INSTITUTO FEDERAL DO RIO GRANDE DO NORTE
DIRETORIA ACADÊMICA DE GESTÃO E TECNOLOGIA DA INFORMAÇÃO
PROGRAMA DE GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

**SAULO DANIEL FERREIRA PONTES**
**WANNDERSON EDUARDO DE OLIVEIRA NOGUEIRA**
**DANRLEY HENRIQUE TENÓRIO SILVESTRE**

# Definindo a Arquitetura de um Computador Simulado

Natal
16/11/2017

## Resumo

Este documento trata da abordagem e definição de requisitos e planejamento de um simulador de computador para posterior construção na linguagem de programação Java acordada pelo grupo que vos escreve.

palavras chave: Java, computador, simulador.

## Abstract

These document deals with the approach and definition of requirements and planning of a computer simulator for later construction in a Java programming language agreed by the group that writes to you.

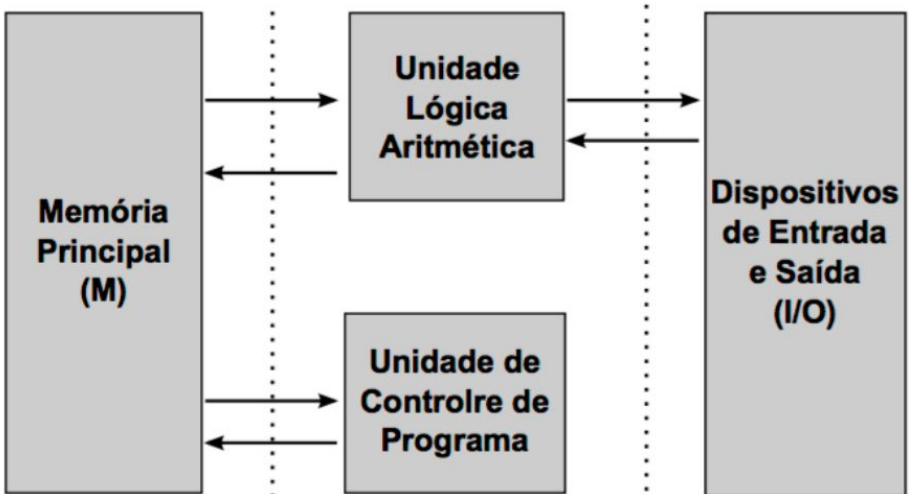
keywords: Java, computer, simulator.

## Introdução

Um computador e uma máquina capaz capaz de fazer a leitura de dados em uma linguagem padronizada de um determinado meio e interpretá-los para gerar diferentes tipos de tratamentos automáticos de informações. Atualmente um computador pode possuir inúmeros atributos, que derivam das funções baseadas em cálculos em grande escala, controle, armazenamento dados, transferência de dados e processamento de dados. Eles são constituídos de diversos componentes a Unidade central de processamento (CPU), os barramentos, as memórias principais, os discos de armazenamento de dados e os periféricos. Neste relatório estão descritos os requisitos necessários para se implementar um simulador de um computador de arquitetura CISC **(**computador com um conjunto de instruções mais complexas) possibilitado ao se utilizar a linguagem de programação Java.

## Metodologia

Atualmente a arquitetura que é utilizada como base para a construção dos computadores e a Arquitetura de Von Neumann, onde temos uma unidade lógico aritmética, uma memória principal, uma unidade de controle de programa e dispositivos de entrada e saída.




*Arquitetura de Von Neumann*

A partir desta arquitetura o atributos visíveis para o programador são:

– Conjunto de instruções, que a unidade lógico aritmética consegue executar;

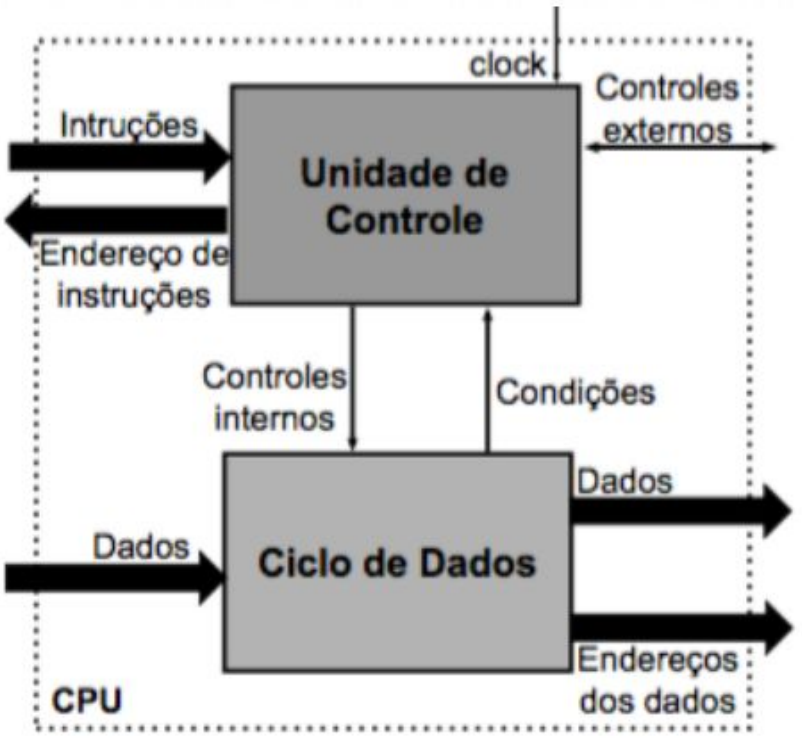
– Número de bits utilizados para representação de dados;

– Técnicas de endereçamento de memória;

– Mecanismos de entrada e saída.

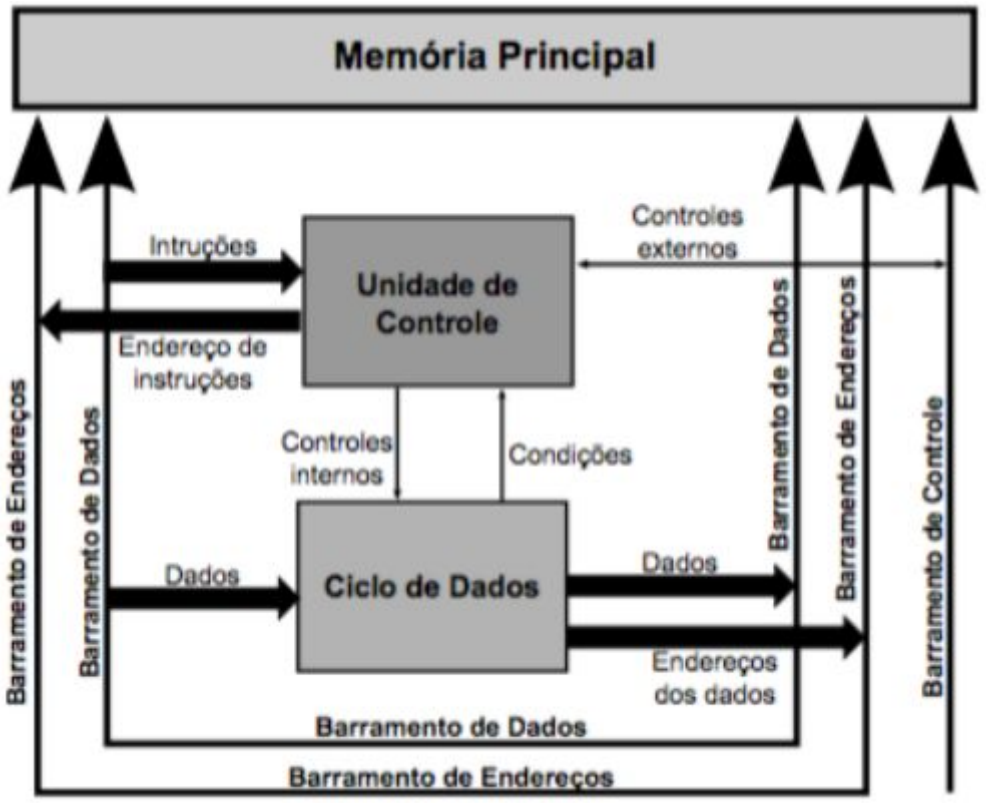
A unidade de controle de programa e responsável pelo controle de execução e sequenciamento das instruções. Ela trabalha com execução de microoperações, onde em cada estágio do Ciclo de Instrução é quebrado em sub estágios. A micro operação é a operação atômica realizada pela UC para a execução de cada estágio do programa.

Uma CPU é constituída de uma unidade de controle e uma unidade de ciclo de dados. Onde seu funcionamento e determinando a partir de um sinal de relógio (clock). A cada transição feita pelo relógio a unidade de cadência a execução de um passo, passa os dados para quem deve, e se prepara para o próximo passo. Quanto mais rápido o sinal de clock mais operações por segundo é possível realizar.



*Unidade de Controle e Unidade de Ciclo de Dados*

A interface de comunicação entre a CPU e memória principal são os Barramentos, neles trafegam todos os dados que vem de um bloco externo ou que são transmitidos após o processamento.




CPU, UC, CD e memória principal conectadas ao barramento de dados

Para que uma operação seja executada na CPU deve haver uma busca de dados que podem ser executada através da busca de Instrução, onde uma nova instrução deve ser buscada da memória e trazida para a CPU para que possa ser decodificada e executada em seguida, ou através da busca Indireta, onde é feita a busca de um dado que está na memória e traz para a CPU para ser utilizado pela instrução.

A CPU é capaz de executar consecutivamente uma série de instruções de um

programa armazenado em uma estrutura de memória. Após compilado, o programa de linguagem de alto nível é transformado em um programa apenas com instruções

de máquina. Onde cada instrução de máquina contém apenas uma instrução a ser executada pela CPU. Para que o programa execute ele deve ser movido para a memória principal.

Tipos de Micro Operações

- Transferência de dados entre registradores
- Transferência de dados de registrador para o exterior da CPU
- Transferência de dados do exterior da CPU para um registrador
- Operação lógica e aritmética

A CPU possui um conjunto de registradores padrões que servem para armazenar dados temporários, informações de manipulação e estados dos processos, eles são:

- PC: Contador de Programa (Program Counter);
- IR: Registrador de Instrução (Instruction Register);
- MAR: Registrador de Endereço (Memory Address Register);
- MBR: Registrador de Dados ( Memory Buffer Register).

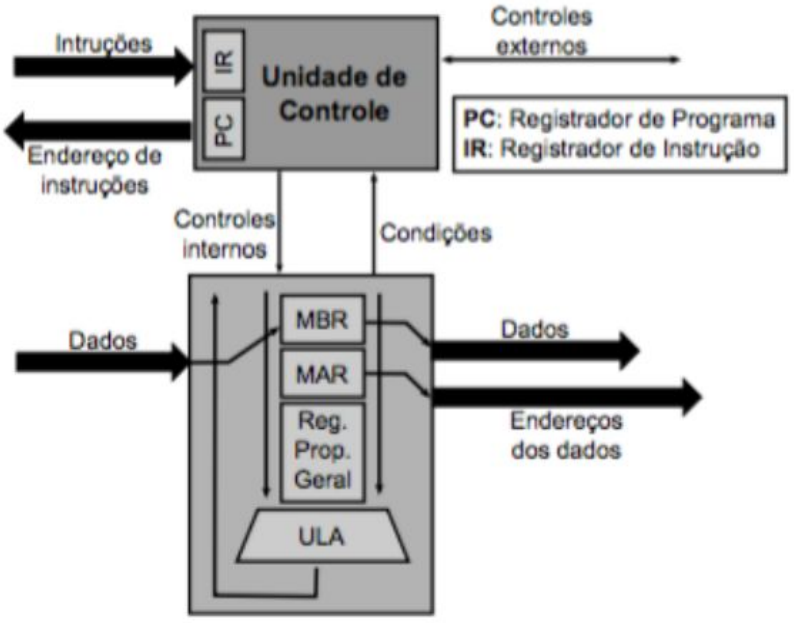



*Diagrama organizacional dos registradores da CPU*

O Contador de Programa, contém o endereço de memória que será utilizado para buscar a próxima instrução a ser executada. A CPU envia o comando para PC e para a memória, a memória envia o conteúdo armazenado para o IR. O Registrador de Instrução, armazena a instrução recebida da memória, Encaminha instrução para a unidade de controle. MAR e MBR possuem funções análogas ao PC e IR mas referentes a dados e não instruções . O endereço do dado é passado para o MAR quando o dado está na memória e não em um registrador . O conteúdo da memória referente a esse endereço é armazenado em MBR . Também pode-se ter registradores de Propósito Geral, para guardar variáveis de programas. A Unidade Lógica Aritmética consegue identificar – soma, multiplicação, divisão, AND, OR, entre outras. Podem operar com números inteiros ou reais. A UC indica os dois registradores que fornecerão os dados de entrada A saída da ULA é armazenada em um registrador indicado pela UC . A unidade de Controle, recebe a instrução armazenada em IR, decodificar e envia os sinais de controle. Sinais de controle podem ser de dois tipos:

- Internos: ULA executar uma subtração, ou para o conteúdo de um registrador ser; transferido para a ULA...
- Externos: para um dispositivo de entrada e saída ou memória principal;

A execução das Microoperações é sempre ordenada pela Unidade de Controle. Isso é feito no estágio de Decodificação, a partir da leitura da instrução presente em IR. Ela pode ter vários estágios de execução. Um exemplo mais simples de ser observado na estrutura de um processador que possui 5 estágios:

– Busca de instrução
– Decodificação de instrução
– Busca de dados
– Execução da instrução
– Salvar resultado

Na Busca da Instrução é necessário que o registrador PC contenha o endereço inicial do bloco de memória de programa(geralmente 0x00) para que a primeira instrução seja buscada e executada.

Na Decodificação a instrução que acabou de ser copiada para IR é analisada e a Unidade de Controle vai então preparar as Microoperações serão executadas nas etapas seguintes.

Na Busca de Dados é feita a busca dos dados necessários para a execução da instrução.

Na Execução a instrução e processada e a manipulação das variáveis é feita.

Ao Salvar Resultados eles são salvos na variável em um Bloco de memória ou registrador.

O Programa em Linguagem de Máquina é composto por instruções em binário, a cada instrução trazida da memória, a CPU lê seu código binário de operação para interpretando o comando a ele vinculado, posteriormente e iniciado o processo de execução. Dependendo da operação a CPU envia ordens para que outros dispositivos atuarem a fim de completar a operação e repassa os resultados para uma nova região de memória.

**Arquitetura do Simulador**

A arquitetura escolhida para implantação do computador simulado, foi a arquitetura baseada em Pilha, onde os dados necessários para a execução das operações pela ULA são provenientes de registradores especiais organizados na forma de uma pilha.

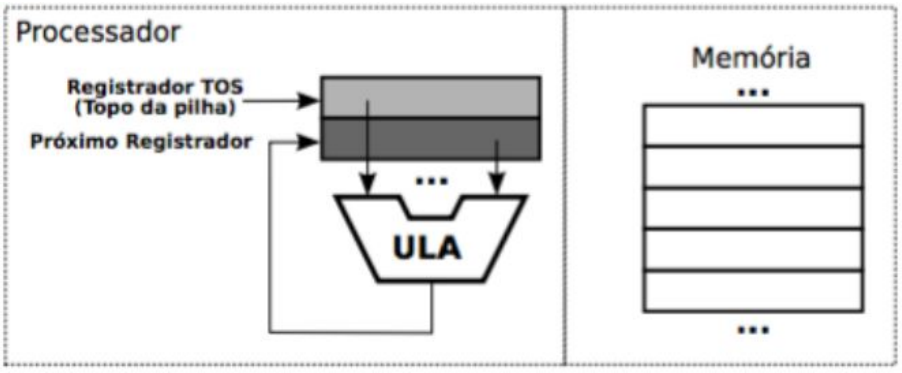



*Diagrama de uma Unidade de Controle com arquitetura em pilha*

A escolha da arquitetura é uma questão importantíssima de projeto e geralmente baseia-se na relação entre o desempenho que se quer atingir e o preço do processador ao final, mas para o propósito deste trabalho a arquitetura baseada em Pilha se mostrou bem interessante pelo tipo de gerenciamento das informações nos registradores e na memória principal do computador.

O projeto do conjunto de instruções possuem cinco pontos a serem contemplados:

– Modelo de Memória;

– Tipos de Dados;

– Formato das Instruções;

– Tipos de Instruções – Modo de Endereçamento;

O Modelo de Memória define para cada instrução de onde vem e para onde vão os dados. Na arquitetura do computador simulado será utilizada a estrutura de memória Alinhada, que é um método de organização de dados em memória para que ele possa ser recuperado o mais rápido possível. O alinhamento correto é composto por dois processos: alinhamento e preenchimento.

O Processador ler e gravar informações em grupos de bytes chamados de "palavras". Os dados correctamente alinhados começam em uma localização na memória, chamado de "endereço", segundo um múltiplo do tamanho da palavra. Por exemplo, o tamanho da palavra de um computador de 32 bits é de 4 bytes. Assim, uma palavra que começa no endereço 8 seriam alinhados, enquanto a memória no endereço 5 não seria .

Os Dados na memória muitas vezes terminam em um ponto que não é um múltiplo do tamanho de palavra. Se isso acontecer e um processador escreve para o próximo endereço disponível, assim os dados ficarão desalinhados. Para evitar isso, o processador deve escrever um número de bytes sem sentido até que chega a um endereço que é um múltiplo do tamanho da palavra.

Os Benefícios dos dados Alinhados permite que o processador possa realizar o menor número de tarefas possíveis durante o acesso e processamento os dados. Por exemplo, usando o tamanho da palavra de 4 bytes, se uma palavra ficar no

endereço 3, o processador deve ler endereços de 0 a 3 para obter os dados em 3, em seguida, ler os endereços 4-7 para obter os dados de 4, 5, e 6 . Se os dados foram alinhados no endereço 4 , uma operação de leitura seria suficiente.

A memória principal conterá 2 Megas de armazenamento divididos em 4 Bytes para as instruções e variáveis onde preferencialmente o endereço do início do programa será armazenado no primeiro byte da array. O programador deverá gerenciar o espaço de memória que pretende utilizar e preferencialmente criar blocos de variáveis antes do ponto de execução inicial do programa.

A ordem dos bytes seguirá o formato padrão das máquinas virtuas Java que intercalam automaticamente entre o Big Endian e o Little Endian, dependendo da arquitetura e, que estão sendo executadas.

Além da memória principal serão adicionados cinco registradores, em que dois deles o P0 e P1 corresponde a os registradores da pilha e três deles o A, B e C são para uso geral. Assim como na memória principal possuirão 4 bytes de armazenamento para cada um.

Os registradores da unidade de controle PC, IR, MAR, MBR também serão emulados.

Os tipos de dados aceitos são números flutuantes de 32bits IEEE 754-2008, ($4.38\times 10^{-16}$ a $4.38\times 10^{16}$) .

Os comandos do assembly serão executados passo a passo e em tempo de execução a partir de um arquivo .txt contendo os comandos em assembly descritos na seção seguinte.

## Resumo da arquitetura do Simulador

Tamanho das memórias:

- 25600 Bites -> 3200 Bystes -> 0,0032 Megabytes - divididos em blocos de 1 Byte que corresponde a 100 posições de memória para inserção de códigos e variáveis.

Modelo de memória:

- Memória Alinhada de 100 posições.

Formas de endereçamento:

- Imediato e Direto.

Projeto de arquitetura:

- Arquitetura de Pilha utilizando dois registradores.

Quantidade de registradores:

- **P0**, **P1** - 4 Bytes (32 bits) (registradores da pilha).
- **A**, **B**, **C** - 4 Bytes (32 bits) (registradores de uso geral).

Registradores da Unidade de Controle:

- **PC** : Contador de Programa (Program Counter).
- **IR** : Registrador de Instrução (Instruction Register).
- **MAR** : Registrador de Endereço (Memory Address Register).
- **MBR** : Registrador de Dados ( Memory Buffer Register).

Tipos de dados:

- **float** - blocos de 4 Bytes (32 bits)

Formato de instruções:

- As instruções atuam na manipulação dos dados da memória principal, na pilha e nos registradores auxiliares. Algumas possuem acesso restrito a algumas estruturas e servem como marcadores de início e fim de uma programa.

Tipos de instruções:

- Caracteres espiral:
  - **#** : sinaliza o acesso a um endereço de memória.
    Exemplo:
    **#1** -> endereço 1 da memória.

- Marcadores de programa:
  - **INIT** : marca o início do programa da posição de memória onde os comandos estão armazenados. (deve ser sempre utilizado apenas uma vez na posição zero da memória).
    Exemplo:
    **INIT #1** -> inicia a execução do código no endereço 1 da memória.
  - **END** : finaliza a execução do programa.

- Movimentação de dados:
  - **PUSH** : adicionar dado na pilha.
    Exemplos:
    **PUSH #1** -> Desloca dado de P0 para P1, registrador P0 recebe o valor contido na posição de memória 1.
    **PUSH 12** -> Desloca dado de P0 para P1, registrador P0 recebe o valor 12.

- **POP** : copia dado da pilha para um registrador ou região de memória.
  Exemplos:
    **POP #1** -> posição de memória 1 recebe o valor contido em P0.
    **POP A** -> registrador A recebe o valor contido em P0.
- **ZERO** : adiciona zero em um registrador ou região de memória.
  Exemplos:
    **ZERO #1** -> posição de memória 1 recebe zero.
    **ZERO A** -> registrador A recebe o valor contido em P0.
- **MOV** : move dado de um registrador ou região de memória para outro registrador ou região e adiciona zero a posição original.
  Exemplos:
    **MOV #1 #2** -> move o conteúdo da posição e memória 1 para a posição e memória 2. Substitui o conteúdo da posição de memória 1 para zero.
    **MOV A #1** -> move o conteúdo do registrador A para a posição e memória 2. Substitui o conteúdo do registrador A para zero.
- **CP** : copia um dado para um registrador ou um local da memória.
  Exemplos:
    **CP #1 #2** -> move o conteúdo da posição e memória 1 para a posição e memória 2.
    **CP A #1** -> move o conteúdo do registrador A para a posição e memória 2.

- Salto:
  - **EQUAL** : se igual salte para alguma região de memória.
    Exemplos:
      **EQUAL A 3 #1** -> se A é igual a 3 pula para a posição de memória 1.
  - **NEQUAL** : se igual salte para alguma região de memória.
      **NEQUAL A 3 #1** -> se A é diferente de 3 pula para a posição de memória 1.

- **JUMP** : pula para uma posição de memória.

  Exemplos:

  **JUMP #1** -> pula para a posição de memória 1.

- Aritmética (atuam exclusivamente nos valores P0 e P1 da pilha):
  - **ADD** : soma ( P0 = P1 + P0 ).
  - **SUB** : subtração ( P0 = P1 - P0 ).
  - **MULT** : multiplicação ( P0 = P1 * P0 ).
  - **DIV** : divisão ( P0 = P1 / P0 ).
  - **EXP** : exponencial ( P0 = exp(P0) ) .
  - **MOD** : resto de divisão ( P0 = P1 % P0 ).

## Considerações Finais

A partir deste documento será criado o simulador e posteriormente poderá ser aperfeiçoado para utilização didática caso seu desempenho seja satisfatório.