Universidade Federal de Pernambuco

Emmanuel Silva Nascimento

Guilherme Vinicius Nigro Braga

Ruan Alexandre Ferreira Da Silva

João Gabriel Valentim Dias

Erick Martins Gomes Vieira

Relatório da APS4

Nível do Sistema de Computador

Recife

2024

NandGame

O primeiro nível é chamado de Combined Memory( Memória Combinada), que consiste na criação de uma memória juntando dois registradores normais e uma RAM(Random Acess Memory, ou Memória de Acesso Aleatório). Uma Combined Memory facilita o acesso de uma grande quantidade bits de informação, podendo possuir mais de uma característica dependendo da forma que é montado.

Como é mostrado na imagem, a Combined Memory possui 5 Inputs(3 Inputs(a,d,\*a) para escolher o Output, 1 input(X) de 16 bits carregando o valor a ser armazenado e o último sendo o Clock(cl)) e 3 Outputs de 16 bits(A,D,A). O primeiro Output (A) é dado por um registro, que está conectado aos inputs (a, X, e cl), porém, o valor do registro só será atualizado quando a borda do sinal cl for negativa (uma mudança de estados, em específico de 1 para 0). O segundo output(D) possui o mesmo funcionamento, porém utiliza o input (d) no lugar do (a). Já o output de \*A é dado por uma Ram, que está conectada a saída do Registrador responsável pela saída (A) e aos bits (\*a, X e cl). O intuito é que o Registrador A forneça um endereço de armazenamento a RAM, fazendo com que ela armazene quantidades de bits diferentes em cada endereço provido pelo Registrador A.Interface gráfica do usuário, Diagrama

Descrição gerada automaticamente

O segundo nível é chamado de Instruction(Instrução), que consiste na criação de uma das partes de um Control Unit(Unidade de Controle), que consiste na junção de um select a um condition e a um ALU (todos de 16 bits). O Instruction possui um total de 4 Inputs(1 input responsável por fornecer ao ALU instruções, 1 input responsável pelo valor de X que o ALU irá receber e 2 inputs sendo responsáveis pelo valor de Y), e 5 Outputs(1 para o resultado do ALU, 3 pelos valor de (a,d e \*a) e um pelo valor de j. O bit I passa por um splitter, facilitando o acesso aos bits de I e proporcionando mais precisão ao ALU,e após isso, determinados bits de I serão responsáveis por aspectos diferentes do ALU. Caso o bit de número 12 esteja ativo, a entrada Y do Alu será \*A, caso esteja desativado, será A. os bits do 10 ao 6 são responsáveis pelas operações do ALU, sendo o bit 10 responsável pelo operados do número, bit 9-8 pela operação matemática a ser realizada e o bit 7 por zerar o resultado e o bit 6 por trocar os valor de X e Y. Já os bit 5 ao 3 são responsáveis pelos valores de a, d e \*a, respectivamente. E finalmente, os bit 2 ao 0 são encarregados pelo valor de j (Bit 2 = Less Than , Bit 1 = Equal to e Bit 0 = Greater Than).

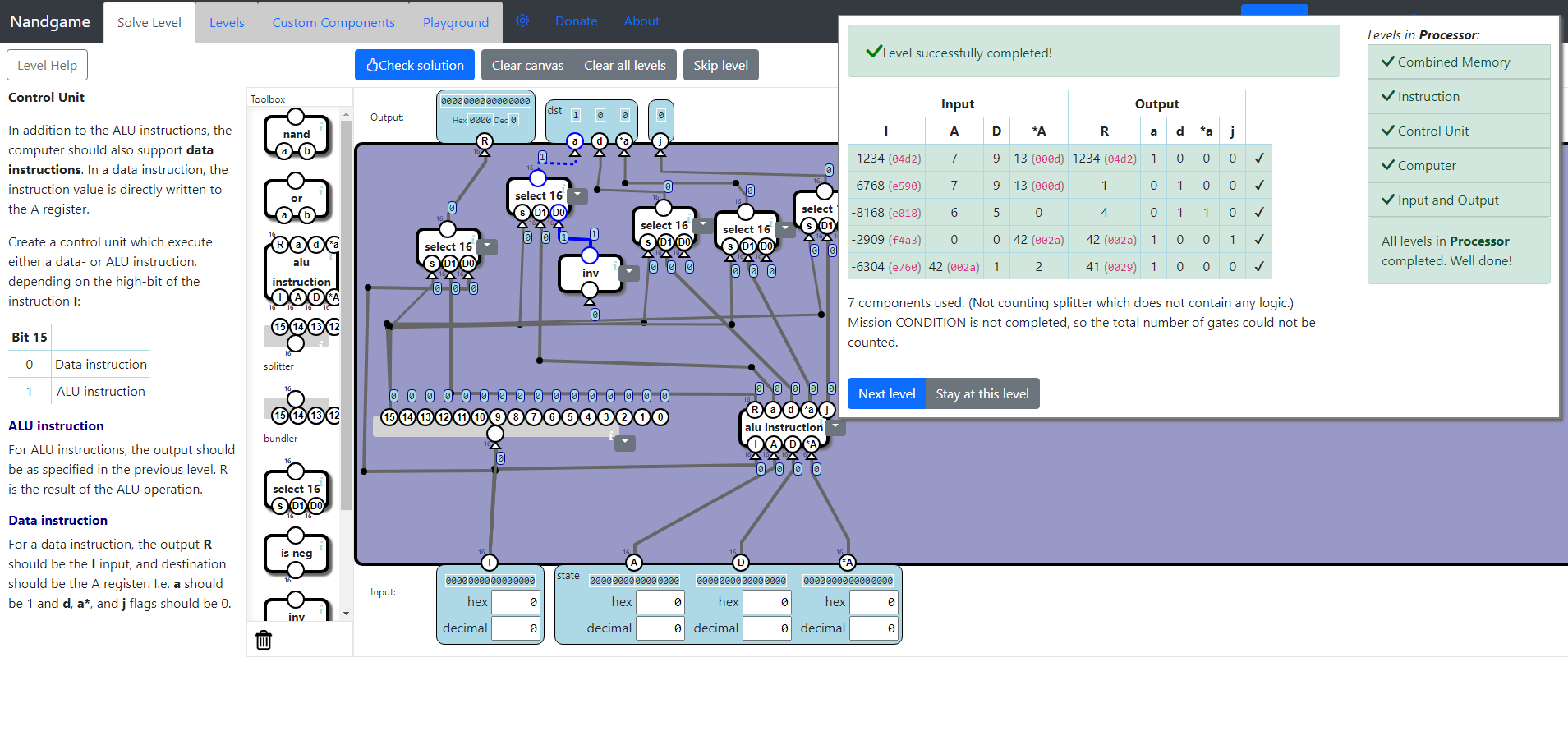
Diagrama

Descrição gerada automaticamente

O terceiro nível é chamado de Control Unit (Unidade de Controle), que consiste na junção de um ALU com o input de I. É utilizado um Seletor para escolher qual dos valores utilizar (Resultado do ALU ou I). Caso o bit 15 do valor de I for 1, então o valor a ser utilizado no output R será o do ALU, caso contrário, será o de I.

Junto a isso, caso o valor do bit 15 de de I for 1, os valores de a, d ,\*a e j serão os mesmos do ALU, caso não, todos exceto a serão 0.

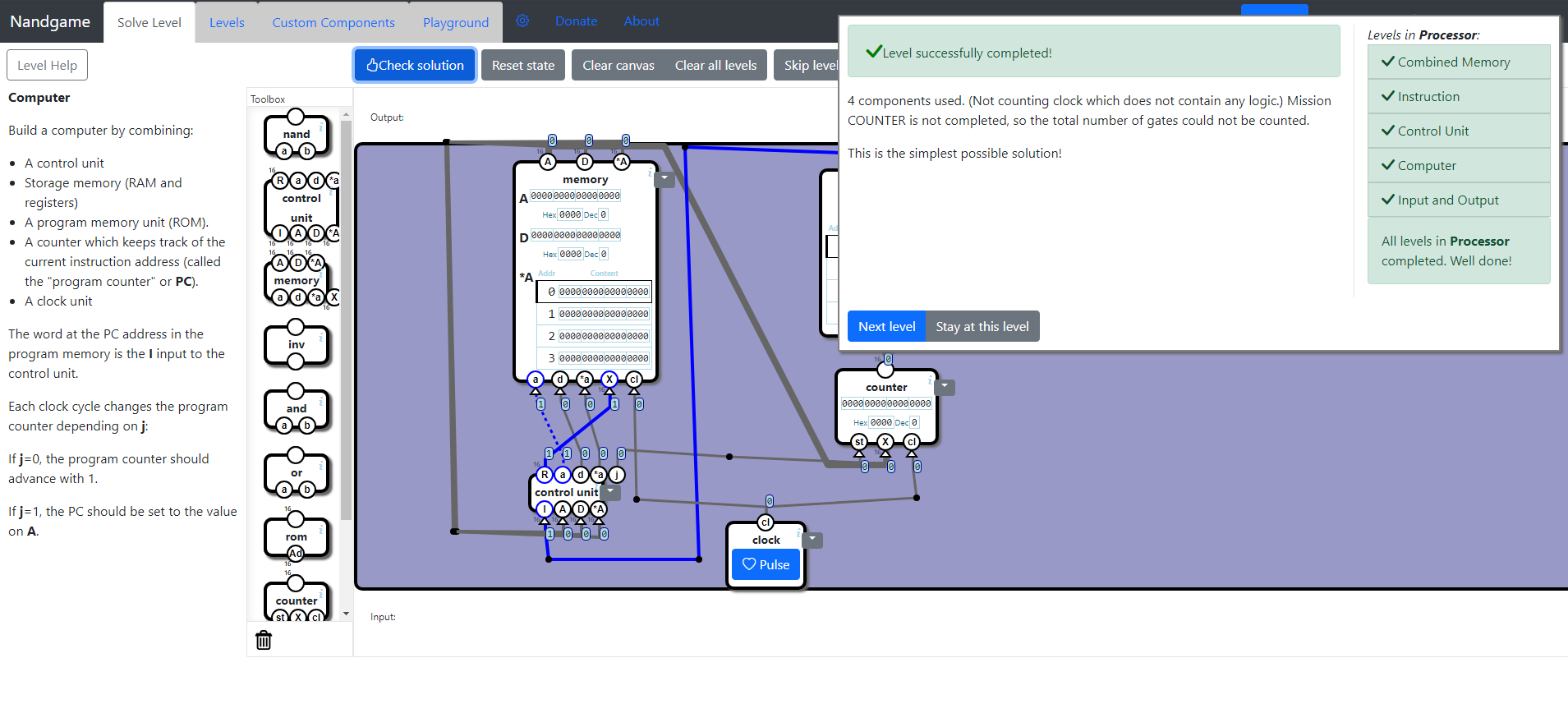
A Unidade de Controle é essencial para o funcionamento de um computador, garantindo as atividades internas da CPU, assegurando que o sistema execute as tarefas de maneira organizada e eficiente. Ela coordena a execução das instruções e o fluxo de dados, possibilitando que o computador realize operações complexas de forma eficaz.



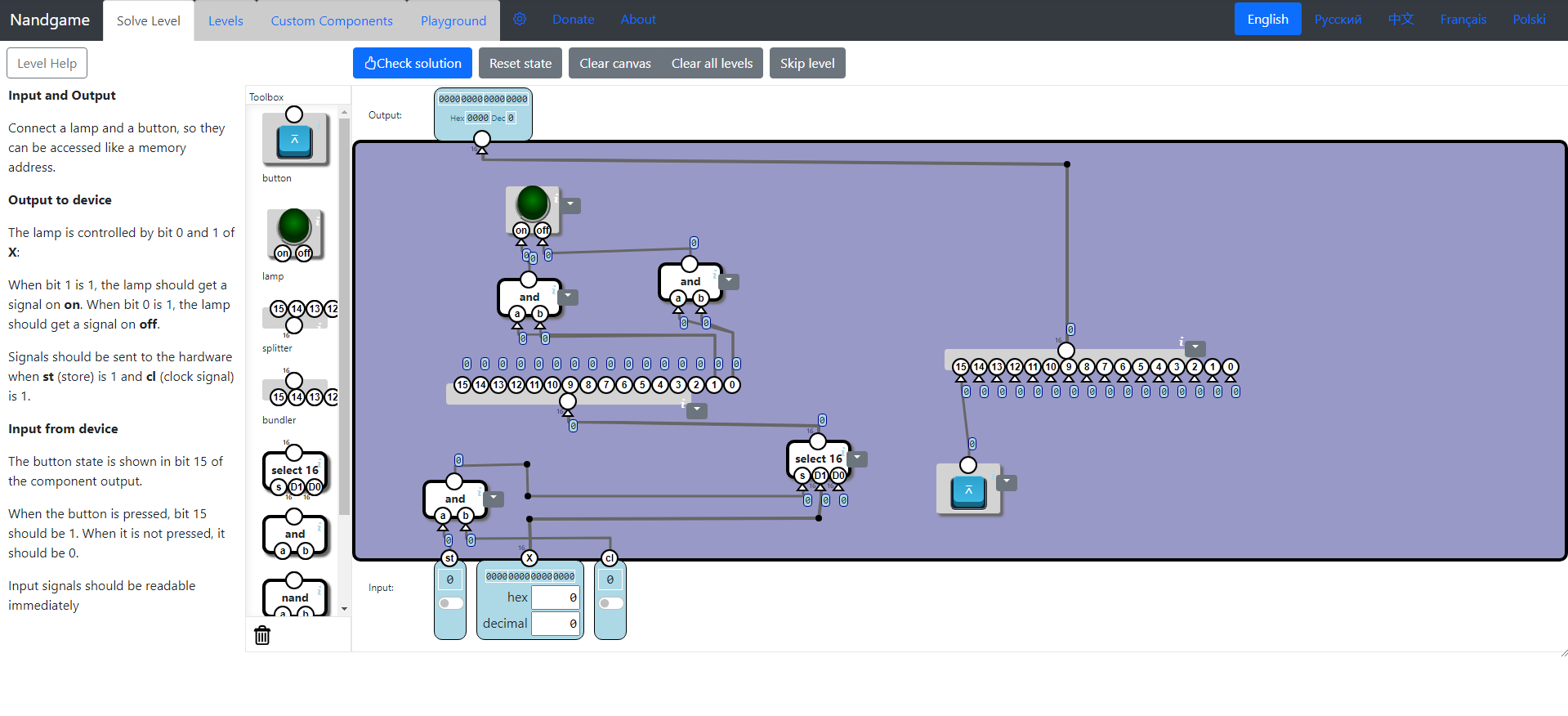
O quarto nível é chamado de Computer, que consiste na criação de um computador utilizando uma RAM, ROM(Read only memory ou Memória Somente Leitura ), PC(Program Counter ou Contador de Programas), CLOCK e CU(Control Unit ou Unidade de Controle).

O clock fornece sinal tanto para o PC quanto para o ROM, garantindo que enquanto o sinal de j do CU não for ativado, o será adicionado 1 no valor do PC, enquanto na ROM, garante que o valor resultante do CU seja alocado nele própio.

Esses componentes trabalham em conjunto para executar programas e processar informações. O CU orquestra a operação do sistema, o PC gerencia a sequência de instruções, e a RAM e ROM fornecem os locais necessários para armazenar dados e código. O CLOCK mantém tudo em sincronia, garantindo que o computador funcione de maneira eficiente e ordenada.



O quinto e último nível é chamado de Input e Output, que consiste no gerenciamento de sinais lógicos para diferentes propósitos. Porém, o diferencial desse nível é a existência de um botão e de uma lâmpada, que devem ser acionados de forma correta para o nível ser concluído. É determinado que para a lâmpada ser acesa, ela depende dos bits 0 e 1 de X, que é um número de 16 bits, e também dos valores lógicos de st e cl. Se cl e st estiverem ativos, então o estado da lâmpada dependerá do split do bit X. Caso o Bit 1 esteja ativo, então a Lâmpada deve ser acesa, caso o bit 0 esteja ativo, então a lâmpada permanece apagada, e caso um número que possua tanto o bit 1 quanto o bit 0 ativos, ela permanece desligada. Agora para o botão, um bundler foi utilizado já que era pedido que apenas o bit de maior significância fosse usado como output.



Memória

A memória é a parte onde guardamos as informações que serão utilizadas pela CPU no funcionamento do computador, ela é separada em 3 partes:

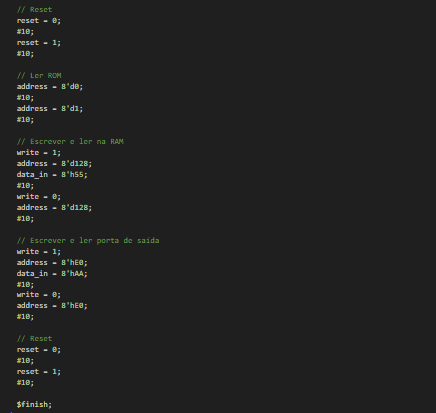
ROM → É a memória onde guardamos as instruções que serão enviadas à CPU.

RAM → É a memória temporária do computador, normalmente utilizada para guardar dados que serão usados pela CPU.

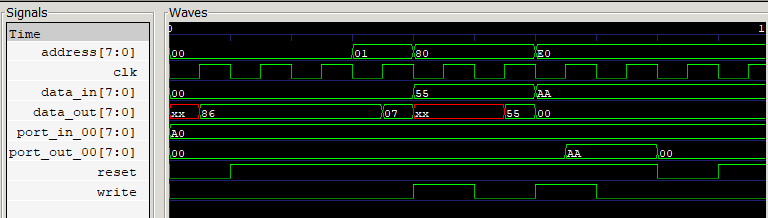
IO → É a memória que guarda as entradas e saídas do do computador, para que possam ser lidas pelo computador (no caso da entrada), e para ser lida por outros componentes fora do computador (no caso da saída).

Após utilizar os módulos fornecidos pelo arquiteto, fomos capazes de juntar todas em memory.v e criar o testbench para a compilação e a simulação em ondas.

Fotos dos testes no testbench:



Fotos das simulações em onda pelo gtkwave:



Agora explicando cada teste:

Primeiro temos um reset para reiniciar a máquina. 10ns para os tempos de espera da máquina, o reset está em negedge então ele ficará como “1” o teste inteiro até ser atualizado para 0 pela borda negativa.

Segundo temos a leitura da ROM onde lemos o endereço 00 na ROM nos dando o valor 86, que significa load\_A imediato, e após 10ns lemos o endereço 01 da ROM, que armazena o valor 07, após isso temos mais 10ns.

Terceiro temos a escrita da RAM, o sinal write sobe para 1, o que significa que devemos armazenar o valor 55 no endereço 80, após isso temos 10ns de espera para poder ler o endereço 80 na RAM, que mostra exatamente o valor 55.

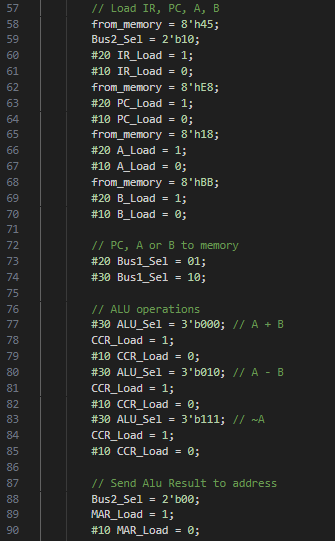
Após isso temos a escrita e leitura na porta de saída, write sobre para 1 e é nos dado o endereço da porta 0 (E0) e o valor a armazenar (AA), após isso temos mais 10ns de espera, write cai para 0 e nos é dado o endereço E0 novamente, só que dessa vez para ler a porta 0, que mostra o valor AA.

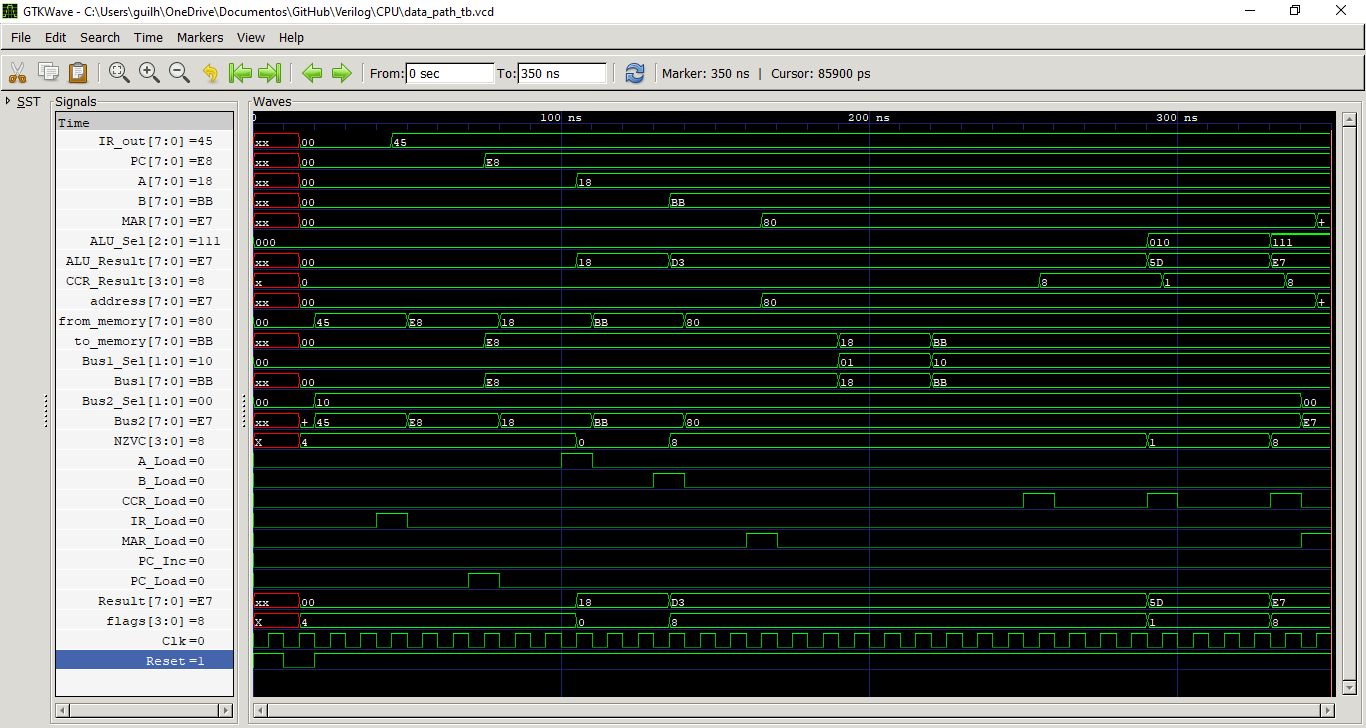
Por último temos mais um reset, ele começa com 0, e quando sobe para 1, todas as saídas caem para 0.

Data Path

O data path é um conjunto de componentes que executam as operações de dados dentro da CPU. Nele estão presentes os elementos responsáveis pelo transporte e processamento dos dados, registradores, ALU, multiplexadores, etc.

Após concluir o módulo data\_path.v fornecido pelo engenheiro, criamos um testbench para testar casos essenciais que o data path deve cobrir. Carregamento dos registradores A, B, PC e IR, enviar dados para a memória via barramento de dados, operações da ALU e endereçamento de dados via MAR.

Testes do testbench:  


Simulando os dados obtidos no GTKWave obtivemos o seguinte resultado:  


Com isso concluímos que ao receber os dados corretos da control unit, o data path consegue realizar suas operações corretamente.

Control Unit

A Control Unit é responsável por dirigir as operações da CPU, gerando os sinais de controle necessários para coordenar e controlar o data path durante a execução das instruções. Ela interpreta as instruções decodificadas e gera sinais que determinam quais operações devem ser realizadas, quais dados devem ser utilizados e para onde os resultados devem ser direcionados.

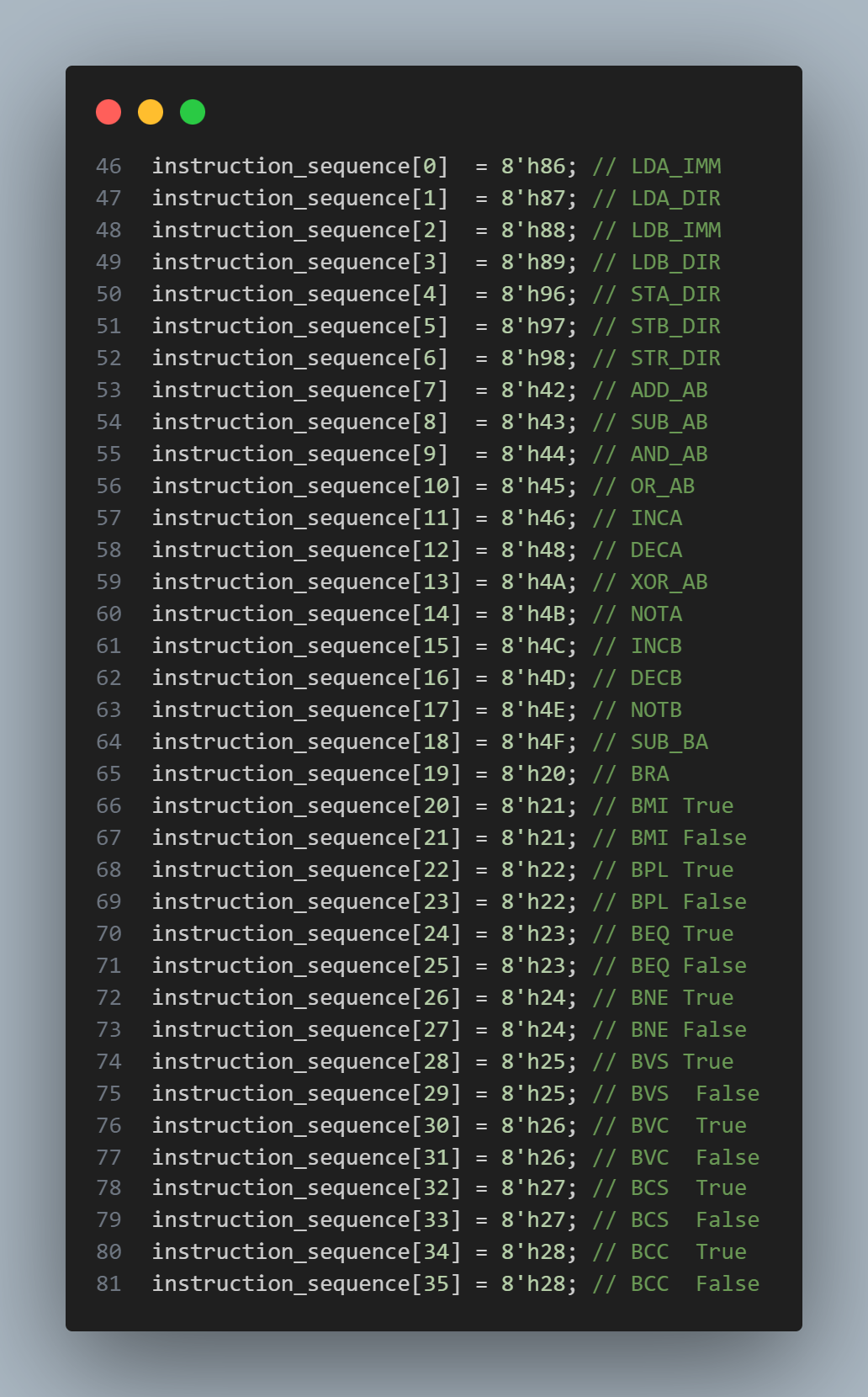
Antes de concluirmos o módulo control\_unit.v fornecido para o engenheiro, definimos as instruções que poderiam ser usadas pela nossa CPU no arquivo instructions.v, dessa forma as definições poderiam ser chamadas em outros arquivos posteriormente.

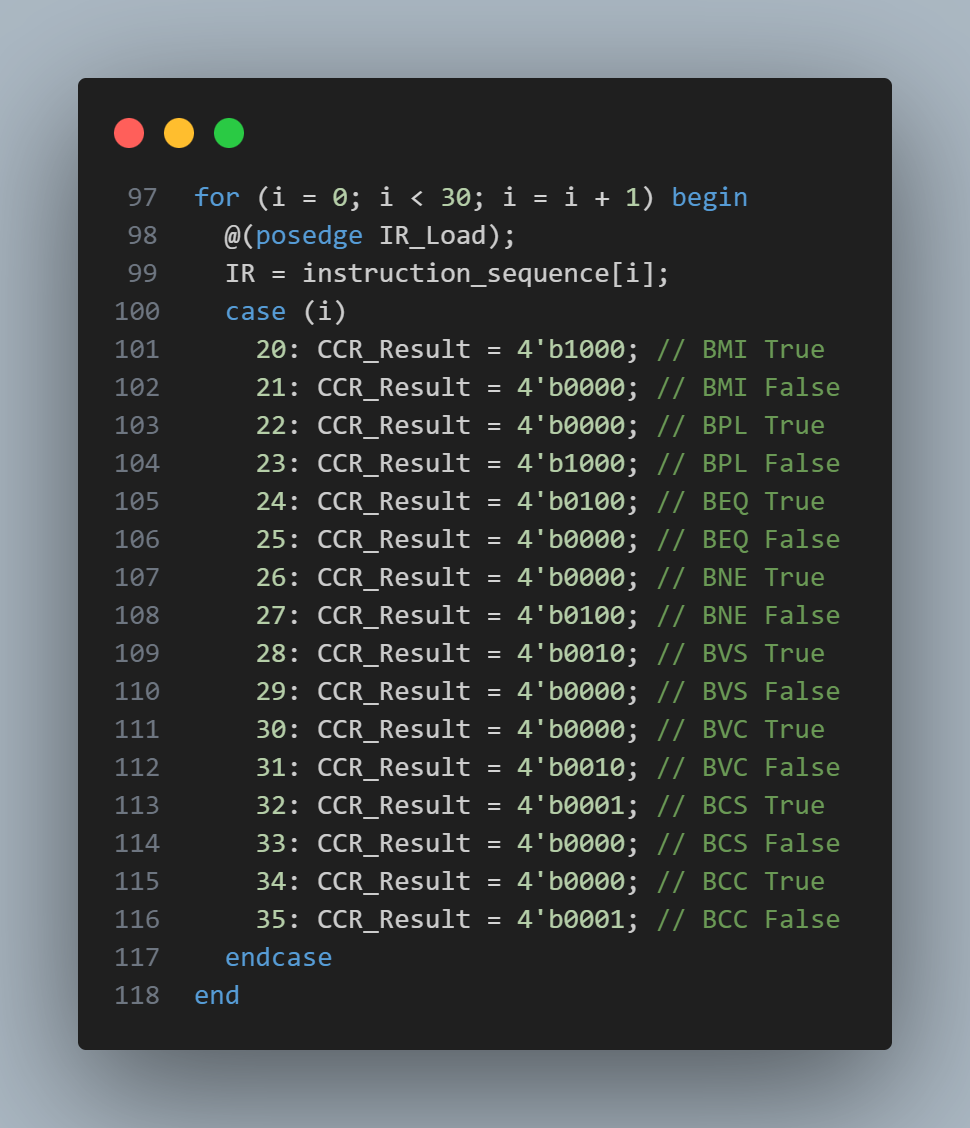
Instruções do instructions.v:



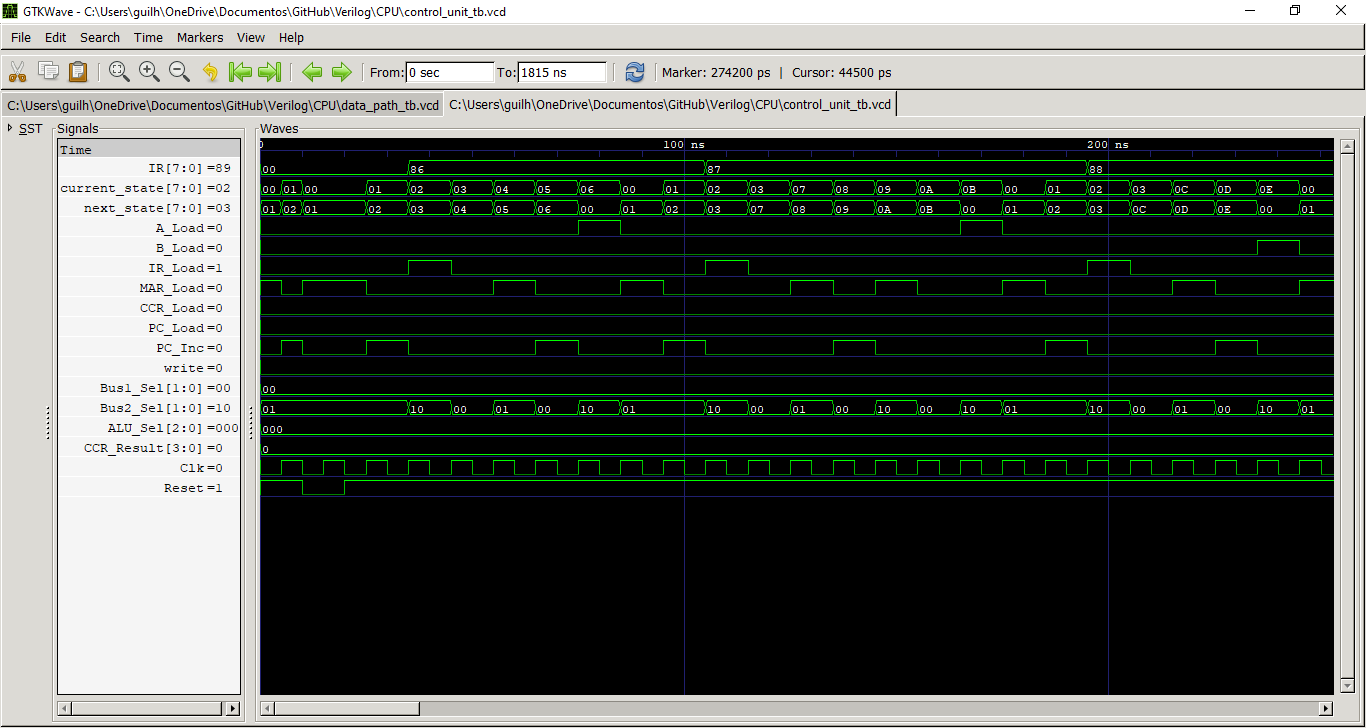
Após isso, concluindo a control\_unit.v, ela era capaz de realizar qualquer uma das instruções definidas.

Para testar a control unit, criamos um testbench que contempla todas as instruções possíveis. Não poderei mostrar todas as formas de onda aqui, porém elas estarão presentes junto aos arquivos do código fornecidos posteriormente.

Instruções do testbench:  


Lógica para leitura das instruções:  


Formas de onda das três primeiras instruções:



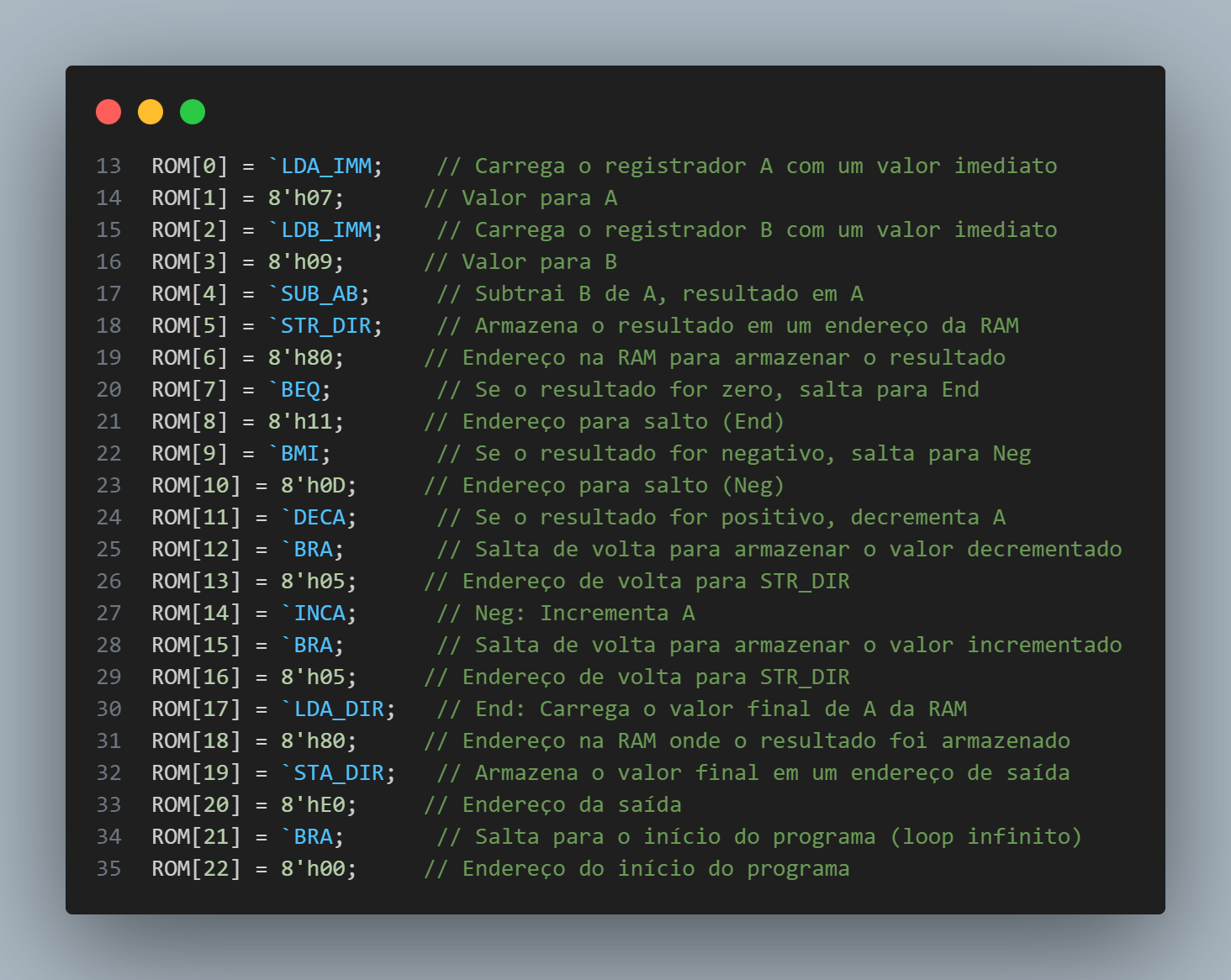
Analisando as formas de onda percebemos que cada instrução está sendo executada perfeitamente.

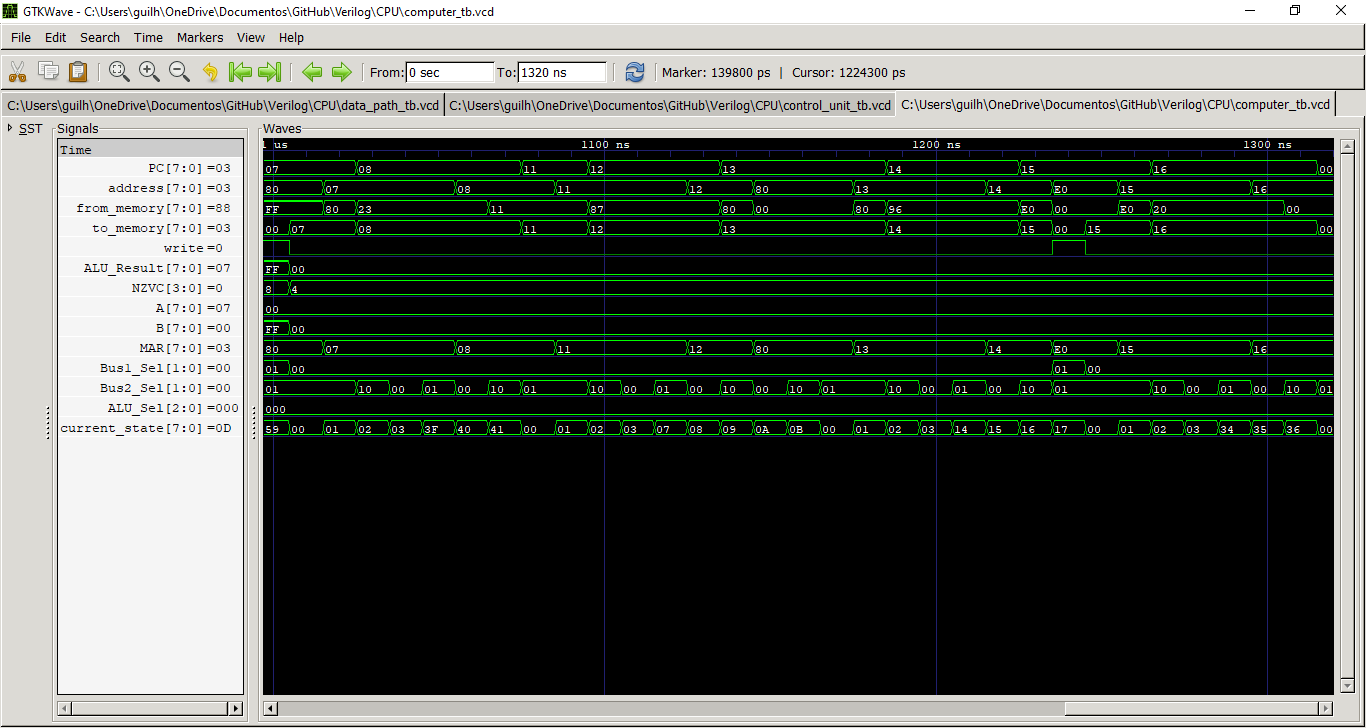
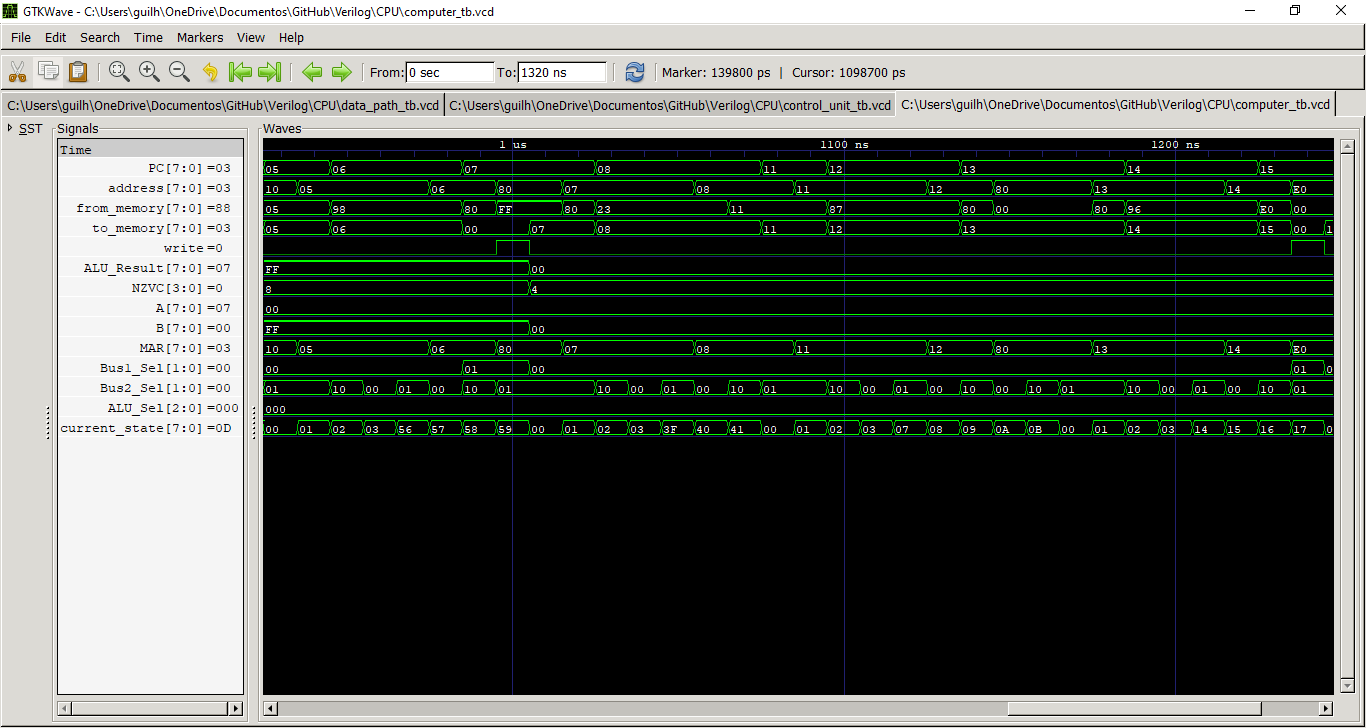
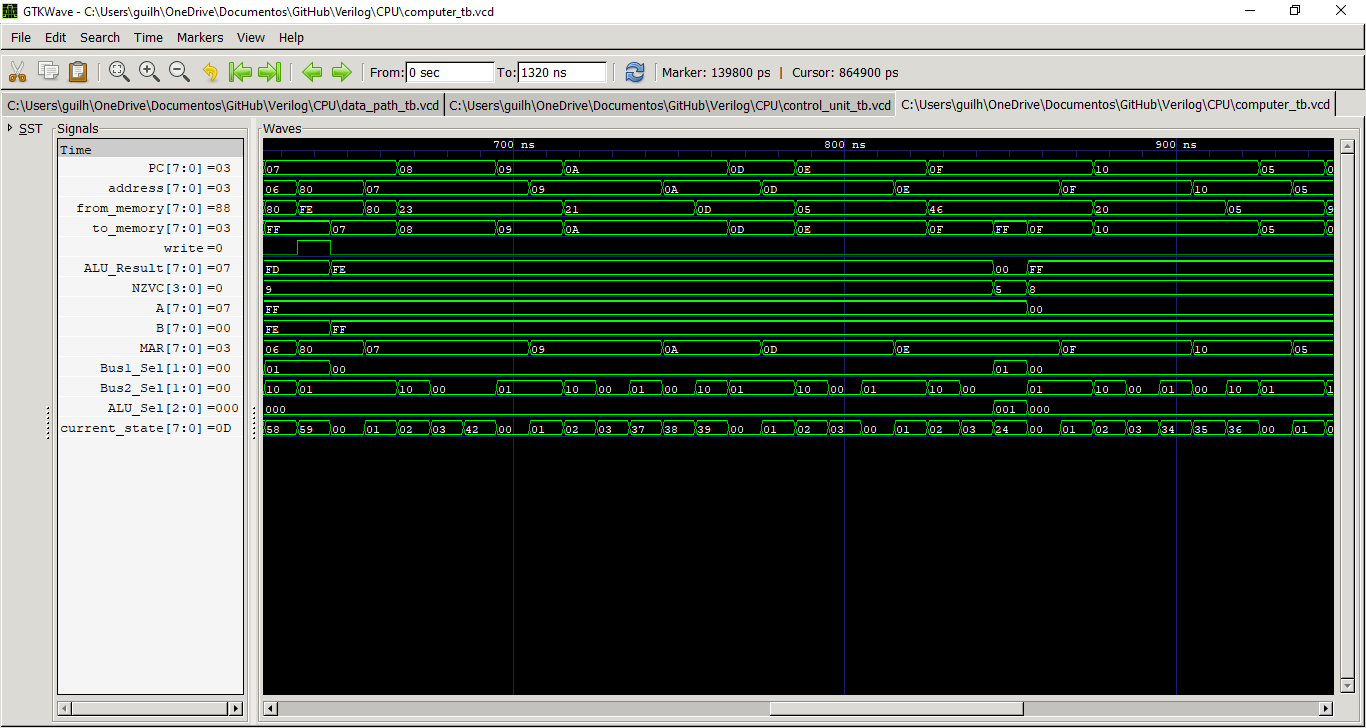
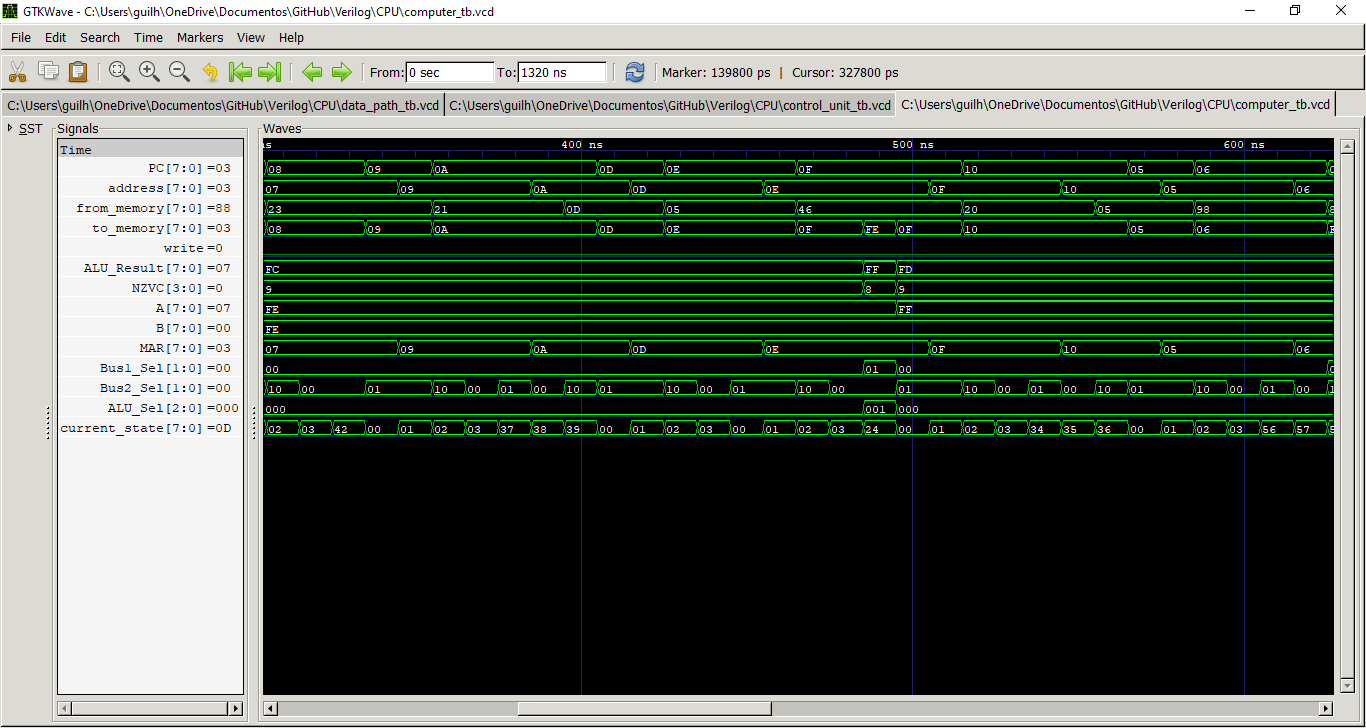
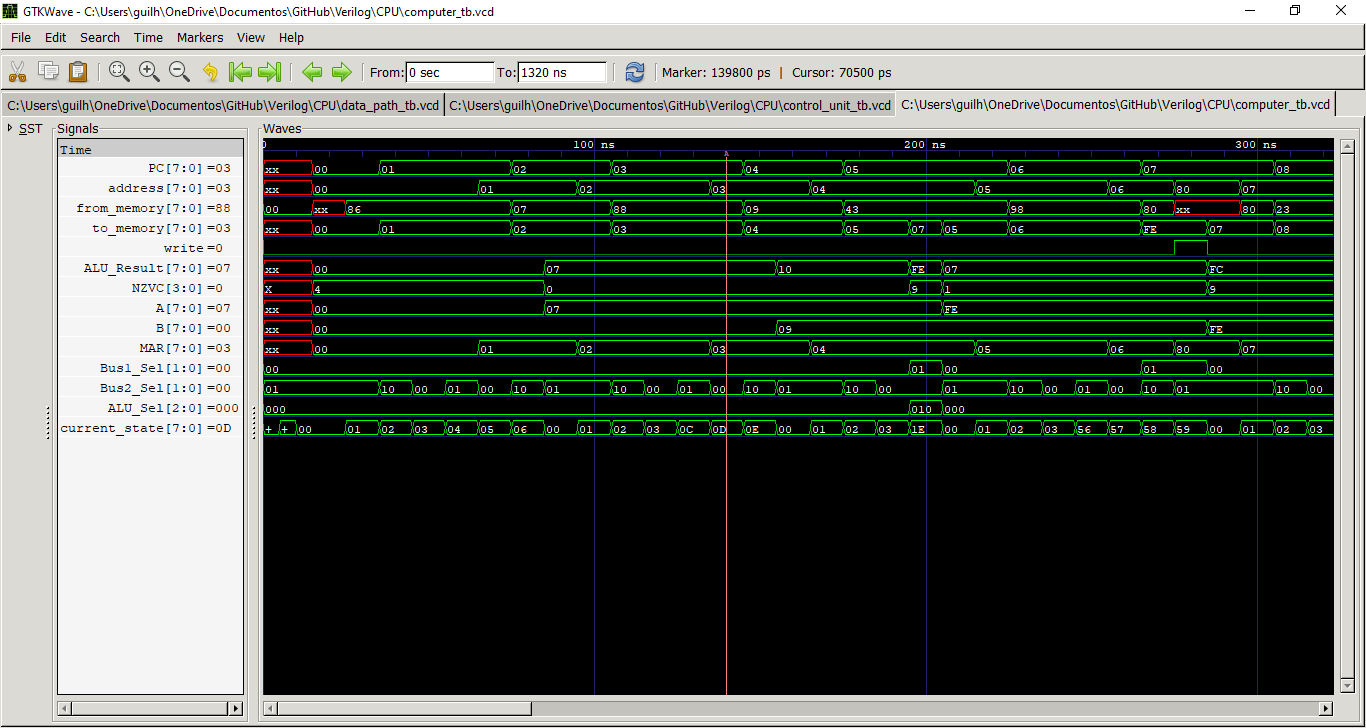
Computer

Finalizando todos os módulos componentes, finalmente estávamos aptos para juntar cada módulo para formar o computador.

Para testar nosso computador, desenvolvemos um programa nulificador de contas. Esse programa usa como input um conta matemática qualquer A - B e é capaz de transformar o resultado em 0 executando um algoritmo que vai ser melhor entendido ao ver o programa. O programa foi armazenado na rom para leitura pela CPU.

Código do programa nulificador de contas:



O testbench de execução do programa apenas resetava o computador para que funcionasse corretamente, pois não usamos portas IO para execução do programa, porém elas poderiam ser utilizadas sem quaisquer problemas.  
  
As formas de onda de resultado do GTKWave mostraram que o computador conseguiu obter um resultado satisfatório, anulando a conta inicial e voltando ao loop do começo do programa. Em um caso de um computador real a subtração A - B seria alterada conforme os inputs do usuário na IO do computador e com isso o computador faria passos diferentes para cada input. Como no nosso caso a subtração A - B continua a mesma, analisar apenas um loop já é satisfatório para nosso teste.  
  
Formas de onda de resultado:  


Para melhor entendimento, as ações que o computador realiza são:  
1 - Gravar um valor no registrador A, um valor no registrador B e realizar a subtração;

2 - Armazena o valor na RAM;  
3 - Verifica que o resultado deu negativo;

4 - Faz um jump para o End (incrementar A);

5 - Faz um jump para armazenar o valor de A novamente;

6 - Faz os passos 3, 4 e 5 novamente, A agora é 0;

7 - Verifica que A é 0 e faz um jump para o End;

8 - Coloca A em um endereço de saída;

9 - Volta para o começo do loop.

Auto Avaliação

O sucesso do projeto foi algo bem satisfatório para a equipe. Analisar um computador por dentro e desenvolvê-lo inicialmente parecia algo extremamente desafiador, mas com o desenvolvimento do projeto vimos que era possível. Quando finalmente conseguimos fazer o computador funcionar achamos muito interessante como cada coisa se relacionava para tudo dar certo no final. Acreditamos que, para melhorar o nosso próximo projeto, talvez nós devêssemos inovar com criatividade, fazendo algo além do que foi pedido. Com base no sucesso que obtivemos, consideramos o nosso projeto nota 9.