INATEL – INSTITUTO NACIONAL DE TELECOMUNICAÇÕES
C208 – LABORATÓRIO DE ARQUITETURA DE COMPUTADORES

ALEXÂNDER AUGUSTO SILVA FERNANDES
VANESSA SWERTS ESTEVES

MÁQUINA VIRTUAL AS_MIC

SANTA RITA DO SAPUCAÍ
2019

ALEXÂNDER AUGUSTO SILVA FERNANDES
VANESSA SWERTS ESTEVES

MÁQUINA VIRTUAL AS_MIC

Projeto de Arquitetura de Computadores apresentado ao Curso de Engenharia da Computação do Inatel como requisito parcial para conclusão da disciplina.

SANTA RITA DO SAPUCAÍ
2019

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A máquina virtual AS_MIC foi desenvolvida como proposta para o projeto de Arquitetura Computacional. Esse projeto tem como principal objetivo realizar e explorar os temas até então estudados ao longo do curso.

Assim, a arquitetura virtual é composta por 32 bits, 12 registradores, sendo 8 registradores de uso geral, 3 temporários e 1 contador de programa (PC), 2 memórias divididas em memória principal e memória cache e 4 instruções, sendo 2 aritméticas e 2 lógicas. Para o desenvolvimento do projeto foi utilizado a linguagem de programação JAVA e a *IDE NetBeans*.

A partir disso, cada elemento exerce uma determinada função. Logo, os registradores são responsáveis por armazenar os dados a serem processados pela arquitetura. As memórias são encarregadas de salvar as instruções, que por sua vez, são constituídas por 32 bits representando a operação a ser decodificada e executada.

2 DESENVOLVIMENTO

Para o melhor entendimento sobre o desenvolvimento do projeto, foi necessário dividi-lo em duas partes principais sendo elas:

- Descrição dos principais componentes;
- Descrição de funcionamento.

2.1 Descrição dos principais componentes

A descrição dos principais componentes foi subdividida nas categorias:

- Registradores;
- Conjunto de Instruções;
-Memória Cache virtual.

2.1.1 Registradores

Os registradores são a memória do processador, usada em operações para a execução de instruções. Existem vários tipos de registradores, cada um com uma função específica e cada grupo com sua finalidade, desta forma a arquitetura AS-MIC é constituída por 12 registradores, sendo estes divididos em 8 de uso geral, 3 temporários e 1 contador de programa (PC).

O PC é um registrador de uso específico, no qual fica armazenado o endereço da memória onde está a próxima instrução a ser executada. Os registradores de uso geral e os temporários são usados para armazenar dados da execução do programa, como os valores que serão operados e o resultado das operações, no entanto estes registradores diferem-se ao fato de que os de uso geral salvam os dados de maneira definitiva, enquanto os temporários armazenam os dados apenas durante a execução da instrução. Estes registradores são determinados com 5bits e sua nomenclatura é representada a seguir na Tabela 1.

Tabela 1 – Tabela de Registradores

Nome	Representação em bits
AS0	00000
AS1	00001
AS2	00010
AS3	00011

continua

Nome	Representação em bits
AS4	00100
AS5	00101
AS6	00110
AS7	00111
temp0	01000
temp1	01001
temp2	01010

Fonte: Equipe

2.1.2 Conjunto de instruções

Instruções são tarefas elementares que podem ser executadas em um processador, como por exemplo: operações aritméticas e operações lógicas. Cada instrução é representada em 32bits por um código de operação. A arquitetura AS-MIC é constituída por 2 tipos de instruções: **Reg** e **Const**.

As instruções do tipo **Reg** são caracterizadas por operações entre dois registradores e contém 5 campos, sendo eles: o **OP** que representa o *op-code* que indica ao processador qual o tipo da instrução que ele irá executar. O **dest** que caracteriza o registrador responsável por armazenar o resultado da operação entre os registradores indicados por **var1** e **var2.** E o **func** que indica o tipo de operação a ser executada. O tamanho de cada campo em bits é representado a seguir.

OP	dest	var1	var2	func
8	5	5	5	9

As instruções do tipo **Const** são caracterizadas por operações entre um registrador e um valor imediato (constante). Esse tipo de instrução contém 4 campos, sendo eles: o **OP** (*op-code*) que indica ao processador qual a operação ele irá executar. O **dest** que caracteriza o registrador responsável por armazenar o resultado da operação entre o registrador indicado por **var** e a constante indicado por **cte**. O tamanho de cada campo em bits é representado a seguir.

OP	dest	var	cte
8	5	5	14

Dessa maneira, a arquitetura AS-MIC é composta por 4 instruções aritméticas e 2 instruções lógicas, sendo 2 do tipo **reg** e 4 do tipo **const**. A seguir, tem-se a descrição das instruções.

- Soma

A instrução soma é do tipo **reg** e representa uma operação aritmética. Esta instrução armazena no registrador **dest** o resultado da adição entre os registradores **var1** e **var2**. Sua sintaxe e seus valores de **OP** e **func** são indicados a seguir.

soma dest, var1, var2

0x00	dest	var1	var2	0x11
8	5	5	5	9

- Mult

A instrução mult é do tipo **reg** e representa uma operação aritmética. Esta instrução armazena no registrador **dest** o resultado da multiplicação entre os registradores **var1** e **var2**. Sua sintaxe e seus valores de **OP** e **func** são indicados a seguir.

mult dest, var1, var2

0x00	dest	var1	var2	0x22
8	5	5	5	9

- Somai

A instrução somai é do tipo **const** e representa uma operação aritmética. Esta instrução armazena no registrador **dest** o resultado da adição entre os registrador **var1** e a constante **cte**. Sua sintaxe e seu valor de **OP** são indicados a seguir.

somai dest, var1, cte

0x17	dest	var	cte
8	5	5	14

- Multi

A instrução multi é do tipo **const** e representa uma operação aritmética. Esta instrução armazena no registrador **dest** o resultado da multiplicação entre os registrador **var1** e a constante **cte**. Sua sintaxe e seu valor de **OP** são indicados a seguir.

multi dest, var1, cte

0x13	dest	var	cte
8	5	5	14

- Load

A instrução load é do tipo **const** e representa uma operação lógica. Esta instrução carrega um dado da memória para o registrador, ou seja, armazena em **dest** o dado localizado na posição de memória indicado pela soma entre **var** e **cte**. Sua sintaxe e seu valor de **OP** são indicados a seguir.

load dest, cte(var)

0x25	dest	var	cte
8	5	5	14

- Store

A instrução store é do tipo **const** e representa uma operação lógica. Esta instrução carrega o valor do registrador para a memória, ou seja, armazena o valor do registrador **dest** na posição de memória indicado pela soma entre o registrador **var** e a constante **cte**. Sua sintaxe e seu valor de **OP** são indicados a seguir.

s*tore dest, cte(var)*

0x27	dest	var	cte
8	5	5	14

2.1.2 Memória Cache virtual

A memória cache é uma memória mais rápida do que a memória principal e tem como objetivo armazenar os dados e instruções mais utilizadas pelo processador, permitindo que estas sejam acessadas rapidamente. A memória cache possui uma capacidade menor de armazenando do que a memória principal e utiliza de um princípio de localidade para determinar qual bloco de dados da memória principal ela deve carregar. O princípio de localidade está relacionado à probabilidade de uso de dados pelo processador no futuro, ou seja, se o dado de um endereço x é solicitada pelo processador, as chances de o processador solicitar o mesmo dado ou algum localizado próximo ao endereço x são altas. Assim, quando o dado do endereço x é solicitado, um bloco dos dados próximos a esse endereço é copiado para a memória cache.

A partir disso, a arquitetura AS-MIC é formada por uma memória principal com capacidade de 16Bytes e uma memória cache de instruções com capacidade de 4Bytes. A memória cache é constituída por 3 campos, sendo eles: *valid, tag* e *data*. O campo *valid* é responsável por indicar se a cache possui o dado que o processador precisa, retornando uma

valor booleano. A *tag* por sua vez, referencia a posição da memória principal em que este dado está armazenado, tornando possui a verificação para saber se o dado solicitado pelo processador se encontra na cache. E por fim, o campo *data* representado o dado armazenado na cache, ou seja, é neste campo que está armazenado a instrução dentro da memória cache.

2.2 Descrição de funcionamento

A descrição de funcionamento foi subdividida nas categorias:

- Ferramentas utilizadas;
- Ciclo de execução;

2.2.1 Ferramentas utilizadas

A arquitetura AS-MIC foi desenvolvida utilizando a linguagem de programação Java e a *IDE NetBeans.* O Java é uma linguagem orientada ao objetivo que conta com características que conferem uma série de vantagens, tornando o ato de programação mais fácil e eficiente. Esta linguagem possui uma grande quantidade de bibliotecas gratuitas para realizar os mais diversos trabalhos, tais como relatórios, gráficos, sistemas de busca, manipuladores de texto, entre outros. Além disso, o Java é uma linguagem multi plataforma, ou seja, ela possui a capacidade de rodar em diferentes sistemas operacionais, tais como o Windows, o Linux e o Android. E também possui ferramentas que possibilitam desenvolver várias aplicações, tornando-a bastante versátil e intuitiva. E diante essas vantagens, foi escolhida o uso desta linguagem para o desenvolvimento do projeto.

O *NetBeans* IDE é um ambiente de desenvolvimento integrado gratuito e de código aberto para desenvolvedores de software. Esta plataforma fornece uma base sólida para a criação de projetos, pois possui um grande conjunto de bibliotecas, módulos e APIs (Application Program Interface) além de uma documentação vasta bem organizada. Tais recursos auxiliam o desenvolvedor a escrever seu software de maneira mais rápida. Além disso, esta IDE é executado em muitas plataformas, como Windows, Linux, Solaris e MacOS.

2.2.2 Ciclo de busca e execução

Ao ligar a máquina virtual, inicialmente ocorre o carregamento das instruções da memória secundária para memória principal. Após esta etapa, a CPU começa a operar realizando a busca das instruções na memória cache, mas como a máquina acabou de ser iniciada, a cache terá seu bit *Valid* falso, ou seja, ela não deve conter nenhuma informação.

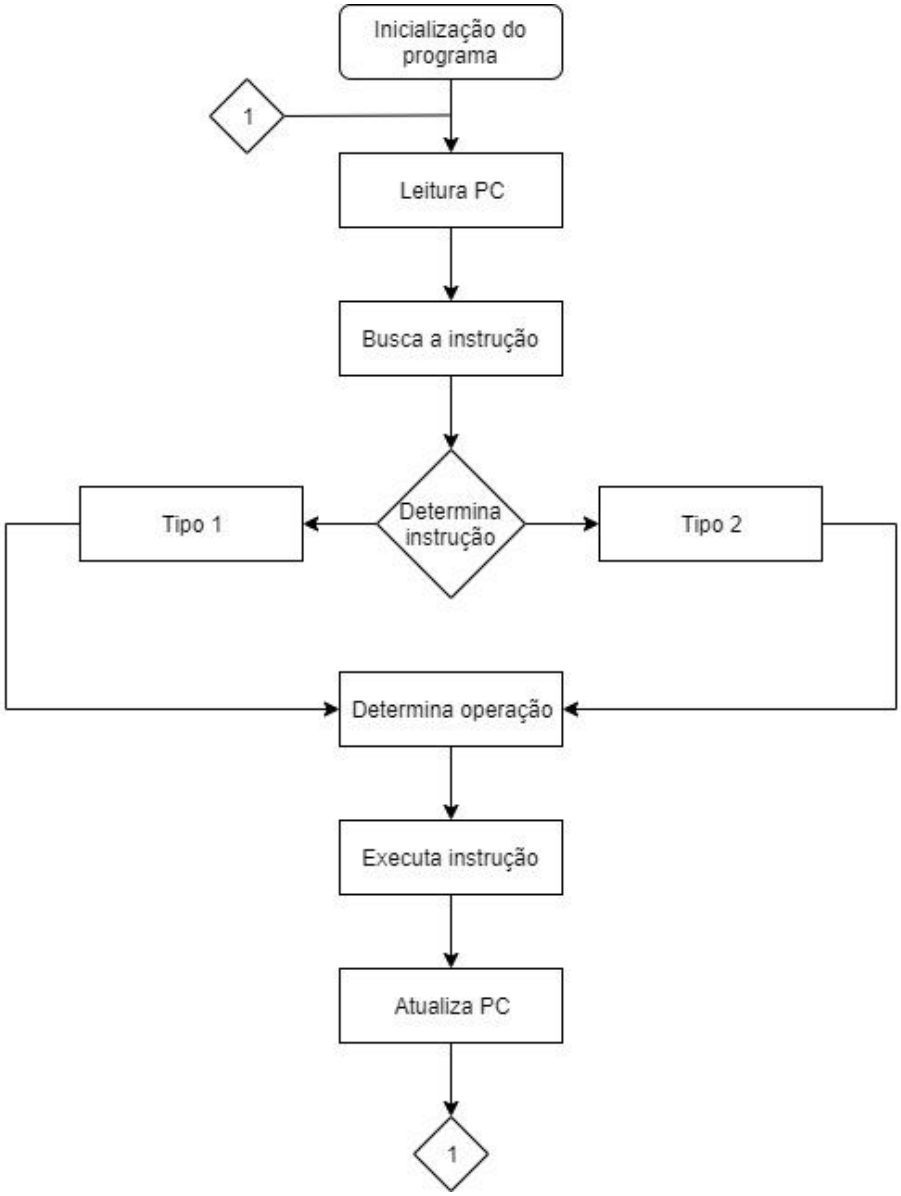
Deste modo acontece o primeiro cache *miss*, fazendo com que esta memória faça sua primeira consulta à memória principal, buscando e armazenando os 4 primeiros blocos de instruções e atribuindo a eles uma *tag* correspondente. Com isso, a cache já possui informações, podendo então enviar à CPU para que ela possa realizar a execução da primeira instrução. Ao executar, incrementa-se o valor de PC, assim partindo para busca da próxima instrução, desta vez a cache possui informações (bit *Valid* verdadeiro) então o que é verificado é se a próxima instrução está contida na cache ou não: Se sim, ocorre um cache *hit* e a CPU executa a instrução, porém caso a instrução não esteja na cache, então outro *cache miss* é enfatizado, fazendo com que ela faça todo processo de buscar os próximos blocos da memória principal novamente. Ao buscar, a CPU executa a instrução, incrementa-se o valor de PC e continua os passos anteriores novamente, até que a máquina seja desligada ou não tenha mais instruções carregadas na memória principal.

Na execução, primeiramente o que acontece é a verificação do tipo de instrução (**Const** ou **Reg**) que fica armazenado no *op-code* dela com 8bits. Caso este *op-code* seja 0x00, então é determinado que a instrução é do tipo **reg**, caso contrário é uma instrução do tipo **const**, após essa determinação as instruções são tratadas separadamente. Após a primeira decodificação, as instruções do tipo **reg** são decodificadas novamente, mas agora ocorre a separação dos valores de bits em seus respectivos campos de acordo com os seus tamanhos, sendo eles: **dest**, **var1**, **var2** em 5bits cada e o campo **func** em 9bits. Após essa etapa, o processador determina o tipo de operação (**soma** ou **mult**) que será executada, de acordo com o campo **func** e subsequente ocorre a execução dessa operação aritmética entre os campos **var1** e **var2** e armazenando seu resultado no registrador determinado por **dest** finalizando a instrução e seguindo para a próxima leitura do registrador PC. No entanto, caso a instrução seja do tipo **const**, a segunda decodificação irá separar os valores de bits nos determinados campos de acordo com seu tamanho: **OP** em 8 bits, **dest** e **var**, em 5bits cada e o campo **cte** em 14bits. A determinação do tipo de operação ocorre com a leitura do campo **OP**, caso este determine uma operação aritmética (**somai** ou **multi**), o processador irá inicialmente converter o valor binário do campo **cte** para um valor decimal e assim irá realizar a operação com o dado armazenado no registrador indicado por **var** e subsequente irá salvar o resultado no registrador determinado por **dest**, finalizando assim a instrução. Mas caso a operação seja do tipo lógica (**load** ou **store**), o processador inicialmente irá converter os valor binário nos campos **var** e **cte** para decimal, realizar a soma destes valores e posteriormente converte o resultado para um valo binário de 5bits, que chamaremos de *binaryValue* para a operação

load e *binaryKey* para o **store**. Após essa etapa, é realizada a operação lógica, no qual o **load** consiste na busca de um dado da memória, ou seja, este operador irá armazenar no registrador indicado por **dest** o valor localizado na posição de memória representada pelo *binaryValue*. Enquanto a operação **store**, consiste em carregar um valor para a memória, ou seja, irá armazenar na posição de memória indicada pelo *binarykey*, o valor que está salvo no registrador **dest**.

3 FLUXOGRAMA

O fluxograma da arquitetura AS-MIC é demonstrado a seguir e contempla os seguintes passos: Após a inicialização do programa ocorre a leitura do registrador PC, que é responsável por indicar a instrução que será executada, posteriormente ocorre a busca da instrução definida pelo PC e assim é realizada a sua decodificação para determinar se a instrução é do tipo 1 ou tipo 2. Subsequente a essa etapa, ocorre à segunda decodificação, no qual é determinado o tipo de operação, aritmética ou lógica, que será efetuada pelo processador. Por fim, a instrução é executada e o PC é incrementado, para uma próxima leitura e a continuidade do ciclo de execução.



4 CONCLUSÃO

A arquitetura AS-MIC, projetada pela equipe com base nos conceitos adquiridos durante a disciplina de Arquitetura Computacional, visou simular o funcionamento de uma arquitetura computacional genérica. Por meio do desenvolvimento deste projeto foi possível obter uma maior compreensão dos processos de busca, decodificação e execução de uma instrução, assim como a importância e o funcionamento de uma memória cache. Com a finalização da proposta, foi notório o ganho de conhecimento proporcionado pela experiência de projetar e desenvolver uma arquitetura, tendo uma visão mais detalhada de seu funcionamento.

REFERÊNCIAS

Anakim, Soraya. **Regras ABNT 2019 para trabalho acadêmico**. Disponível em: <https://cursoseempregos.com/regras-abnt-2019-para-trabalho-academico/ >. Acesso em: 23 nov. 2019.

Profa. Bruschi, Sarita Mazzini. **Arquitetura de Computadores**. Disponível em:< https://edisciplinas.usp.br/pluginfile.php/4408580/mod_resource/content/1/3aula%20-%20Revisao%20MIPS%20monociclo.pdf/ >. Acesso em: 24 nov. 2019.

NetBeans . **NetBeans IDE - A Forma Mais Inteligente e Rápida de Codificar**. Disponível em:< https://netbeans.org/features/index_pt_BR.html/ >. Acesso em: 25 nov. 2019.

Redação Oficina. **O que é o NetBeans?**. Disponível em:< https://www.oficinadanet.com.br/artigo/1061/o_que_e_o_netbeans/ >. Acesso em: 25 nov. 2019.

Wikidot. **Assembly -Tudo sobre linguagem Assembly**. Disponível em:< http://assemblytutorial.wikidot.com/registers/>. Acesso em: 24 nov. 2019.

ANEXO A – Programa

Link do Github: http://twixar.me/S5DT