Escola Politécnica da Universidade de São Paulo

PCS 3732 – Laboratório de Processadores

16 de Agosto de 2024

WebProc

Nome Completo	**nºUSP**
Arthur Milani Giovanini	12745146
Gustavo Hess Vaz de Lima	12550601
Keith Ando Ogawa	12550772

1. Motivação e Inspiração

A motivação do projeto é oriunda de duas fontes: Uma dificuldade para os primeiros passos no conteúdo e a Wiki disponível para acompanharmos a disciplina. Primeiro, observamos que muitos dos tópicos apesar de extremamente interessantes apresentam uma certa barreira inicial para ser possível realizar um aprendizado rápido dos conceitos. Conglomerados de conhecimentos da matéria tem aplicações práticas divertidas e factíveis de forma que o aprendizado se torna fluído ao amadurecer seu conhecimento, entretanto, no início, acreditamos que as pequenas porções de conteúdo não são tão facilmente praticáveis, o que torna o estudo menos eficiente e traz o desafio de conhecer as partes enquanto se monta o todo.

Por outro lado, um grande aliado e direcionador do aprendizado durante a disciplina foi a Wiki, a qual, aula a aula, trouxe, de forma unificada, conteúdos que pudessem ser aplicados na mesma semana, amenizando a dificuldade citada.

Portanto, inspirado na Wiki e no regime de aulas, e buscando maximizar a profundidade e eficiência durante o aprendizado, WebProc buscar ser uma plataforma digital interativa para aprendizado dos conteúdos da disciplina de Laboratório de Processadores, de modo que, intermediando por interações prática, torna os passos necessários para se conseguir aplicar o conteúdo menores.

2. Problema

O Problema que encontramos, é que alguns dos assuntos abordados necessitam de uma visualização e interação para serem compreendidos de uma forma mais clara, assim, um site interativo permite aos alunos compreenderem esses temas de forma mais completa.

3. Protótipo

Inicialmente construímos três páginas abordando os seguintes temas: Instruções do ARM, MESI (Modified, Exclusive, Shared, Invalid) e Memory Management Unit (MMU)

a. Página de Instruções

Com o objetivo de tornar mais palpável o funcionamento e formato das instruções durante a leitura inicial ou mesmo para ser uma consulta rápida de comportamento, a página de instruções foi desenvolvida de modo a abordar, inicialmente, os principais tópicos do conjunto de instruções estudado do A32.

O primeiro bloco didático aborda a execução condicional, mecanismo presente em todas as instruções do conjunto. No diagrama interativo deste bloco, é possível interagir tanto com os bits de condição quanto com as flags do registrador CPSR, tendo a informação se há a execução ou não em cada caso.

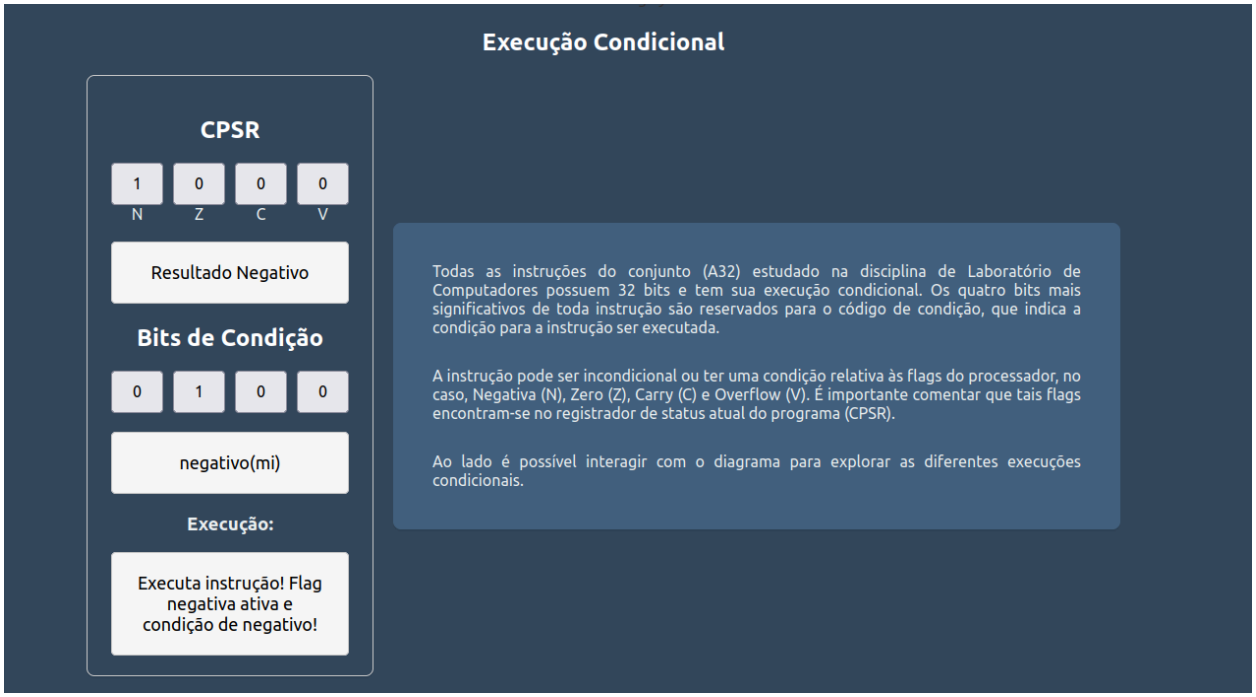

Imagem 1: Diagrama Interativo de Execução Condicional

O segundo bloco didático aborda o formato das instruções lógico-aritméticas, tendo as partes mais relevantes da instrução como partes interativas para os usuários inserirem os bits desejados. Tendo informações sobre cada instrução, e como cada mudança de bit altera o funcionamento dela.

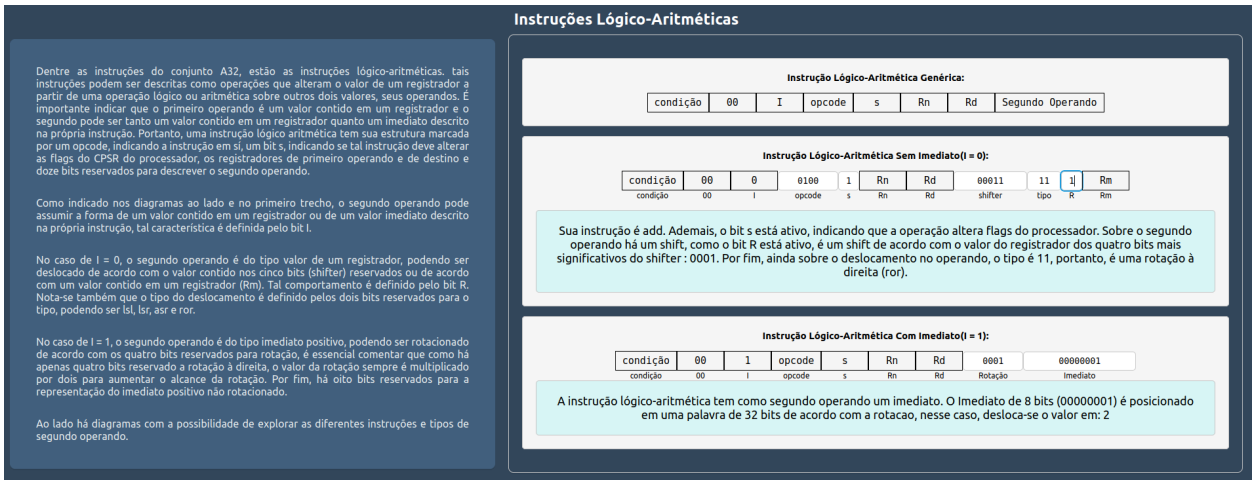

Imagem 2: Diagrama Interativo de Formato de Instruções Lógico-Aritméticas

O terceiro bloco didático aborda os saltos relativos, mecanismo de desvio de fluxo de execução presente no conjunto de instruções A32. O bloco didático busca explicitar o formato da instrução branch e a forma de desvio relativo, indicando como o offset do desvio aparenta na instrução e o resultado do desvio de forma relativa, alterando a simulação do PC presente no bloco de acordo com o offset inserido pelo usuário.

Imagem 3: Diagrama Interativo de Salto Relativo

O quarto bloco didático aborda também saltos, entretanto, agora absolutos. Este bloco procura mostrar o funcionamento da instrução MOV, que copia de um registrador a outro, de modo que o usuário possa interagir com um banco de registradores simulado para entender o mecanismo de salto absoluto que é construído a partir dessa instrução.

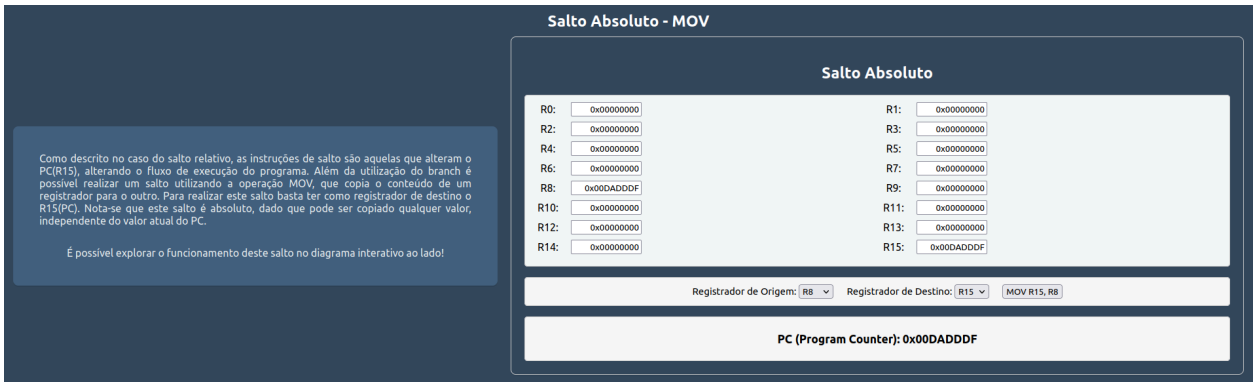

Imagem 4: Diagrama Interativo de Salto Absoluto

Por fim, o último bloco didático aborda as instruções de acesso à memória, que, no A32, por o ARM ser um processador RISC, são apenas as de load e store. Nesta seção busca-se explicitar o formato das instruções de load e store, explicando como cada parte que o usuário insere altera a instrução final e explicando se o formato escolhido é pós ou pré fixado.

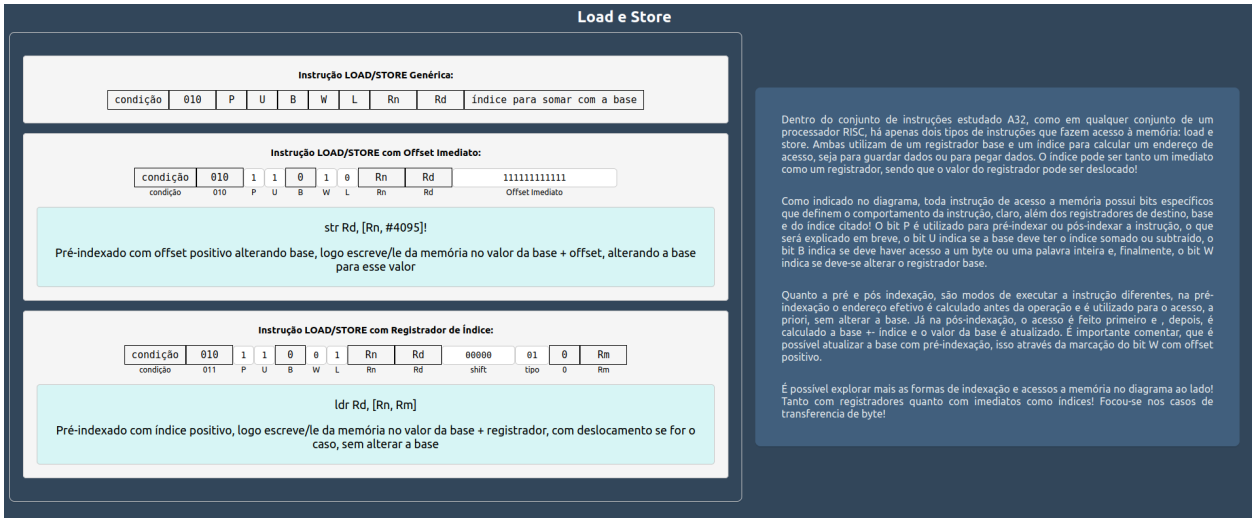

Imagem 5: Diagrama Interativo de Acesso a Memória

b. Página de MESI

Com o objetivo de facilitar o entendimento do protocolo de coerência entre caches MESI, foi construída a página simulando o protocolo com duas caches de processadores e uma memória principal. A página consiste em duas memórias cache L1, relativas a cada processador, e uma memória física mais externa compartilhada entre os processadores. Ademais, tem-se uma interface para interação do usuário com as caches por meio de acessos simulados à mesma por escrita e leitura.

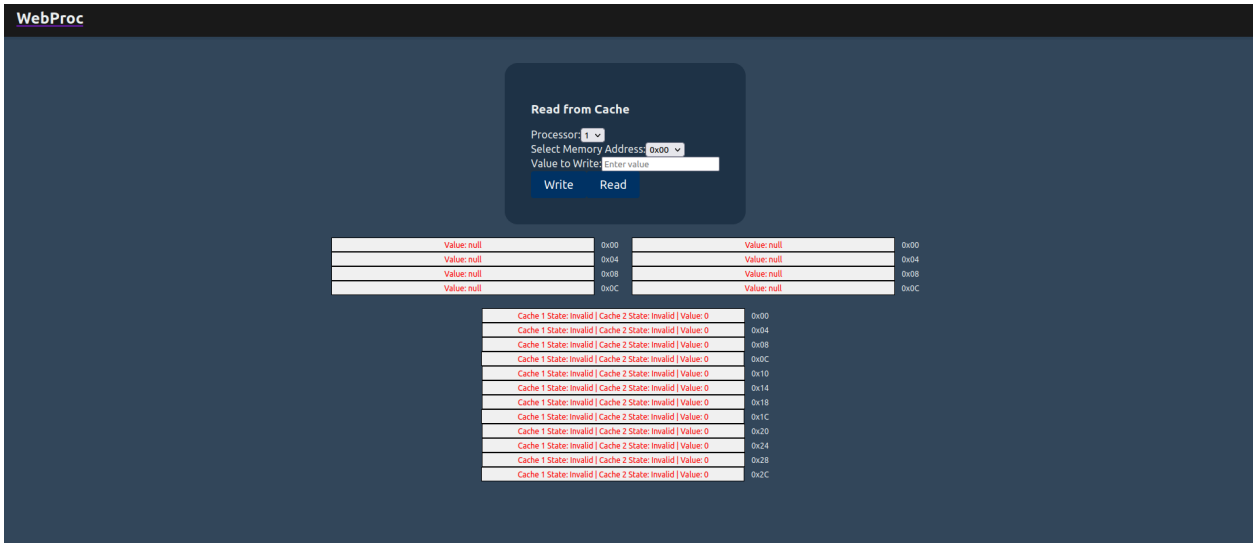

Imagem 6: Diagrama MESI

Caches: As caches são inicializadas sem valores, o que pode ser notado pelos valores nulos em cada posição de memória das mesmas e os estados inválidos mostrados em cada endereço da memória compartilhada.
Memória: A memória compartilhada possui algumas informações relevantes, como o valor de cada posição de memória, os estados de cada bloco na cache do processador 1 e na cache do processador 2.
Interface: A interface de comunicação com os usuários possui três campos principais, sendo eles o de endereço, responsável por selecionar a região de memória a ser acessada, o de processador, indicando qual cache irá ser utilizada, e o de valor de escrita, que consiste no valor a ser escrito na posição de memória pelo processador selecionado. Note que há dois botões, um para acesso a uma posição de memória por escrita, enquanto o outro realiza um acesso por meio de leitura.

Funcionamento:
O MESI se baseia na atribuição de estados a cada bloco em cada cache e nas transições entre as mesmas para garantia de coerência entre os valores das caches. Os estados são Modified, Exclusive, Shared e Invalid:

- **Invalid:** O estado inválido representa um bloco que não está presente na cache em questão ou está desatualizado em virtude de uma escrita em outra cache.
- **Exclusive:** O estado exclusivo indica que um bloco está presente exclusivamente em uma das caches, e apenas com permissão para leitura sem transição de estados. Para ocorrer uma escrita, seria necessária a transição para o estágio *Modified*, conforme explicado posteriormente.
- **Shared:** o estado compartilhado é semelhante ao *Exclusive* no quesito de permissões, dado que é possível apenas executar escritas em tal bloco sem transição de estados. Ademais, tal estado ocorre quando um bloco está presente em mais de uma cache simultaneamente.
- **Modified:** Por fim, o estágio modificado ocorre apenas quando ocorre uma escrita no bloco em questão. Desta forma, há permissão tanto para escrita, quanto para leitura sem transição de estágios. Mas note que se outra cache tentar ler tal bloco, ele irá para o estágio *shared* novamente, bloqueando a escrita novamente.

Principais Transições de Estado:

Neste trecho será explicitado as transições de estados mais importantes para tal método, além de explicitar algumas considerações no projeto:

- **Invalid → Exclusive:** Esta transição ocorre apenas quando, em um momento inicial, um bloco de memória não está presente em nenhuma das caches, se alterando para *Exclusive* ao ocorrer uma leitura.
- **Invalid → Shared:** Esta transição ocorre quando, em um momento inicial, um bloco de memória está presente em algumas das caches e, posteriormente, ocorre uma leitura do mesmo bloco por uma cache que não o possuía.
- **Shared → Modified:** Ocorre quando há, inicialmente, mais de uma cache com um bloco em questão e, posteriormente, ocorre solicitação de escrita em tal bloco por algum processador. Desta forma, o estado da cache que solicita a escrita irá para Modificado, enquanto os demais estados relativos às outras caches serão invalidados.
- **Modified → Shared:** Ocorre quando há uma solicitação de leitura quando algum dos blocos está no estágio Modificado, Desta forma, o bloco que estava no estágio modificado transita para o compartilhado, enquanto o bloco da cache que solicitou escrita transitará do estágio inválido para o compartilhado.
- **Shared → Invalid && Exclusive → Invalid:** Por fim, ambas transições ocorrem quando ocorre uma solicitação de escrita por parte de uma outra cache. Para que haja coerência entre as caches, o estado das demais caches (que não solicitam escrita) devem ser invalidados, visto que para serem lidos novamente precisam ser lidos da memória novamente.
- **WriteBack:** Não necessariamente é um estágio, mas sim uma ação. O *write back* consiste na escrita do valor atualizado de uma cache na memória para que as demais possam ler o valor atualizado. Tal evento sempre deve ocorrer quando um bloco sair do estágio modificado para algum outro estágio qualquer. Portanto, é possível afirmar que tal ação é uma das principais responsáveis por garantir que os valores estejam coerentes.

Uma observação importante sobre a implementação do MESI é que, no *hardware*, uma transição direta do estágio *Shared* para o *Modified* e do *Invalid* para o *Modified* não são possíveis, sendo necessária uma transição para o estado *Exclusive* antes do mesmo. Entretanto, por questões de simplificação, optou-se por representar uma transição direta entre tais estágios, afinal, tal transição não seria perceptível dada a velocidade de cada ciclo de relógio (Seria perceptível apenas se houvesse uma execução etapa por etapa das transições, o que não foi implementado).

OBS: Tal parte do projeto não foi 100% finalizada, não havendo, pois, uma identidade visual agradável, descrições com explicações sobre o algoritmo, e alguns poucos erros de lógica. Isto ocorreu porque decidimos priorizar MMU e Instruções, deixando o MESI mais como um extra, por questões de tempo.

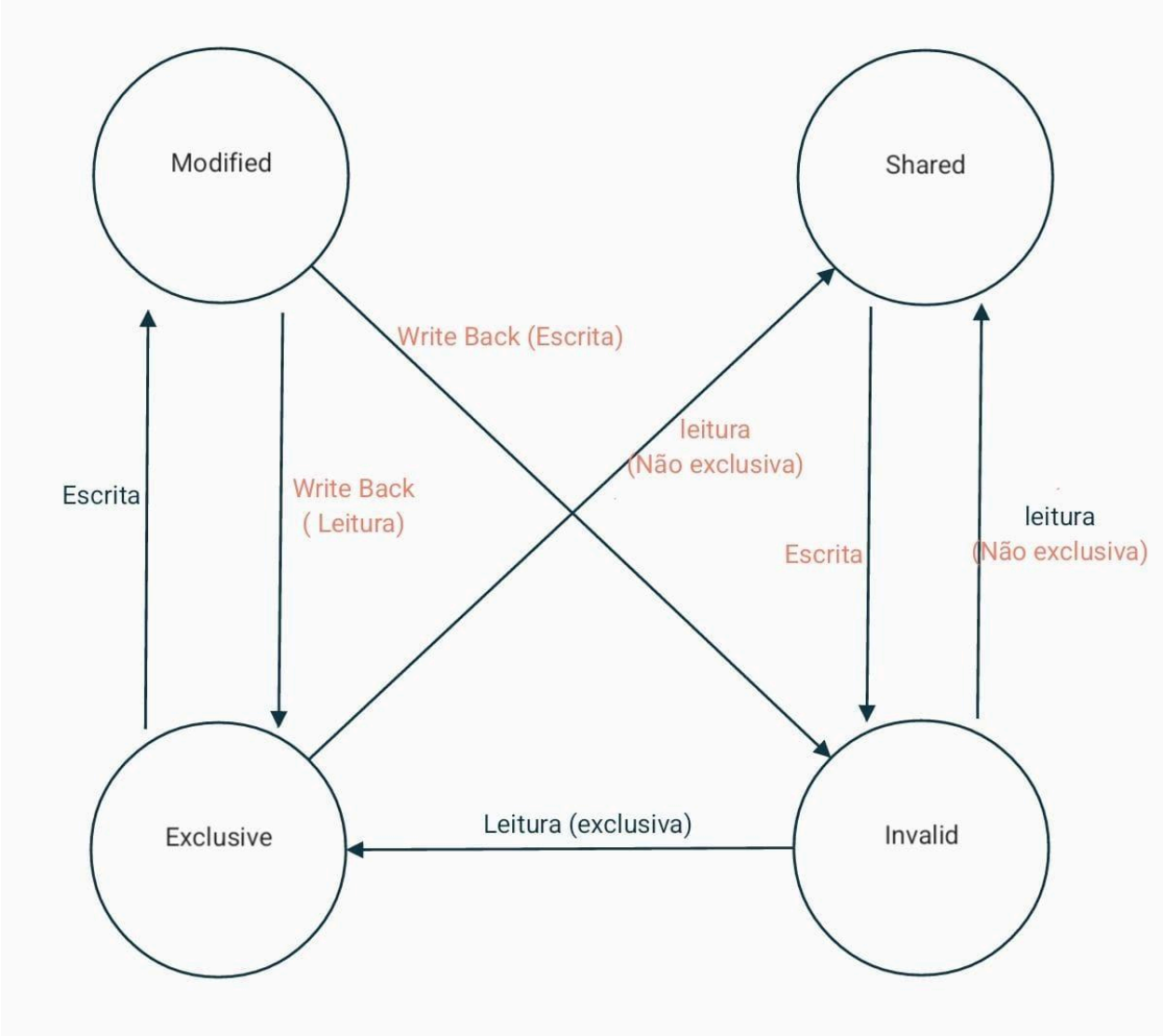

Imagem 7: Transição de Estados MESI simplificado

c. Página de MMU

A página de MMU tem objetivo de introduzir o usuário ao conceito de MMU de uma forma sintetizada e direta. Ensinando sobre o que faz esse gerenciador e como ele faz.

Ela inicia justamente com essa tomada teórica da MMU, passando pelos tópicos de:

- Visão Geral do Gerenciador de Memória (MMU)
- Organização das Tabelas de Páginas e o Papel do TLB
- Alinhamento de Páginas e Tabelas de Páginas
- Proteção de Memória Implementada pela MMU

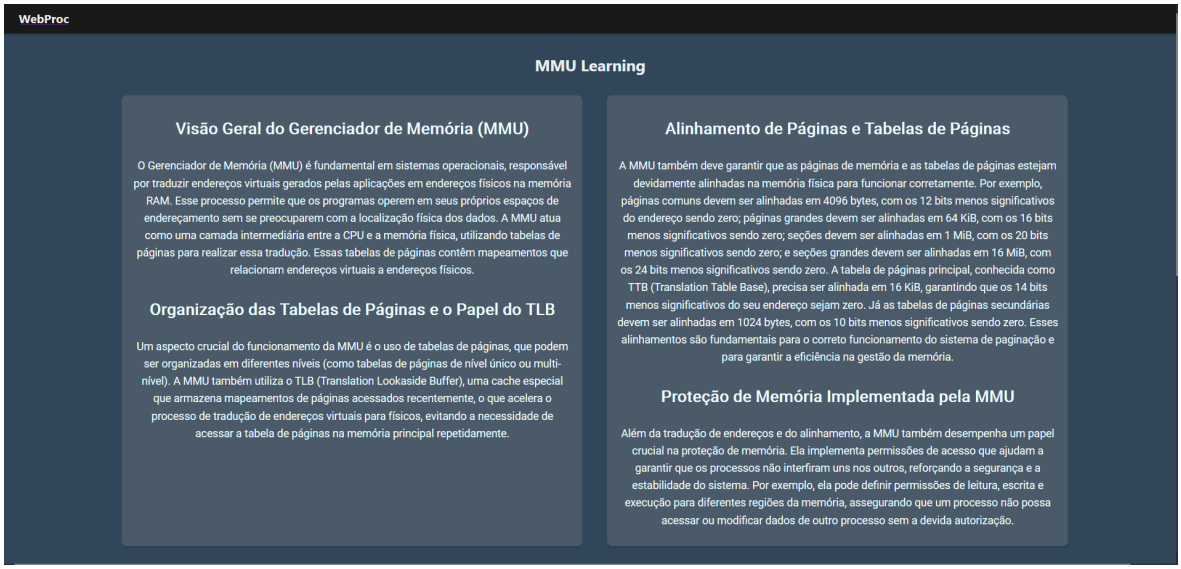

Imagem 8: Textos da página de MMU

Foi criada então uma seção interativa para o usuário, onde é possível adicionar novas páginas e quadros de memória ou pesquisar uma página para determinar seu endereço físico. O limite máximo de páginas e quadros foi estabelecido em 6, e cada quadro físico contém 400 bits. Ao criar uma nova página, o usuário pode definir o quadro de memória associado a ela. Além disso, é possível invalidar uma página simplesmente clicando sobre ela, e revalidá-la com o quadro original ao clicar novamente.

Imagem 9: Exemplo inicial do aplicação

4. Considerações Finais

Acreditamos que o WebProc é uma ótima iniciativa para melhorar ainda mais o aprendizado durante a disciplina de Laboratório de Processadores. Aliado ao mecanismo de acompanhamento semanal criado pelo fórum e à Wiki como uma fonte confiável e direcionada de conhecimento, temos convicção de que as pequenas interações do WebProc podem se tornar as conexões necessárias para tornar o caminho entre o entendimento de um conceito teórico e sua aplicação mais fluído e fácil.

Além disso, acreditamos que durante o desenvolvimento do projeto tivemos a oportunidade de observar a disciplina da perspectiva de quem está transmitindo o conhecimento, assim, conseguimos obter um entendimento mais profundo e sólido dos conteúdos que abordamos. Pensando nos principais pontos de dúvidas e entendendo as escolhas de abordagem didáticas utilizadas na Wiki, percebemos algumas lacunas próprias de conhecimento que não conhecíamos antes e construímos as interações pensando nessas falhas de teoria que normalmente ficam ocultas até termos que utilizar o conhecimento relativo a elas.