GABARITO AD1 - Organização de Computadores 2012.1
Data de entrega 13/03/2012

"Atenção: Como a avaliação a distância é individual, caso seja constatado que provas de alunos distintos são cópias umas das outras, independentemente de qualquer motivo, a todas será atribuída a nota ZERO. As soluções para as questões podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser individual."

1. (1,0) Um computador hipotético possui uma capacidade máxima de memória principal com 2G bytes, cada célula é capaz de armazenar uma palavra de 64 bits.

   a) Qual é o maior endereço em decimal desta memória?

   Como M = quantidade de bits em uma célula = 64bits = 8 bytes
   T = capacidade total da memória = 2Gbytes = 2^{31}bytes
   T = N x M => N = T / M => N = 2^{31}bytes / 8 bytes => N = 2^{28} células
   Maior endereço em decimal = N – 1 = 2^{28} -1 = ***268.435.455***

   b) Qual é o tamanho do barramento de endereços deste sistema?

   N = 2^{28} => e = 28 => O barramento terá de ter 28 bits

   c) Quantos bits podem ser armazenados no RDM e no REM?

   Consideremos que uma célula seja transferida pelo barramento de dados durante um processo de leitura/escrita, então palavra = tamanho da célula, e o barramento de dados = tamanho da palavra, concluindo barramento de dados = 64 bits
   RDM = barramento de dados = 64 bits

   Barramento de endereços = REM = 28 bits

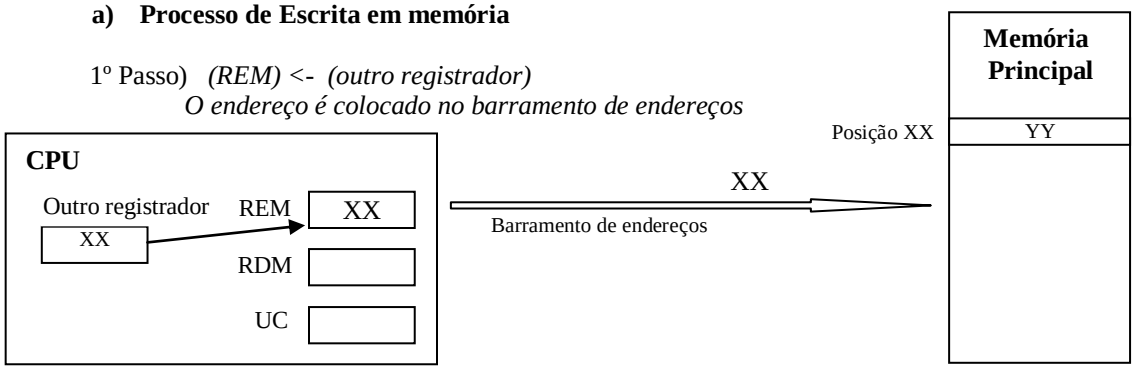
   d) Qual é o número máximo de bits que pode existir na memória?

   T = capacidade total da memória = 2Gbytes = 2^{31}bytes = 2^{31}bytes x 8bits = 2^{34}bits

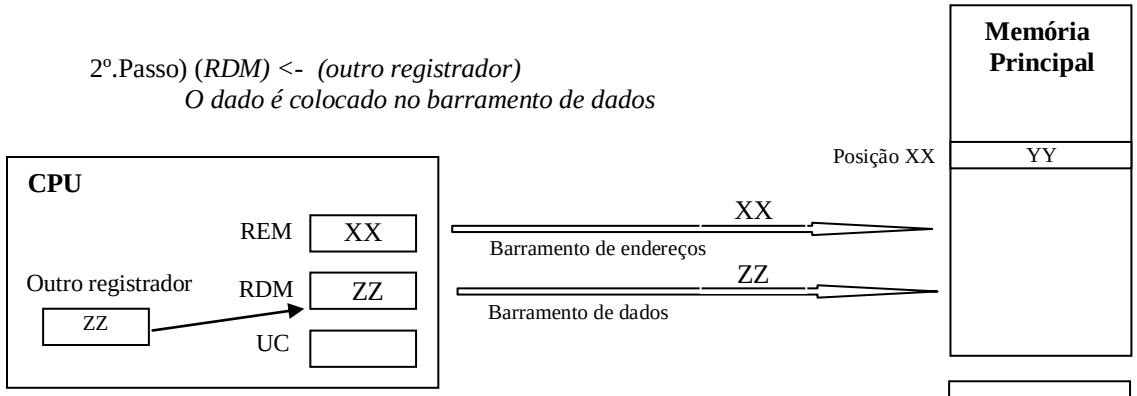
2. (1,0) Descreva e esquematize graficamente passo a passo as operações de leitura da memória e de escrita na memória, indicando como os registradores RDM e REM são utilizados e como a unidade de controle gera os sinais necessários.

a) Processo de Escrita em memória

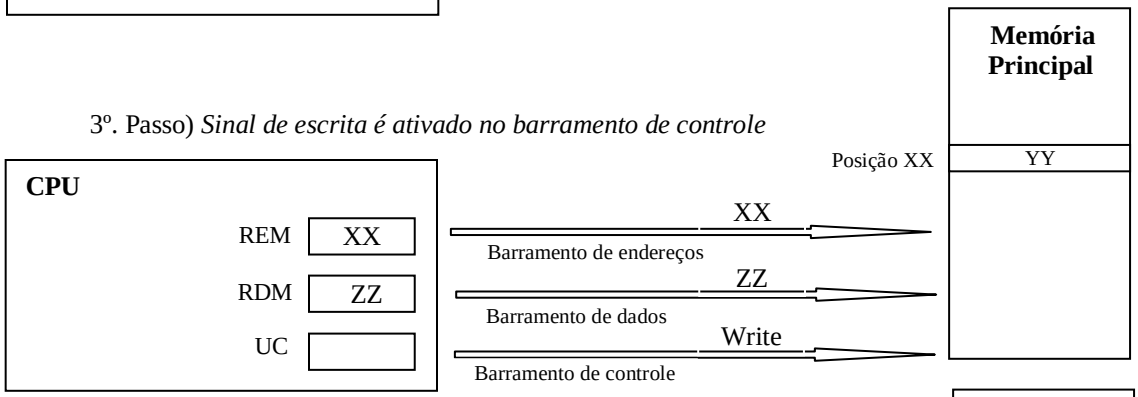
1º Passo) *(REM) <- (outro registrador)*
O endereço é colocado no barramento de endereços



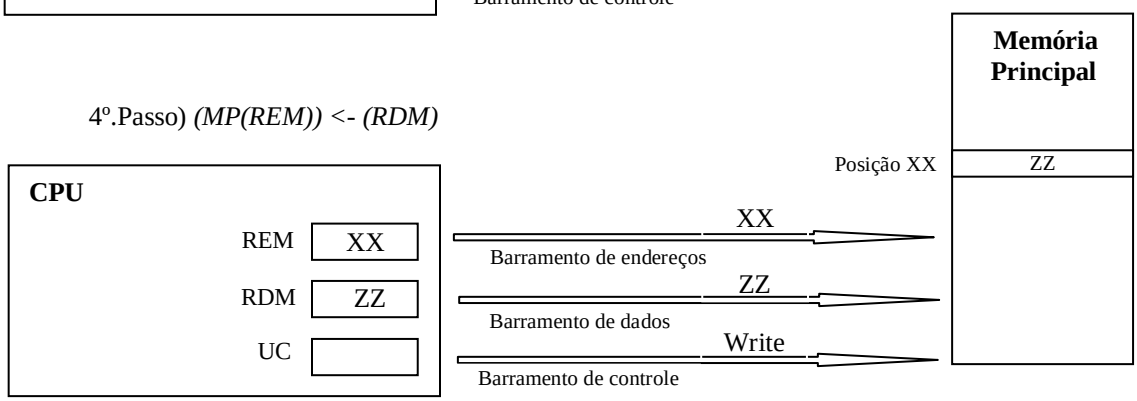
2º.Passo) (*RDM) <- (outro registrador)*
O dado é colocado no barramento de dados



3º. Passo) *Sinal de escrita é ativado no barramento de controle*

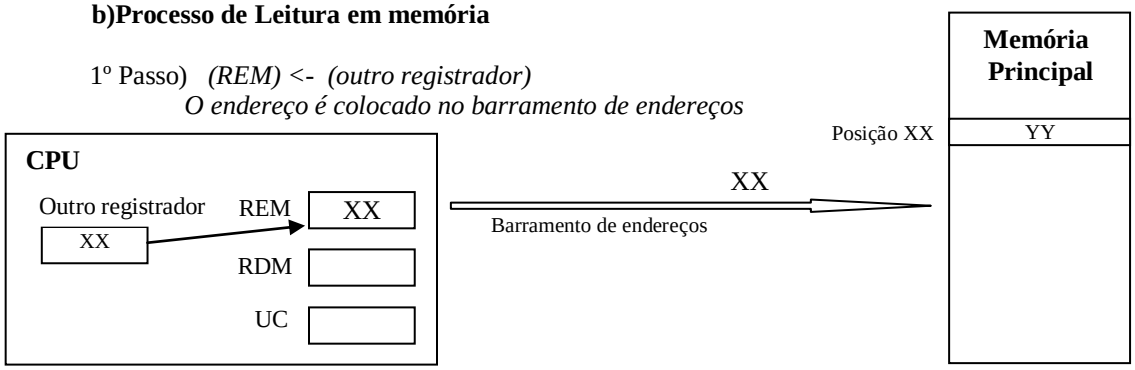


4º.Passo) *(MP(REM)) <- (RDM)*

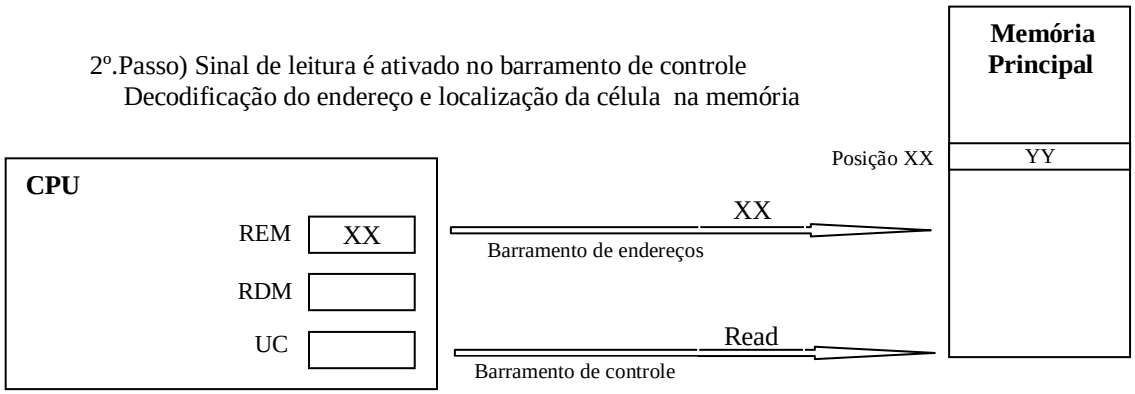


b)Processo de Leitura em memória

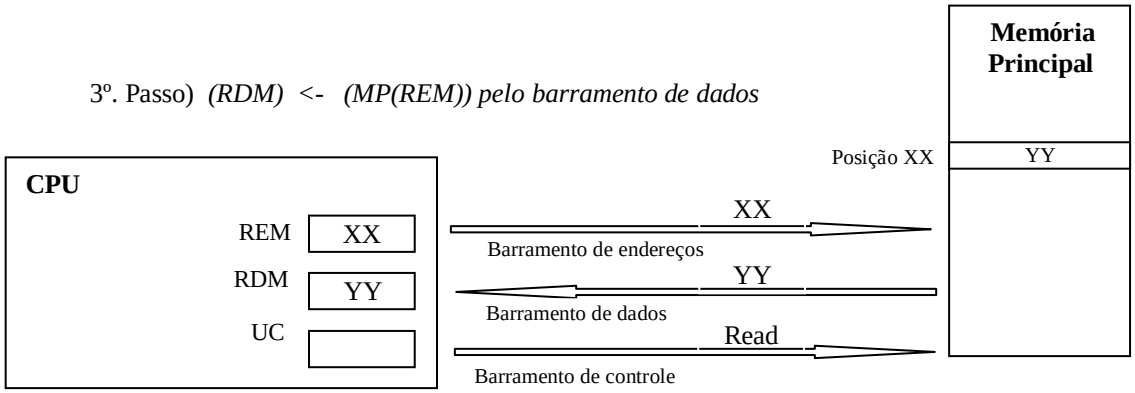
1º Passo) *(REM) <- (outro registrador)*
O endereço é colocado no barramento de endereços



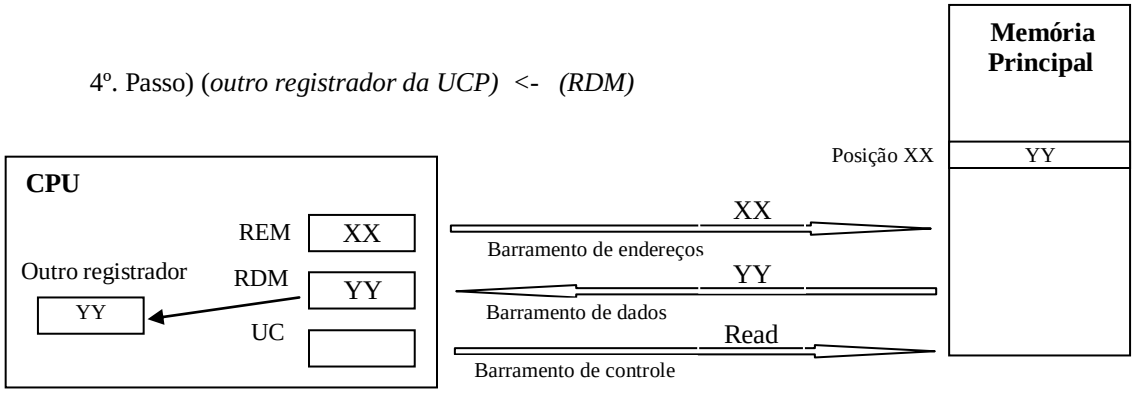
2º.Passo) Sinal de leitura é ativado no barramento de controle
Decodificação do endereço e localização da célula na memória



3º. Passo) *(RDM) <- (MP(REM)) pelo barramento de dados*



4º. Passo) (*outro registrador da UCP) <- (RDM)*



3. (1,0) Considere uma máquina que possa endereçar 4 Gbytes de memória física, utilizando endereço referenciando byte, e que tenha a sua memória organizada em blocos de 512bytes. Ela possui uma memória cache que pode armazenar 256K blocos, sendo um bloco por linha. Mostre o formato da memória cache, indicando os campos necessários (tag, bloco) e o número de bits para cada campo, e o formato de um endereço da memória principal, indicando os bits que referenciam os campos da cache, para os seguintes mapeamentos:

   a) Mapeamento direto.

Memória Principal

⇒ *Tamanho da memória (em bytes) = 4Gbytes, como 1 célula referencia a 1 byte, temos N = 4 G células*

⇒ *Será organizada em blocos de 512 bytes, como 1 célula = 1 byte, temos cada bloco = 512 células, K = 512*

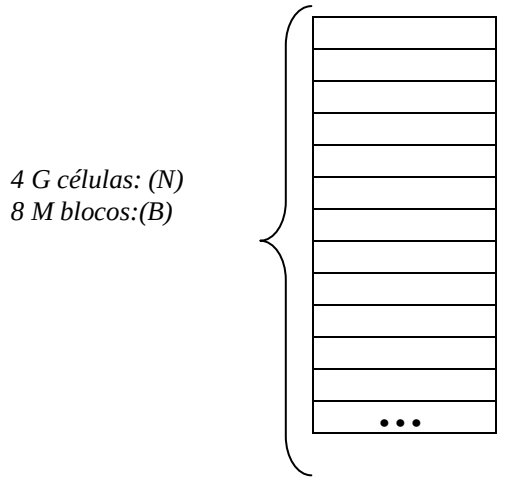
⇒ *Sendo N o tamanho endereçável da memória e K que é a quantidade de células por blocos temos:*
N = 4 G células e K = 512 células / blocos o total de blocos da MP (B) será:
Total de blocos: B = N / K => B = 4 G células / 512 células/bloco => B = 8M blocos

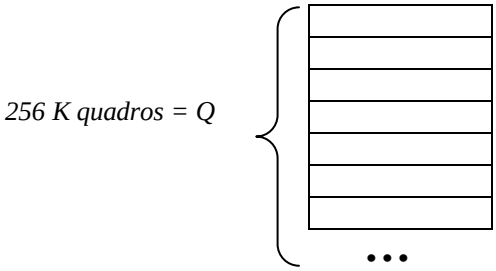
Memória Cache

OBS: O K (quantidade de células/bloco) tem de ser igual a MP.

⇒ *Tamanho da memória cache em blocos = 256K linhas que podem armazenar 256 K blocos*

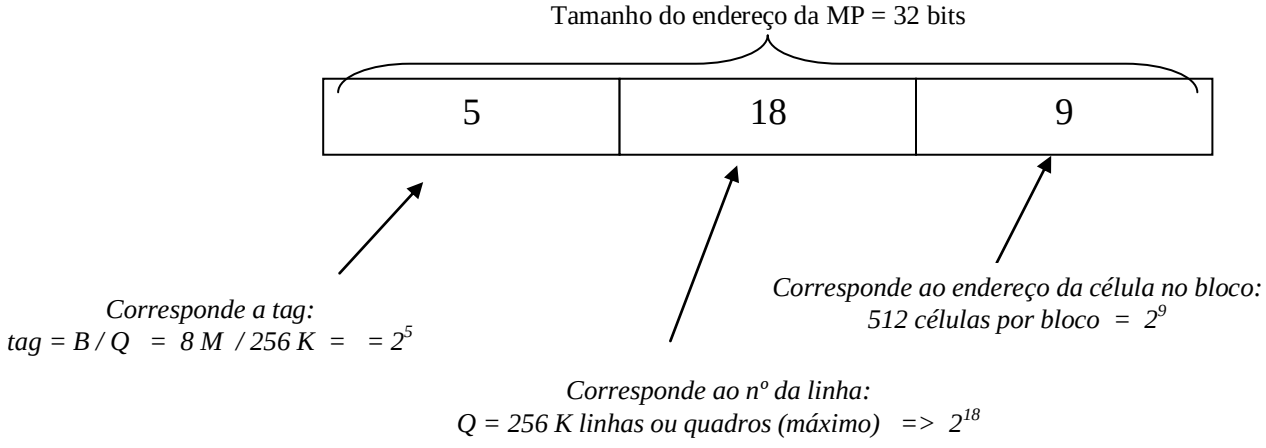
⇒ *Tamanho da memória cache em células = 256K blocos x 512 células/bloco = 128 M células*





Cada célula corresponde a 1 byte (8bits)
Cada bloco (MP) e linha (cache) corresponde a 256K células (K)

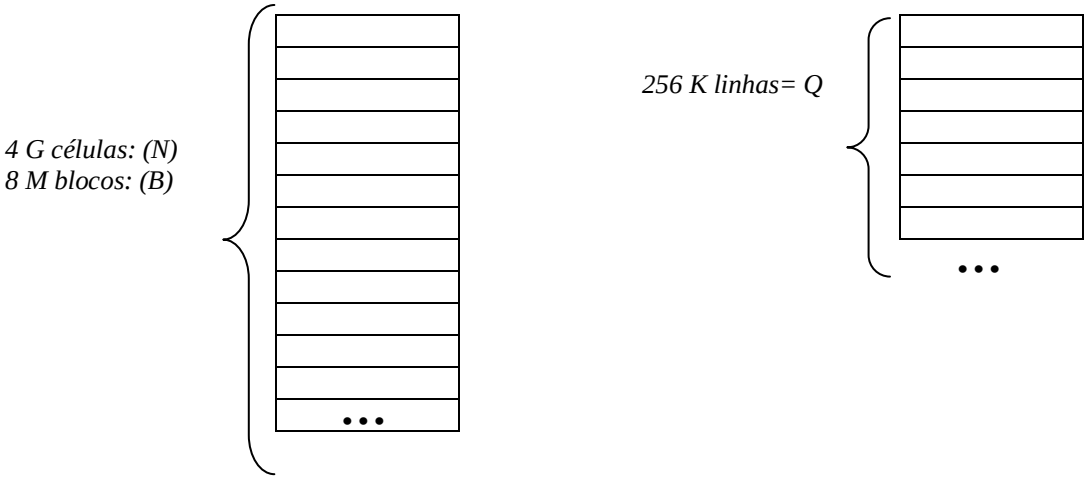
Para endereçarmos toda a MP precisamos da seguinte quantidade de bits (E)
sendo $N = 2^E$ => N = 4G células => $N = 2^{32}$ => E = 32 bits



b) Mapeamento totalmente associativo.

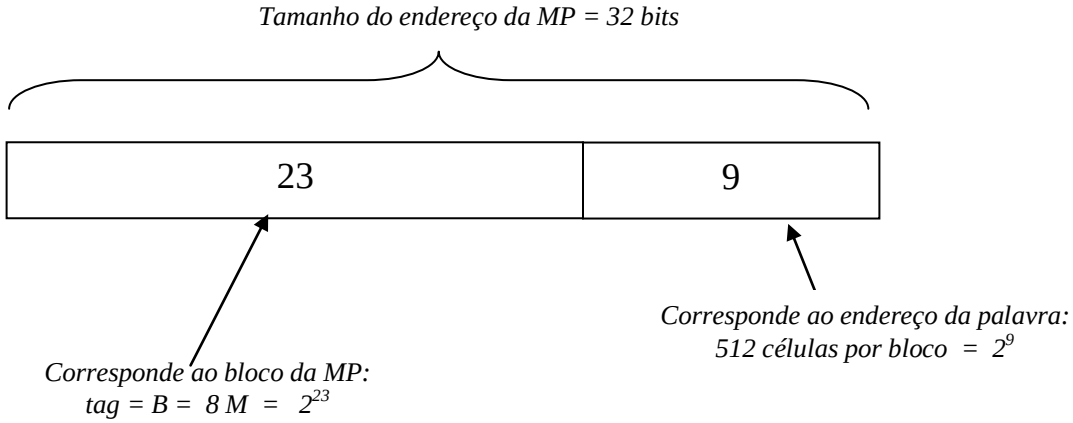
Memória Principal
=> N = 4G células
=> K = 512 células por bloco
=> B = 8 M blocos

Memória Cache
OBS: O K (quantidade de células/bloco) tem de ser igual a MP.
=> Q =256 K linhas ou 256K blocos
=> Tamanho da memória cache = 128 M células



Para endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 32 bits

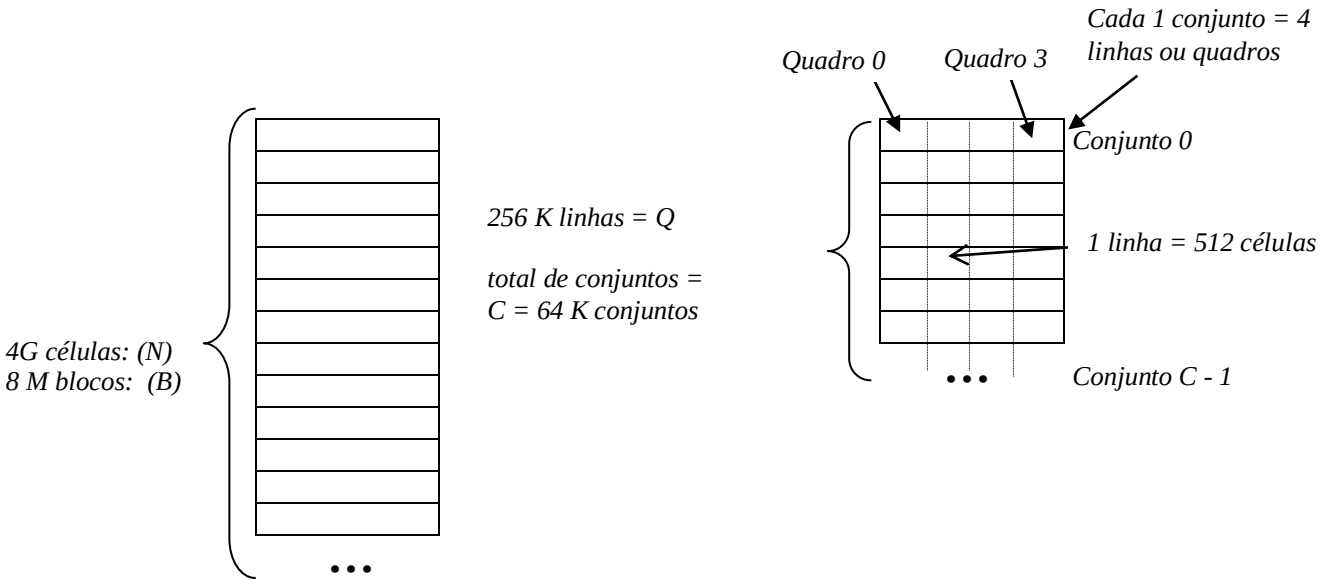
Como o bloco pode ser alocado em qualquer posição da memória cacha a tag indicará qual dos blocos da MP está alocado naquela posição da memória cachê



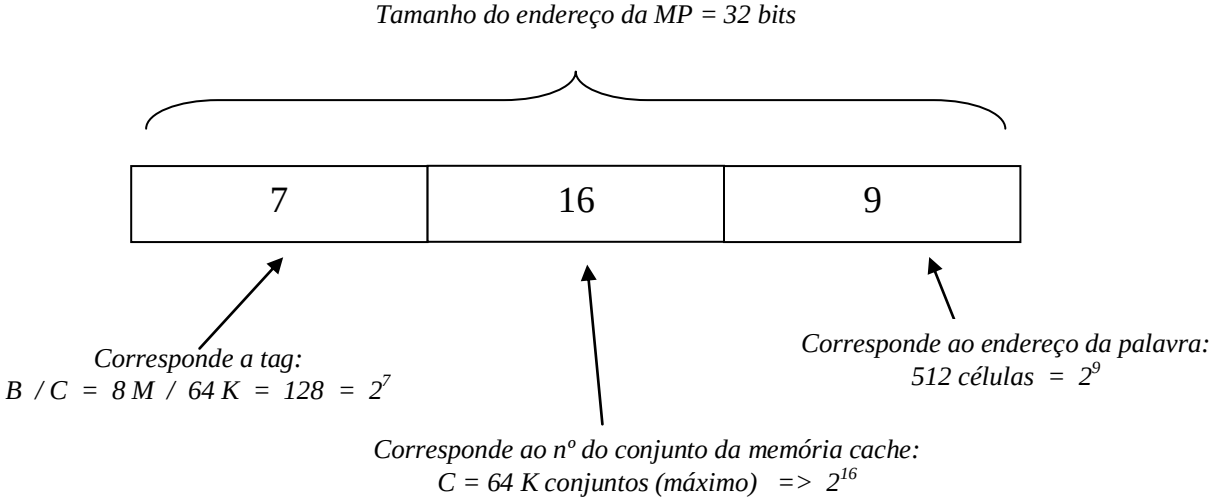
c) Mapeamento associativo por conjunto, onde cada conjunto possui quatro linhas, cada uma de um bloco.

Memória Principal
=> N = 4 G células
=> K = 512 células
=> B = 8 M blocos

Memória Cache
OBS: O K (quantidade de células/bloco) tem de ser igual a MP.
=> Q =256 K linhas ou 256K blocos
=> Tamanho da memória cache = 128M células
=> 1 conjunto = 4 linhas (ou quadros) =>
Total de conjuntos => C = 256K células / 4 => ***C = 64 K conjuntos***

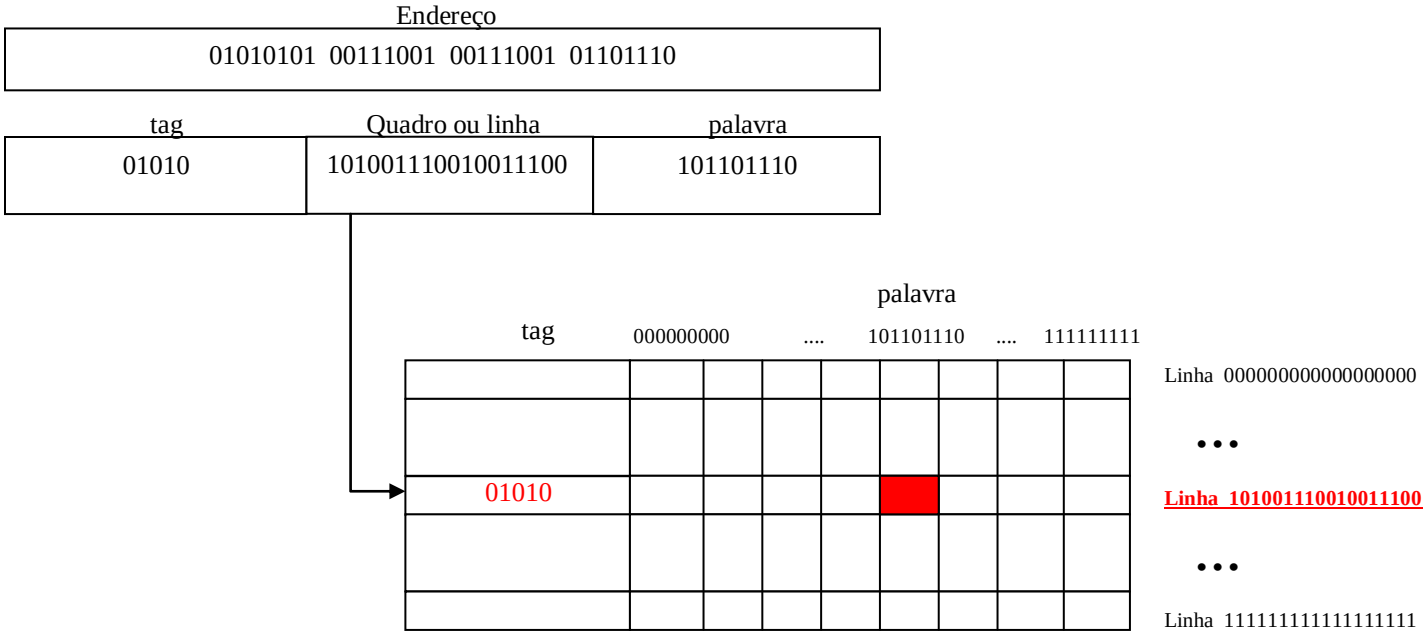


Para endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 32 bits



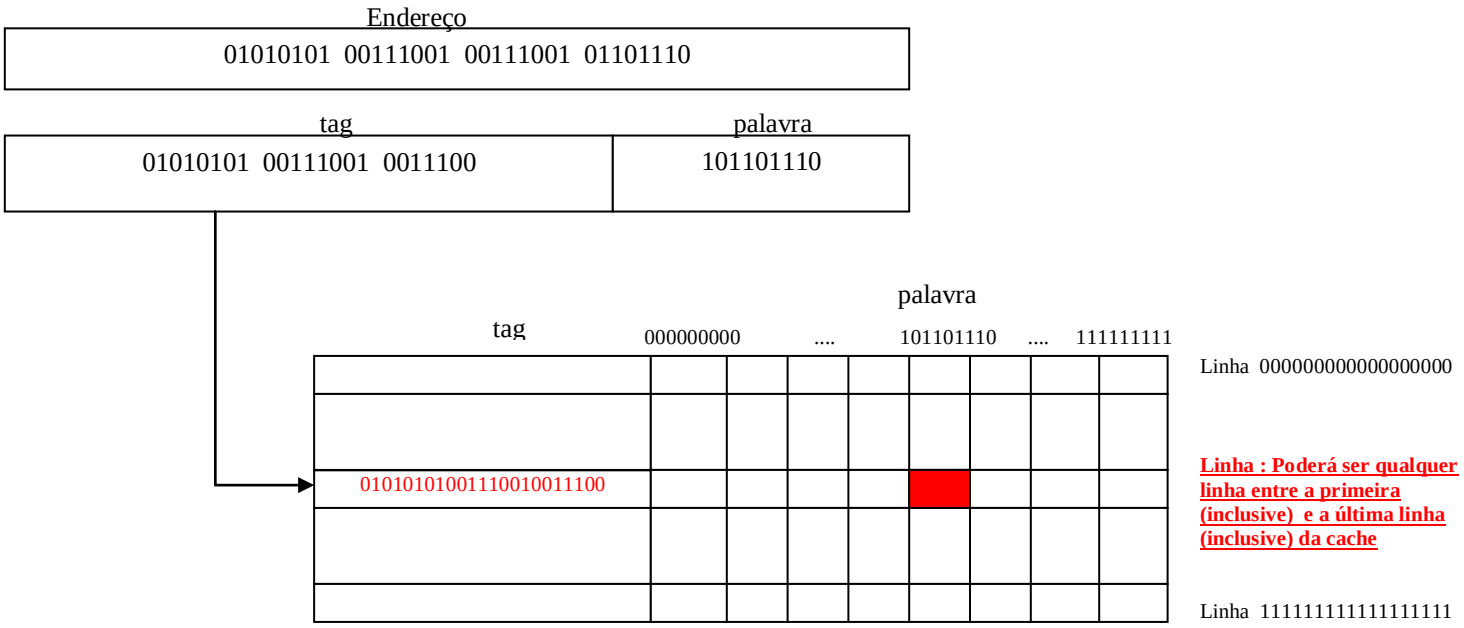
d) Em que linha, para cada um dos mapeamento dos itens anteriores, estaria contido o byte armazenado no seguinte endereço da MP: 01010101 00111001 00111001 01101110.

i) Para o mapeamento direto



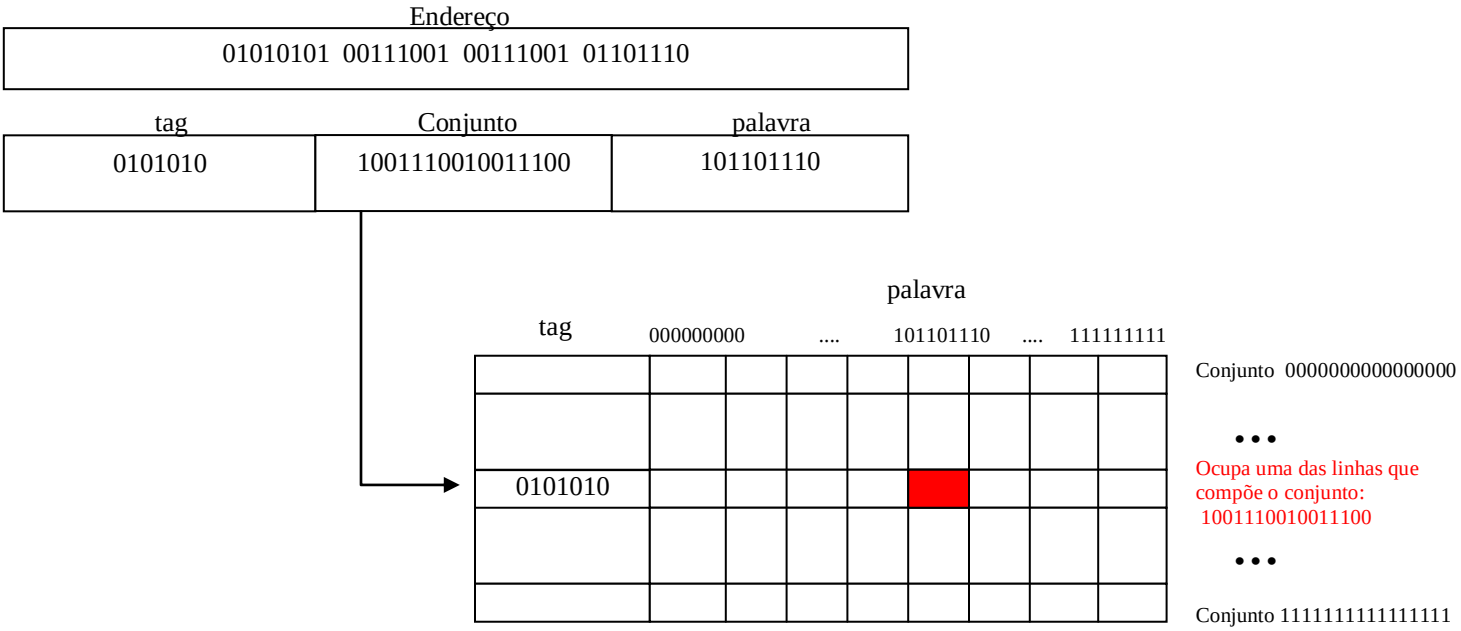
OBS: Com o endereço fornecido, será conferido o campo tag do endereço com a tag da linha indicada, caso não sejam iguais, teremos um cache miss, ou seja a palavra não está na memória cache.

ii) Para o mapeamento totalmente associativo



OBS: Com o endereço fornecido, o conteúdo do campo tag do endereço será procurado linha a linha da cache, caso não encontrado, teremos um cache miss, ou seja a palavra não está na memória cache.

iii) Para o mapeamento associativo por conjuntos



OBS: Com o endereço fornecido, o conteúdo do campo tag do endereço será procurado linha a linha do conjunto, caso não encontrado, teremos um cache miss, ou seja a palavra não está na memória cache.

4. (1,0) Explique em detalhes a organização hierárquica do subsistema de memória e a localização física nos computadores atuais. Descreva particularmente o processador da Intel I5 de segunda geração, descrevendo a cache, velocidade do relógio, número de núcleos, tipos de memória e detalhes da CPU.

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. As MPs são mais lentas que a cache e mais rápidas que a memória secundária, possuem capacidade bem superior à da cache e os dados ou instruções permanecem na MP enquanto durar a execução do programa.

Localização física nos computadores atuais.

1º. Nível: Registradores presentes apenas no interior do núcleo das CPUs.

2º. Nível: Memória Cache. Antes do Pentium II era dividida nos subníveis L1, L2 e L3 o que corresponde respectivamente às caches presentes na CPU, em chips próprios na placa mãe, e em placas conectadas a placa mãe através de slots próprios. Nos processadores atuais, estes subníveis são dispostos internamente no chip da CPU, sendo a L1 individual a cada core, a L2 pode ou não ser compartilhada entre cores, e no caso da haver a L3, esta poderá ser ou não compartilhada pelos núcleos.

3º. Nível: A memória principal é disposta na forma de placas de memória que são conectadas a placa mãe em slots próprios obedecendo às especificações da memória.

4º. Nível: Memória secundária ou auxiliar, temos aí as memórias conectadas, em geral, através de barramentos (IDE, SATA, USB, SCSI) como as unidades de disco rígido, dispositivos com memória flash como pen drives, entre outros.

Processador I5 2ª. geração:
Fonte: http://www.clubedohardware.com.br/artigos/Por-Dentro-da-Microarquitetura-Intel-Sandy-Bridge/2146/1

O processador I,5 como o I3 e I7 da 2ª. geração utilizam a microarquitetura Sandy Bridge. Esta arquitetura tem como principais características:

- *A ponte norte, que controla memória, vídeo e controlador do barramento PCI Express, está integrada no mesmo chip do processor.*
- *Possui arquitetura em Anel, onde os componentes internos do processador se comunicam. Quando o componente quer "conversar" com outro, ele coloca a informação no anel para que ela chegue até o destinatário.*
- *Possui uma cache de instruções decodificadas denominada cache L0, capaz de armazenar cerca de 1536 instruções, em torno de 6KB.*
- *A cache L1 não diferencia da microarquitetura da 1ª. Geração (Nehalem), é dividida em 2: cache de instruções com cerca de 32KB, e cache de dados de igual tamanho. A cache L2 foi renomeada para cache intermediária com 256KB por núcleo. A cache L3 também foi renomeada, esta para cache de último nível, não mais unificada, e é compartilhada entre os núcleos, ou seja, não é ligada a um núcleo em particular. Qualquer núcleo pode usar qualquer uma das caches L3. As caches L3 podem ser utilizadas pelo componente gráfico para armazenar dados, em especial texturas aumentando o desempenho 3D, sem a necessidade de buscar dados na RAM.*
- *Para acesso a memória principal, possui controlador de memória DDR3 de dois canais, suportando memórias de até DDR3-1333*
- *Introdução de um conjunto de instruções AVX (Advanced Vector Extensions ou Extensões de Vetor Avançadas), estas instruções utilizam o conceito SIMD, armazenando dados vetorialmente e processá-los em uma única instrução de processamento. Este conjunto compõe-se de cerca de 12 novas instruções.*
- *Possui processador de vídeo integrado. A microarquitetura Sandy Bridge permite até 12 unidades de execução gráficas.*
- *Possui uma nova versão da tecnologia Turbo Boost que faz overclock no processador quando este demanda mais poder de processamento, e na arquitetura Sandy Bridge, esta tecnologia permite exceder seu TDP (termal Design Power) por até 25 segundos.*

Sobre o I5 2ª. Geração

- *Possui cerca de 995 milhões de transistores. Destes, 114 milhões são reservados para a GPU, também chamada pela Intel de Processador Gráfico.*
- *Processamento Quad Core: 4 núcleos*
- *Frequência base do processador: 3.1 ou 3.3GHz*
- *Frequência do Turbo Boost: chega até 3,7GHz*
- *Intel HD Graphics 3000: 12 unidades gráficas trabalhando em 850Mhz, podendo chegar a 1100Mhz via TurboBoost 2.0;*
- *Controlador de Memória Integrado: Oferece suporte para dois canais de memória DDR3-1333 com dois DIMMs por canal*

5. (1,0) Considere uma máquina com arquitetura semelhante àquela apresentada em aula. Pode-se endereçar no máximo 1G células de memória onde cada célula armazena uma palavra e cada instrução tem o tamanho de uma palavra. Esta máquina possui uma capacidade de armazenamento de 2G bytes **(corrigido para 4Gbytes).** Todas as instruções desta máquina possuem o mesmo formato: um código de operação e um operando que indica um endereço de célula de memória.

a) Qual o tamanho mínimo do REM?

REM = Barramento de endereços, este terá a capacidade de endereçar 1G células = N
N = 1G células => $N = 2^{30}$ => e = 30 bits
REM = barramento de endereços = 30 bits

b) Qual o tamanho mínimo do CI?

CI terá o tamanho necessário para endereçar toda a memória = REM = 30 bits

c) Qual o tamanho do barramento de endereços?

Barramento de endereços = REM = 30 bits

d) Qual o tamanho mínimo do RI?

O tamanho de RI deverá ser o tamanho de uma instrução
Cada instrução tem o tamanho de uma palavra = tamanho de célula
N = 1G células , T (total de bits) = 4G bytes ou 32Gbits , M (tamanho da célula) = ?
T = N x M => M = T / N => M = 32Gbits / 1Gcelulas => M = 32bits / célula
Concluindo, RI = tamanho de uma célula = 32 bits.

e) Qual o número máximo de códigos de operação?

Instrução = código de operação + operando
Operando = um endereço de uma célula, portanto terá que ter pelo menos 30 bits, então:
32bits = código de operação + 30bits => Código de operação = 2 bits
Com 2 bits poderemos ter até 2^2 = 4 operações diferentes

6. (0,7) Um computador possui instruções que são constituídas de dois campos: um para o código de operação e outro para o endereço de memória. Existem 30 códigos de operação diferentes e sua unidade central de processamento tem capacidade de endereçar 2048 células de memória. Cada célula de memória armazena 16 bits.

a) Calcule a capacidade mínima de armazenamento em bits do REM, considerando que os bits armazenados no REM são utilizados para endereçar uma célula de memória.

REM = Barramento de endereços, este terá a capacidade de endereçar 2048 células
Número de bits para endereço (N) = $\log_2$2048 células => $N = \log_2 2^{11}$ => N = 11 bits
REM = barramento de endereços = 11 bits

b) Calcule o número de bits que devem ser transmitidos no barramento de endereços em cada acesso à memória.

REM = barramento de endereços = 11 bits

c) Calcule o número mínimo de bits utilizado para o campo código de operação.

Se tivermos 30 códigos diferentes, necessitaríamos de no mínimo 5 bits.
O código de operação com 5 bits permite até 32 códigos diferentes.

d) Indique o tamanho do RI (Registrador de Instruções) utilizando o que você calculou nos itens anteriores.

RI terá que ter o tamanho de uma instrução
O tamanho da instrução = código de operação + operando.
Tamanho da instrução = 5 + 11 = 16 bits => RI = 16 bits

e) Calcule a capacidade máxima de armazenamento da memória deste sistema em bits.

T = N x M => T = 2048 x 16 => T = 32 Kbits

f) Calcule o número de células que uma instrução ocupa.

Como cada instrução tem 16 bits, será necessária 1 célula para armazenar 1 instrução.

g) Calcule a capacidade mínima em bits do CI (Contador de Instrução), considerando que os bits armazenados no CI são utilizados para endereçar a primeira célula de uma instrução armazenada na memória.

CI terá o tamanho mínimo para endereçar qualquer posição da memória.
Neste caso CI = REM = 11bits

7. (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) LDA 10
a) RI <- Instrução lida
b) CI <- CI + 1
c) Decodificação do código de operação
d) Busca do operando na memória
- A Unidade de controle (UC) emite sinais para que o valor do campo operando = 10 seja transferido para o REM
*- A UC emite sinais para que o valor contido no REM seja transferido para o barram*ento de endereços
- A UC ativa a linha READ do barramento de controle
- O conteúdo da posição da memória, conforme endereço contido no barramento de endereços (10), é transferido através do barramento de dados para o RDM
- O conteúdo do RDM é transferido para o registrador acumulador (ACC <- RDM)

b) JZ 20
a) RI <- Instrução lida
b) CI <- CI + 1
c) Decodificação do código de operação
d) UC emite sinal para transferir conteúdo do acumulador para UAL
- UAL <- ACC
e) Executa operação de comparação
e.1) Resultado = verdadeiro, isto é, ACC = 0
CI <- Operando (CI <- 20)

8. (0,3) Considere a máquina apresentada na aula 4. Uma variável X está armazenada no endereço 20 da memória e uma variável Y está armazenada no endereço 15 de memória. Considere a seguinte expressão: Y=X+2, que significa que o valor da variável X será adicionado de 2 e armazenado na variável Y. Traduza esta expressão em um programa composto das instruções estudadas na aula 4.

Seja X uma variável armazenada no endereço 20
Seja Y outra variável armazenada no endereço 15

Para a solução nos basearemos nas seguintes instruções da aula 4

```
LDA Op. => ACC <- (Op.).
STR Op. => (Op.) <- ACC
ADD Op. => ACC <- ACC + (Op.)
```

OBS: Como as instruções acima trabalham com operandos que correspondem a endereços de memória, o valor a ser acrescido (2) terá que estar em algum endereço também de memória. Consideremos o endereço 10 contendo o conteúdo 2.

Para Atender a expressão Y = X + 2 teremos a seguinte solução

```
LDA 20 => ACC <- (20)
ADD 10 => ACC <- ACC + (10)
STR 15 => (15) <- ACC
```

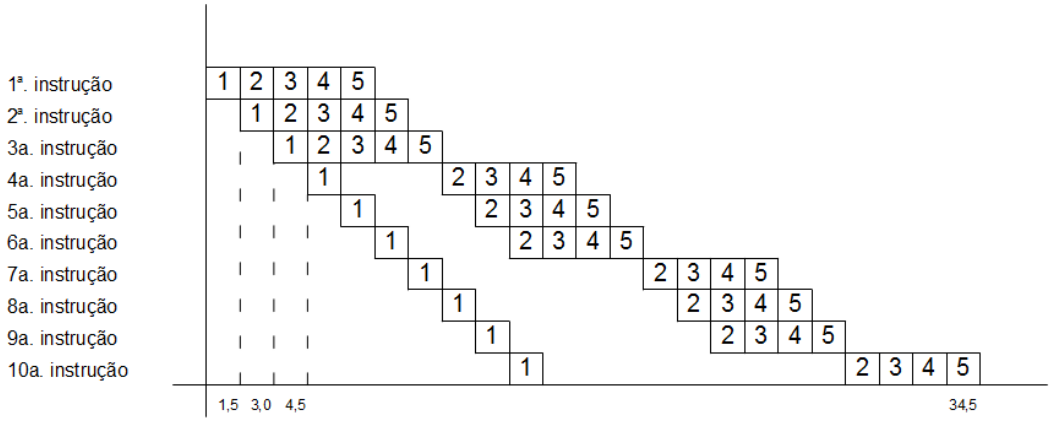
9. (1,0) Considere uma máquina cujo relógio possui uma freqüência de 2 GHZ e um programa no qual são executadas 10 instruções desta máquina.

Tempo de um ciclo de relógio = 1/2.000.000.000 = 0,000 000 000 5 seg ou 0,5 ns (nanosegundos)

a) Calcule o tempo para executar este programa, considerando que cada instrução é executada em 6 ciclos de relógio e a execução de uma instrução só se inicia quando a execução da instrução anterior é finalizada.

Tempo de execução de 1 instrução = 6 ciclos de relógio = 6 x 0,5ns = 3ns
*10 instruções executadas sequencialmente = 10 x 3ns = **30ns***

b) Uma nova implementação dessa máquina utiliza um pipeline de 5 estágios, todos de duração igual a 3 ciclos de relógio. Calcule o tempo para executar este programa, considerando que existe conflito entre os estágios 2 e 5, ou seja, estes estágios não podem ser executados simultaneamente.



Tempo para um estágio = 3 ciclos de relógio = 3 x 0,5ns = 1,5ns
Tempo de execução do programa=23 estágios=23 x 1,5 = 34,5ns

c) Uma nova implementação dessa máquina utiliza um pipeline de 5 estágios, todos de duração igual a 3 ciclos de relógio. Calcule o tempo para executar este programa, considerando que não existem conflitos de qualquer tipo.

Tempo para um estágio = 3 ciclos de relógio = 3 x 0,5ns = 1,5ns
Para execução da 1ª instrução = 5 estágios x 1,5ns = 7,5ns
Para execução das instruções posteriores = tempo de 1 estágio devido ao pipeline = 1,5ns
*Tempo total para execução das 10 instruções = 7,5ns + 9 x 1,5ns = **21ns***

10. (1,0) Explique detalhadamente como funciona o barramento PCI Express. Indique sua fonte de consulta (sugestão para consulta: www.clubedohardawre.com.br)

Texto retirado: http://www.clubedohardware.com.br/artigos/Barramento-PCI-Express/1060/3

O PCI Express se caracteriza com sendo um barramento serial ao contrário do PCI que é paralelo. Antes, os circuitos eletrônicos eram mais lentos e a forma de fazer com que os barramentos ficassem mais rápidos era adicionar mais trilhas ao barramento e transmitir vários bits de cada vez. O barramento paralelo operando em altas freqüências criava ruído eletromagnético e fazia ocorrer problemas de sincronismo, devido ao grande número de trilhas em paralelo, fatores que limitavam a freqüência e distâncias destes barramentos.

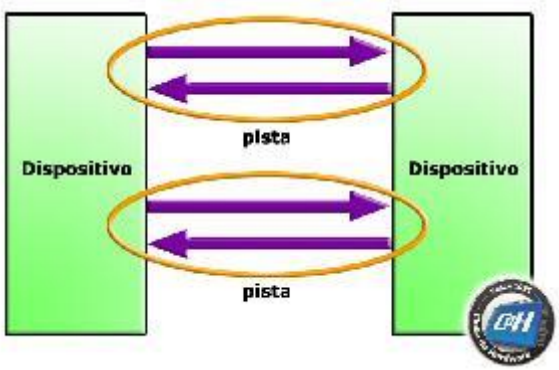
Com o avanço tecnológico, as comunicações seriais, onde os bits são transmitidos um após o outro em uma única trilha, alcançaram altas taxas de transmissão e permitiram eliminar o problema do ruído e interferência.

Outra característica fundamental do PCI Express é que ele é um barramento ponto a ponto, onde cada periférico possui um canal exclusivo de comunicação com o chipset. No PCI tradicional, o barramento é compartilhado por todos os periféricos ligados a ele, o que pode criar gargalos.

O barramento PCI Express é um barramento serial trabalhando no modo full-duplex. Os dados são transmitidos nesse barramento através de dois pares de fios chamados pista. Cada pista permite obter taxa de transferência máxima de 250 MB/s em cada direção, quase o dobro da do barramento PCI.

*O barramento PCI Express pode ser construído combinando várias pistas de modo a obter maior desempenho (vide figura abaixo). Podemos encontrar sistemas PCI Express com 1, 2, 4, 8, 16 e 32 pistas, que equivalem ao x1, x2, x4, x8, x16 e 32x respectivamente. Por exemplo, a taxa de transferência de um sistema PCI Express com 8 pistas (x8) é de 2 GB/s (250 * 8).*

O X1 (ou 1x) possui taxa de 250MB/s, o X2 de 500MB/s, o X4 de 1000MB/s, o X16 de 16MB/s e o X32 de 8000 MB/s. "



(imagem retirada do site: www.clubedohardware.com.br em 19/03/08)

11. (1,0) Descreva detalhadamente o funcionamento de uma unidade de controle microprogramada e uma unidade implementada em hardware.

Unidade de controle microprogramada

A unidade de controle microprogramada é utilizada para se desenvolver a implementação de um conjunto de instruções que apresenta muita complexidade para ser implementado somente em hardware. A execução de uma instrução é composta da execução de microinstruções referentes a ela e a unidade de controle microprogramada é projetada de modo a executar estas microinstruções. Ela é composta por: Memória de controle, Contador de microprograma e Sequenciador.
A Memória de controle armazena as microinstruções que compõem uma instrução e o Contador de microprograma armazena a localização da próxima microinstrução a ser executada. O Sequenciador é o componente que controla a sequencia de execução das microinstruções, informando o local da próxima microinstrução que deve ser executada e armazenada no Contador de microprograma.

Unidade de controle por hardware

Em uma unidade controlada por hardware, o seu desenvolvimento consiste essencialmente no projeto de circuitos combinatórios. Os sinais lógicos de entrada na unidade devem ser transformados em um conjunto lógico de sinais que controlam a execução da instrução. Para implementar a unidade, necessita-se derivar, para cada sinal de controle a ser gerado para que cada instrução seja executada de forma correta, uma expressão booleana que defina esse sinal em função dos sinais de entrada referentes à instrução.

Nesta unidade de controle, as microinstruções serão executadas diretamente pelo hardware.