**Projeto EC208 – Arquiteturas de Computadores II**

**Carlos Bryan Andrade de Paiva**

**Carlos Henrique Jacinto**

**Daniel Liz Fonseca Castro Borges**

**Jonathan Brendon Eugênio**

**Pedro Henrique Carvalho Alves**

As instruções terão 16 bits cada.

Os 4 primeiros bits indicam a operação a ser realizada:

-Load -> 0100

-Store -> 0101

-Soma -> 0000

-Subtração -> 0001

-Multiplicação - > 0010

-Divisão - > 0011

Após a indicação do tipo de operação, os próximos bits dependem da operação escolhida, caso seja:

- Store, os 6 bits posteriores indicarão o local na memória onde o valor será armazenado, e os outros 6 bits indicam o valor a ser inserido.

- Load, os 6 bits posteriores indicarão o registrador onde o valor será armazenado temporariamente, e os outros 6 bits, a linha do arquivo da memória onde o valor será buscado.

- Operações Aritméticas (soma, subtração, multiplicação e divisão), os 6 bits posteriores indicam o primeiro registrador utilizado na operação, e os 6 bits finais, o segundo registrador.

Existe um caso específico para a operação Store, caso os 6 bits finais da instrução sejam 1 (111111) o qual indica que deverá salvar o valor do registrador c na memória, no local indicado pelos 6 bits anteriores (segundo conjunto de bits).

As instruções serão salvas no arquivo “instru.txt”, e a memória será o arquivo “memoria.txt”. A arquitetura desenvolvida possui 5 registradores e um Program Counter (pc), que indica o número da instrução a ser executada.

Ao executar a instrução Load, antes de buscar o valor na memória, é verificado se o valor procurado está disponível na memória cache. Se o valor for encontrado, o programa pega o valor direto da cache e incrementa a variável cache hit, senão, a cache busca o valor na memória, e armazena o valor juntamente com um conjunto de valores próximos ao endereço do valor buscado na memória e incrementa o valor da variável cache miss.

A memória cache projetada possui 4 bits de índice resultando em 16 blocos, e 2 bits de Tag, resultando em 4 palavras por bloco.

O controle dos dados buscados na memória principal para a memória cache é feito através de uma matriz 16x4 (16 linhas e 4 colunas), onde as colunas representam o índice, bit de validação, tag e dado. Para manter-se a coerência de cache, toda vez que é realizada a operação Store, verifica-se se o respectivo valor da memória principal está contido na memória cache e se estiver ele é atualizado também na memória cache.

Abaixo, a tabela exemplifica como a memória cache está sendo representada no código.

|  |  |  |  |
| --- | --- | --- | --- |
| Índice | Validação | Tag | Dado |
| 0000 | S | 01 | 21 |
| 0001 | N | 00 | 0 |
| 0010 | S | 11 | 12 |
| 0011 | S | 01 | 32 |
| 0100 | N | 00 | 0 |
| 0101 | N | 00 | 0 |
| 0110 | N | 00 | 0 |
| 0111 | N | 00 | 0 |
| 1000 | N | 00 | 0 |
| 1001 | N | 00 | 0 |
| 1010 | S | 01 | 37 |
| 1011 | N | 00 | 0 |
| 1100 | N | 00 | 0 |
| 1101 | N | 00 | 0 |
| 1110 | N | 00 | 0 |
| 1111 | S | 11 | 26 |

Considerando que a cache já está carregada conforme a tabela acima, as operações abaixo serão exemplificadas:

0100 000000 110010

0100 000001 111111

0000 000000 000001

0101 000100 111111

A primeira linha carregará o registrador A, com o valor correspondente ao endereço 110010 (linha 25). Quando o valor for buscado, como ele já está na cache, não será necessário o acesso à memória principal.

O mesmo ocorre na segunda instrução, que carrega o registrador B, com o valor correspondente ao endereço 111111 (63).

A terceira instrução somará o valor dos registradores A e B, armazenando o resultado no registrador C.

A quarta instrução salvará o valor do registrador C na linha 4 da memória, e verificará se a memória cache contém o valor correspondente à linha 4 e atualizará os dados da memória cache.

No exemplo acima, obteve-se 100% de cache hit, ou seja, as duas instruções de Load encontraram o valor na cache.

A figura abaixo exemplifica a Arquitetura descrita:

