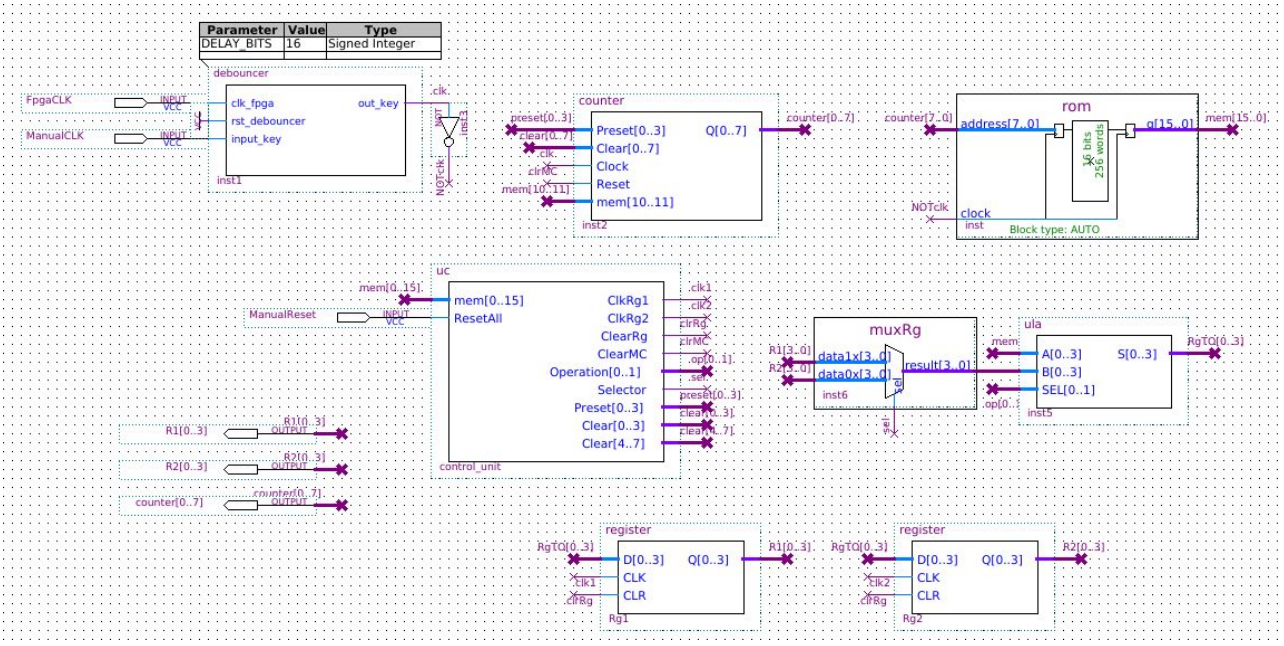
Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação
Departamento de Sistemas de Computação
SSC108 - Prática de Sistemas Digitais

Relatório Unidade Central de Processamento

Ana Luisa Teixeira Costa - 11218963
Nathan Rodrigues de Oliveira - 11218938
Otto Cruz Fernandes - 11219196

1. Circuito principal

O projeto da Unidade Central de Processamento (CPU) consiste em um circuito que interpreta e processa um arquivo de memória do tipo .mif. Nele estão integrados um contador para endereçar a memória; uma unidade de controle (UC), que realiza o processamento da memória; uma unidade lógica e aritmética (ULA); e dois registradores para armazenar os valores de saída. Os comandos admitidos pela CPU são: realizar uma operação na unidade lógica aritmética, apagar o conteúdo dos registradores, reiniciar o contador ou pular para uma determinada região da memória. A arquitetura geral do circuito pode ser observada a seguir:



a. Entradas

O circuito conta com apenas três entradas:

- **ManualClock**: funciona como o clock do circuito, processando uma instrução a cada ciclo de clock;
- **ManualReset**: reseta o contador e todos os registradores de forma assíncrona;

- **FPGA_CLK**: utiliza o clock interno da fpga para corrigir falhas físicas das chaves.

b. Saídas

Na saída, temos:

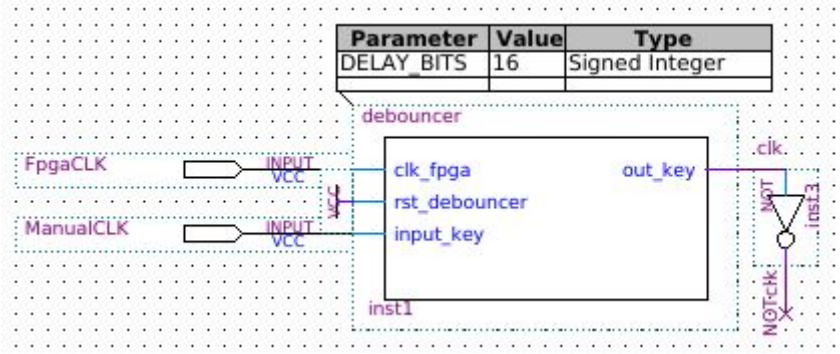
- Valor atual do contador;
- Valores armazenados nos registradores;
- Representação hexadecimal da memória.

c. Clock

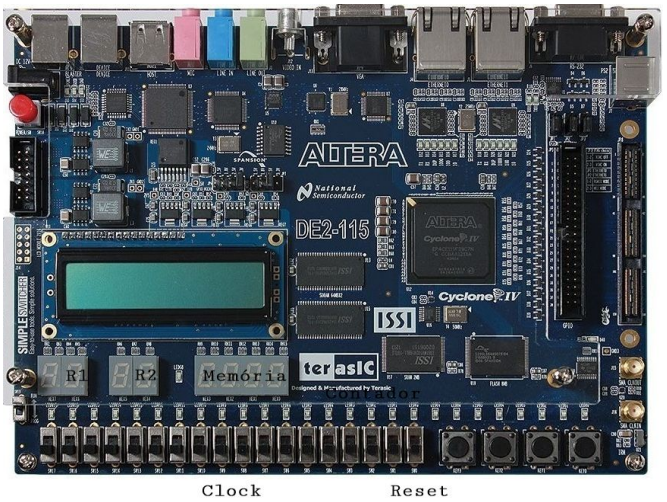
Para que o circuito funcione de forma integrada, é preciso que todos os componentes estejam em sincronia e que cada um deles tenha tempo de enviar e receber suas respectivas informações. Essa sincronia se dá através do clock. Em nosso projeto, o contador e os registradores funcionam em borda de subida e a memória, em borda de descida.

Ele foi projetado desta forma pois por padrão, todas as chaves começam em 0 e o circuito é resetado (melhor descrição desse sistema a frente) e a primeira subida de clock já faz o endereçamento da primeira posição de memória. Após isso, o valor do contador vai para a memória que tem seu output na borda de descida, que é processada pela ULA em conjunto com a unidade de controle e é armazenada na próxima subida de clock, juntamente com o próximo endereçamento de memória.

Além disso, como o pulso de clock é dado de forma manual, foi necessário o uso de um circuito para filtrar os sinais enviados pela placa já que ao mudar a chave física de 0 para um, o circuito pode interpretar um número muito alto de sinais e encerrar o programa em apenas “um clock”, esse circuito é chamado de debouncer e requer o clock interno da placa para funcionar adequadamente.



2. Configuração da Placa



A placa usada na implementação da CPU foi a Altera DE2-115. Dos pinos disponíveis, foram usadas 2 chaves físicas (SW[9]:Clock) e (SW[0]:Reset); 6 displays de 7 segmentos (HEX[6]: Registrador 1), (HEX[4]: Registrador 2) e (HEX[0]-HEX[3]: Memória); e 8 leds (LEDR[0]-LEDR[7]: Contador).

3. Arquivo de memória

A memória consiste em um arquivo tipo .mif (Memory Initialization File) de 256 linhas de 16 bits, seguindo o seguinte formato:

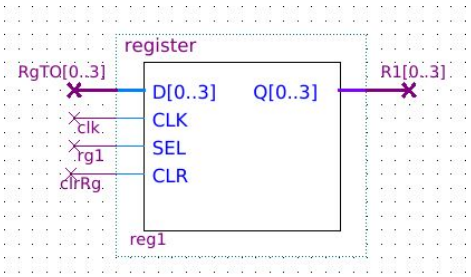
15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01
RgTo RgIn Jmp ULA Disponível Operando

- **RgTO**: Representa o registrador de destino, ou seja, o registrador que recebe a operação da ULA, sendo 01 o registrador 1 e 10 o registrador 2.
- **RgIN**:Representa o registrador de origem, ou seja, o registrador que fornecerá sua saída para a ULA, sendo 01 o registrador 1 e 10 o registrador
- **Jmp**: Indica qual operação o circuito deverá realizar
  - 00 - Operação de ULA (descrita abaixo);
  - 01 - Reinicia os registradores assincronamente;
  - 10 - Reset o contador de memória (endereça 0);
  - 11 - Jump assíncrono para a posição do operando.
- **ULA**: indica qual operação deverá ser realizada pela ULA
  - 00 - RgTO < RgIN + Operando;
  - 01 - RgTO < Operando * 2;
  - 10 - RgTO < RgIN - Operando;
  - 11 - RgTO < Operando / 2.
- **Disponível**: Bits extra para adicionar mais instruções conforme necessidade.
- **Operando**: 4 bits que funcionam como número, tanto para as operações de ULA, quanto para as operações de Jump.

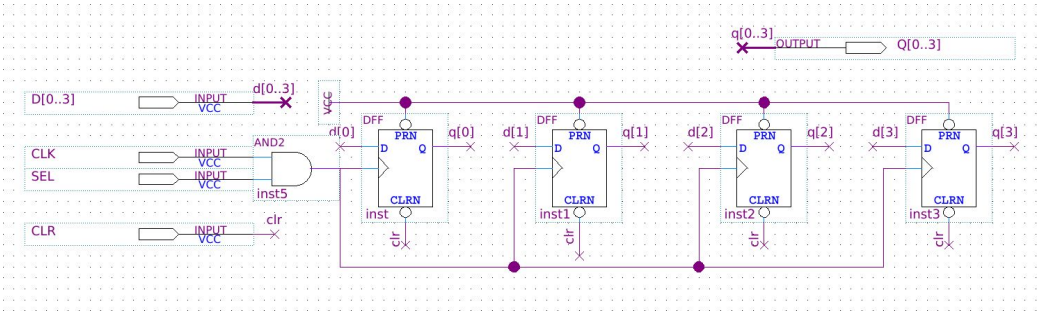
4. Componentes

a. Registradores

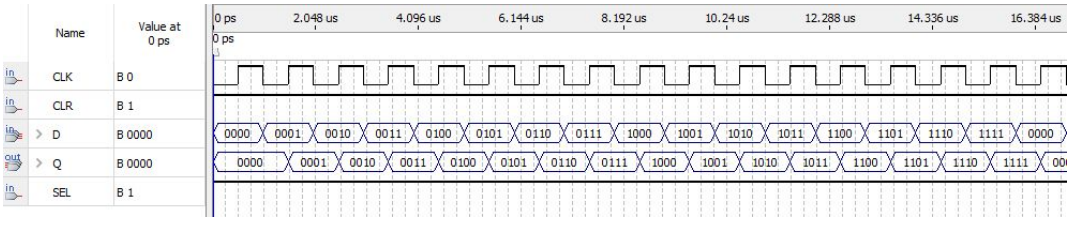
Os registradores têm a função de armazenar os resultados das operações realizadas pela ULA. Para isso, recebem como entrada um número de 4 bits a ser armazenado e, mediante sinal de subida clock, muda sua saída para tal número recebido.



Internamente os registradores são formados por flip-flops do tipo D, que fazem bit a bit a função de um registrador. Os presets e clears necessitam estar recebendo sempre 1 para manter o funcionamento do sistema. Caso um clear receba 0, ele imediatamente manda 0 para a saída do flip flop de forma assíncrona.
No circuito, os registradores estão adaptados para apenas receber clock mediante um sinal de autorização enviado pela unidade de controle através de uma porta AND entre tal sinal e o clock geral do circuito.

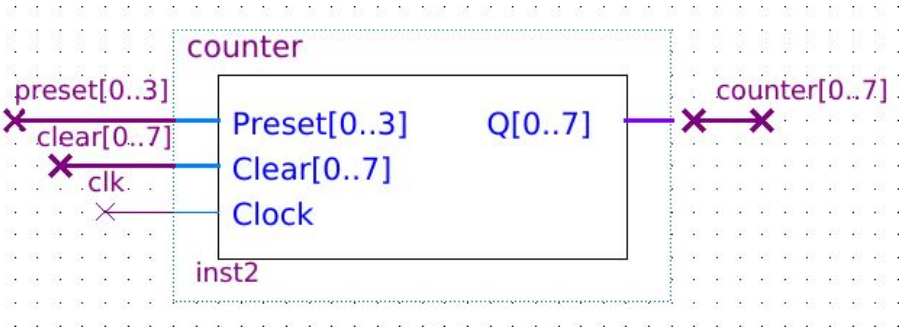


A imagem abaixo mostra uma simulação do funcionamento do registrador. Geramos uma entrada que é incrementada a cada borda de descida de clock, e como podemos observar, na borda de subida, o valor da saída do registrador é atualizado.

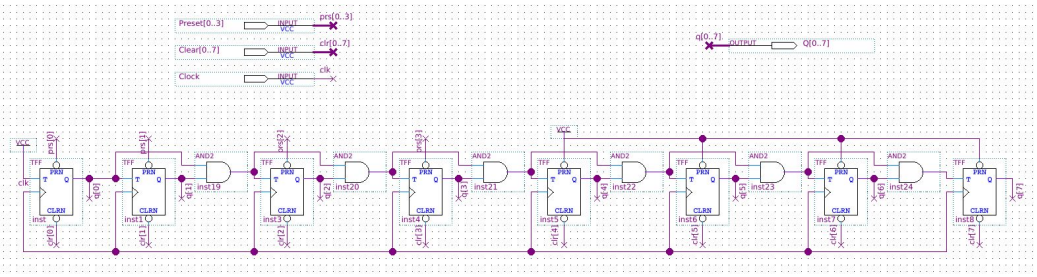


b. Contador

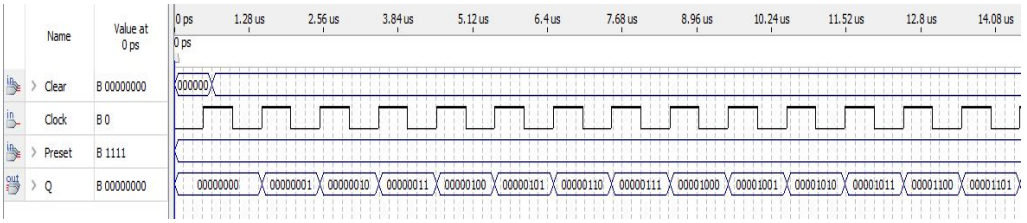
O contador tem como objetivo fornecer os endereços de memória para a leitura do programa. Para isso, seu circuito incrementa o valor de saída em 1 mediante sinal de subida de clock.



Internamente o contador usa flip-flops do tipo T, que invertem o sinal de saída mediante sinal de subida de clock caso sua entrada seja 1. No contador, as saídas são alinhadas para que o sinal de entrada seja adiado em 2^k para cada flip flop em fila. Além disso, existem entradas de preset e clear assíncronos para o uso em operações de Jump de Reset que são reguladas pela unidade de controle.

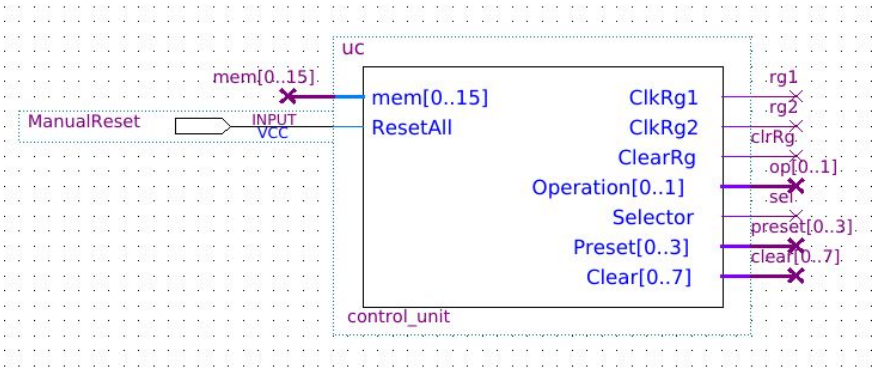


A imagem abaixo mostra uma simulação do funcionamento do contador. A cada pulso de clock, o valor da saída do contador é incrementado.



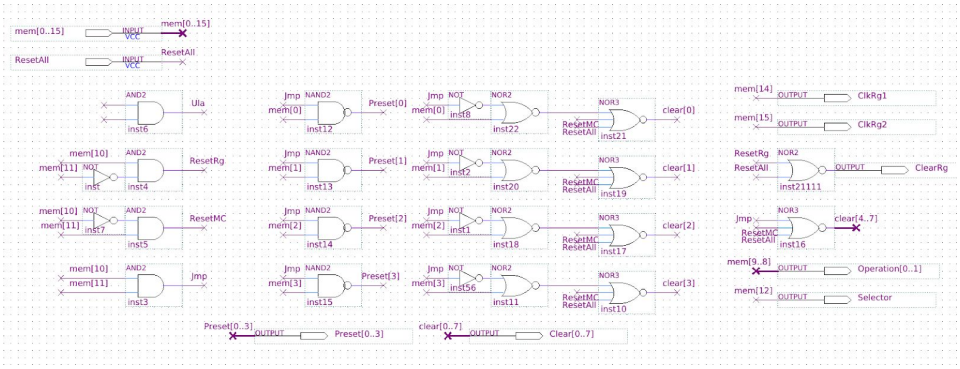
c. Unidade de Controle

A unidade de controle é responsável por interpretar cada bit de entrada do arquivo de memória e através de uma lógica de seleção, regular as saídas que farão todo o endereçamento de dados pela CPU. para uma explicação mais clara, sua execução será dividida em tópicos a seguir.



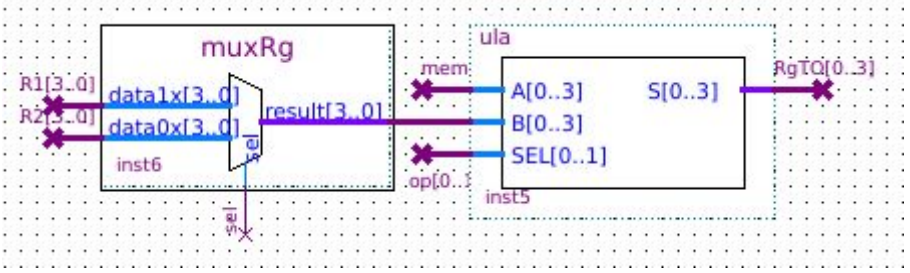
- Os bits 10 e 11 da memória são analisados atráves de portas AND, que apresentam 1 como saída para a operação a ser realizada e 0 para o restante.
- **ClkRg1 e 2**: Os clocks dos registradores são baseados nos bits 14 e 15 (RgTo) que definem se cada registrador deve ou não receber clock.
- **ClearRg**: Os registradores devem ser resetados se receberem o comando de ResetAll (Chave assíncrona de clear) ou de ResetRG (operação “01” dos bits de JMP). Como o clear desses componentes funcionam em sinal baixo, utilizamos a porta NOR;
- **Operation[0..1]**: Como os bits [9] e [8] indicam qual operação a ULA deve realizar, essa entrada é repassada para o circuito como selecionadores na ULA;
- **Selector:** Selector é o bit que entrará como o seletor de um multiplex que por sua vez escolhe qual registrador irá enviar sua saída para a ULA.
- **Preset[0..7]**: Quando há necessidade de realizar o Jump, se um bit de endereço for 1, é preciso ativar o preset (ativo sinal baixo) do flip-flop correspondente, e para isso utilizamos uma porta NAND;
- **Clear[0..7]**: O Clear no contador de memória possui 3 condições: Ou o ResetAll (Chave assíncrona de clear) está ativa, ou a operação de ResetMC (operação “10” dos bits de

JMP) está ativa e, se a operação de JMP for “11”, o clear deve ser ativo nos bits do operando que são 0, logo not JMP NOR mem[0..3].



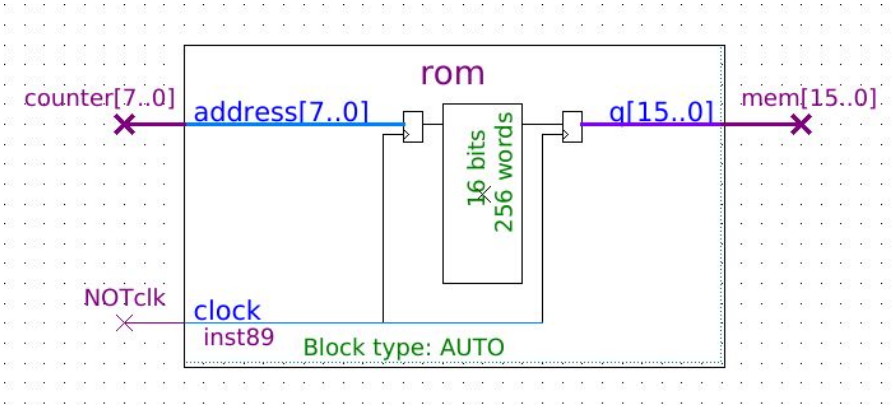
d. Unidade Lógica Aritmética

A Unidade Lógica Aritmética (ULA) é responsável por fazer as operações matemáticas através de lógica combinacional de circuitos. Ela recebe 2 números A e B e 2 bits para selecionar a operação (+,-,*2 ou /2) e gera um resultado também de 4 bits para ser armazenado em um dos registradores.



e. ROM

A unidade de memória é responsável por armazenar o arquivo que contém as instruções a serem processadas pelo circuito. Ela foi criada a partir de uma ferramenta do software usado (MegaWizard ROM: 1-PORT), que carrega um arquivo .mif e retorna um bloco que: dado um endereço de memória de 8 bits e um sinal de clock, devolve como saída os 16 bits programados no arquivo .mif. Para integrar a ROM ao circuito, como seus dados de endereço vêm diretamente do contador, ela deve estar em uma borda oposta ao contador para evitar uma leitura atrasada de dados, logo, ela se encontra em borda de descida.



5. Conclusão

Após passar 2 meses construindo esse projeto, fica evidente para nós como é complexo o funcionamento de uma CPU. As matérias de Sistemas Digitais e Prática em Sistemas Digitais nos mostram como a visão de um usuário do computador é limitada, e a falta de uma visão desde o mais baixo nível pode impedir um programador de utilizar o melhor desempenho da máquina que opera.
Várias dificuldades foram encontradas ao longo do trabalho: o complicado funcionamento do Quartus 2, fios e barramentos wi-fi que recebiam os nomes ou ordens errados e geravam sequências invertidas, que aparentemente eram erradas e aleatórias, debugar o projeto, com as dezenas de portas lógicas que tivemos que usar, entre outros. Mesmo assim, com bastante esforço e com a ajuda do professor e do monitor, os problemas foram ficando mais claros e foram sendo resolvidos, até resultar no projeto da CPU pronto e funcionando.

6. Links

O projeto pode ser acessado pelo link: https://github.com/anacst/CPU.
Para maior resolução das imagens presentes no documento: https://github.com/anacst/CPU/tree/master/Relat%C3%B3rio.