada para armazenar dados temporários, endereços de retorno de funções e outras informações relacionadas à execução do programa.

GP (Global Pointer): É usado para acessar dados globais. O registrador GP armazena um ponteiro para uma área de memória que contém dados globais compartilhados entre várias partes do programa.

TP (Thread Pointer): É usado para acesso a dados específicos da thread. Em sistemas com suporte a multithreading, o registrador TP pode ser usado para armazenar um ponteiro para uma área de memória específica da thread em execução.

FP (Frame Pointer): Também conhecido como S0 (Saved Register 0), é usado como um ponto de referência para acessar variáveis locais e registros salvos no quadro atual. O FP aponta para o início do quadro de ativação da função atual e é usado para acessar os parâmetros da função, variáveis locais e registros salvos.

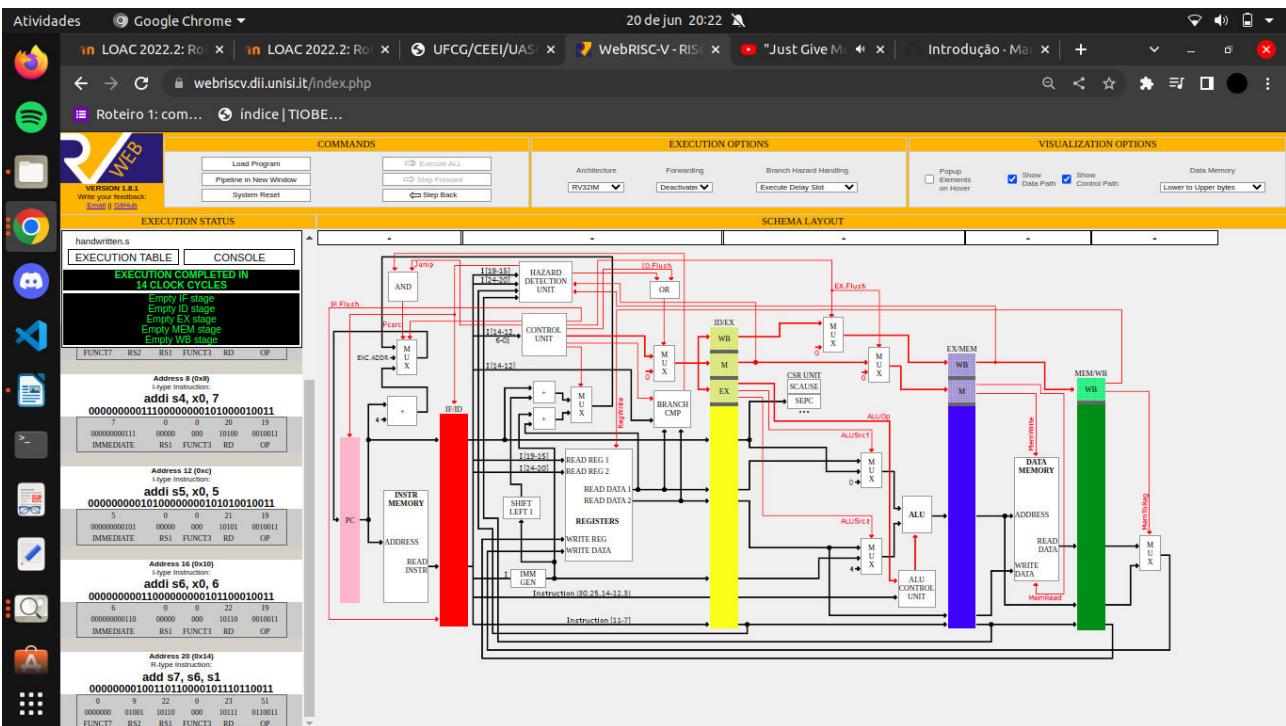
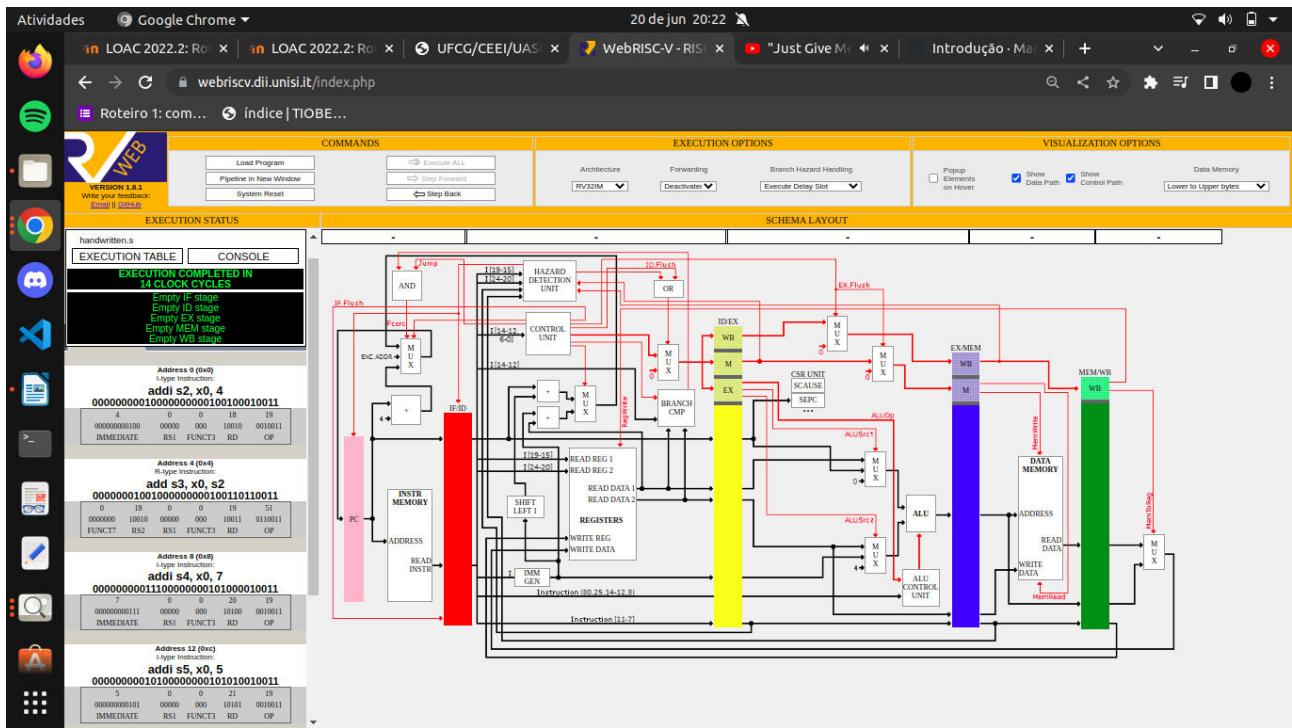




Final da execução:

Instruction Memory:

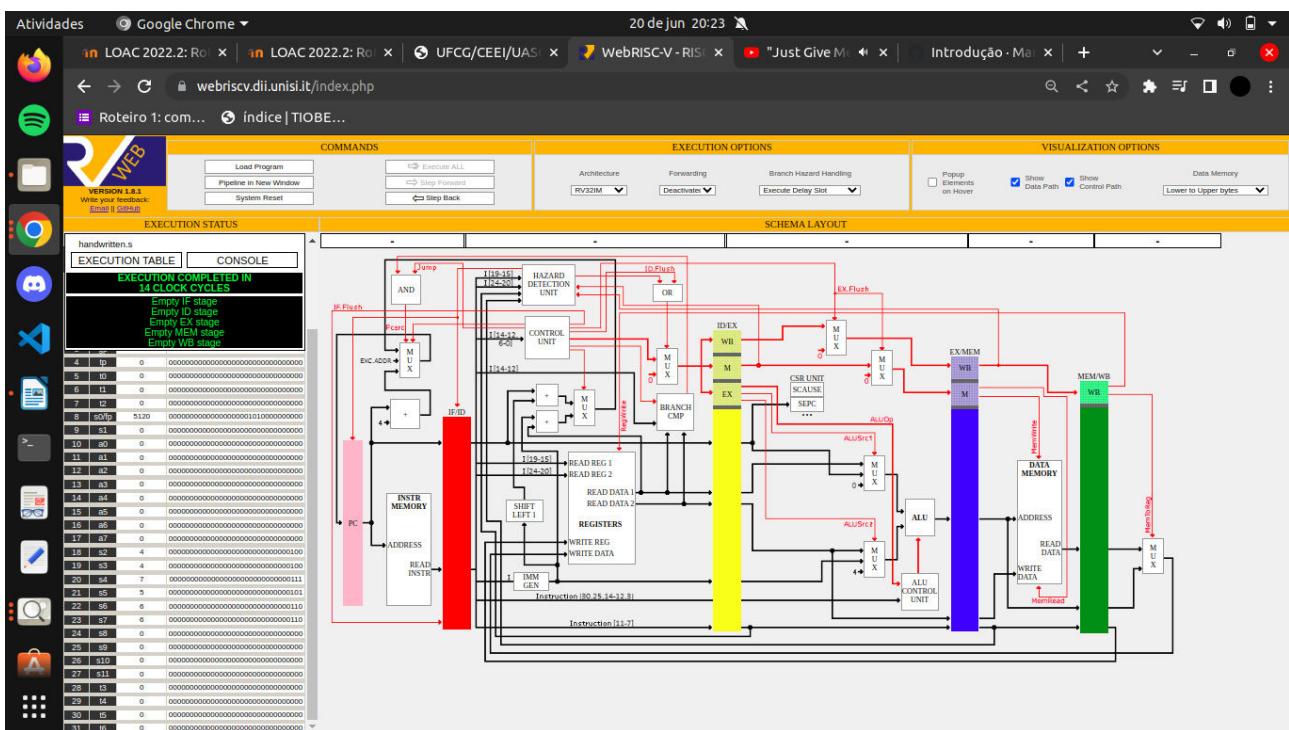
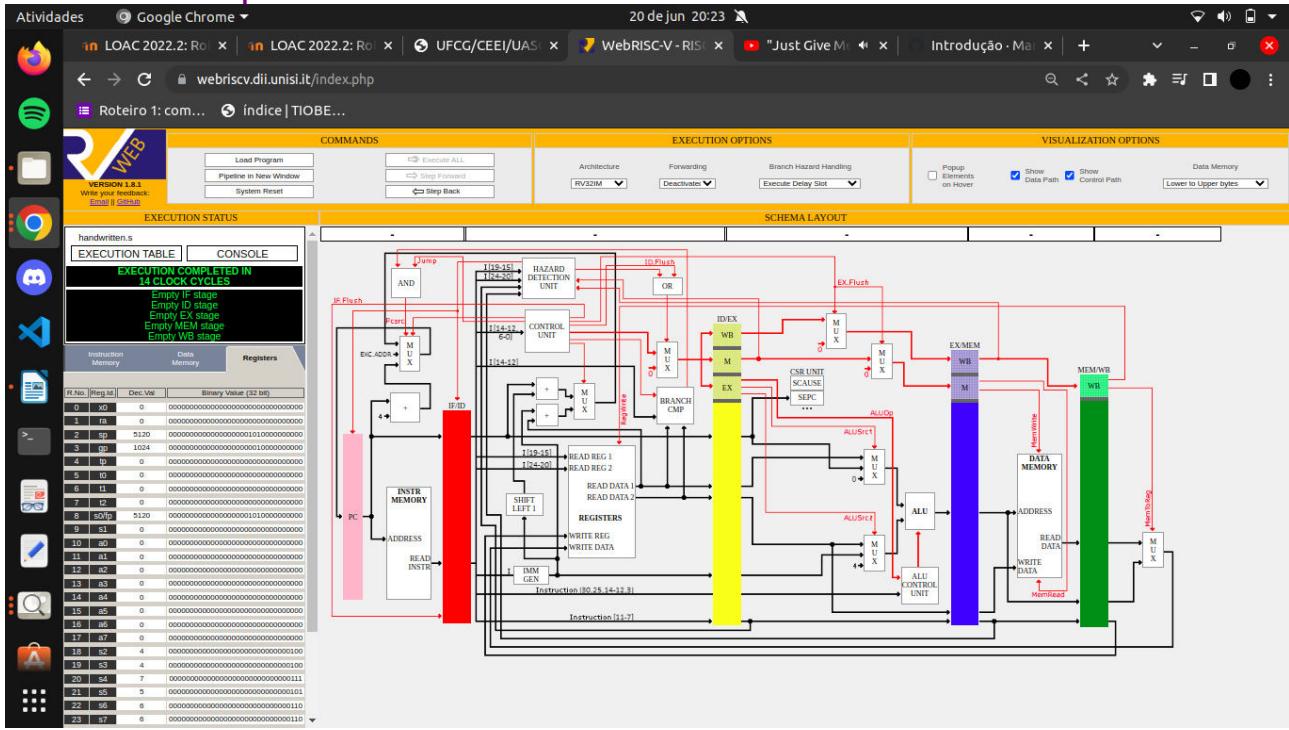
A "Instruction Memory" é uma região de memória reservada para armazenar as instruções do programa, e uma vez que as instruções são carregadas nessa área de memória, elas permanecem lá inalteradas até o final da execução do programa. As instruções são buscadas sequencialmente na memória para serem executadas pelo processador, mas a própria memória de instruções não é modificada durante esse processo.



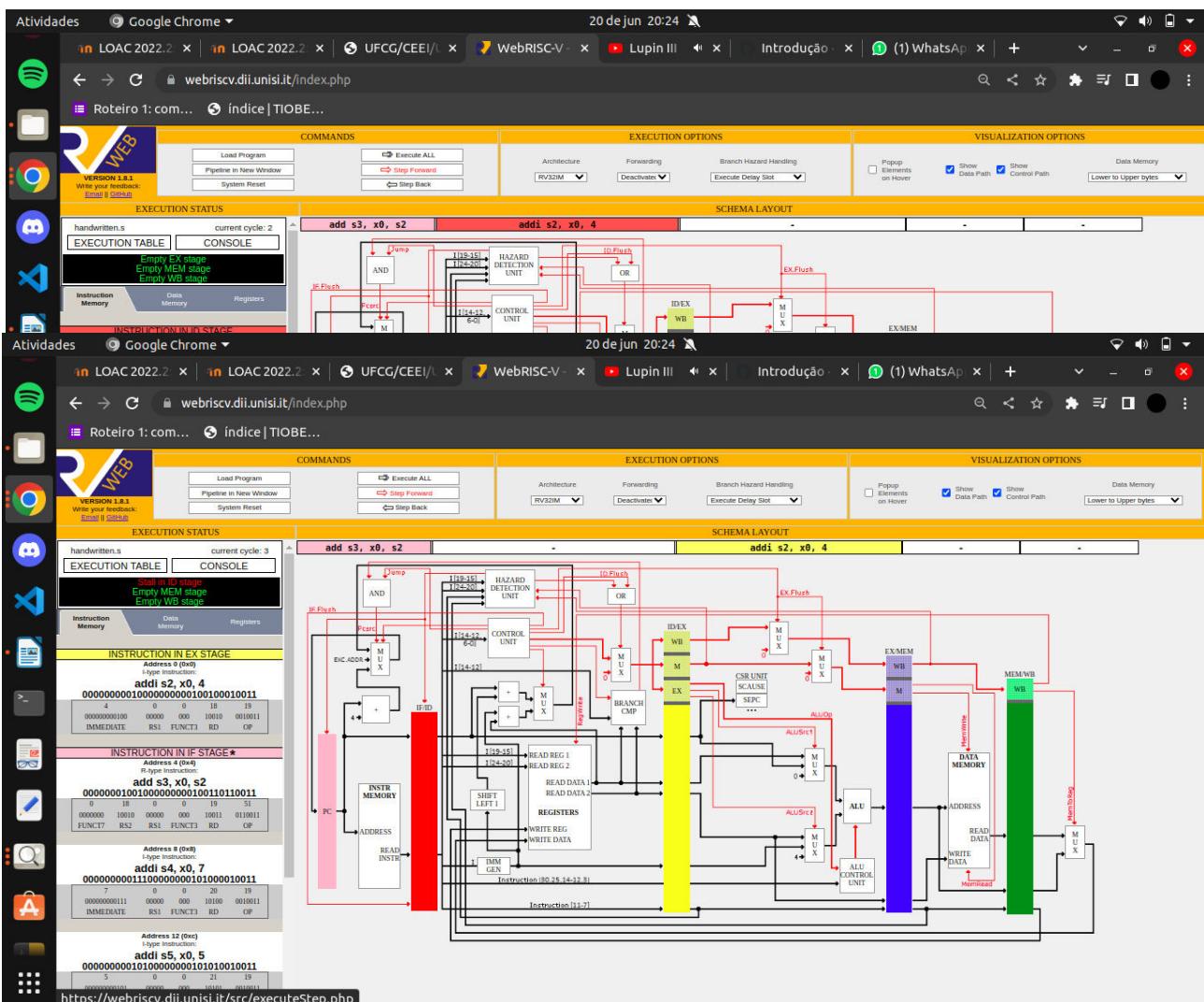
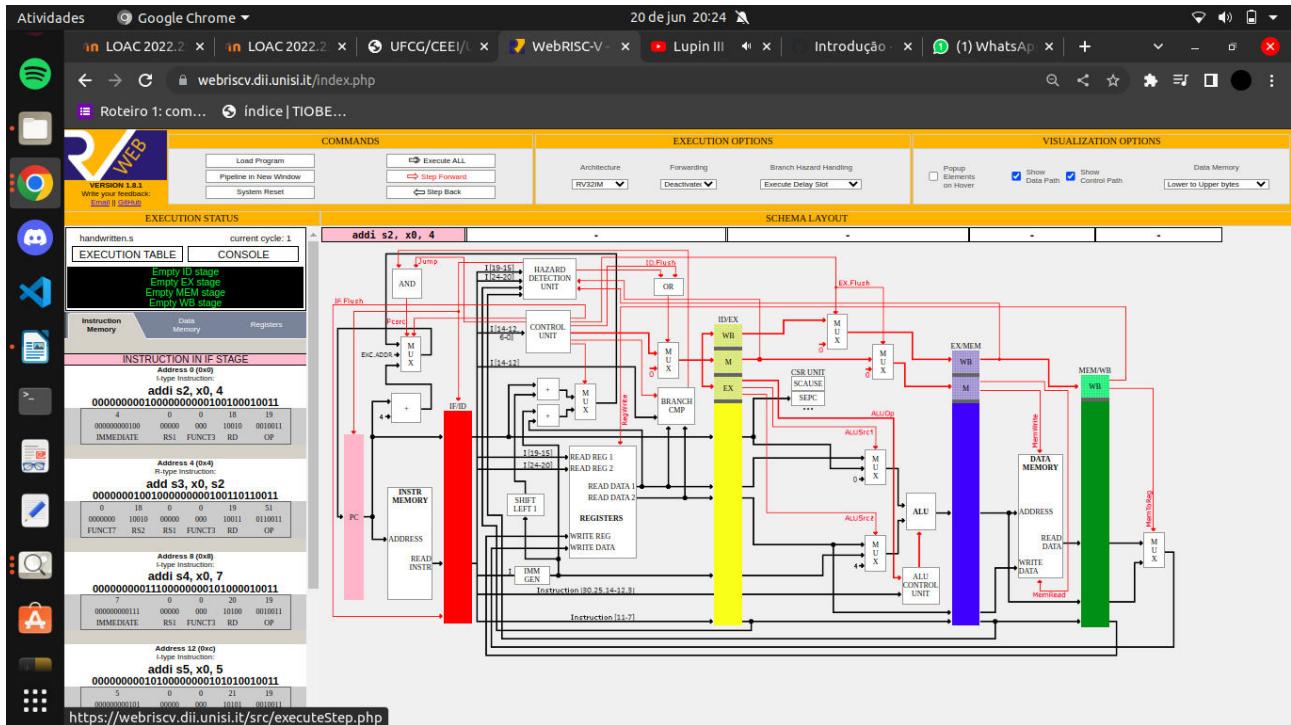
Registers:

s2 atualizado para o valor 4
s3 atualizado para o valor 3

s4 atualizado para o valor 7
 s5 atualizado para o valor 5
 s6 atualizado para o valor 6
 s7 atualizado para o valor 6



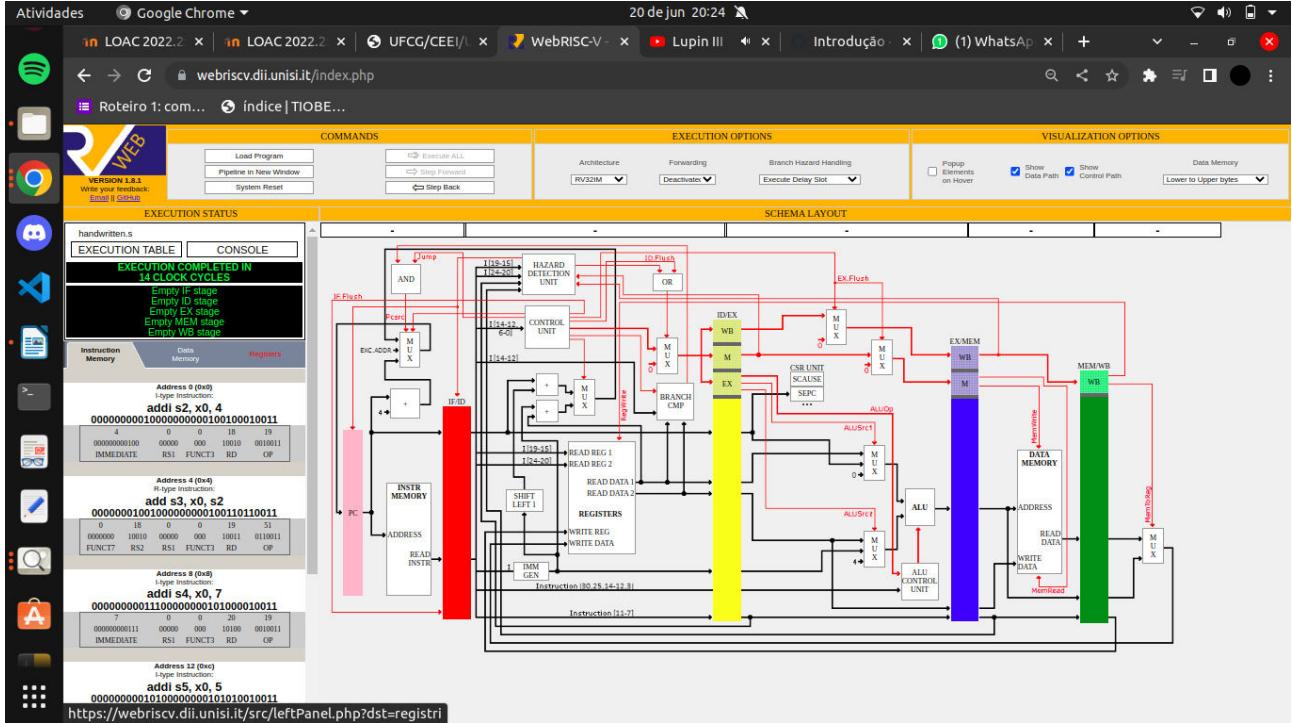
B1) Passagem em três estágios representativos do Pipeline (“SCHEMA LAYOUT”):



D1) Resultado final da execução em Pipeline, por meio da Tabela da Execução do Programa (“EXECUTION TABLE”):

Instruction	CPU Cycles													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
addi \$2, x0, 4	F	D	X	M	W									
addi \$3, x0, \$2		F	-	-	D	X	M	W						
addi \$4, x0, 7			F	D	X	M	W							
addi \$5, x0, 5				F	D	X	M	W						
addi \$6, x0, 6					F	D	X	M	W					
addi \$7, \$6, \$1						F	-	-	D	X	M	W		

E1) Ciclos de CPU necessários para executar esse programa: Resposta: 14 Ciclos. 14 ciclos = 1 * 5 + 5 + 4

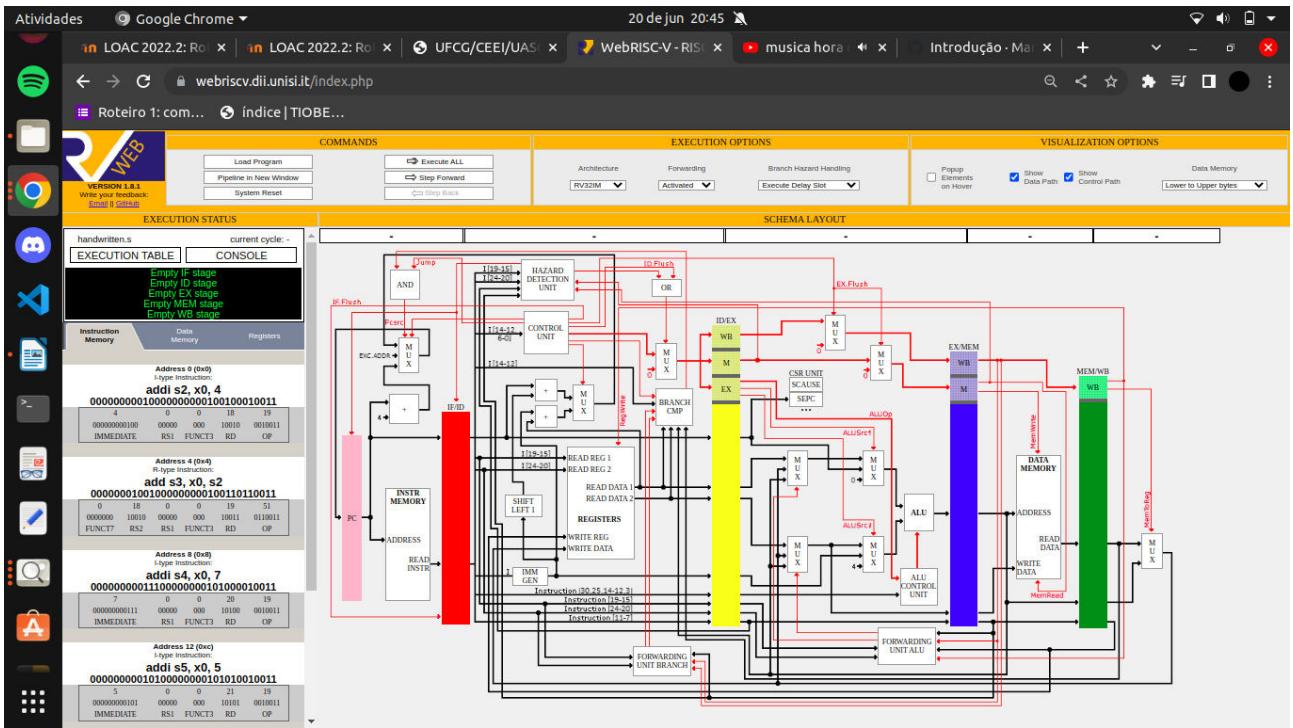


Parte 2:

Architecture: RV32IM

Forwarding: ACTIVATED

Branch Hazard Handling: EXECUTE DELAY SLOT



A2) Início da execução:

Instruction Memory:

Endereço 0 (0x0): addi s2, x0, 4

Essa instrução adiciona o valor imediato 4 ao registrador x0 e armazena o resultado no registrador s2.

Endereço 4 (0x4): addi s3, x0, 3

Essa instrução adiciona o valor imediato 3 ao registrador x0 e armazena o resultado no registrador s3.

Endereço 8 (0x8): addi s4, x0, 7

Essa instrução adiciona o valor imediato 7 ao registrador x0 e armazena o resultado no registrador s4.

Endereço 12 (0xc): addi s5, x0, 5

Essa instrução adiciona o valor imediato 5 ao registrador x0 e armazena o resultado no registrador s5.

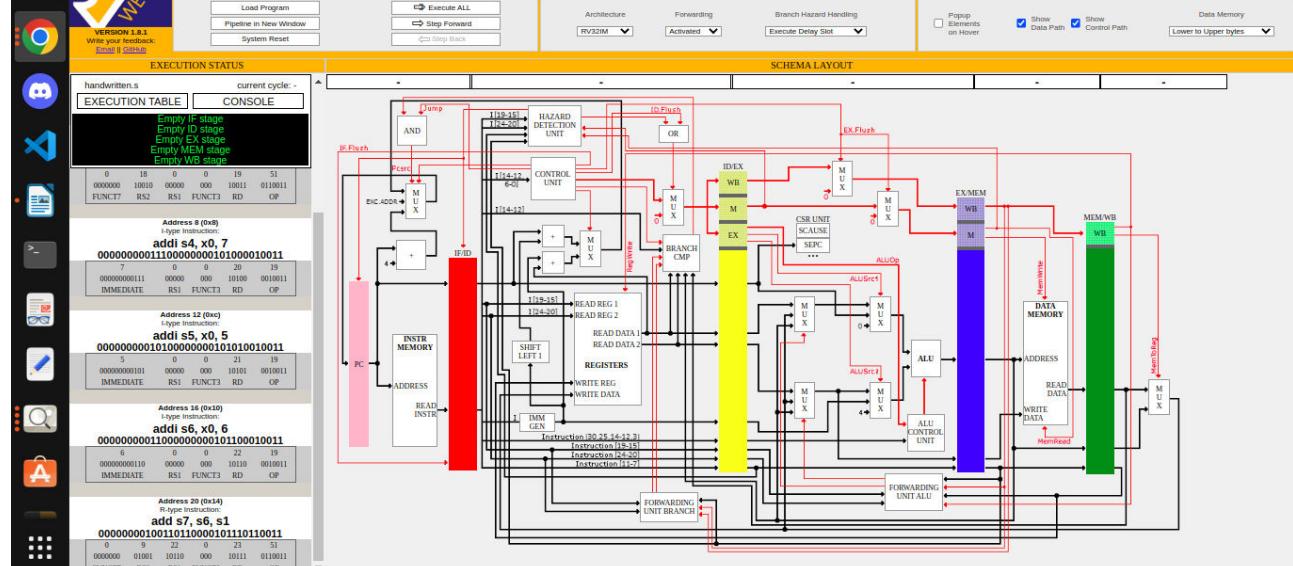
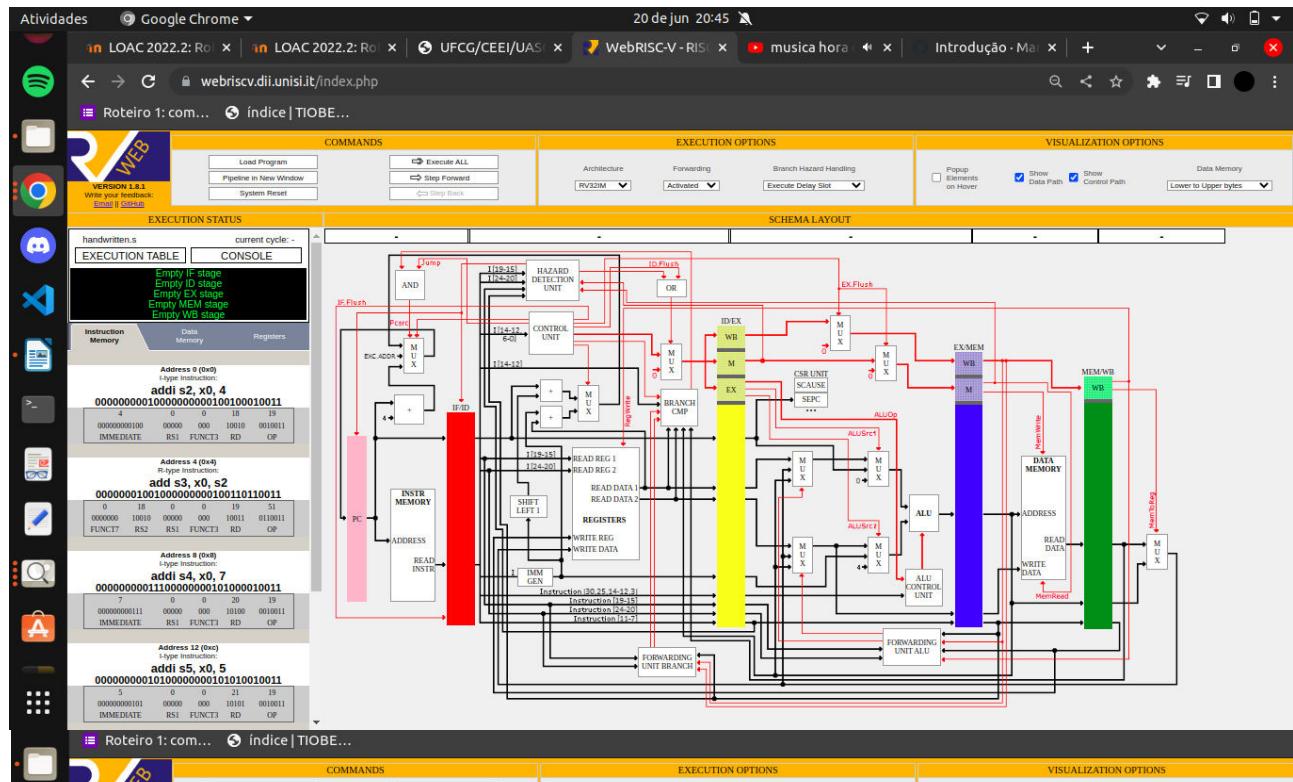
Endereço 16 (0xc): addi s6, x0, 6

Essa instrução adiciona o valor imediato 6 ao registrador x0 e armazena o resultado no registrador s6.

Endereço 20 (0x10): add s7, s2, s3

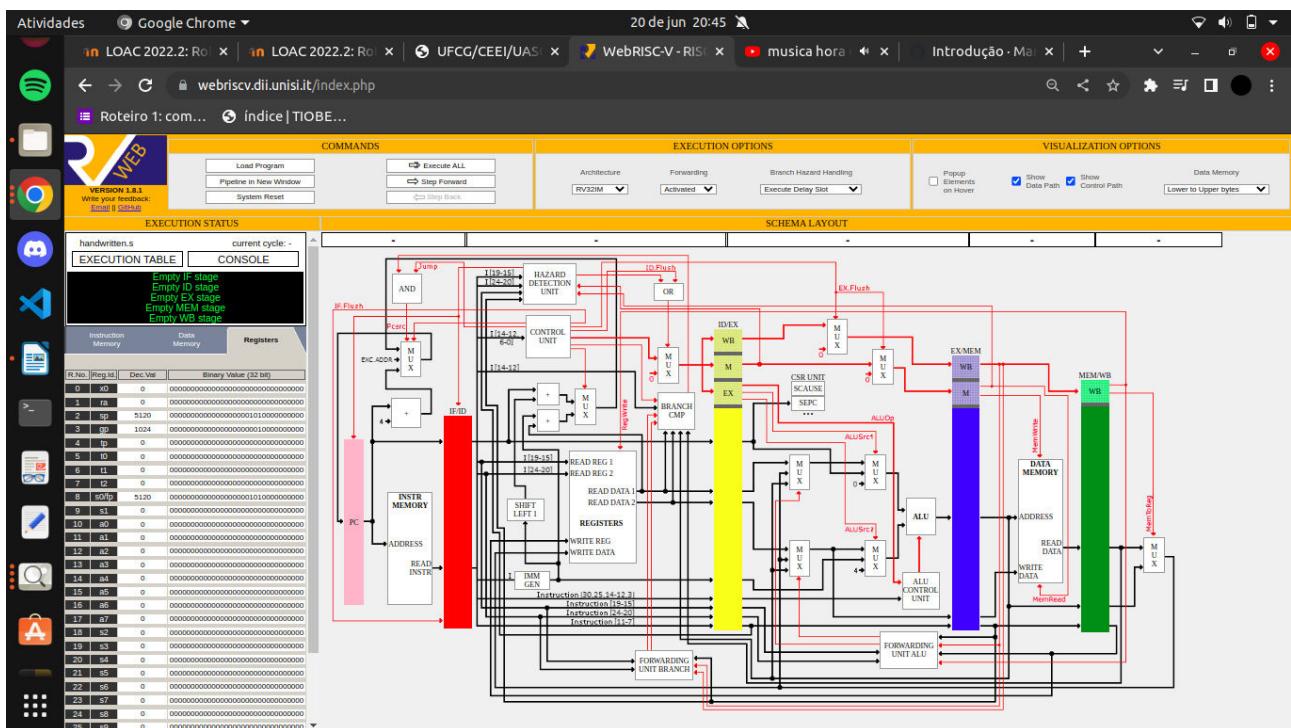
Por fim, adiciona o valor do registrador s2 ao valor do registrador s3 e armazena o resultado no

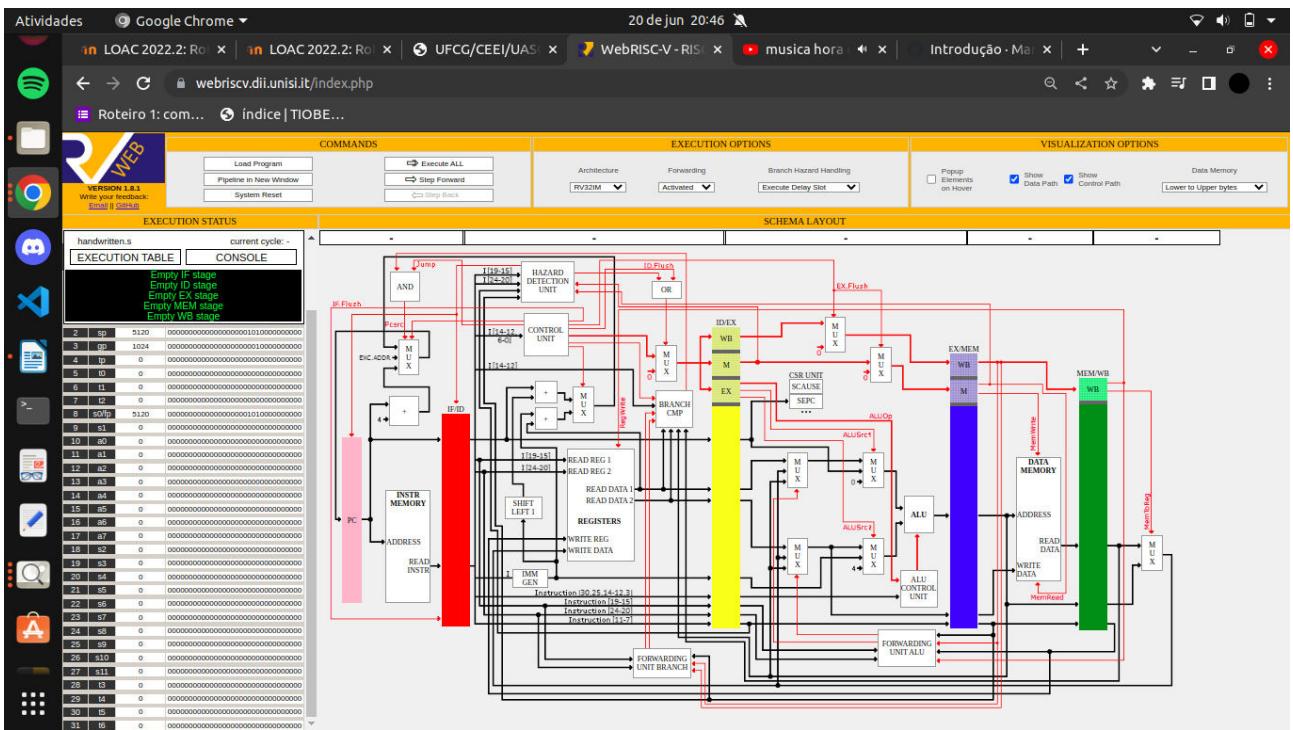
registrador s7.



Registers:

sp (Stack Pointer): É usado para rastrear o topo da pilha. A pilha é uma região de memória utilizada para armazenar dados temporários e informações relacionadas à execução do programa. gp (Global Pointer): É usado para acessar dados globais. Ele aponta para uma área de memória onde estão localizados os dados globais, que são acessíveis por todo o programa. tp (Thread Pointer): É usado para acesso a dados específicos da thread. Cada thread em um programa pode ter sua própria área de memória dedicada, e o tp é usado para apontar para essa área específica. fp (Frame Pointer): Também conhecido como s0 (saved register 0), é usado como um ponto de referência para acessar variáveis locais e registros salvos no quadro atual. O fp é usado para manter o controle do quadro de ativação (activation frame) ou pilha de chamadas, que contém informações sobre as chamadas de função em execução.

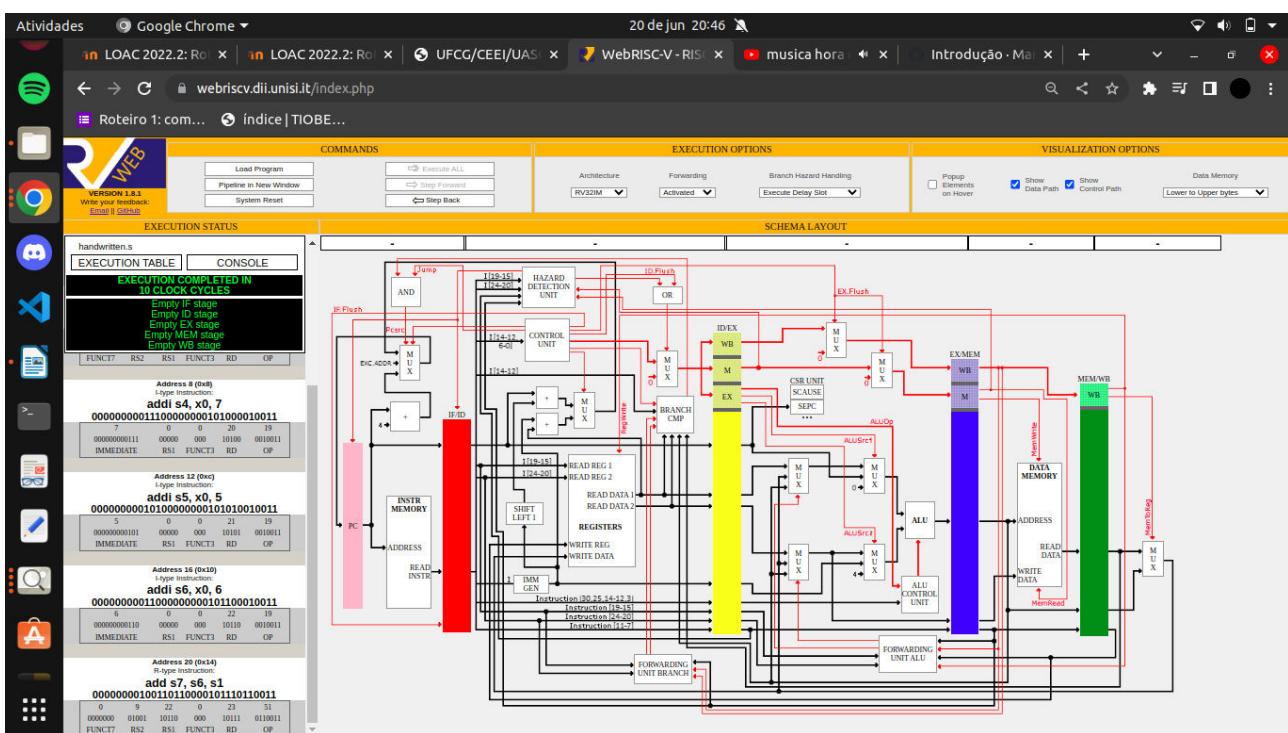
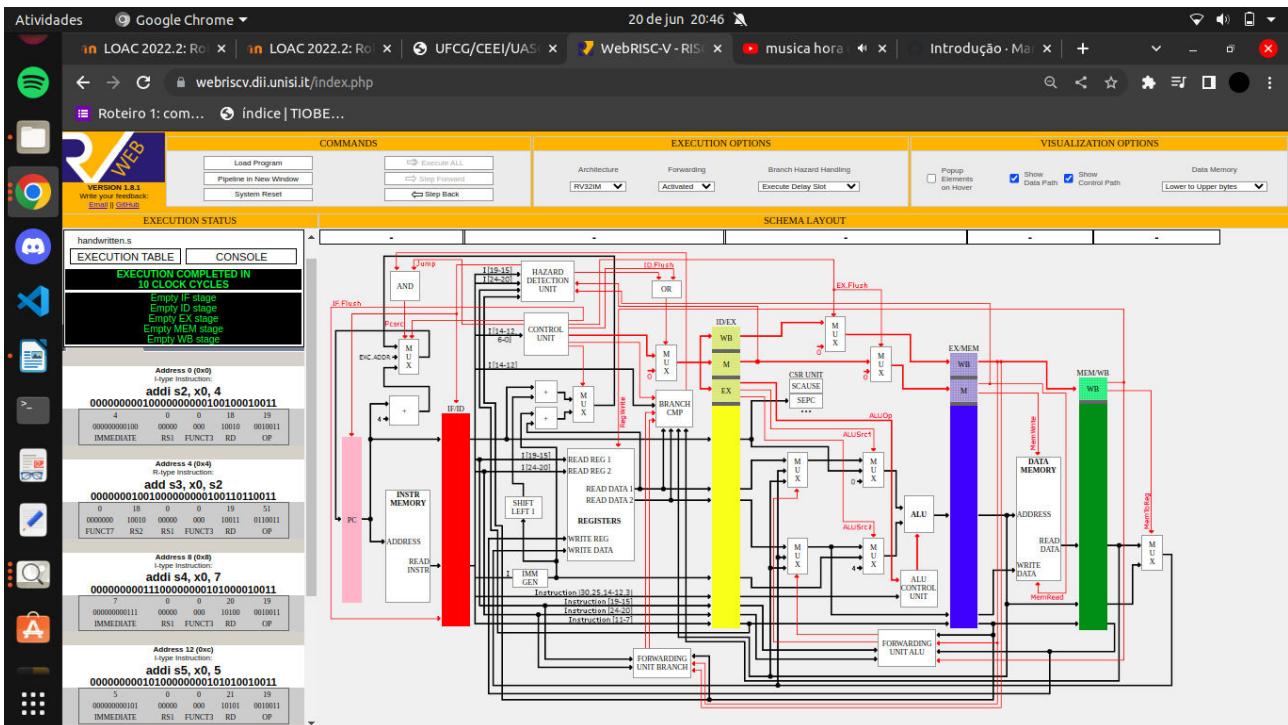




Final da execução:

Instruction Memory:

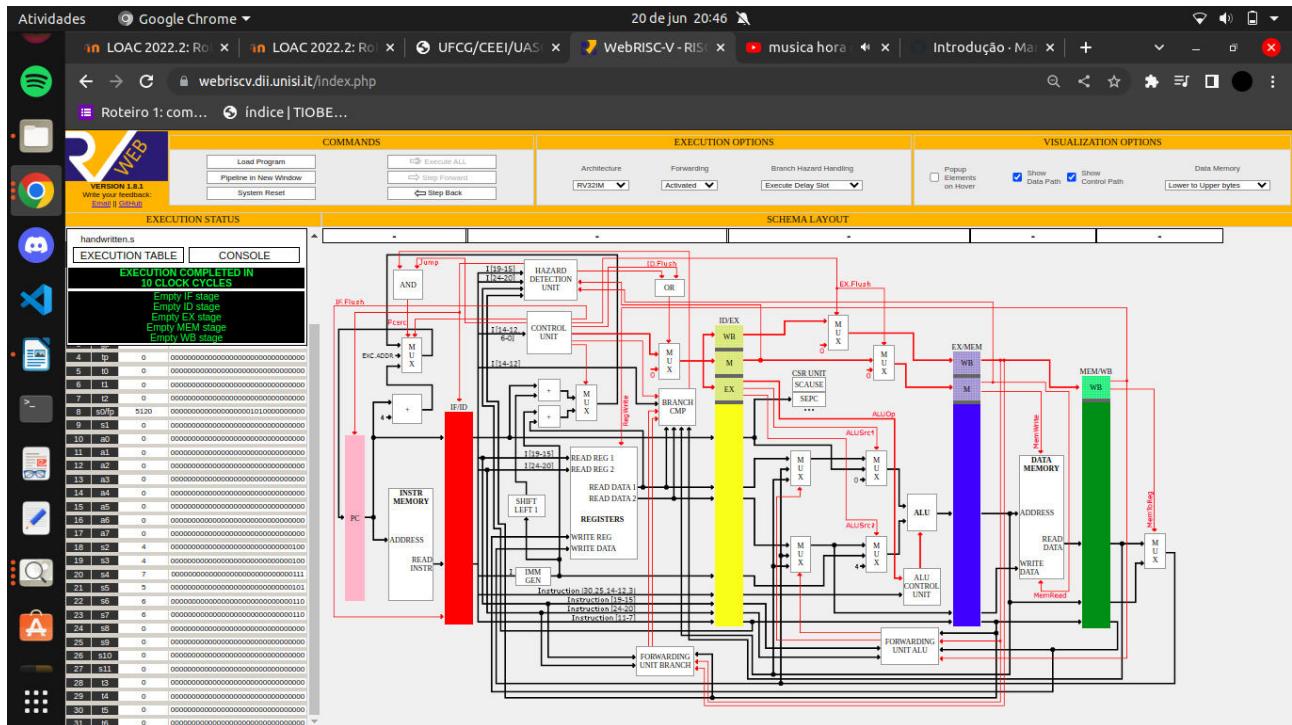
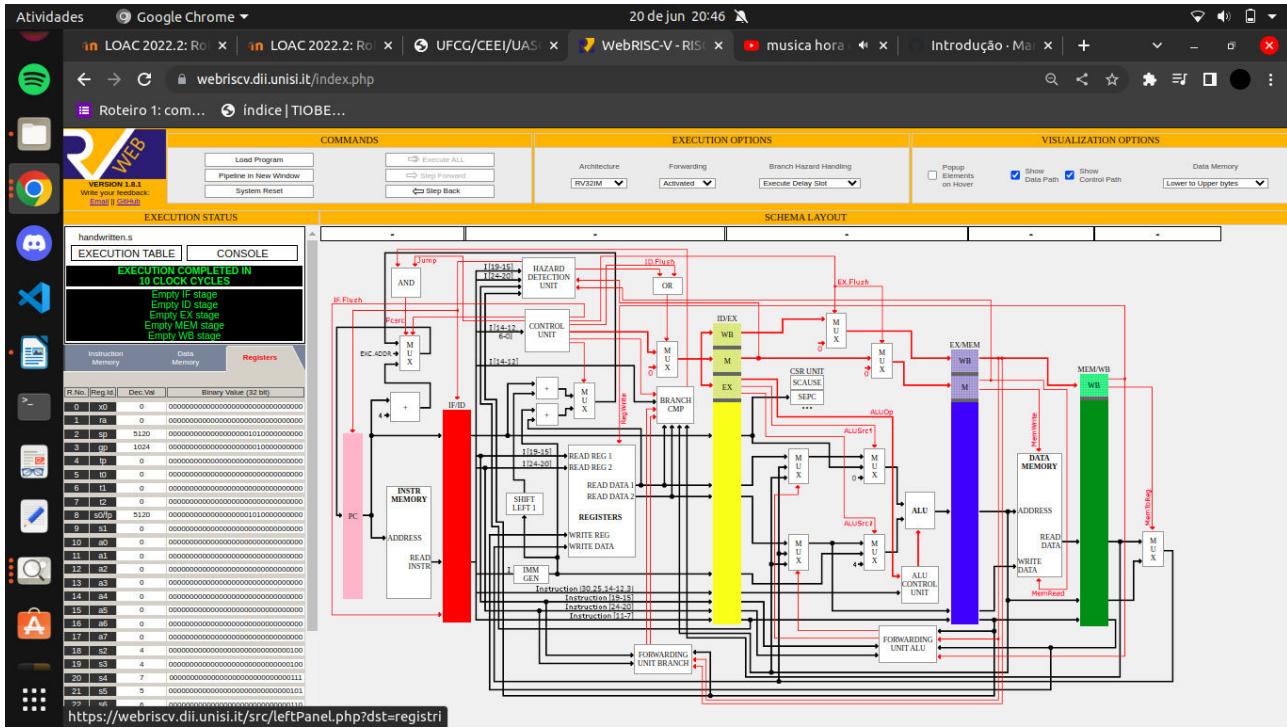
A "Instruction Memory" é uma área de memória onde as instruções do programa são armazenadas durante a execução. Uma vez que as instruções são carregadas na memória, elas permanecem lá e não são modificadas durante a execução do programa. As instruções são buscadas sequencialmente da "Instruction Memory" e executadas pelo processador.



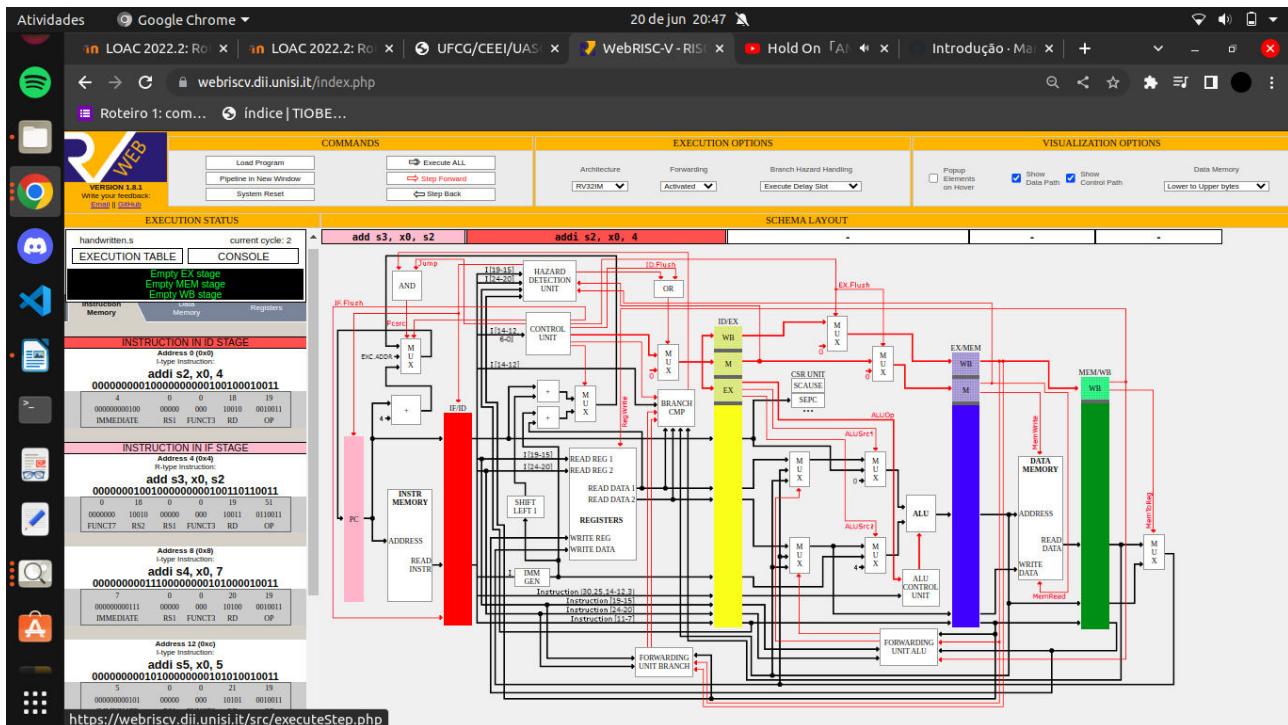
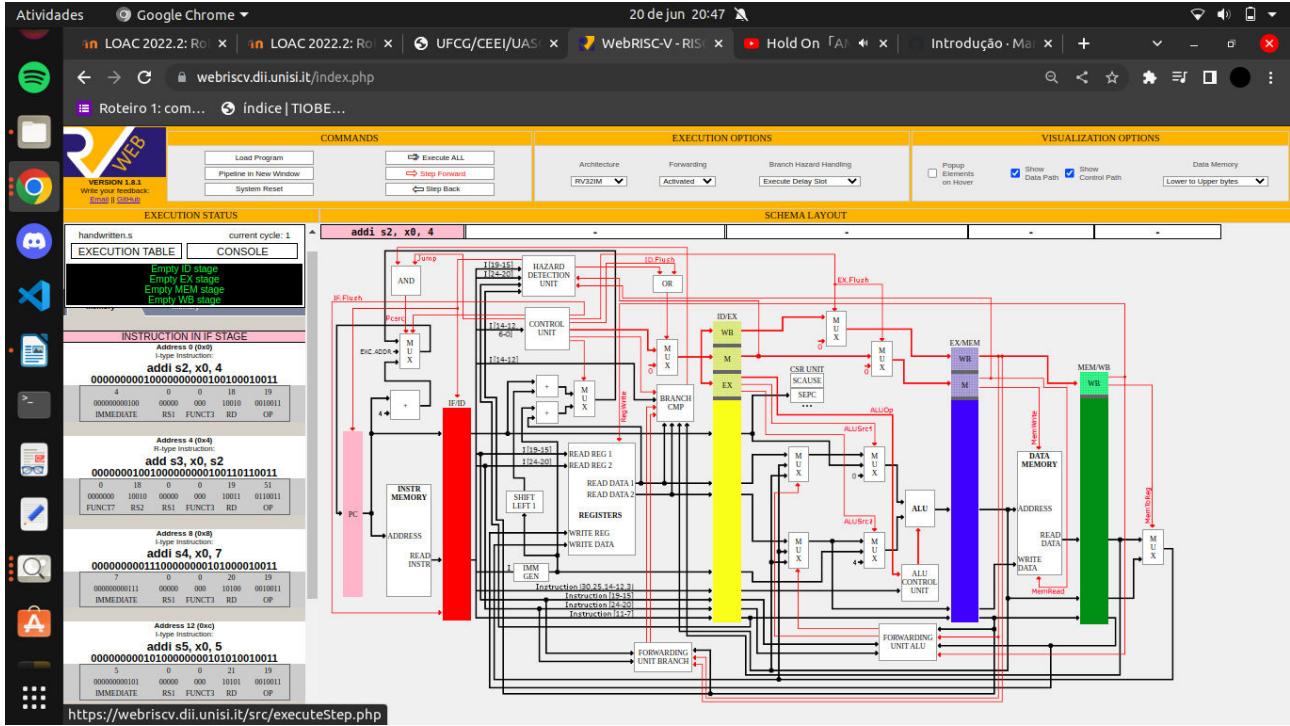
Registers:

s2 atualizado para o valor 4.
s3 atualizado para o valor 4.

s4 atualizado para o valor 7.
s5 atualizado para o valor 5.
s6 atualizado para o valor 6.
s7 atualizado para o valor 6.



B2) Passagem em três estágios representativos do Pipeline (“SCHEMA LAYOUT”):



D2) Resultado final da execução em Pipeline, por meio da Tabela da Execução do Programa (“EXECUTION TABLE”):

The screenshot shows a Google Chrome window titled "WebRISC-V - RISC-V PIPELINED DATAPATH SIMULATION ONLINE - Google Chrome". The URL is webriscv.dii.unisi.it/src/pipeTable.php. The page displays an "EXECUTION TABLE" for a sequence of instructions. The table has columns for "Instruction" and "CPU Cycles" (1 through 10). The "CPU Cycles" column is further divided into stages: F (Fetch), D (Decode), X (Execute), M (Memory), and W (Write-back). The table shows the following data:

	CPU Cycles									
FULL LOOPS	1	2	3	4	5	6	7	8	9	10
Instruction										
add s2, x0, 4	F	D	X	M	W					
add s3, x0, s2	F	D	X	M	W					
addi s4, x0, 7	F	D	X	M	W					
addi s5, x0, 5	F	D	X	M	W					
addi s6, x0, 6		F	D	X	M	W				
add s7, s6, s1			F	D	X	M	W			

E2) Ciclos de CPU necessários para executar esse programa: Resposta: 10 Ciclos. $1 * 5 + 5$

