AD1 - Organização de Computadores 2017.1
Data de entrega 21/03/2017

1. (1,5) Explique em detalhes como funciona a execução da instrução *STR op* na arquitetura mostrada em aula.
    a. *RI <- (CI)*
    b. *CI <- CI + 1*
    c. *Decodificação do código de operação*
        i. *Recebe os bits do código de operação*
        ii. *Produz sinais para a execução da operação de escrita*
    d. *A UC emite sinais para que o valor do campo operando = op seja transferido para o REM*
    e. *Conteúdo do Acumulador (ACC) é transferido para o RDM (RDM ← ACC)*
    f. *A UC ativa a linha WRITE do barramento de controle*
    g. *O REM passa o conteúdo para o barramento de endereços*
    h. *O RDM passa o conteúdo para o barramento de dados*
    i. *A memória grava o dado recebido pelo barramento de dados no endereço enviado através do barramento de endereços*

2. (2,0) Considere uma máquina cujo relógio possui uma frequência de 2.4 GHZ e um programa no qual são executadas 1.200 instruções desta máquina.

    a) Calcule o tempo de UCP utilizado para executar este programa, considerando que cada instrução é executada em dois ciclos de relógio e a execução de uma instrução só se inicia quando a execução da instrução anterior é finalizada. (0,8)

    2,4GHz = 2.400.000.000 Hz
    Tempo de um ciclo de relógio = 1/2.400.000.000 = 0,000 000 000 4167 seg ou 0,4167ns (nanosegundos)
    Tempo de execução de 1 instrução = 2 ciclos de relógio = 0,833ns
    *1200 instruções **executadas sequencialmente** = 1200 × 0,833ns = **1000ns (para executar 1200 instruções)***

    b) Considere que essa máquina utilize um pipeline de 5 estágios, todos de igual duração. Calcule o tempo máximo que o estágio deve durar para que o tempo de execução do programa seja menor do que o tempo calculado no item anterior. (1,2)

    *Tempo total para execução das 1200 instruções deverá ser < **1000ns** (tempo do item a)*
    Consideremos t como sendo o tempo para execução de um estágio
    Tempo para execução de uma instrução = 5 estágios × t = 5t
    *Tempo para execução das demais em **pipeline** = 1199 × t = 1199t*
    Tempo para execução das 1200 instruções = 5t (primeira instrução) + 1199t (para as demais) = 1204t

    O tempo total para execução do 2º. Caso < tempo total execução do 1º. Caso =>
    *1204t < **1000ns** => t < 1000/1204 **=> t < 0,83ns ou t_{max} = 0,83ns** (tempo para um estágio)*

3. (2,5) Considere uma máquina com arquitetura semelhante àquela apresentada em aula. Pode-se endereçar no máximo 128M células de memória onde cada célula armazena uma palavra e cada instrução tem o tamanho de uma palavra. Todas as instruções desta máquina possuem o mesmo formato: um código de operação, que permite a existência de um valor máximo de 1024 códigos, e dois operandos, que indicam endereços de memória.

    a) Qual o tamanho mínimo do REM ? (0,3)

    REM = Barramento de endereços, este terá a capacidade de endereçar 128Mcélulas = N
    N = 128Mcélulas => $N = 2^{27}$ => e = 27 bits

*REM = barramento de endereços = **27 bits***

b) Qual o tamanho mínimo do CI ? (0,3)

*CI terá o tamanho necessário para endereçar toda a memória = REM = **27** **bits***

c) Qual o tamanho do barramento de endereços ? (0,3)

*REM = barramento de endereços **= 27 bits***

d) Qual o tamanho mínimo do RI ? (0,5)

O tamanho mínimo para RI deverá ser o tamanho de uma instrução

Cada instrução tem o tamanho de uma palavra = tamanho de célula
CodOper deverá permitir 1024 códigos diferentes (instruções). CodOper = 10 bits
Operando corresponde a 1 endereço de memória = 27 bits
Instrução = CodOper + 2 Operandos => Instrução = 10 + 2 × 27 => Instrução = 64 bits
*O tamanho mínimo para RI deverá ser **64 bits***

e) Qual a capacidade máxima da memória em bits ? (0,5)

T = M x N => T = tamanho da célula x quantidade de células da memória principal =>
tamanho da célula (M) = 64 bits
total de endereços da memória (N) = 128M células
T = 64 bits/célula x 2^{27} células => T = 8192 M bits (ou 1024Mbytes)

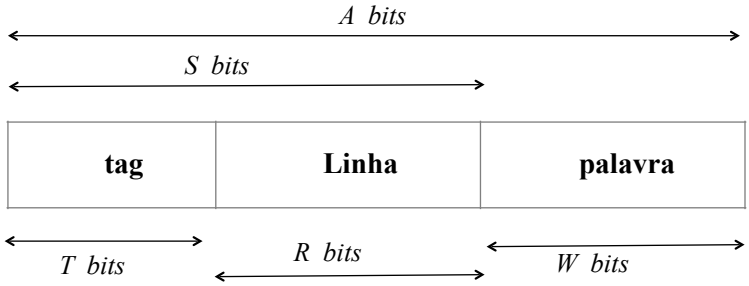
f) Se a largura do barramento de dados desta máquina for igual à metade do tamanho de uma instrução, como funcionará o ciclo de busca ? (0.6)

Seriam necessários 2 ciclos de busca para transferir uma instrução completa.

4. (1,0) Explique em detalhes as três formas de organização da memória cache e como é realizada a localização dos dados em cada uma delas a partir de um endereço de memória.

*Para detalhar as 3 formas de mapeamento, considere que uma CPU deseja ler uma palavra da MP cujo endereço seja representado com **A** bits.*

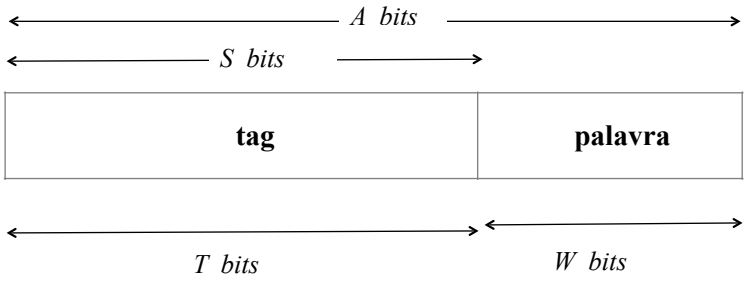
*No **mapeamento direto**, cada bloco da memória principal é mapeado em uma única linha da cache. Nos acessos à memória cache, um endereço da memória principal pode ser visto como constituído de três campos (tag ou rótulo, linha, palavra). Os **W** bits menos significativos identificam uma única palavra dentro de um bloco da memória principal. Os **S** bits restantes especificam um dos 2^S blocos da memória principal. Na memória cache estes **S** bits são interpretados como compostos de 2 campos (tag, linha). O campo linha, com **R** bits, identifica a linha da cache onde poderá ser encontrada a palavra correspondente ao endereço da MP, já que um bloco da MP só poderá estar em uma única linha da cache. Para verificar se a palavra desejada está na linha da cache, o conteúdo do campo tag do endereço da MP com **S – R** bits (ou **T**) deverá ser igual ao conteúdo do campo tag da linha da cache.*



Para exemplificar este mapeamento considere o endereço 1010100010_2 de uma palavra da MP a ser lida pelo processador. No acesso à memória cache, os 3 campos a serem vistos no endereço (1010100 010) serão: 3 bits para tag (101), 4 bits para a linha (0100) e 3 bits para palavra (010). A palavra desejada, que faz parte do bloco 1010100, só poderá ser encontrada na linha 0100 da cache. Mas nesta linha, além do bloco 1010100, também poderiam ser

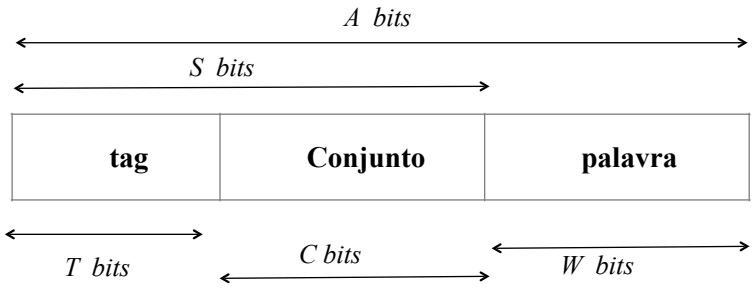
alocados os blocos 0000100, 0010100, 0100100, 0110100, 1000100, 1100100, 1110100. Para saber se o bloco alocado na linha 0100 é o desejado, precisaremos comparar o conteúdo do campo tag da linha 0100 com o do endereço (101), caso sejam iguais temos um cache hit, caso contrário temos um cache miss.
A técnica do mapeamento direto é simples e tem custo de implementação baixo, porém possui a desvantagem de que cada bloco só poderá ser mapeado em uma posição fixa na memória cache.

*O **mapeamento associativo** permite que cada bloco da memória principal possa ser carregado em qualquer linha de memória cache. A lógica de controle da memória cache interpreta um endereço de memória como constituído simplesmente de um rótulo (ou tag) com **T** bits e um campo de palavra com **W** bits. A tag identifica um bloco da memória principal de modo unívoco. Para determinar se um bloco está na memória cache, a lógica de controle da memória cache deve comparar simultaneamente o campo de rótulo de endereço do bloco acessado com os rótulos de todas as linhas. A principal desvantagem do mapeamento associativo é a complexidade do conjunto de circuitos necessários para a comparação das tags de todas as linha da memória cache em paralelo.*



Para exemplificar este mapeamento considere o endereço 1010100010_2 de uma palavra da MP a ser lida pelo processador. No acesso à memória cache, os 2 campos a serem vistos no endereço (1010100 010) serão: 7 bits para tag (1010100) e 3 bits para palavra (010). A palavra desejada, que faz parte do bloco 1010100, poderá estar em qualquer linha da cache. Assim, para saber se a palavra desejada está na cache, a lógica de controle da memória cache deve comparar simultaneamente o campo de rótulo de endereço do bloco acessado com os rótulos de todas as linhas. Caso tenha resposta positiva na comparação, temos um cache hit, caso contrário temos um cache miss.

*O **mapeamento associativo por conjunto** combina as vantagens do mapeamento direto e do mapeamento associativo e diminui suas desvantagens. Nesse mapeamento, a memória cache é dividida em um número de conjuntos com K linhas. Um bloco poderá ser alocado em qualquer uma das K linhas de um conjunto. Similar ao mapeamento direto, no acesso à memória cache um endereço da memória principal pode ser visto como constituído de três campos (tag ou rótulo, conjunto, palavra).*
*Os **W** bits menos significativos identificam uma única palavra dentro de um bloco da memória principal. O conjunto com **C** bits, identifica o conjunto de linhas da cache onde poderá ser encontrada a palavra correspondente ao endereço da MP. Para verificar se a palavra desejada está na cache, a lógica de controle da memória cache deve comparar simultaneamente o campo tag do endereço do bloco acessado com as tags das K linhas do conjunto constante do endereço da MP.*



Para exemplificar este mapeamento considere o endereço 1010100010_2 de uma palavra da MP a ser lida pelo processador. Considere um conjunto com K=2 linhas. No acesso à memória cache, os 3 campos a serem vistos no endereço (1010100 010) serão: 4 bits para tag (1010), 3 bits para o conjunto (100) e 3 bits para palavra (010). A palavra desejada, que faz parte do bloco 1010100, só poderá ser encontrada em uma das 2 linhas do conjunto 100 da cache. Para saber se o bloco alocado no conjunto 100 é o desejado, a lógica de controle da memória cache deve comparar simultaneamente o campo de tag de endereço do bloco acessado com as tags das K linhas do conjunto 100. Caso tenha resposta positiva na comparação, temos um cache hit, caso contrário temos um cache miss.

5. (1,0) Ao se verificar a organização de um sistema de memória em um computador, observa-se que a memória cache tem uma capacidade de armazenamento muito menor que a memória principal e, ainda assim sabe-se que em 100 acessos do processador a memória cache a taxa de acertos da memória cache é de 95 a 98%. Isso ocorre devido ao princípio da localidade. Explique esse princípio.

Por causa do Princípio de localidade. Observa-se que os programas, na sua grande maioria, são executados em lotes de instruções que são frequentemente acessadas pelo processador e que há uma grande chance de que essas instruções, uma vez acessadas, sejam novamente acessadas em um curto espaço de tempo.

O princípio da localidade pode ser temporal ou espacial. O primeiro (temporal) baseia-se na ideia de que se um Bloco foi acessado recentemente, há grandes chances de que ele seja novamente acessado em breve, durante a execução de um programa (como em um loop, por exemplo). O segundo (espacial) baseia-se na premissa onde uma Palavra foi acessada recentemente, há uma grande probabilidade de que, no próximo acesso à Memória Principal se dê em busca de Palavras (blocos) subjacentes.

6. (1,0) Considere um sistema de armazenamento onde a MP é endereçada por byte, que utiliza o método de mapeamento direto na sua cache e onde o formato dos endereços interpretados pelo sistema de controle é: tag: 8bits ; Linha: 12 bits ; Palavra 8 bits;

a. Qual a capacidade, em bytes, de armazenamento da MP?

Tamanho do endereço = tag (8bits) + linha(12bits) + palavra (8bits) = 28bits
Tamanho da MP = 2^{28} = 256M células
Como cada célula = 1 bytes, Capacidade da MP = 256M bytes

b. Quantas linhas possui a memória cache?

Quantidade de linhas da cache (Q) = 2^e , e = tamanho do campo da linha = 12bits
Quantidade de linhas da cache (Q) = 2^{12} = 4K linhas

c. Qual é a capacidade de armazenamento das linhas?

Quantidade de armazenamento da cada linha = tamanho de um bloco
Quantidade de armazenamento de cada linha = 2^e , e = tamanho do campo da palavra = 8bits
Quantidade de armazenamento de cada linha = 2^8 => bloco = 256 palavras = 256 bytes
Como a cache possui 4K linhas, o total a ser armazenado na cache = 4K × 256 bytes = 1024K bytes

d. Qual a quantidade de blocos da MP atribuídos a uma linha da memória cache?

A quantidade de blocos que pode ser atribuído a uma linha da cache = 2^e , sendo e = tamanho da tag
A quantidade de blocos que podem ser atribuídos a uma linha da cache = 2^8, ou seja,
256 blocos podem ocupar, não simultaneamente, uma linha da cache.

7. (1,0) Supondo que o sistema exposto no exercício anterior utilize o método de mapeamento associativo por conjuntos, onde um conjunto seja formado por 4 linhas, e que o formato de endereços interpretados pelo sistema de controle da cache seja: tag: 8 bits ; Conjunto 8 bits; Palavra 8 bits;

a. Qual a capacidade, em bytes, de armazenamento da MP ?

Tamanho do endereço = tag (8bits) + conjunto(8bits) + palavra (8bits) = 24bits
Tamanho da MP = 2^{24} = 16M células
Como cada célula = 1 bytes, Capacidade da MP = 16M bytes

b. Quantas linhas possui a memória Cache?

Quantidade de linhas da cache (Q) = 2^e × 4 linhas por conjunto, e = tamanho do campo conjunto = 8bits
Quantidade de linhas da cache (Q) = 2^8 × 4 = 1 K linhas

c. Quantos conjuntos possui a memória Cache?

Quantidade de conjuntos da cache (C) = 2^e , e = tamanho do campo conjunto = 8bits
Quantidade de conjuntos da cache (C) = 2^8 = 256 conjuntos

d. Qual é a capacidade de armazenamento das linhas?

Quantidade de armazenamento de cada linha = tamanho de um bloco
Quantidade de armazenamento de cada linha = 2^8 = 256 palavras = 256 bytes
Como a cache possui 2K linhas, o total a ser armazenado na cache = 1K × 256 bytes = 256K bytes

e. Qual a quantidade de blocos da MP atribuídos a uma linha da memória cache?

A quantidade de blocos que pode ser atribuído a um conjunto = 2^e , sendo e = tamanho da tag
A quantidade de blocos que podem ser atribuídos a um conjunto = 2^8 = 256 blocos
Cada linha de um conjunto poderá receber qualquer um dos 256 blocos que são atribuídos ao conjunto