UNIVERSIDADE ANHEMBI MORUMBI

ENZO FERREIRA AGUIAR

GUSTAVO CARAMASQUI DA SILVA

JEAN SILVA FREITAS

MARIA EDUARDA FRAGOSO DA SILVA

YGOR PEREIRA SOUSA SANTOS

ESPECIFICAÇÃO DE EQUAÇÃO MATEMÁTICA RELACIONADA AO CICLO DE MICROINSTRUÇÕES DE UM PROCESSADOR

São Paulo

2022

ENZO FERREIRA AGUIAR

GUSTAVO CARAMASQUI DA SILVA

JEAN SILVA FREITAS

MARIA EDUARDA FRAGOSO DA SILVA

YGOR PEREIRA SOUSA SANTOS

ESPECIFICAÇÃO DE EQUAÇÃO MATEMÁTICA RELACIONADA AO CICLO DE MICROINSTRUÇÕES DE UM PROCESSADOR

Atividade Avaliativa apresentada à Universidade Anhembi Morumbi utilizada como nota parcial para a conclusão da Unidade Curricular.

Orientadores: Professor Mestre Edquel Bueno Prado Farias e Professor Doutor Marcel Stefan Wagner.

São Paulo

2022

RESUMO

O presente trabalho tem como objetivo apresentar a resolução da equação escolhida previamente, que posteriormente foi colocada no *Von Neumann Machine Simulator (VNSimulator)* para a realização do projeto. E por fim fazer a explicitação dos processos durante a execução da operação matemática.

Palavras-Chave: Arquitetura de Von Neumann; Ciclo de microinstruções do processador; Equação matemática.

ABSTRACT

This project aims to present the resolution of the previously chosen equation, which was later placed in the Von Neumann Machine Simulator (VNSimulator) for project conclusion. And finally, make the explanation of the processes during the execution of the mathemathical operation.

Keywords: Von Neumann architecture; Processor microinstructions cycle; Mathemathical equation.

ÍNDICE DE ILUSTRAÇÕES

LISTA DE SIGLAS E ABREVIAÇÕES

ALU - *Arithmetic Logic Unit*

MQ – *Multiplier Quotient*

SUMÁRIO

1.0. INTRODUÇÃO

Esse documento tem como finalidade mostrar a resolução da equação matemática apresentada na figura abaixo e realizar a explicação da mesma durante a execução no *Von Neumann Machine Simulator*.

Grupo 20:

Variáveis a serem consideradas para a A3:
$T1 = A = 2$
$T2 = B = 2$
$T3 = C = 4$
$T4 = D = 8$

Equação alterada para uso no VNSimulator da A3:
$X = \frac{(A.B + C.D)/(C + D.(C/A - B) + A)}{\big((A.B + A)/(C/A + B.A)\big) + A}$

Resposta:
$X = 2$

Figura 1 - Equação escolhida

2.0. RESOLUÇÃO DA EQUAÇÃO

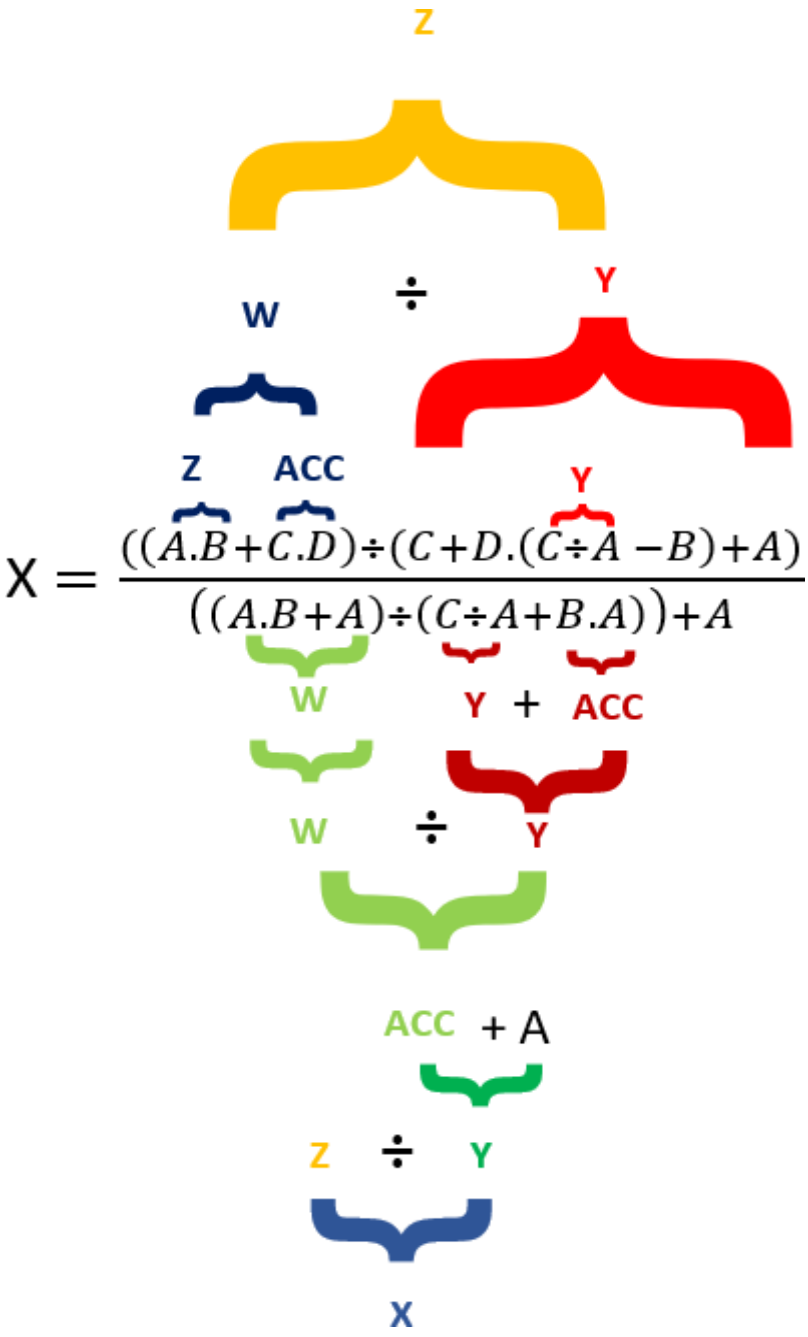

Figura 2 - Resolução da equação

2.1. IMAGENS FINAIS DO VON NEUMANN SIMULATOR

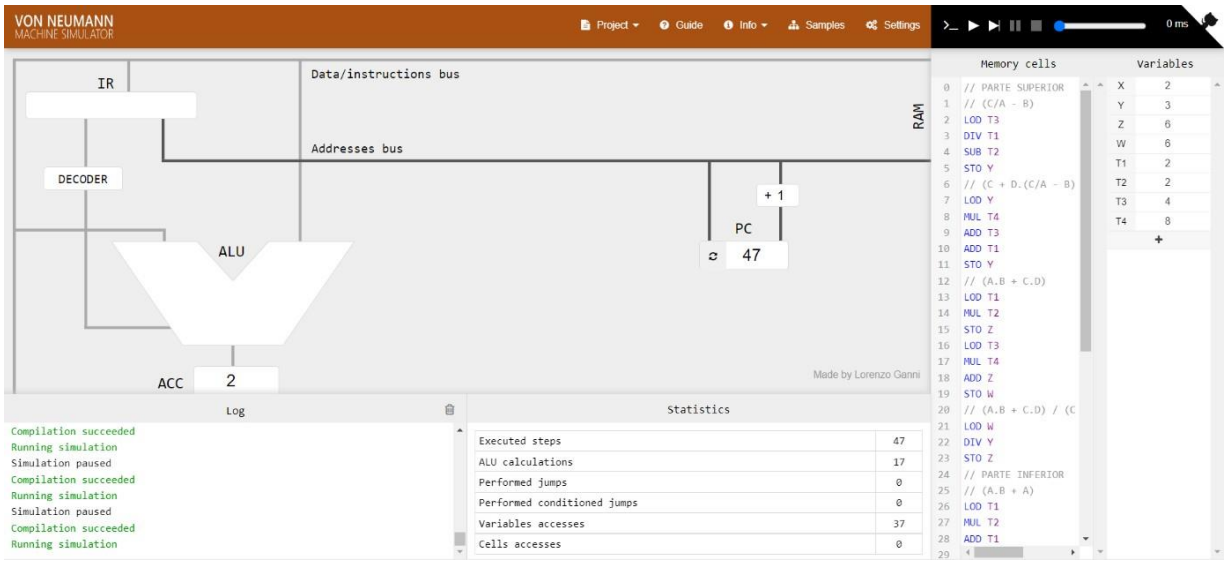

Figura 3 - Estatísticas finais da equação

2.1.1. EXPLICITAÇÃO DOS PASSOS DA EQUAÇÃO

Após a realização e a transcrição da equação para a linguagem Assembly, foi desenvolvida a explicação de cada etapa em relação aos componentes internos do sistema, como barramentos, Unidade Lógica e Aritmética (ULA), entre outros.

3.0. PARTE SUPERIOR

3.1. BLOCO 01

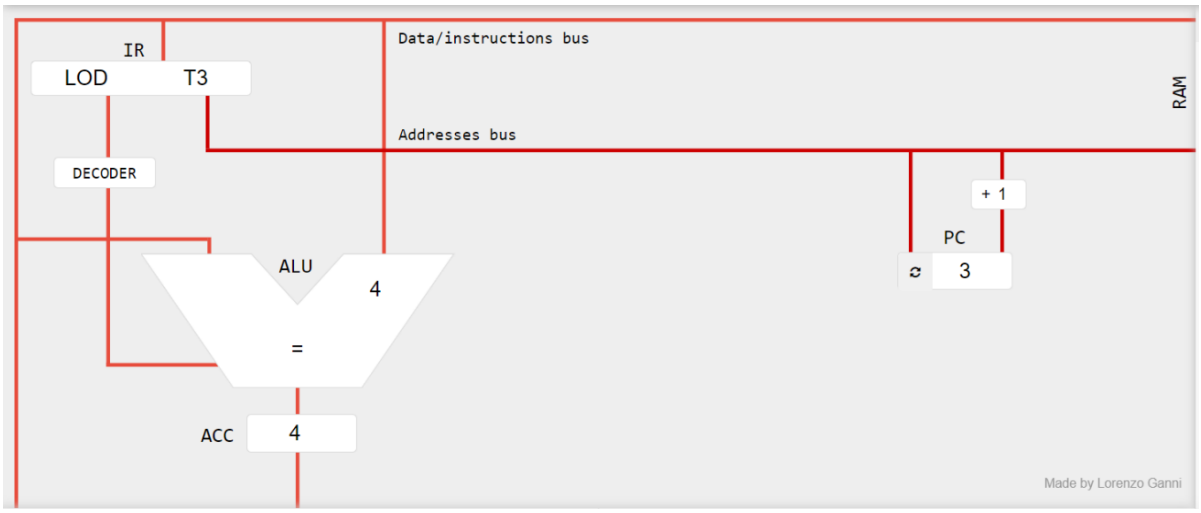

Figura 4 – Execução da primeira instrução

Os primeiros passos presentes no Contador de Programas (PC), são devido às duas primeiras linhas de comentários (instruções NOP), que não são considerados instruções.

A instrução LOD T3 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 02. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso LOD T3. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

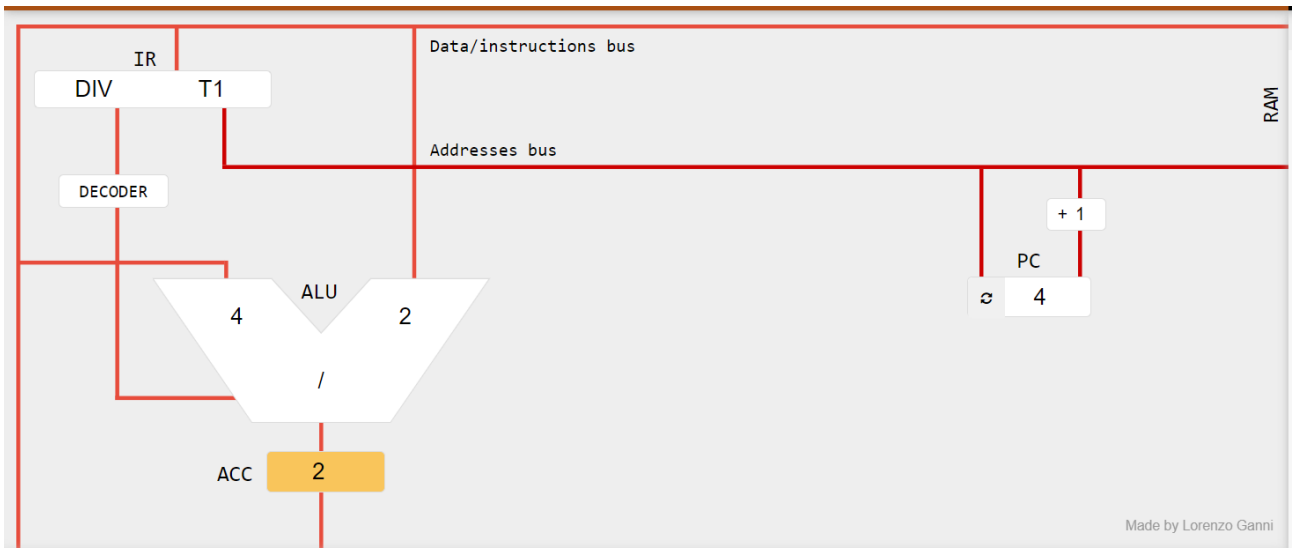

Figura 5 – Execução da segunda instrução

A instrução DIV T1 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 03. Já o Registrador de Buffer de Memória (MBR) recebe a instrução DIV da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso DIV T1. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução DIV T1 é feita pelo *Decoder*, que envia a instrução para a ALU por meio dos barramentos internos, a operação que será realizada.

A operação de divisão é realizada pela ALU com a instrução que estava salva no Acumulador (LOD T3) depois disso, o resultado e os operandos da operação são salvos temporariamente e novamente no Acumulador e no Quociente Multiplicador.

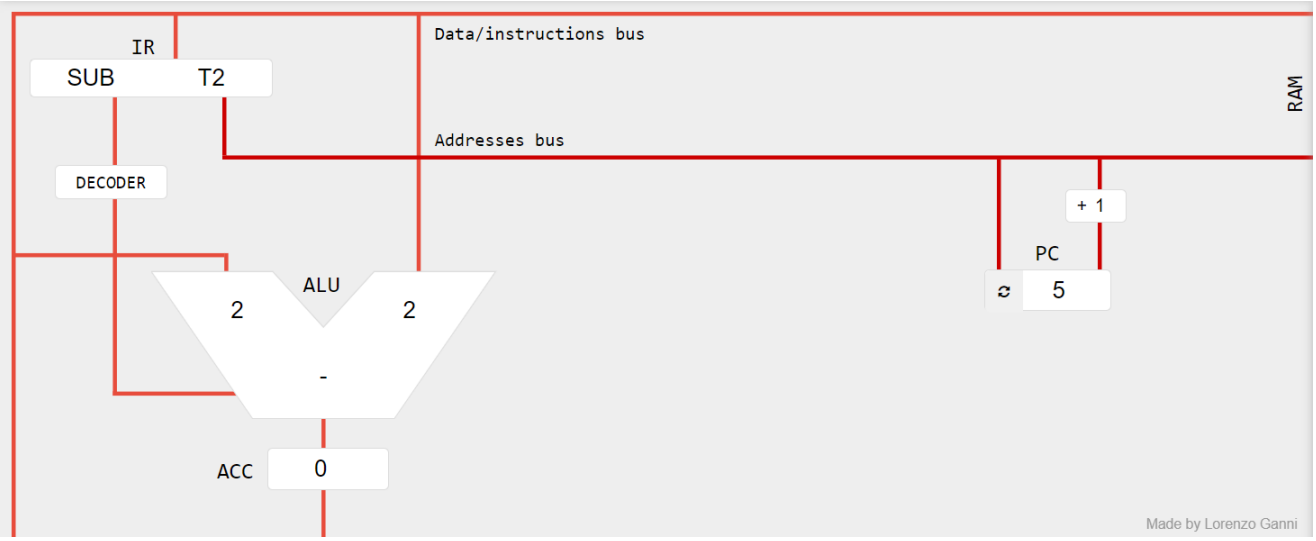

Figura 6 – Execução da terceira instrução

A instrução SUB T2 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 04. Já o Registrador de Buffer de Memória (MBR) recebe a instrução SUB da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso SUB T2. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução SUB T2 é feita pelo *Decoder*, que envia a instrução para a ALU por meio dos barramentos internos, a operação que será realizada. A operação de subtração é realizada pela ALU, com o resultado da expressão anterior (LOD T3 DIV T1), que já estava armazenado temporariamente no Acumulador e no Quociente Multiplicador, com a instrução SUB T2. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 05. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma

temporária a próxima instrução que será executada, no caso STO Y. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável Y.

3.2. BLOCO 02

Figura 7 – Execução da primeira instrução

A instrução LOD Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 07. Já o

Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso LOD Y. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

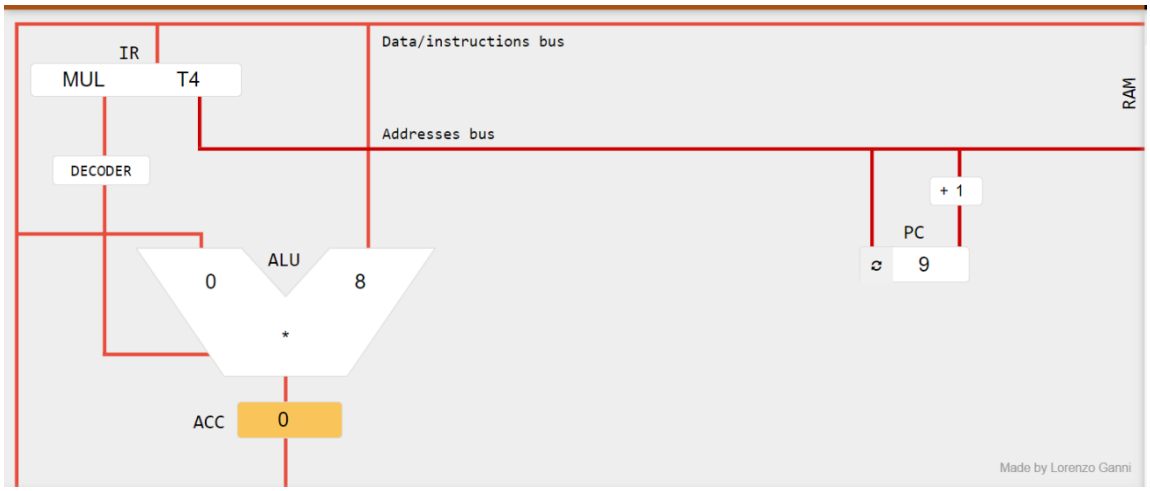

Figura 8 – Execução da segunda instrução

A instrução MUL T4 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 08. Já o Registrador de Buffer de Memória (MBR) recebe a instrução MUL da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso MUL T4. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução MUL T4 é realizada pelo Decodificador que envia pelos barramentos internos para a ALU, a instrução que será executada.

A operação de multiplicação é realizada pela ALU, com a instrução anterior (LOD Y), que já estava armazenado temporariamente no Acumulador e no Quociente Multiplicador, com a instrução MUL T4. Depois disso o resultado é alocado temporariamente no Quociente Multiplicador e no Acumulador.

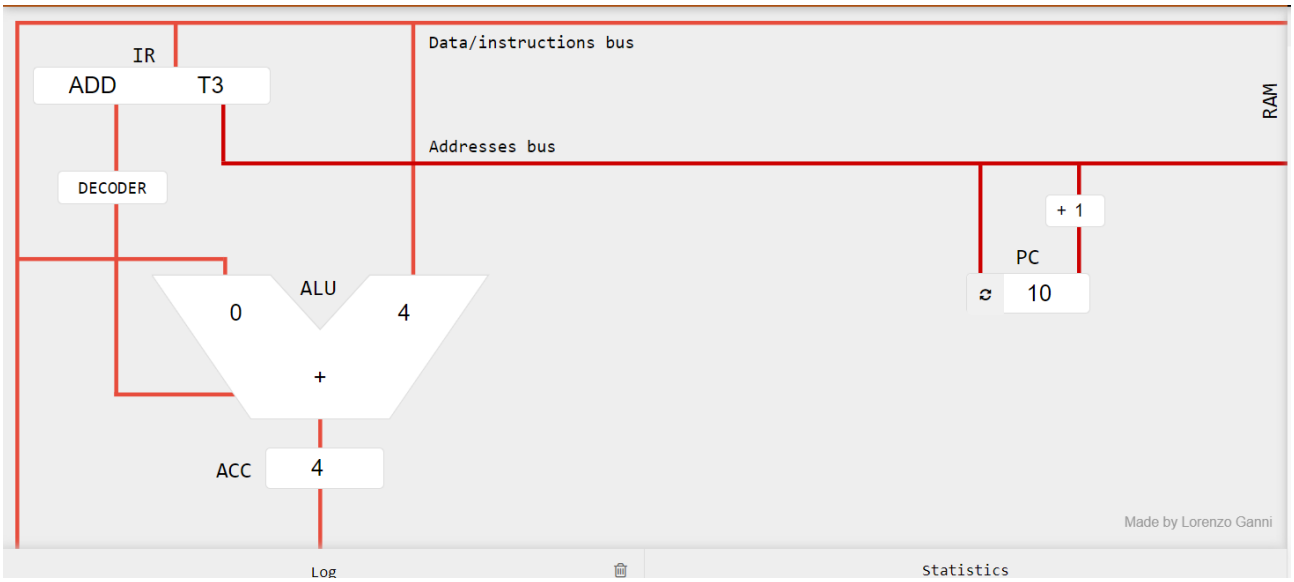

Figura 9 – Execução da terceira instrução

A instrução ADD T3 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 09. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso ADD T3. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo Decodificador, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

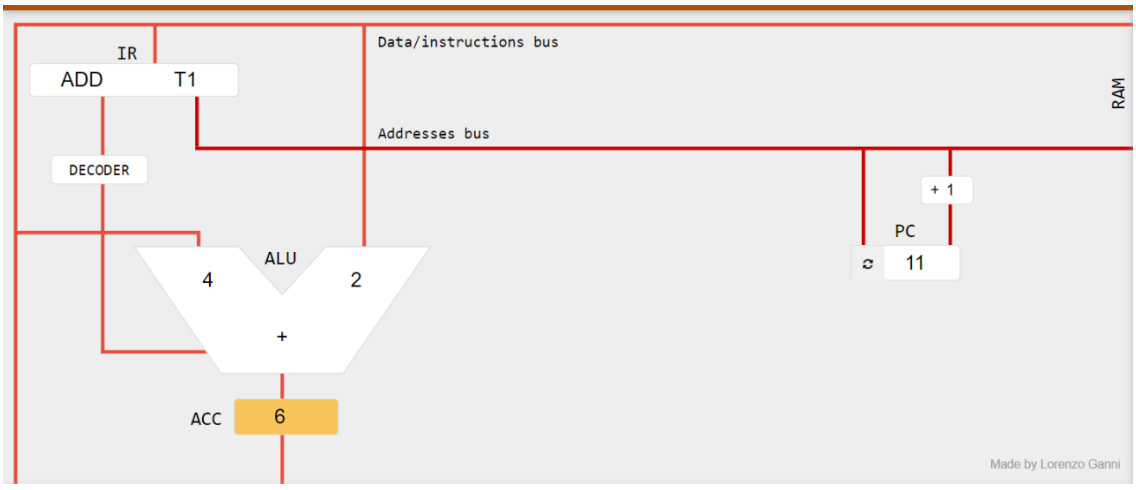

Figura 10 – Execução da quarta instrução

A operação é realizada com o resultado da operação anterior que já estava no Acumulador e no Quociente Multiplicador (LOD Y MUL T4), com o valor da nova instrução (ADD T3), o resultado dessa operação será armazenado de forma temporária novamente no Quociente Multiplicador e no Acumulador.

A instrução ADD T1 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 10. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso ADD T1. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de adição é realizada pela ALU, com o resultado das expressões anteriores (LOD Y MUL T4 ADD T3) que já estavam armazenados temporariamente no Acumulador e no Quociente Multiplicador, com a instrução SUB T2. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 11. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso STO Y. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável Y.

3.3. BLOCO 03

Figura 11 – Execução da primeira instrução

A instrução LOD T1 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 13. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso LOD T1. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

A instrução MUL T2 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 14. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso MUL T2. Após isso, o Contador de Programas (PC) é incrementado.

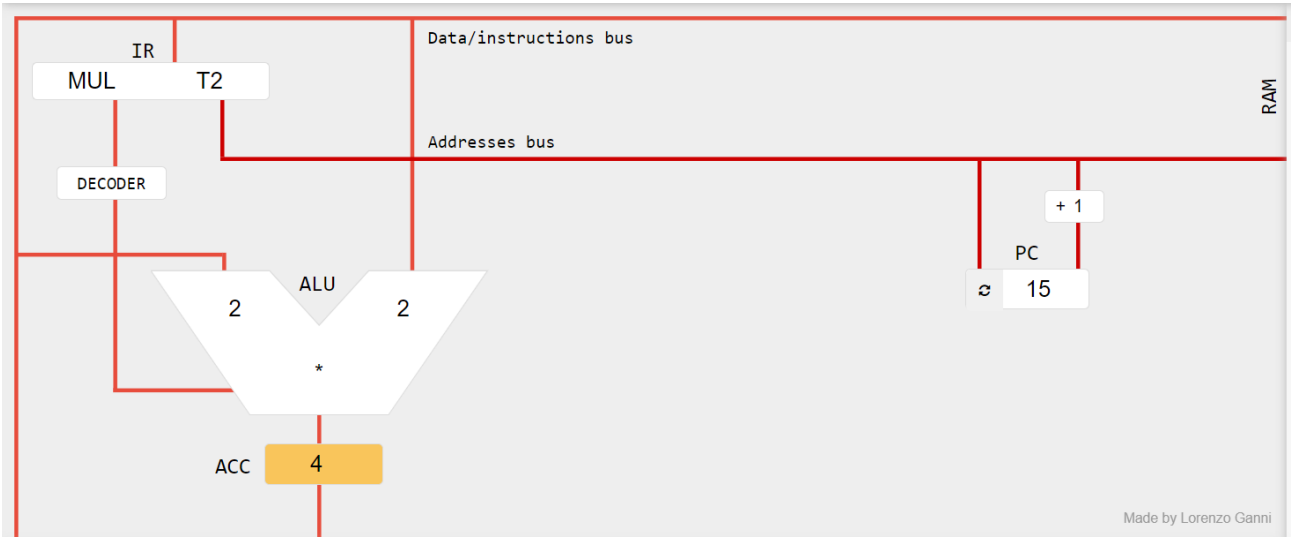

Figura 12 – Execução da segunda instrução

A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada. A operação de multiplicação é realizada pela ALU, com o resultado da expressão anterior (LOD T1), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução MUL T2. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO Z é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 15. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso STO Z. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável Z.

A instrução LOD T3 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 16. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso LOD T3. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

A instrução MUL T4 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 17. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso MUL T4. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada. A operação de multiplicação é realizada pela ALU, com o resultado da expressão anterior (LOD T3), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução MUL T4. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução ADD Z é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 18. Já o

Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso ADD Z. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de adição é realizada pela ALU, com o resultado da expressão anterior (LOD T3 MUL T4), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução ADD Z. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO W é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 19. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso STO W. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável W.

3.4. BLOCO 04

A instrução LOD W é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 21. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso LOD W. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

A instrução DIV Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 22. Já o Registrador de Buffer de Memória (MBR) recebe a instrução DIV da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso DIV Y. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de divisão é realizada pela ALU, com o resultado da expressão anterior (LOD W), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução DIV Y. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO Z é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 23. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso STO Z. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável Z.

Figura 13 – Variável Z

4.0 PARTE INFERIOR

4.1. BLOCO 01

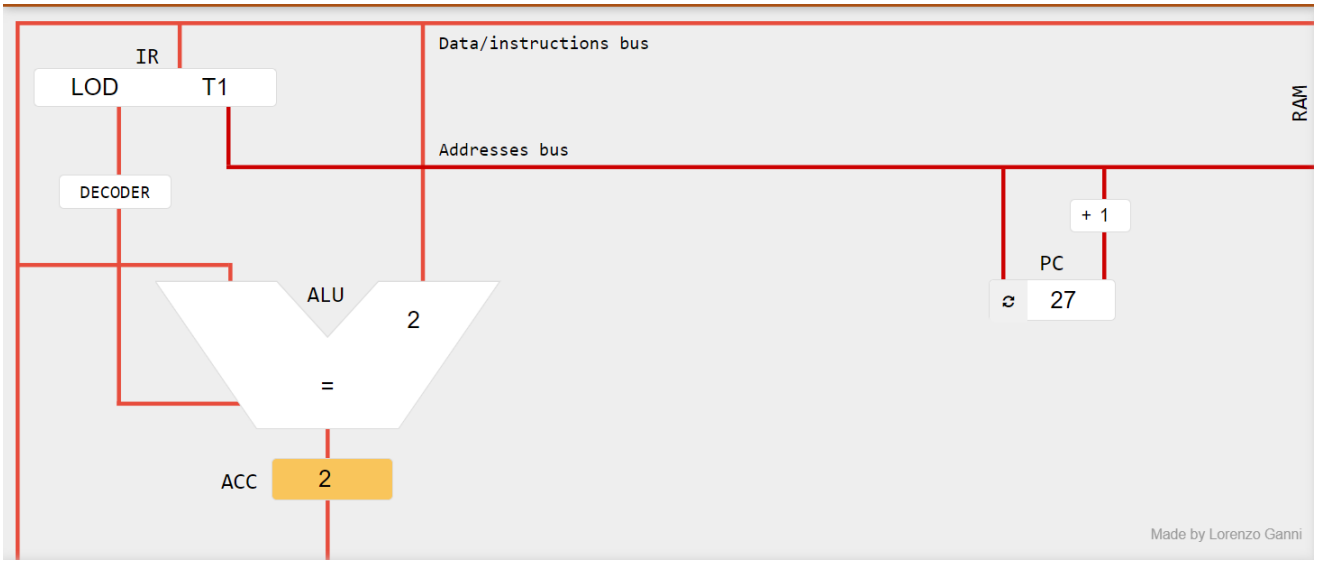

Figura 14 – Execução da primeira instrução

A instrução LOD T1 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 26. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso LOD T1. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

A instrução MUL T2 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 27. Já o Registrador de Buffer de Memória (MBR) recebe a instrução mul da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso MUL T2. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de multiplicação é realizada pela ALU, com o resultado da expressão anterior (LOD T1), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução MUL T2. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução ADD T1 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 28. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso ADD T1. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de multiplicação é realizada pela ALU, com o resultado da expressão anterior (LOD T1 MUL T2), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução ADD T1. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO W é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 29. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso STO W. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável W.

4.2. BLOCO 02

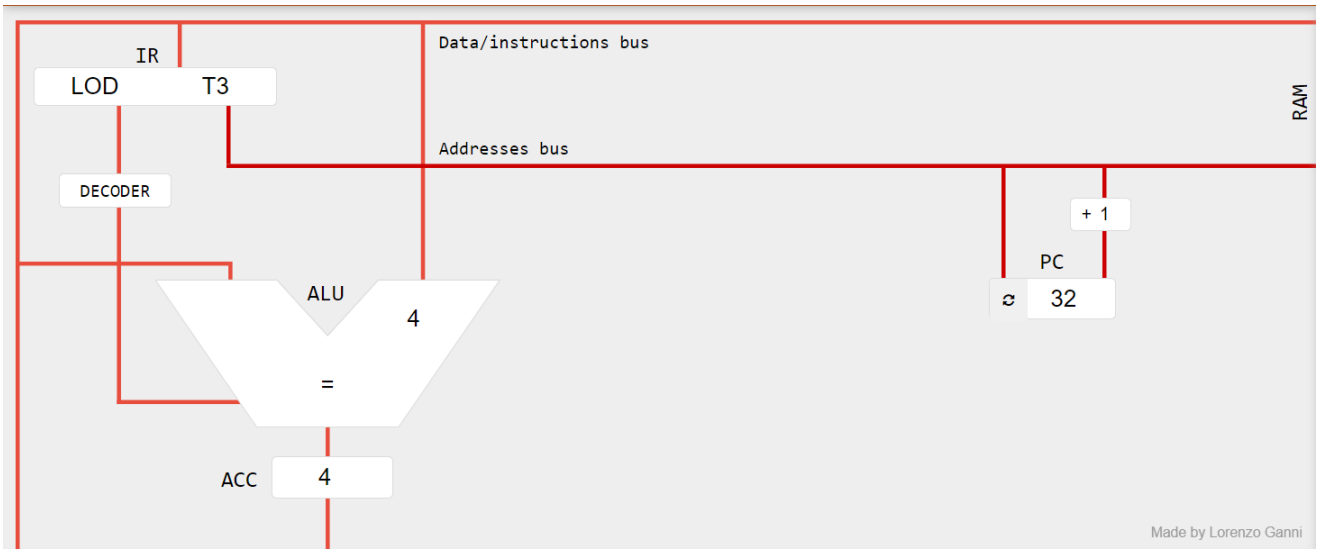

Figura 15 – Execução da primeira instrução

A instrução LOD T3 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 31. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso LOD T3. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

A instrução DIV T1 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 32. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso DIV T1. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de divisão é realizada pela ALU, com o resultado da expressão anterior (LOD T3), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução DIV T1. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 33. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso STO Y. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável Y.

A instrução LOD T2 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 34. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma

temporária a próxima instrução que será executada, no caso LOD T2. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

A instrução MUL T1 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 35. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso MUL T1. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de multiplicação é realizada pela ALU, com o resultado da expressão anterior (LOD T2), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução MUL T1. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução ADD Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 36. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso ADD Y. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de multiplicação é realizada pela ALU, com o resultado da expressão anterior (LOD T2 MUL T1), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução ADD Y. Depois disso o

resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 37. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso STO Y. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável Y.

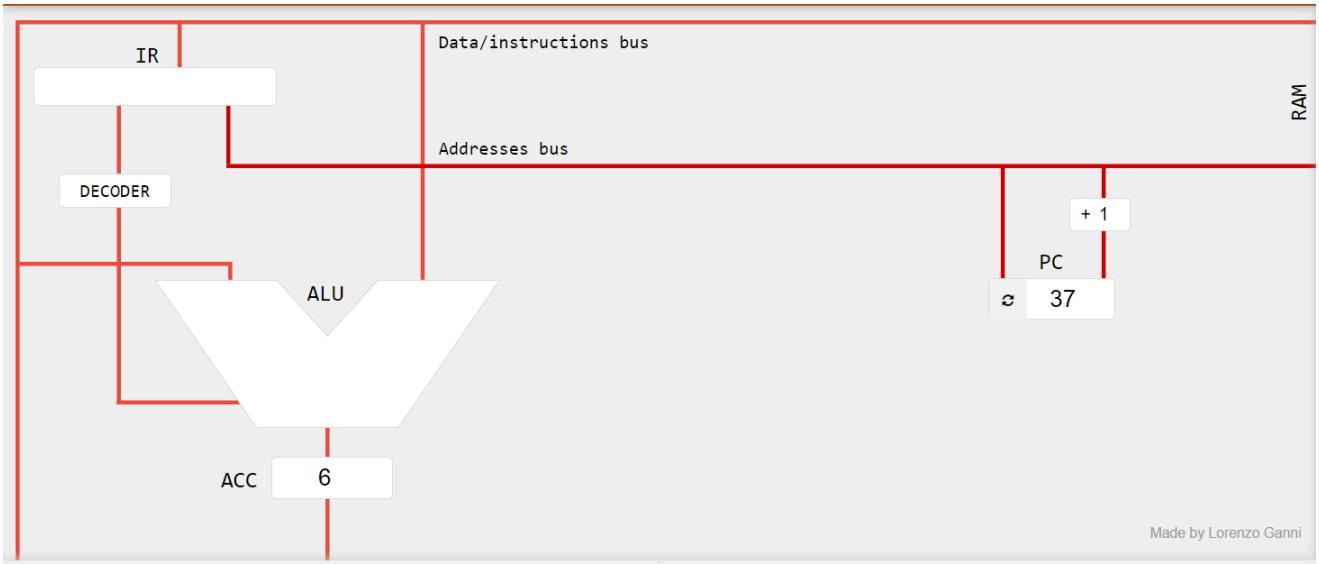

Figura 16 – Resultado final do bloco 02

4.3. BLOCO 03

A instrução LOD W é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 39. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a a próxima instrução que será executada, no caso LOD W. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

A instrução DIV Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 40. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso DIV Y. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de multiplicação é realizada pela ALU, com o resultado da expressão anterior (LOD W), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução DIV Y. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução ADD T1 é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 41. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso ADD T1. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada. A operação de adição é realizada pela ALU, com o resultado da expressão anterior (LOD W DIV Y), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução ADD T1. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 42. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso STO Y. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável Y.

Figura 17 – Execução da quinta instrução

A instrução LOD Z é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 43. Já o Registrador de Buffer de Memória (MBR) recebe a instrução LOD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso LOD Z. Após isso, o Contador de Programas (PC) é incrementado.

A decodificação da instrução é realizada pelo *Decoder*, que transferirá a instrução pelos barramentos internos para a ALU, a operação que será realizada, nesse caso de carregamento, e o resultado da operação é armazenado temporariamente no Quociente Multiplicador e no Acumulador.

A instrução DIV Y é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 40. Já o Registrador de Buffer de Memória (MBR) recebe a instrução ADD da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma

temporária a próxima instrução que será executada, no caso DIV Y. Após isso, o Contador de Programas (PC) é incrementado. A decodificação da instrução é realizada pelo *Decoder*, que transfere para a ALU, por meio dos barramentos internos, a operação que será realizada.

A operação de multiplicação é realizada pela ALU, com o resultado da expressão anterior (LOD Z), que já estava armazenada temporariamente no Acumulador e no Quociente Multiplicador, com a instrução DIV Y. Depois disso o resultado e os operandos são alocados temporariamente no Quociente Multiplicador e no Acumulador.

A instrução STO X é enviada da memória principal (RAM) para o Registrador de Endereços (IR) por meio dos barramentos internos. O Registrador de Endereço da Memória (MAR) especifica o endereço da posição na memória, nesse caso o 46. Já o Registrador de Buffer de Memória (MBR) recebe a instrução STO da memória. O Registrador de Buffer de Instrução (IBR) está presente para guardar de forma temporária a próxima instrução que será executada, no caso STO X. Após isso, o Contador de Programas (PC) é incrementado.

A operação de decodificação da instrução é realizada pelo *Decoder*, que transfere pelos barramentos internos para a ALU, a operação que será executada. Após o processo, o valor e os operandos são armazenados temporariamente no Quociente Multiplicador e no Acumulador, depois, o resultado é armazenado na variável X.

Após todo o processo, a instrução HLT é executada, que encerra a execução do código, incrementando mais um passo ao Contador de Programas (47).

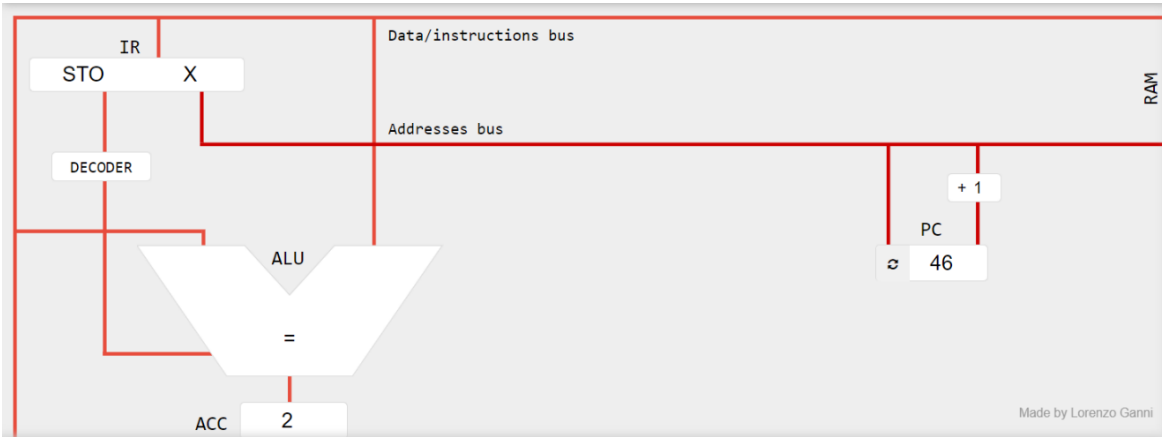

Figura 18 – Armazenamento do resultado final

REFERÊNCIAS BIBLIOGRÁFICAS

MONTEIRO, Mario. **Introdução à Organização de Computadores.** 5. ed. Editora LTC, 2007.