AD1 - Organização de Computadores 2015.2
Data de entrega 08/09/2015

1. (1,5) Considere uma máquina com arquitetura semelhante àquela apresentada em aula. Pode-se endereçar no máximo 256M 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 612 códigos, e dois operandos, que indicam endereços de memória.

a) Qual o tamanho mínimo do CI ?

Barramento de endereços terá a capacidade de endereçar 256M células = N

N = 256Mcélulas = 2^e = 2^{28} => e = 28 bits

CI terá o tamanho necessário para endereçar toda a memória = ***28*** **bits**

b) Qual a capacidade máxima da memória em bits ?

Cada instrução tem o tamanho de uma palavra = tamanho de célula
Instrução = CodOper + 2 Operandos
CodOper deverá permitir 612 códigos diferentes (instruções). CodOper = 10 bits
Operando corresponde a 1 endereço de memória = 28 bits
Instrução = CodOper + 2 Operandos => Instrução = 10 + 2 x 28 => Instrução = 66 bits
tamanho da célula (M) = tamanho da palavra = tamanho instrução = 66 bits

T = M x N => T = tamanho da célula x memória principal =>
tamanho da célula (M) = 66 bits
total de endereços da memória (N) = 2^{28} células
T = 66 bits/célula x 2^{28} células => T = 17.716.740.096 bits

c) Qual o tamanho mínimo do REM ?

REM = Barramento de endereços = ***28 bits***

d) Qual o tamanho mínimo do RI ?

O tamanho mínimo para RI deverá ser o tamanho de uma instrução
O tamanho mínimo para RI deverá ser ***66 bits***

e) Qual o tamanho do barramento de endereços ?

REM = barramento de endereços ***= 28 bits***

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 ?

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

2. (1,5) Escreva um programa que utilize as instruções de linguagem de montagem apresentadas na aula 4 para executar o seguinte procedimento. O conteúdo da memória cujo endereço é 70 é lido e verifica-se se o seu valor é menor que 0. Caso seu valor seja menor que 0, o conteúdo de memória cujo endereço é 60 é adicionado ao conteúdo de memória cujo endereço é 70 e o resultado é armazenado no endereço 70. Caso contrário, o conteúdo de memória cujo endereço é 80 é multiplicado por 3 e o resultado é armazenado no endereço 60. Além de apresentar seu programa escrito em linguagem de montagem, apresente também o programa traduzido para linguagem de máquina.

Endereço	*Instrução*	*Descrição*	*Linguagem Máquina (bin / hexa)*
10	*LDA 70*	*ACC ← (70)*	*(000101110000 / 170)*
11	*JN 17*	*CI ← 17 se ACC<0*	*(011100010111 / 717)*
12	*LDA 80*	*ACC ← (80)*	*(000110000000 / 180)*
13	*ADD 80*	*ACC ← ACC + (80)*	*(001110000000 / 380)*
14	*ADD 80*	*ACC ← ACC + (80)*	*(001110000000 / 380)*
15	*STR 60*	*(60) ← ACC*	*(001001100000 / 260)*
16	*HLT*	*Encerra*	*(000000000000 / 000)*
17	*ADD 60*	*ACC ← ACC + (60)*	*(001101100000 / 360)*
18	*STR 70*	*(70) ← ACC*	*(001001110000 / 270)*
19	*HLT*	*Encerra*	*(000000000000 / 000)*

OBS1: Considerando endereços em hexadecimal

*OBS2: Para realizar a operação de multiplicação por 3 (conforme pedido no título da questão), utilizamos as instruções apresentadas no slide 5 da aula 4, mas teríamos outras opções como considerar uma célula (90) com valor 003 e efetuar a multiplicação pela instrução MUL 90 (ACC ← ACC * (90)) . Outra opção é criarmos uma instrução com operando imediato, ou seja, o valor do operando não é um endereço de memória, mas o valor a ser utilizado diretamente na operação. Ex: MUL 03, que corresponde a ACC ← ACC * 03*

3. (1,0) Considere uma máquina cujo relógio possui uma frequência de 2 GHZ e um programa no qual são executadas 500 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.

Tempo de um ciclo de relógio = 1/2.000.000.000 = 0,000 000 000 5 seg ou 0,5ns (nanosegundos)
Tempo de execução de 1 instrução = 2 ciclo de relógio = 2 x 0,5ns = 1,0ns
*500 instruções executadas sequencialmente = 500 x 1,0ns = **500ns ou 0,5µs***

b) Considere que essa máquina utilize um pipeline de 4 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

Tempo total para execução das 500 instruções deverá ser < que 500ns que foi o tempo do 1o.caso
Consideremos t como sendo o tempo para execução de um estágio
Tempo para execução de uma instrução = 4 estágios x t = 4t
Tempo para execução das demais em pipeline = 499 x t = 499t
Tempo para execução das 500 instruções = 4t (primeira instrução) + 499t (para as demais) = 503t
O tempo total para execução do 2o. Caso < tempo total execução do 1o. Caso =>
*503t < 500ns => t < 500/503 => t < 0,99ns ou t max = **0,99ns (tempo para um estágio)***

4. (1,0) Explique detalhadamente como funciona uma unidade de controle microprogramada e explique as diferenças existentes entre microinstruções verticais e horizontais.

Em uma arquitetura microprogramada, a unidade de controle é especificada por um microprograma que consiste de uma sequência de instruções de uma linguagem de microprogramação. Estas instruções são muito simples e especificam microoperações. Uma unidade de controle microprogramada é implementada com circuitos lógicos e é capaz de seguir uma sequência de microinstruções gerando sinais de controle para que cada uma delas seja executada. Os sinais de controle gerados por uma microinstrução são usados para causar transferências de dados entre registradores e memória e execução de operações pela ULA.

As microinstruções horizontais tem como característica ter funções distintas para cada bit que a compõe, como por exemplo, controlar uma linha de controle interna da UCP, controlar uma linha

de barramento externo de controle, definir condição de desviro e endereço de desvio entre outras. Tem a vantagem de ser simples e direto possível, podendo controlar várias microoperações em paralelo, além de uma eficiente utilização do hardware. E possui a desvantagem de maior ocupação de espaço de memória de controle em relação à microinstrução vertical.

As microinstruções verticais se caracterizam por possuir um decodificador extra para identificar quais as linhas que serão efetivamente ativadas. Sua principal vantagem é reduzir o custo da Unidade de controle em função do menor tamanho da instrução, o que poderá ser necessário uma maior quantidade de instruções. Tem como principal desvantagem a redução do tempo devido à necessidade da decodificação dos campos de cada microinstrução.

5. (1,0) Considere a máquina apresentada na aula 4. Descreva detalhadamente (do mesmo modo que é apresentado na aula 4) como é realizada a execução das seguintes instruções:
a) STA 200

Passo 1: RI <- Instrução lida
Passo 2: CI <- CI + 1
Passo 3: Decodificação do código de operação
Passo 3.1: recebe os bits do código de operação
Passo 3.2: produz sinais para a execução da operação de escrita em memória

Passo 4: Armazenando conteúdo no endereço apontado contido como operando
Passo 4.1: A UC emite sinais para que o valor do registrador ACC seja transferido para a RDM.
Passo 4.2: A UC emite sinais para que o valor do registrador ACC seja transferido para a RDM
Passo 4.3: A UC emite sinais para que o valor do campo operando = 200 seja transferido para a REM
Passo 4.4: A UC emite sinais para que o valor contido no REM seja transferido para o barramento de endereços
Passo 4.5: A UC ativa a linha WRITE do barramento de controle
Passo 4.6: Conteúdo do RDM é transferido através do barramento de dados para a posição da memória (200) contido no barramento de endereços

b) JNZ 40
Passo 1: A CPU verifica se o valor contido no ACC é diferente de zero (ACC != 0)
Passo 1.1: Caso seja verdadeira a verificação: CI <- Op. Sendo Op = 40.
Será executada a instrução do endereço 40.
Passo 1.2: Caso seja falsa a verificação: CI <- CI + 1.
Será executada a instrução do endereço seguinte

OBS: Outra opção é JNZ corresponder a (ACC <= 0), assim outra resposta aceita:

Passo 1: A CPU verifica se o valor contido no ACC é diferente de zero (ACC <= 0)
Passo 1.1: Caso seja verdadeira a verificação: CI <- Op. Sendo Op = 40.
Será executada a instrução do endereço 40.
Passo 1.2: Caso seja falsa a verificação: CI <- CI + 1.
Será executada a instrução do endereço seguinte

6. (0,5) Explique a hierarquia de memória dos sistemas computacionais atuais, detalhando cada nível da hierarquia.

Organização hierárquica dos subsistemas de memória:

O subsistema de memória é interligado de forma bem estruturada e organizado hierarquicamente em uma pirâmide com os níveis descritos a seguir.
No topo da pirâmide teríamos os registradores, que são pequenas unidades de memória que

armazenam dados na UCP. São dispositivos de maior velocidade com tempo de acesso em torno de 1 ciclo de memória, menor capacidade de armazenamento além de armazenar as informações por muito pouco tempo.

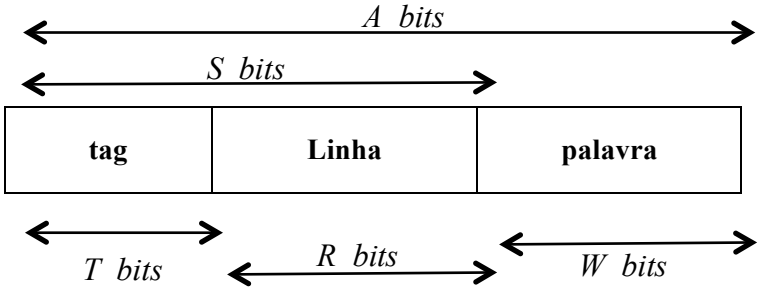
Em um nível abaixo teríamos a memória cache, cuja função é acelerar a velocidade de transferência das informações entre UCP e MP e, com isso, aumentar o desempenho do sistema. A UCP procura informações primeiro na Cache. Caso não as encontre, as mesmas são transferidas da MP para a Cache. A cache possui tempo de acesso menor que a da Memória principal, porém com capacidade inferior a esta, mas superior ao dos registradores e o suficiente para armazenar uma apreciável quantidade de informações, sendo o tempo de permanência do dado menor do que o tempo de duração do programa a que pertence.

Abaixo da memória cache teríamos a memória básica de um sistema de computação, que é a memória principal. Dispositivo onde o programa (e seus dados) que vai ser executado é armazenado para que a UCP busque instrução por instrução para executá-las. A MP são mais lentas que a cache e mais rápidas que a memória secundária, possui capacidade bem superior ao da cache e os dados ou instruções permanecem na MP enquanto durar a execução do programa.

7. (0,5) 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.*

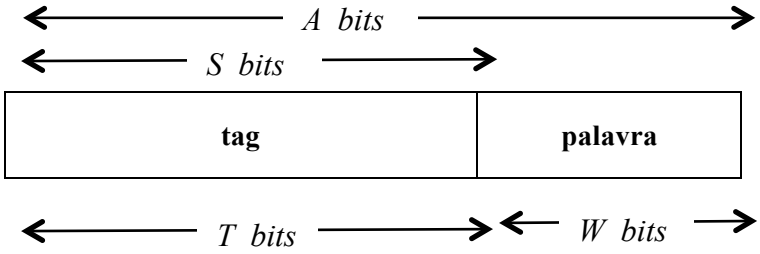
*No **mapeamento direto**, cada bloco da memória principal é mapeado em uma única linha da cache. Nos acessos à memória cache, um endereço da memória principal pode ser visto como constituído de três campos (tag ou rótulo, linha, palavra). Os **W** bits menos significativos identificam uma única palavra 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.

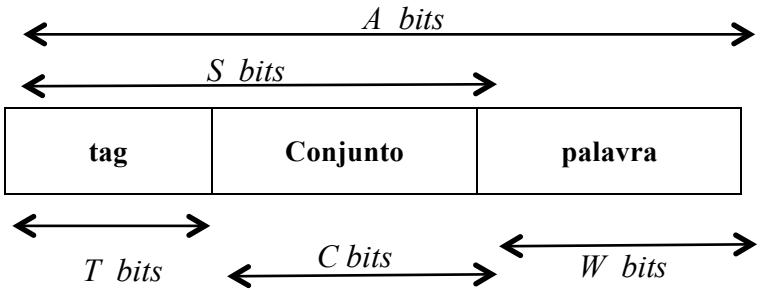
*O **mapeamento associativo** permite que cada bloco da memória principal possa ser carregado em qualquer linha de memória cache. A lógica de controle da memória cache interpreta um endereço de memória como constituído simplesmente de um rótulo (ou tag) com **T** bits e um campo de palavra com **W** 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.

8. (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%. Por quê?

> *Por causa da 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 acessadas novamente em um curto espaço de tempo.*
> *O princípio da localidade pode ser temporal ou espacial. O primeiro (localidade temporal) baseia-se na idéia 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 (loop). O segundo (localidade espacial) baseia-se na premissa onde uma Palavra foi acessada recentemente, há grandes probabilidades de que, o próximo acesso à Memória Principal se dê em busca de Palavras (blocos) subjacentes.*

9. (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: 8 bits ; Linha: 12 bits ; Palavra 4 bits;

a. Qual a capacidade, em bytes, de armazenamento da MP?
Tamanho do endereço = tag (8bits) + linha(12bits) + palavra (4bits) = 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 , 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 = 4bits
Quantidade de armazenamento de cada linha = 2^4 => bloco = 16 palavras = 16 bytes
Como a cache possui 4K linhas, o total a ser armazenado na cache = 4K x 16 bytes = 64K bytes

d. Qual a quantidade de blocos da MP atribuídos á uma linha da memória cache?
A quantidade de blocos que pode ser atribuído à uma linha da cache = 2^e , sendo e = tamanho da tag
A quantidade de blocos que pode ser atribuído à uma linha da cache = 2^8 = 256 blocos podem ocupar, não simultaneamente, uma linha da cache.

10. (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 4 bits;

a. Qual a capacidade, em bytes, de armazenamento da MP ?
Tamanho do endereço = tag (8bits) + conjunto(8bits) + palavra (4bits) = 20bits
Tamanho da MP = 2^{20} = 1M células
Como cada célula = 1 bytes, Capacidade da MP = 1M bytes

b. Quantas linhas possui a memória Cache?
Quantidade de linhas da cache (Q) = 2^e x 4 linhas por conjunto, e = tamanho do campo conjunto = 8bits

Quantidade de linhas da cache (Q) = 2^8 x 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 da cada linha = tamanho de um bloco
Quantidade de armazenamento de cada linha = 2^4 = 16 palavras = 16 bytes
Como a cache possui 1K linhas, o total a ser armazenado na cache = 1K x 16 bytes = 16K bytes

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

A quantidade de blocos que pode ser atribuído à um conjunto = 2^e , sendo e = tamanho da tag
A quantidade de blocos que pode ser atribuído à um conjunto = 2^8 = 256 blocos
Cada linha de um conjunto poderá receber qualquer um de 256 blocos