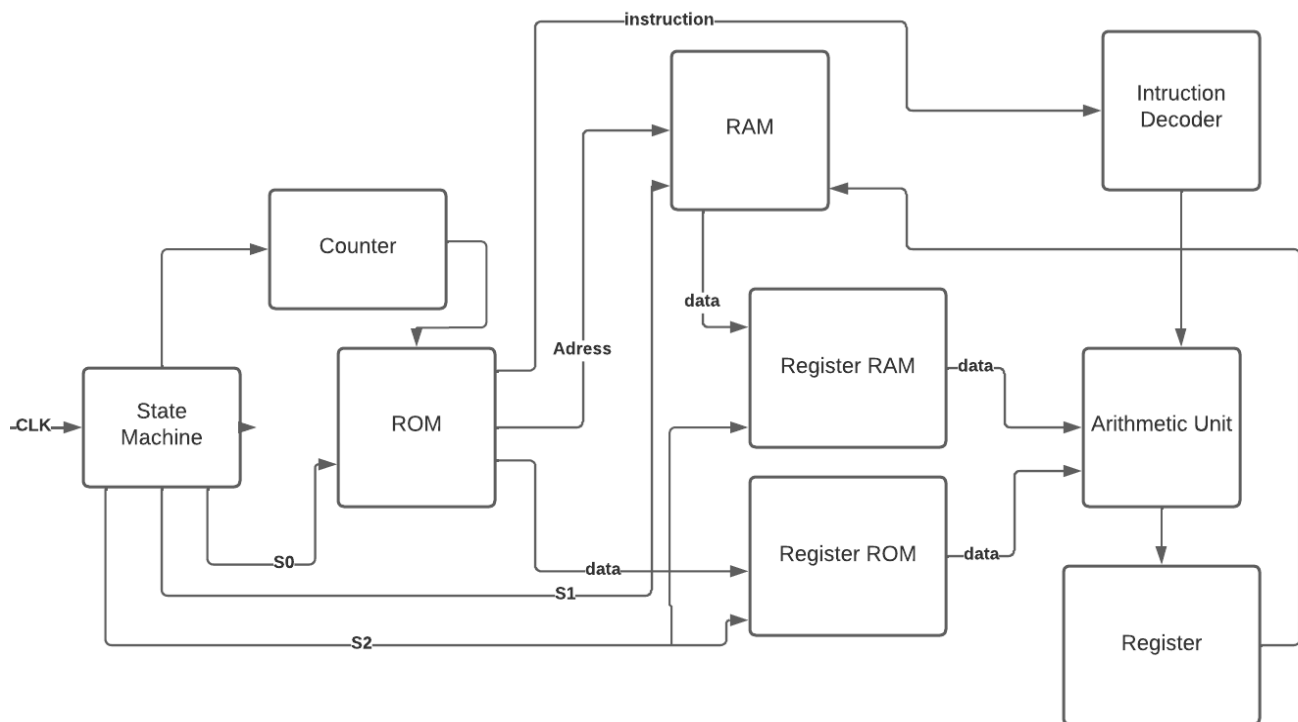


INTRODUÇÃO

O projeto desenvolvido teve como objetivo simular um modelo simplificado de um uProcessador. Foram implementados algumas instruções que o mesmo poderia realizar, sendo elas : carregar um valor na memória, somar, subtrair e algumas operações lógicas(AND, XOR e OR). Para que isso fosse possível, foram implementados diferentes componentes para gerenciar estados e realizar operações. Sendo assim, o esquemático abaixo descreve a comunicação do componentes



INSTRUÇÕES

As instruções anteriormente mencionadas, são armazenadas na memória ROM e tem a seguinte estrutura: 4 bits para o tipo de instrução, 4 bits para o endereçamento, e 8 bits para o operando. Para a instrução “B358”, por exemplo, é instruído que realize o carregamento(B) no endereço de memória 3 o valor “58”. A figura abaixo mostra o arquivo hex que armazena as instruções na ROM.

```

≡ ROM.hex
1  :10000000B358B22BB11FB025502353015208A1CCD5
2  :10008000A0CCC2FFC100C3F17001711103FF02FF50
3  :00000001FF
    
```

A correspondência de cada instrução para o número binário é descrito abaixo:

Instrução	Valor(binário)	Valor(Hexadecima l)
LOAD	1011	B
ADDER	0101	5
SUBTRACT	0111	7
AND	1100	C
XOR	1010	A
OR	0000	0

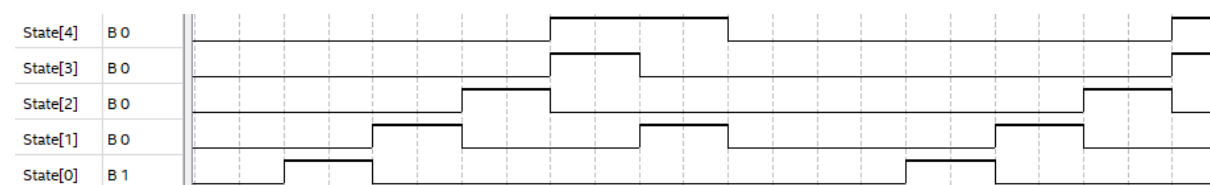
FUNCIONAMENTO GERAL

Como dito antes, as instruções estão armazenadas na ROM, logo é necessário haver um contador para executar cada uma delas, sendo assim, o projeto conta com um contador, além de possuir uma máquina de estados para executar cada ciclo no tempo correto.

O uProcessador em questão funciona em ciclos, descritos a seguir: carrega a instrução proveniente da ROM, tendo como saída a operação, endereço e valor, logo em seguida é realizado o carregamento do valor que encontra-se na RAM para o endereço da instrução, com esses dois operandos é realizado o salvamento em um registrador PIPO, com estes dois dados realiza a operação na unidade aritmética e salva em outro registrador PIPO, e, por fim, salva o resultado na RAM.

MÁQUINA DE ESTADOS

A máquina de estados controla o processo que vai estar sendo executado em cada ciclo, tendo o seguinte funcionamento:



State(0) - Clock na ROM;

State(1) - Clock na RAM;

State(2) - Clock nos Registradores PIPO que salvam os dados de saída da ROM e RAM;

State(3) - Clock nos Registradores PIPO que salvam os dados de saída da Unidade Aritmética;








State(4) - Habilita a escrita na RAM;

REGISTRADORES

Os registradores implementados foram do tipo PIPO(paralelo/paralelo) de 8 bits, foi necessário seu uso pelo fato de requerer o armazenamento de um dado até alguns ciclos fossem realizados, o mesmo armazena o valor de entrada em um flip-flop d quando realizado um clock.

UNIDADE ARITMÉTICA

Já na Unidade Aritmética é responsável por poder realizar todas as operações, inicialmente são realizadas todas as operações possíveis, em seguida é detectado o tipo e com um multiplexador é escolhido o resultado, por fim, salva o resultado no registrador.

▼	 ArithmeticUnit:UnitArithmetic1
	 ANDoperation:and1
	 FullAdder8bits:fulladder1
	 FullAdder8bits:fulladder2
	 ORoperation:or1
	 AccumulatorFFD:register1
	 XORoperation:xor1

CONCLUSÃO

Neste presente trabalho foram utilizados todos os conceitos vistos nesta disciplina, desde lógica booleana até máquinas de estados. Desta forma foi possível criar um modelo simplificado de um uprocessador, com 6 operações diferentes. Além disso, este trabalho tem uma sinergia com disciplinas posteriores como Arquitetura e Organização de Computadores.