AD1 - Organização de Computadores 2011.2
Data de entrega: 20/08/2011
Atenção:

1. **ADS enviadas pelo correio devem ser postadas cinco dias antes da data final de entrega estabelecida no calendário de entrega de ADs.**

2. **Como a avaliação a distância é individual, caso seja constatado que provas de alunos distintos sejam 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 3G células, cada uma capaz de armazenar uma palavra de 12 bits.

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

N = quantidade de células = 3G células = 3 x 2^{30} = 3.221.225.472 células
Maior endereço em decimal = N – 1 = 3.221.225.472 – 1 = ***3.221.225.471***

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

Observe,
um barramento de 30 bits atenderia no máximo a uma memória de até 2^{30} células = 1G células
um barramento de 31 bits atenderia no máximo a uma memória de até 2^{31} células = 2G células
um barramento de 32 bits atenderia no máximo a uma memória de até 2^{32} células = 4G células
Concluindo, para atender a especificação do computador hipotético, a de ter uma memória máxima de 3G células, deveremos ter um barramento de endereços de no mínimo ***32 bits.***

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

Barramento de endereços = REM = 32 bits
RDM = barramento de dados, como o barramento de dados = tamanho da palavra que será transferida durante um processo de leitura/escrita => RDM = 12 bits

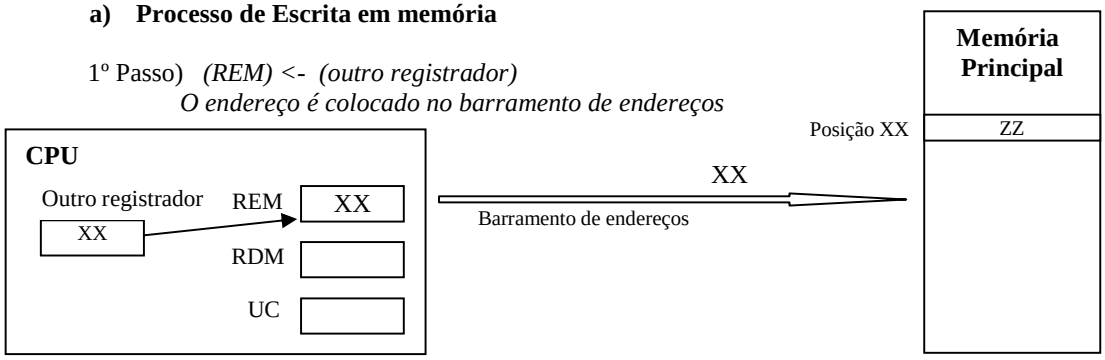
d) Qual é o número máximo de bits que pode existir na memória?

Como M = quantidade de bits em uma célula, neste computador, cada célula = palavra
então M = 12bits e N = 3 x 2^{30}
Total de bits da memória = T = N x M
T = N x M = 3 x 2^{30} x 12 = 36 G bits

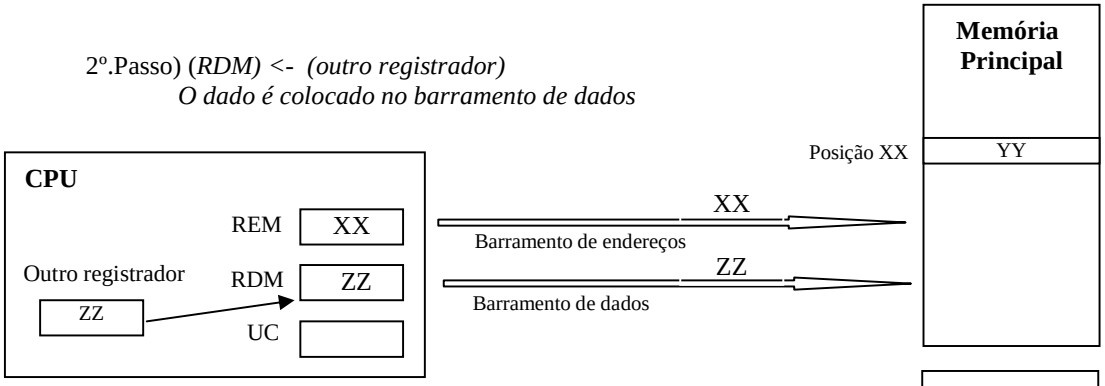
2. (1,0) Descreva e esquematize graficamente passo a passo as operações de leitura da memória e de escrita na memória, indicando como os registradores RDM e REM são utilizados e como a unidade de controle gera os sinais necessários.

a) Processo de Escrita em memória

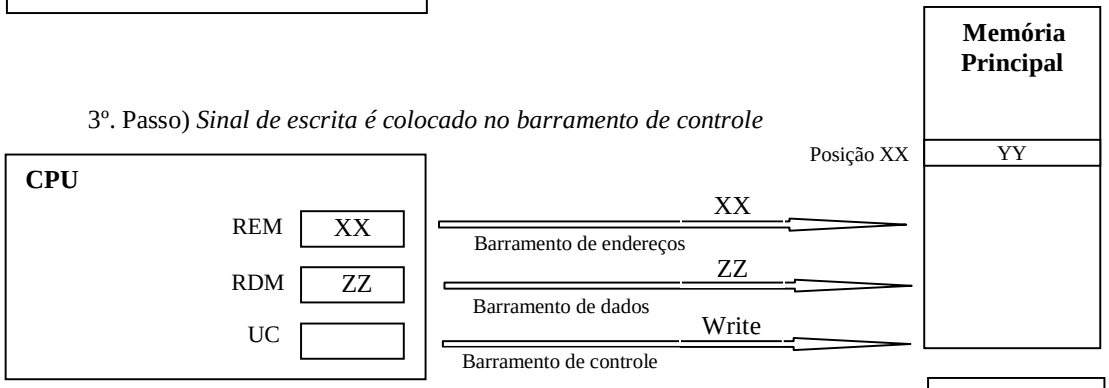
1º Passo) *(REM) <- (outro registrador)*
O endereço é colocado no barramento de endereços



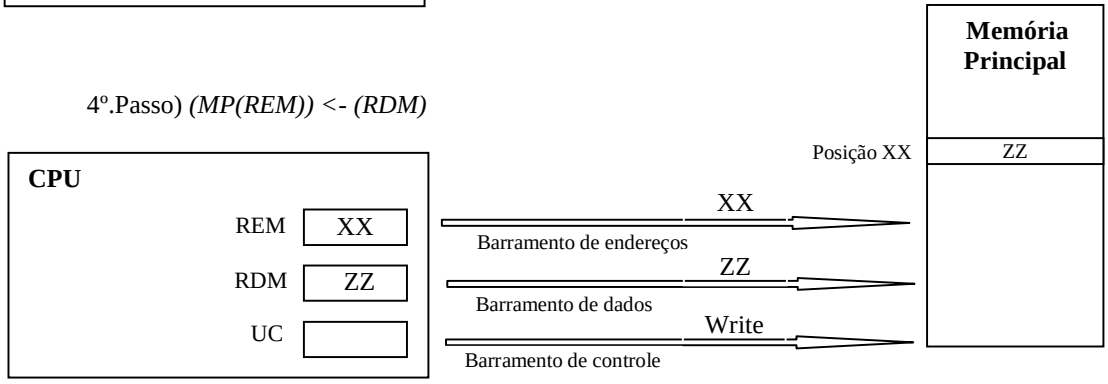
2º.Passo) (*RDM) <- (outro registrador)*
O dado é colocado no barramento de dados



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

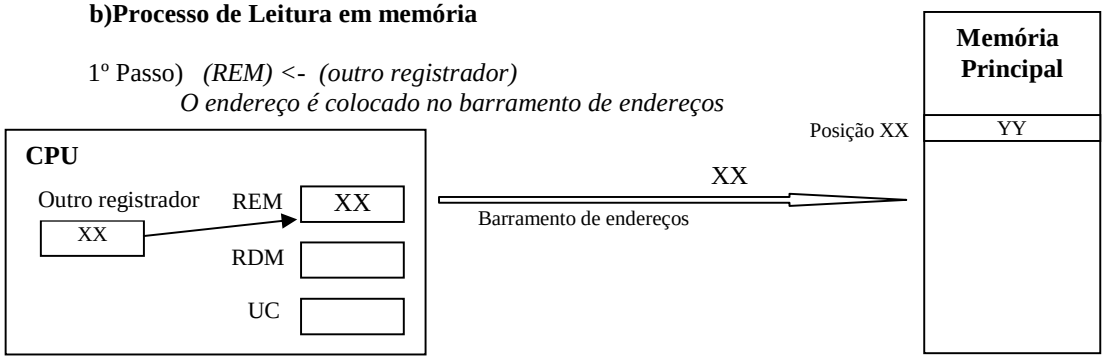


4º.Passo) *(MP(REM)) <- (RDM)*

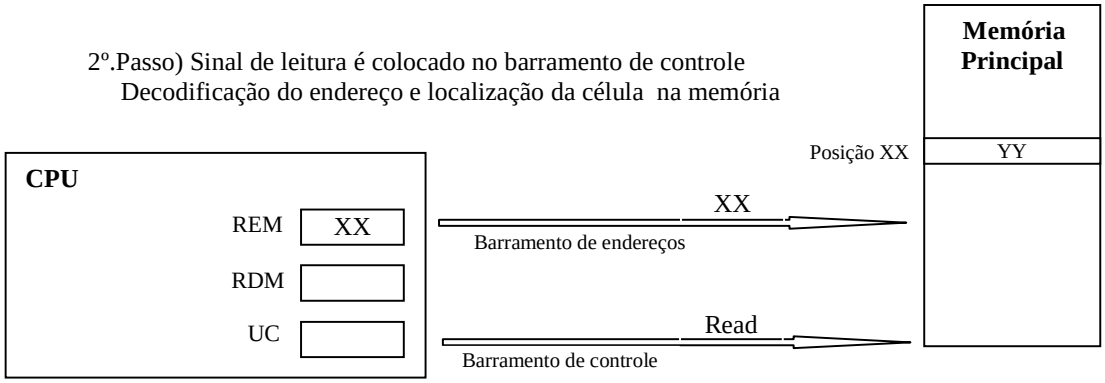


b)Processo de Leitura em memória

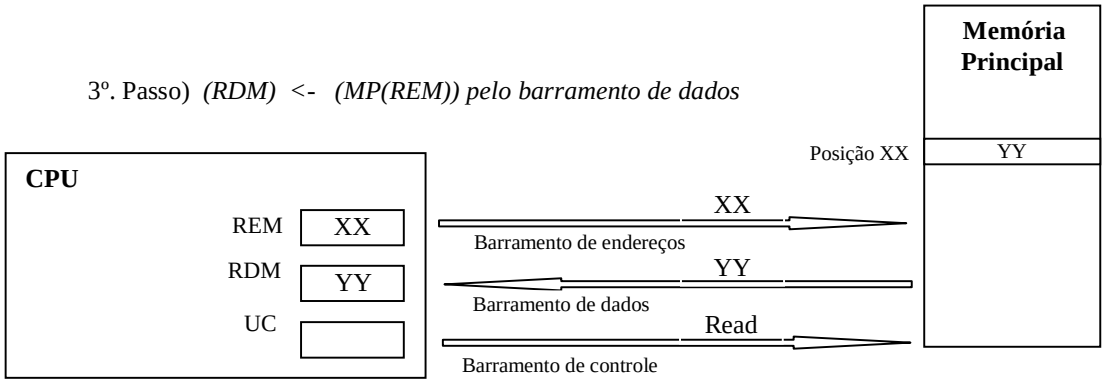
1º Passo) *(REM) <- (outro registrador)*
O endereço é colocado no barramento de endereços



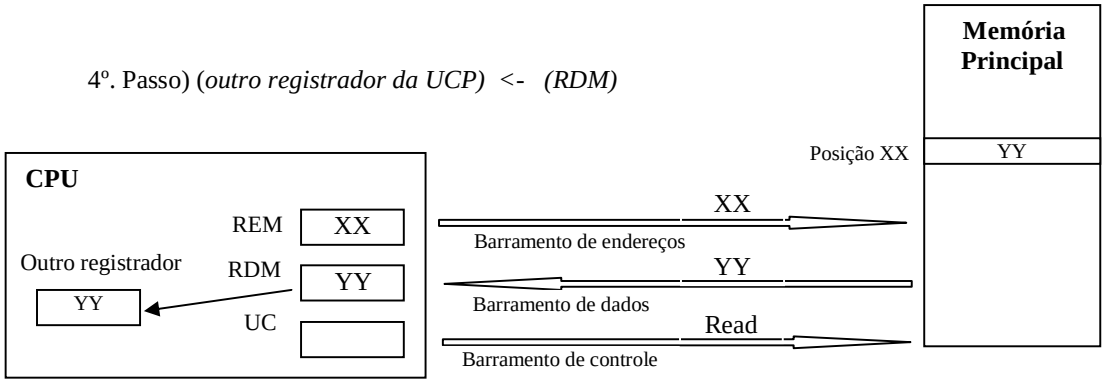
2º.Passo) Sinal de leitura é colocado no barramento de controle
Decodificação do endereço e localização da célula na memória



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*

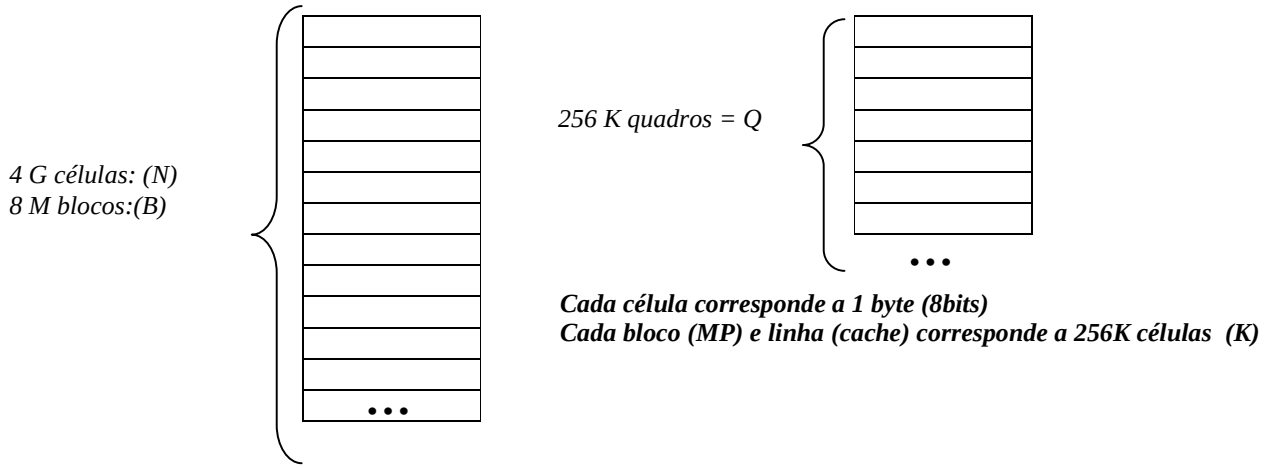
⇒ *Sendo N o tamanho endereçável da memória e K que é a quantidade de células por blocos temos:*
N = 4 G células e K = 512 células / blocos o total de blocos da MP (B) será:
Total de blocos: B = N / K => B = 4 G células / 512 células/bloco => B = 8M blocos

Memória Cache

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

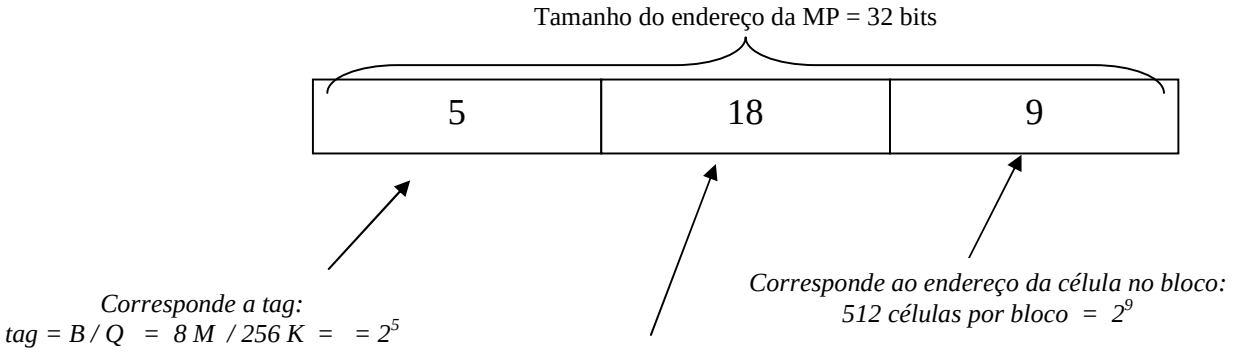
⇒ *Tamanho da memória cache em blocos = 256K linhas que podem armazenar 256 K blocos*

⇒ *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)

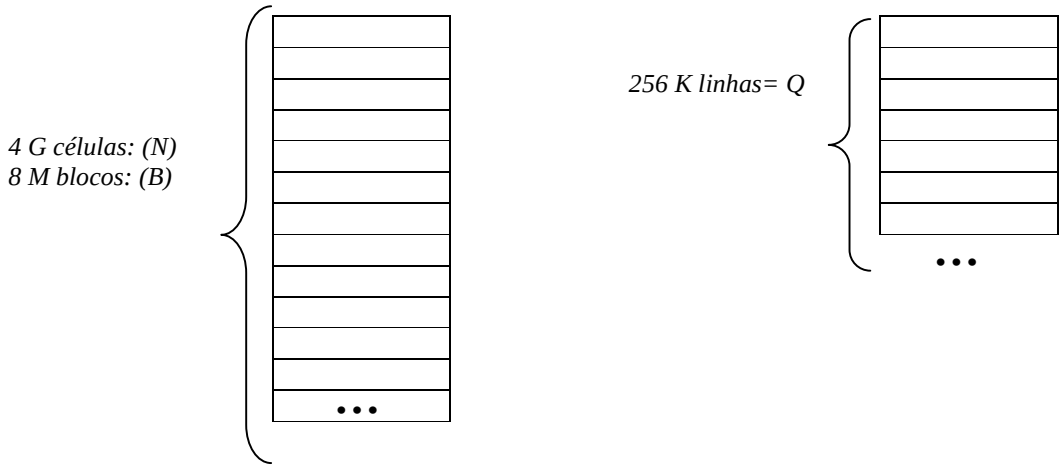
Para endereçarmos toda a MP precisamos da seguinte quantidade de bits (E)
sendo $N = 2^E$ => N = 4G células => $N = 2^{32}$ => E = 32 bits



b) Mapeamento totalmente associativo.

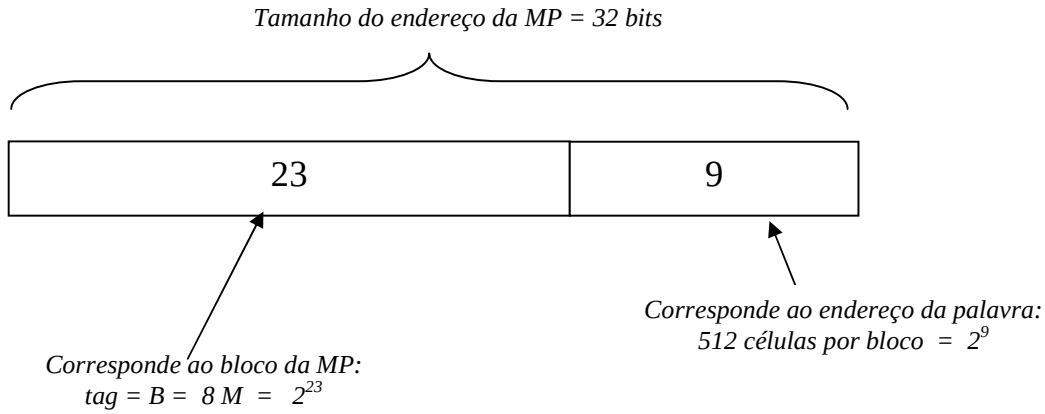
Memória Principal
=> N = 4G células
=> K = 512 células por bloco
=> B = 8 M blocos

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

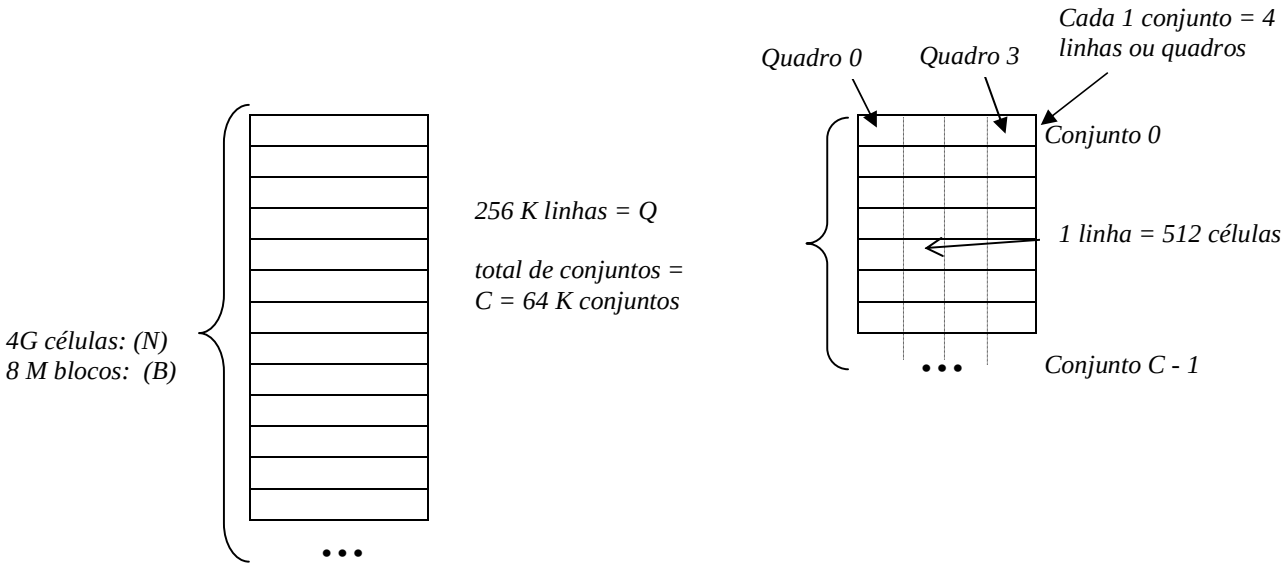
Como o bloco pode ser alocado em qualquer posição da memória cache a tag indicará qual dos blocos da MP está alocado naquela posição da memória cachê



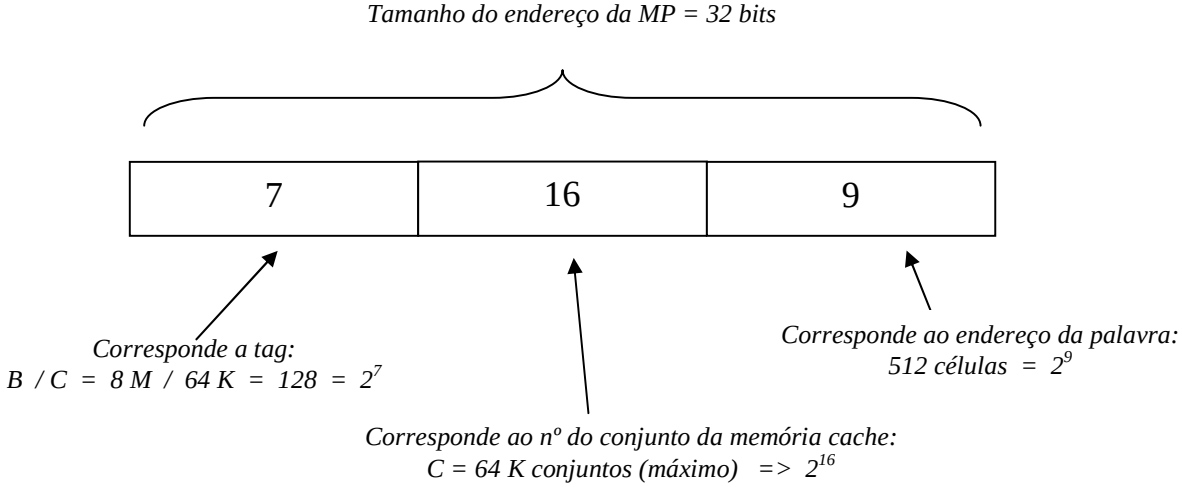
c) Mapeamento associativo por conjunto, onde cada conjunto possui quatro linhas, cada uma de um bloco.

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***

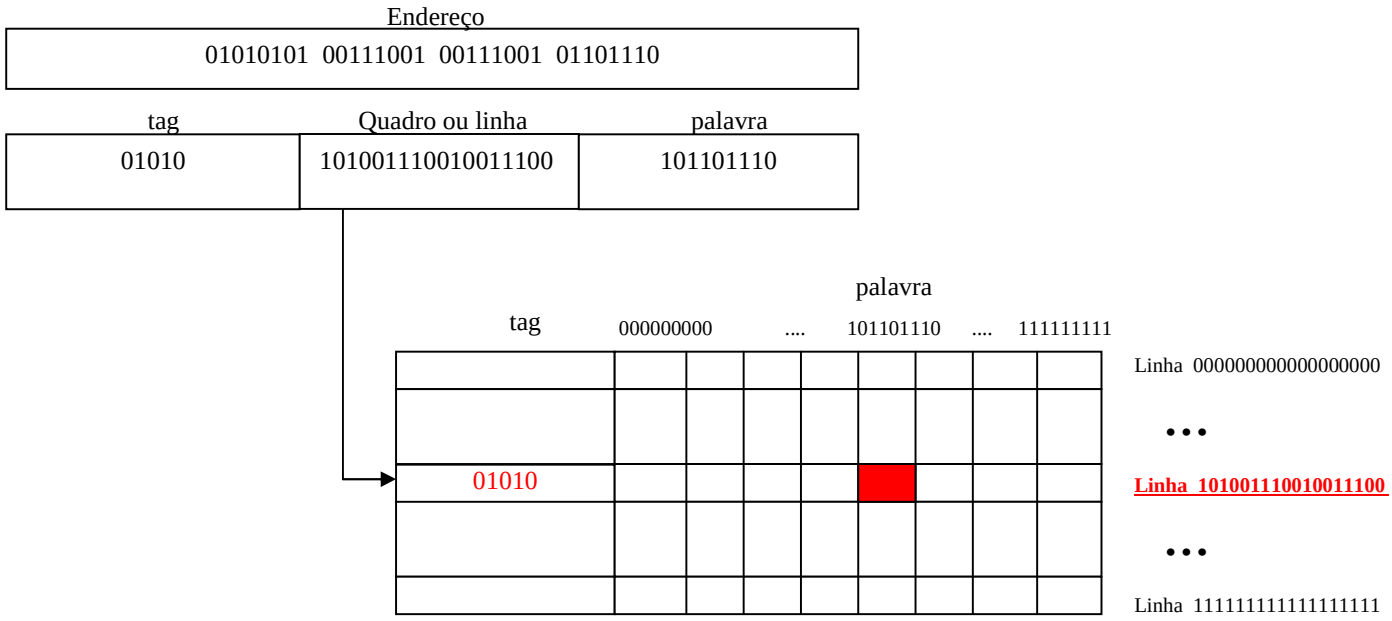


Para endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 32 bits



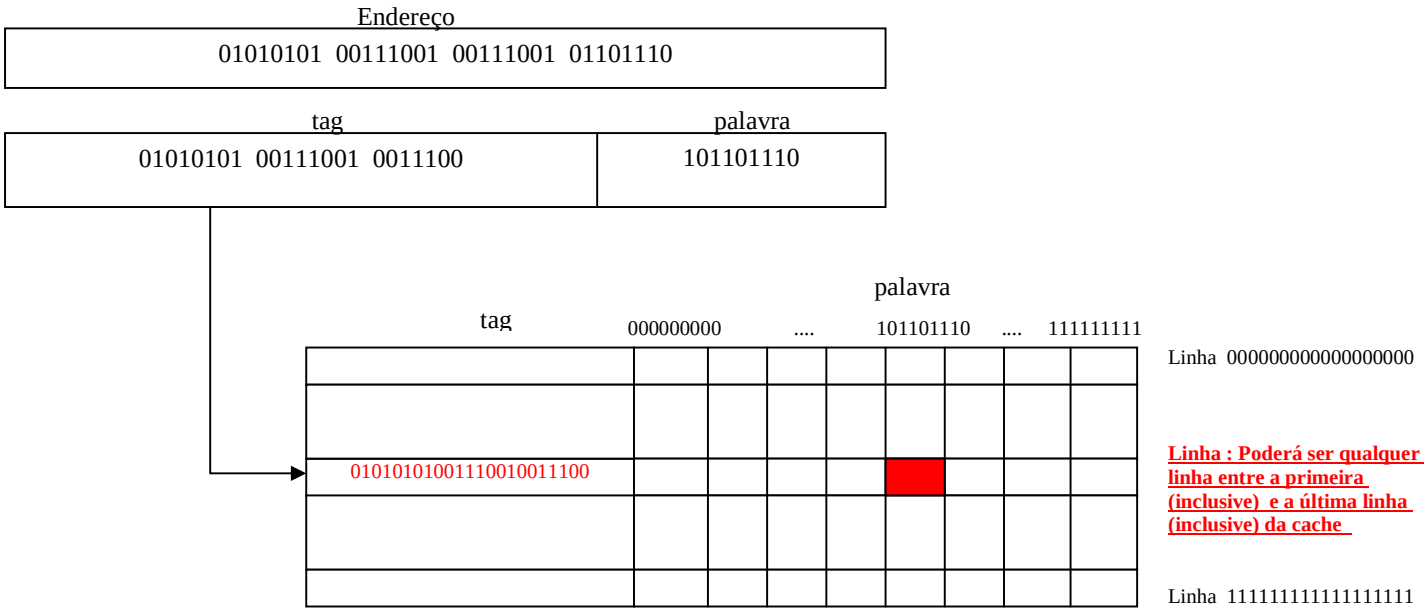
d) Em que linha, para cada um dos mapeamentos dos itens anteriores, estaria contido o byte armazenado no seguinte endereço da MP: 01010101 00111001 00111001 01101110.

i) Para o mapeamento direto



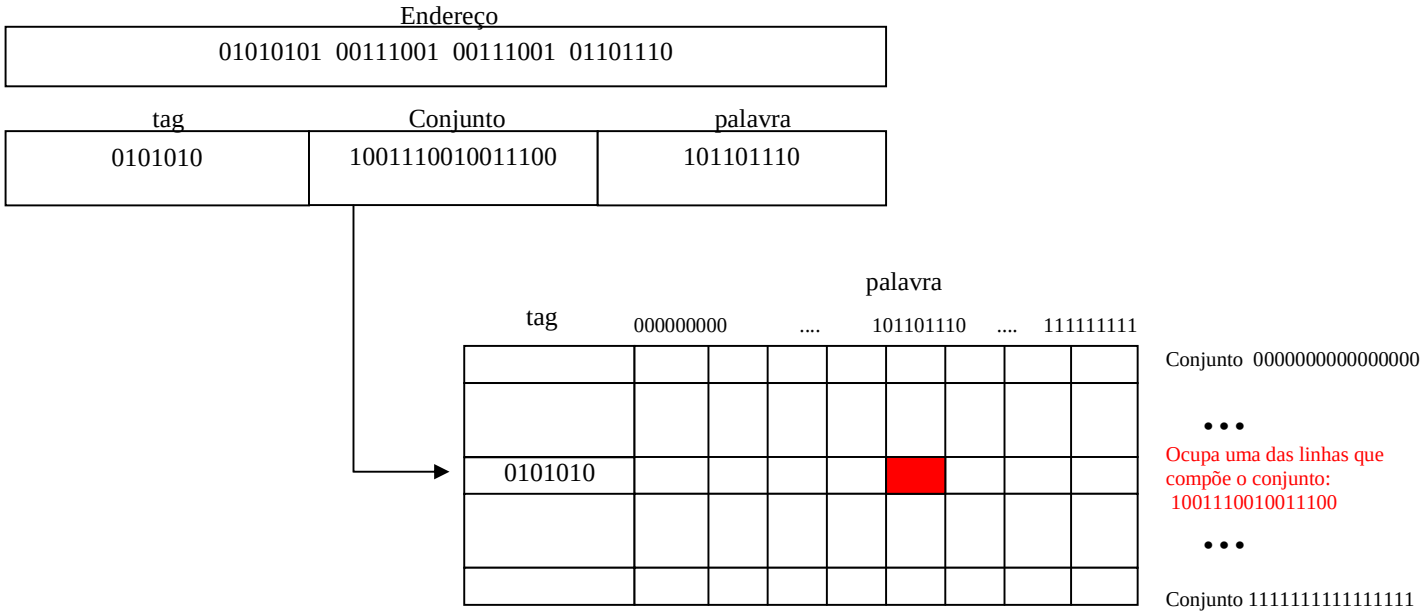
OBS: Com o endereço fornecido, será conferido o campo tag do endereço com a tag da linha indicada, caso não sejam iguais, teremos um cache miss, ou seja, a palavra não está na memória.

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.

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.

4. (1,0) Explique em detalhes a organização hierárquica do subsistema de memória e a localização física nos computadores atuais. Descreva como é organizada a memória cache no I3, I5 e I7 e quais os registradores que eles possuem.

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, possui capacidade bem superior ao 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 eram divididas 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. Nivel: 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.

Organização da cache do I3, I5 e I7

Os primeiros processadores da linha I3, I5 e inicialmente nos I7, eram baseados na arquitetura Nehalem. A organização das caches desta arquitetura consistia em: L1 de 64KB (32Kb para instruções e 32 KB para dados), L2 individuais para cada núcleo com cerca de 256Kb e L3 compartilhado com cerca de 8MB (nos primeiros modelos lançados).

Em 2011 a Intel passou a utilizar uma nova arquitetura, a Sandy Bridge, introduzida nos processadores I7 e a ser usada nos Core I5 e Core I3. A cache desta arquitetura consistia em um novo cachê de micro instruções decodificadas (cachê L0, capaz de armazenar 1.536 microinstruções, o que equivale a mais ou menos 6KB), cachê L1 de 64Kb (32KB para instruções e 32KB para dados), L2 (cache intermediário) com 256KB, e L3 (cache de último nível) compartilhado entre os núcleos do processador e o processador gráfico.

Registradores Intel I3, I5 e I7

Registradores de uso geral (32bits) EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP
Registradores de segmento (16 bits): CS, DS, SS, ES, FS, GS
Registrador de controle e status do programa (32 bits): EFLAGS
Ponteiro de instrução (32bits): EIP
Registradores de gerenciamento de memória: Global Descriptor Table Register (GDTR – 64 bites), Local Descriptor Table Register (LDTR), Interrupt Descritor Table Register: (IDTR - 64 bits), Task Register: (TR - 64 bits)
Registradores de controle: CR0, CR1, CR2, CR3, CR4 e CR8 (64 bits)
Registrador de controle estendido: XCR0 (64 bits)

5. (1,0) Considere uma máquina com arquitetura semelhante àquela apresentada em aula. Pode-se endereçar no máximo 1M 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 3M bytes. 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 1M células = N
N = 1M células => $N = 2^{20}$ => e = 20 bits
REM = barramento de endereços = 20 bits

b) Qual o tamanho mínimo do CI?

CI terá o tamanho necessário para endereçar uma célula de memória = REM = 20 bits

c) Qual o tamanho do barramento de endereços?

Barramento de endereços = REM = 20 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 = 1M células , T (total de bits) = 3M bytes ou 24Mbits , M (tamanho da célula) = ?
T = N x M => M = T / N => M = 24Mbits / 1Mcelulas => M = 24bits / célula
Concluindo, RI = tamanho de uma célula = 24 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 20 bits, então:
24bits = código de operação + 20bits => Código de operação = 4 bits
Com 4 bits poderemos ter até 2^4 = 16 códigos de operação diferentes

6. (1,0) Um computador possui instruções que são constituídas de dois campos: um para o código de operação e outro para o endereço de memória. Existem 62 códigos de operação diferentes e sua memória tem capacidade de endereçar 1024 células de memória. Cada célula de memória armazena 8 bits.

a) Calcule a capacidade mínima de endereçamento 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 1024 células = N
N = 1024 células => $N = 2^{10}$ => e = 10 bits
REM = barramento de endereços = 10 bits

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

REM = barramento de endereços = 10 bits

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

Se tivermos 62 códigos diferentes, necessitaríamos de no mínimo 6 bits.
O código de operação com 6 bits permite até 64 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 = 6 + 10 = 16 bits => RI = 16 bits

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

T = N x M => T = 1024 x 8 => T = 8 Kbits

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

Como cada instrução tem 16 bits, serão necessárias 2 células 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 uma célula da memória.
Neste caso CI = REM = 10bits

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) STR 3
a) RI <- Instrução lida
b) CI <- CI + 1
c) Decodificação do código de operação

d) Busca do operando na memória
- A UC emite sinais para que o valor do campo operando = 3 seja transferido para o REM
- A UC emite sinais para que o valor do registrador acumulador seja transferido para a RDM
- A UC ativa a linha WRITE do barramento de controle
- Conteúdo do RDM é transferido, através do barramento de dados para o endereço 3 da memória, endereço este transferido do REM à memória pelo barramento de endereços.

b) JN 10

a) RI <- Instrução lida
b) CI <- CI + 1
c) Decodificação do código de operação
d) UC emite sinal para transferir conteúdo acumulador para UAL
- UAL <- ACC
e) Executa operação de comparação
e.1) Resultado = verdadeiro, isto é, ACC < 0
CI <- Operando (CI <- 10)
d) Inicia o procedimento de leitura da instrução contida no endereço que consta em CI

8. (0,5) Considere um computador com uma memória de 256 células em que cada célula contém 8 bits. A tabela a seguir ilustra o conteúdo de alguns endereços onde a coluna da esquerda indica o endereço e a da direita o conteúdo da memória.

Endereço	Contéudo
0	00000000
1	10101010
2	11111111
164	11110000
165	10000000
188	10101000
201	10000000
254	11100011
255	11110010

a) Calcule a capacidade total de armazenamento da memória em bits.

Capacidade total de armazenamento em bits (T) = total de células (N) x bits por célula (M)
N = 256 células, M = 8 bits / célula
T = 256 x 8 = 2048 bits

b) Supondo que no início de um ciclo de busca e execução de instrução o conteúdo do CI (Contador de Instrução) seja 164 e que cada instrução ocupa uma única célula, indique a instrução que será executada.

A instrução que será executada será a do endereço 164: 11110000

c) Supondo que o conteúdo do REM (Registrador de Endereços de Memória) tenha o valor 254 e que um sinal de leitura seja enviado da UCP para a memória, indique o conteúdo do RDM (Registrador de Dados da Memória) ao final do ciclo de leitura.

O conteúdo será o do endereço 254: 11100011

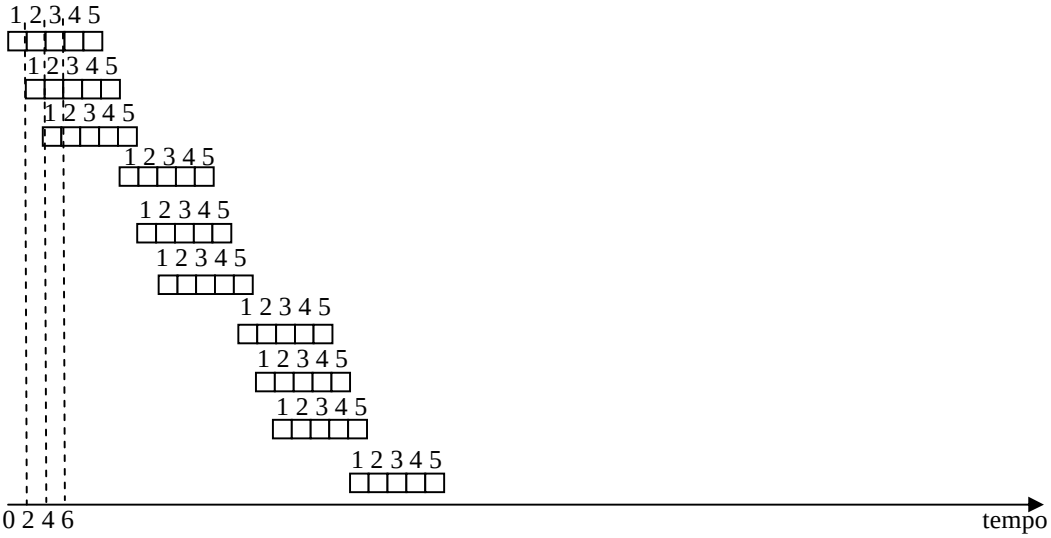
9. (1,0) Considere uma máquina cujo relógio possui uma freqüência de 1 GHZ e um programa no qual são executadas 10 instruções desta máquina.

Tempo de um ciclo de relógio = 1/1.000.000.000 = 0,000 000 001 seg ou 1ns (nanosegundos)

a) Calcule o tempo para executar este programa, considerando que cada instrução é executada em 8 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 = 8 ciclos de relógio = 8 x 1ns = 8ns
*10 instruções executadas sequencialmente = 10 x 8ns = **80ns***

b) Uma nova implementação dessa máquina utiliza um pipeline de 5 estágios, todos de duração igual a 2 ciclos de relógio. Calcule o tempo para executar este programa, considerando que existe conflito entre os estágios 1 e 4, ou seja, estes estágios não podem ser executados simultaneamente.



Tempo para um estágio = 2 ciclos de relógio = 2 x 1ns = 2ns
Para execução da 1ª instrução = 5 estágios x 2ns = 10ns
Verificamos pela figura acima que quando uma instrução alcança o 4º. Estágio, não poderemos executar o 1º.Estágio até todas as instruções já terem executado o 4º. Estágio. Desta forma separamos a execução em blocos. Cada bloco terá 3 instruções, e somente quando a 3ª. Instrução do bloco estiver no 5º. estágio é que poderemos executar a 1ª instrução do próximo bloco (sobreposição de 1 estágio entre execução de blocos sucessivos), então:
Bloco = tempo de 1 instrução + tempo de um estágio para execução de cada uma das outras 2 instruções
Bloco = 5estágios x 2ns + 2 x 2ns = 14ns
Tempo total será = 3 blocos de 3 instruções + 1 bloco com 1 instrução
Tempo total será = bloco 1 + bloco 2 + bloco 3 + 1 instrução – 3 tempos de sobreposição (1 estágio)
*Tempo total será = 14ns + 14ns + 14ns + 10ns – 3x2ns = **46ns***

c) Uma nova implementação dessa máquina utiliza um pipeline de 5 estágios, todos de duração igual a 2 ciclos de relógio. Calcule o tempo para executar este programa, considerando que não existem conflitos de qualquer tipo.

Tempo para um estágio = 2 ciclos de relógio = 2 x 1ns = 2ns
Para execução da 1ª instrução = 5 estágios x 2ns = 10ns
Para execução das instruções posteriores = tempo de 1 estágio devido ao pipeline = 2ns
*Tempo total para execução das 10 instruções = 10ns + 9 x 2ns = **28ns***

10. (0,5) Descreva a execução de uma operação de leitura em um barramento PCI.

Antes de descrever uma operação de leitura em um barramento PCI, segue abaixo um quadro com os sinais e as respectivas descrições (texto retirado de http://www.laercio.com.br/artigos/hardware/hard-013/hard-013b.htm).

AD0-AD31	*Barramento de dados e endereços multiplexados. No início de uma transferência, este barramento indica o endereço, e na fase seguinte, os dados. Como muitas transferências são feitas em modo burst, não existe queda de desempenho perceptível pelo fato de ser usado um único barramento para dupla função.*
C/BE0-C/BE3	*Durante a fase de endereço, esses 4 sinais indicam o comando a ser realizado (leitura, escrita, etc.). Na fase de dados, esses 4 bits indicam quais bytes dos 32 bits do barramento de dados devem ser levados em conta. Isso permite, por exemplo, acessar bytes individuais, apesar do barramento de dados ter 32 bits.*
FRAME	*O Bus Master ativa este sinal para dar início a um ciclo de transferência.*
IRDY	*Initiator Ready. Indica que o Master está pronto para ler ou enviar dados. Quando este sinal não é ativado, o Target irá esperar tantos wait states quanto forem necessários.*
TRDY	*Target Ready. Indica que o Target está pronto para receber dados (escrita) ou que o dado lido já está disponível (leitura). Quando este sinal não é ativado, o Master irá gerar tantos wait states quando forem necessários.*
DEVSEL	*Ativado pelo Target quando reconhece o seu endereço. Desta forma o Master pode saber se o dispositivo Target está ativo ou presente no barramento.*
REQ	*Requisição enviada ao Bus Arbitrer, para que o dispositivo se torne Bus Master. Cada dispositivo tem seu próprio sinal REQ.*
GNT	*Grant. Através deste sinal o Bus Arbitrer indica ao dispositivo solicitante que o barramento está liberado, permitindo assim que se torne Bus Master. Cada dispositivo tem seu próprio sinal GNT.*
INTA, INTB, INTC, INTD	*São linhas de interrupção a serem usadas pelos dispositivos PCI. Cada dispositivo e cada slot é ligado a um desses sinais, que podem ser compartilhados, ou seja, uma mesma linha INT pode ser usada por mais de um slot. O padrão PCI prevê o compartilhamento de interrupções.*
AD32-AD63	*Continuação do barramento de dados e endereços nos slots PCI de 64 bits.*
C/BE4-C/BE7	*Continuação do barramento de comando e habilitação de bytes nos slots PCI de 64 bits.*
REQ64	*Requisição de transferência de 64 bits.*
ACK64	*Indica que o Target está apto a realizar transferência de 64 bits.*

As operações são sincronizadas pelo clock. Durante o período em que o barramento AD traz endereços, temos a fase de endereçamento (address phase). Uma vez determinado o endereço, são feitas as transferências de dados, entrando então na fase de dados (data phase). Wait states podem ser gerados por solicitação do Target ou do próprio Master, através dos controles IRDY e TRDY. Os eventos que ocorrem nesta transferência são os seguintes:

I. O Bus Master inicia a transferência ativando o sinal FRAME, que permanece ativo até que o Target termine sua última fase de dados. O Master também fornece o endereço (AD0-AD31) e o comando (C/BE0-C/BE3).

II. O Target reconhece seu endereço e prepara-se para fornecer os dados.

III. O Master para de indicar o comando nas linhas C/BE e passa a indicar os controles habilitadores dos bytes desejados.

IV. O Target ativa a linha DEVSEL para indicar que foi endereçado, fornece o primeiro dado e ativa a linha TRDY para indicar que o dado está pronto. Wait states podem ser gerados se necessário, bastando retardar a ativação de TRDY.

V. O Master lê o dado e altera as linhas C/BE, se necessário.

VI. O Target desativa a linha TRDY enquanto busca o próximo dado, gerando mais um wait state. Isto pode ser necessário quando os circuitos do Target não são suficientemente velozes.

VII. Caso o Master ainda não esteja pronto para ler o próximo dado, ele desativa o sinal IRDY, gerando mais wait states que farão o Target manter os dados por mais ciclos.

VIII. Após receber o último dado, o Master finaliza a transferência, desativando a linha FRAME.

IX. O Target é desativado, liberando o barramento de dados e desativando os sinais TRDY e DEVSEL.

11. (0.5) Descreva 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 em projetar 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.

12. (0,5) Explique os dois métodos de formatar e usar uma microinstrução na arquitetura microprogramada: microinstruções verticais e microinstruções horizontais.

Em uma arquitetura microprogramada, a unidade de controle é especificada por um microprograma que consiste de uma seqüência 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 seqüência 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 possuem um bit para gerar cada um dos sinais de controle, como por exemplo, sinal para controlar uma linha de controle interna da UCP, sinal para controlar uma linha de barramento externo de controle, sinal para definir condição de desvio e endereço de desvio. Tem a vantagem de ser simples, 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.

Em microinstruções verticais, não utiliza-se um bit para cada sinal de controle, mas uma combinação de bits da instrução indica a ação que deve ser efetuada. Para isso, deve existir um decodificador extra que traduz este código em sinais de controle individuais que serão efetivamente ativados. Sua principal vantagem é reduzir o custo da Unidade de controle em função do menor tamanho da instrução. Tem como principal desvantagem o aumento do tempo de execução da instrução devido à necessidade da decodificação dos campos de cada microinstrução.