GABARITO DA AD1
Organização de Computadores 2017.2

1. (2,5) Considere uma máquina com arquitetura semelhante àquela apresentada em aula. Pode-se endereçar no máximo 8 M 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 32 códigos, e dois operandos, que indicam um endereço de memória e um registrador. Existem 13 registradores.

   a) Qual o tamanho mínimo do REM ? (0,3)
   REM = Barramento de endereços, este terá a capacidade de endereçar 8Mcélulas = N
   N = 8Mcélulas => $N = 2^{23}$ => e = 23 bits
   REM = barramento de endereços = ***23 bits***

   b) Qual o tamanho mínimo do CI ? (0,3)
   CI terá o tamanho mínimo necessário para endereçar toda a memória = REM = ***23*** **bits**

   c) Qual o tamanho do barramento de endereços ? (0,3)
   REM = barramento de endereços ***= 23 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 32 códigos diferentes (instruções). CodOper = 5 bits
   1º.operando corresponde a 1 endereço de memória = 23 bits
   2º.operando corresponde a um endereço de registrador, como são 13 registradores, são necessários 4 bits para endereçar todos os 13 registradores
   Instrução = CodOper + Operando1 + Operando2 => Instrução = 5 + 23 +4 = 32 bits
   O tamanho mínimo para RI deverá ser ***32 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) = 32 bits
   total de endereços da memória (N) = 8M células
   T = 32 bits/célula x 2^{23} células => T = 256 M bits (ou 32Mbytes)

   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

2) (2,1) 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 20
   a) RI <- Instrução lida
   b) CI <- CI + 1
   c) Decodificação do código de operação
   - *recebe os bits do código de operação*
   - *produz sinais para a execução da operação de gravação em memória*

d) Envio de sinais pela UC

- *A UC emite sinais para que o valor do campo operando (20) seja transferido para a REM*
- *Conteúdo do REM é transferido para o barramento de endereços.*
- *A UC emite sinais para que o valor do registrador acumulador seja transferido para a RDM*
- *Conteúdo do RDM é transferido para o barramento de dados.*
- *A UC ativa a linha WRITE do barramento de controle*

e) Armazenamento na MP

- *A MP armazena no endereço 20 (conteúdo do barramento de endereços) a palavra recebida através do barramento de dados*

b) LDA 226

a) RI <- Instrução lida
b) CI <- CI + 1
c) Decodificação do código de operação
- recebe os bits do código de operação
- produz sinais para a execução da operação de leitura em memória
d) Busca do operando na memória
- A UC emite sinais para que o valor do campo operando = 226 seja transferido para a 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
- Conteúdo da posição da memória, conforme endereço contido no barramento de endereços (226), é transferido através do barramento de dados para o RDM
- O conteúdo do RDM é transferido para o registrador acumulador (ACC <- RDM)

c) JP 410

a) RI <- Instrução lida
b) CI <- CI + 1
c) Decodificação do código de operação
- recebe os bits do código de operação
- produz sinais para a execução da operação de salto condicional
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 <- 410)
d) Inicia o procedimento de leitura da instrução contida no endereço que consta em CI

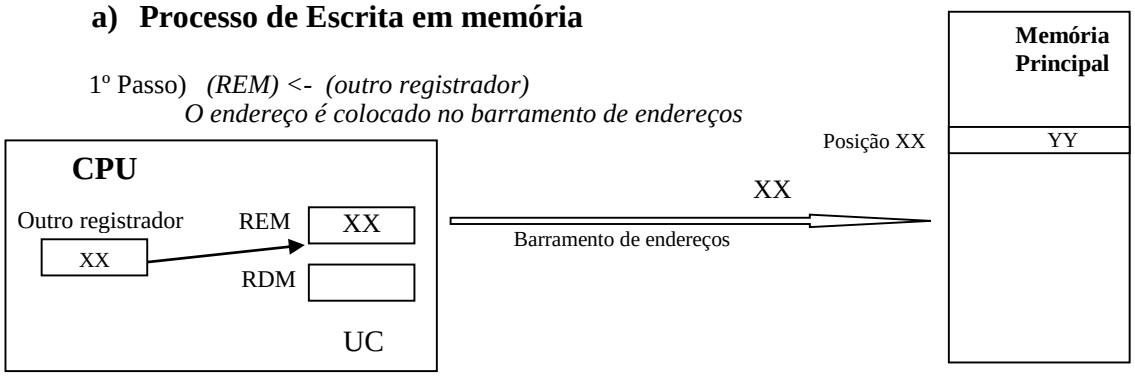
3) (1,4) Escreva um programa que utilize as instruções de linguagem de montagem apresentadas na aula 4 para executar o seguinte procedimento. O conteúdo da memória cujo endereço é 300 é lido e verifica-se se o seu valor é 0. Caso seu valor seja 0, o conteúdo de memória cujo endereço é 350 é somado ao conteúdo de memória cujo endereço é 150 e o resultado é armazenado no endereço 200. Caso contrário, o conteúdo de memória cujo endereço é 350 é subtraído do conteúdo de memória cujo endereço é 150 e o resultado é armazenado no endereço 200. Além de apresentar seu programa escrito em linguagem de montagem, apresente também o programa traduzido para linguagem de máquina.

Endereço (hexa)	***Instrução***	***Descrição***	***Linguagem Máquina (bin / hexa)***
000	*LDA 300*	*ACC <- (300)*	*(0001 0011 0000 0000 / 1300)*
0 01	*JZ 006*	*se ACC = 0, CI <- 006*	*(0110 0000 0000 0110 / 6006)*
002	*LDA 350*	*ACC <- (350)*	*(0001 0011 0101 0000 / 1350)*
003	*ADD 150*	*ACC <- ACC + (150)*	*(0011 0001 0101 0000 / 3150)*
004	*STR 200*	*(200) <- ACC*	*(0010 0010 0000 0000 / 2200)*
005	*JMP 009*	*CI <- 009*	*(1000 0000 0000 1001 / 8009)*
006	*LDA 150*	*ACC <- (150)*	*(0001 0001 0101 0000 / 1150)*
007	*SUB 350*	*ACC <- ACC - (350)*	*(0100 0011 0101 0000 / 4350)*
008	*STR 200*	*(200) <- ACC*	*(0010 0010 0000 0000 / 2200)*
009	*HLT*	*Encerra Procedimento*	*(0000 0000 0000 0000 / 0000)*

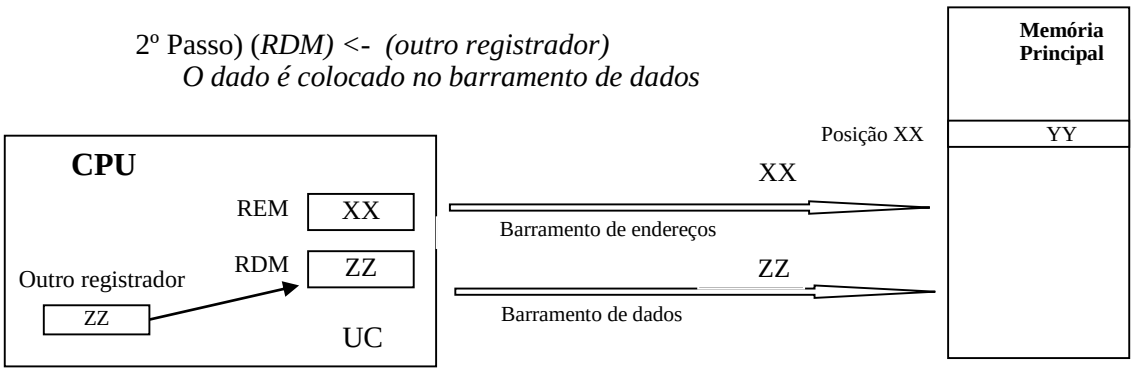
4) (1,0) Descreva e esquematize graficamente passo a passo as operações de leitura da memória e de escrita na memória, indicando como os registradores RDM e REM são utilizados e como a unidade de controle gera os sinais necessários.

a) Processo de Escrita em memória

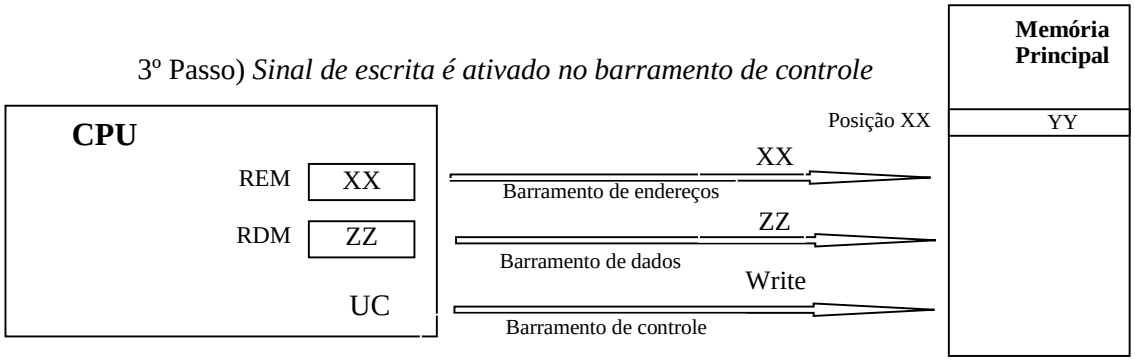
1º Passo) *(REM) <- (outro registrador)*
O endereço é colocado no barramento de endereços



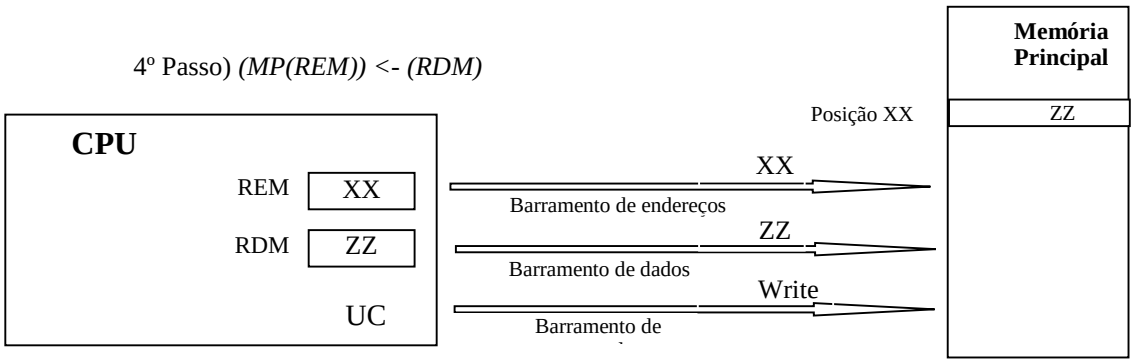
2º Passo) (*RDM) <- (outro registrador)*
O dado é colocado no barramento de dados



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

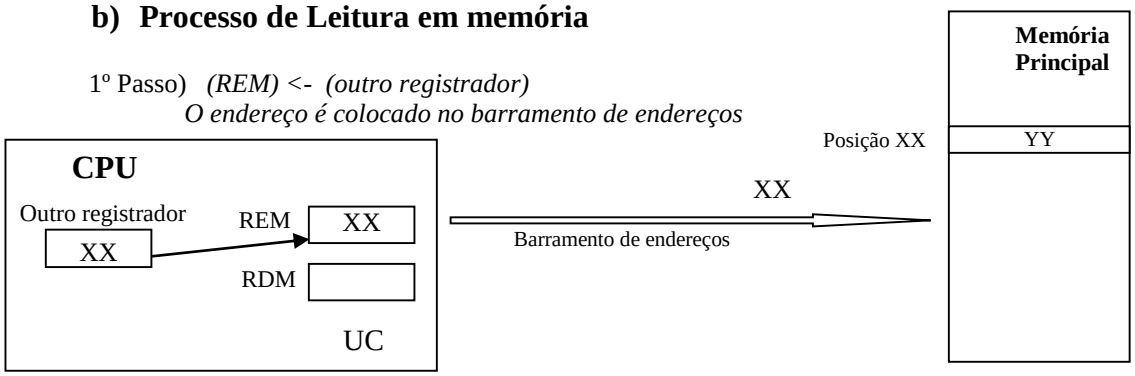


4º Passo) *(MP(REM)) <- (RDM)*

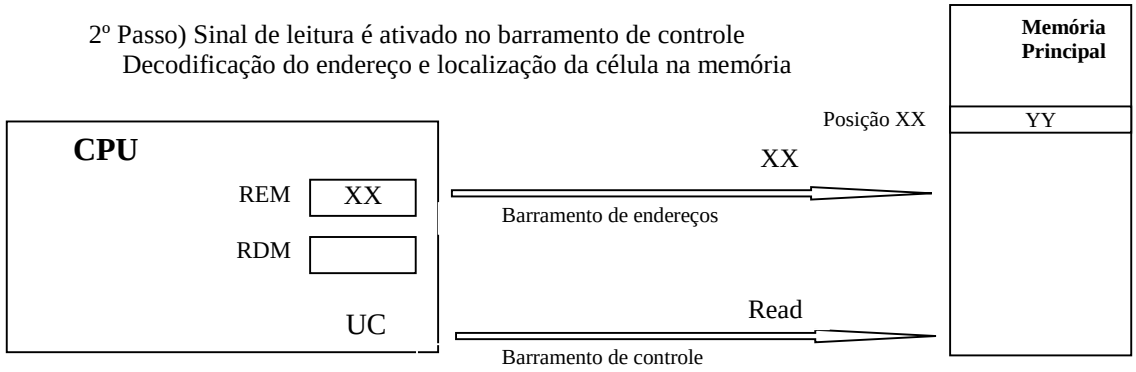


b) Processo de Leitura em memória

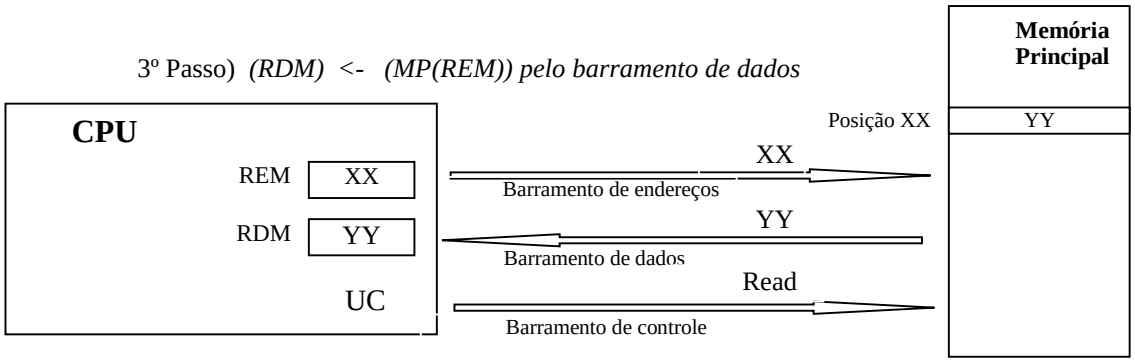
1º Passo) *(REM) <- (outro registrador)*
O endereço é colocado no barramento de endereços



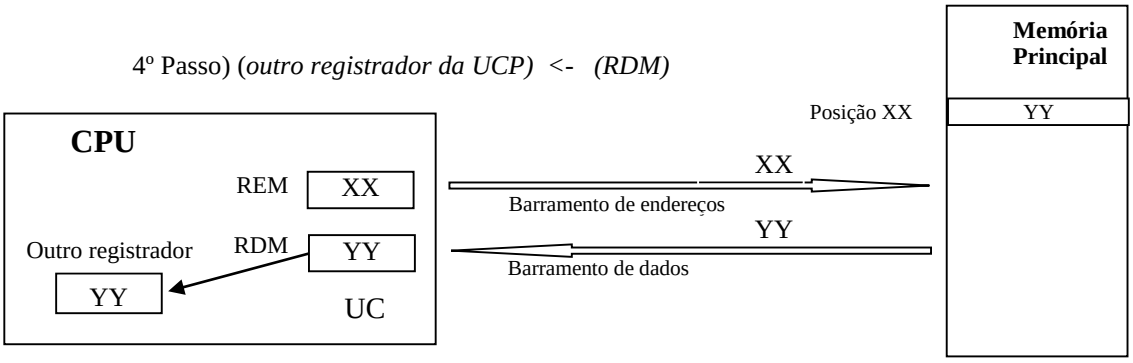
2º Passo) Sinal de leitura é ativado no barramento de controle
Decodificação do endereço e localização da célula na memória



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



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



5) (2,0) 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*

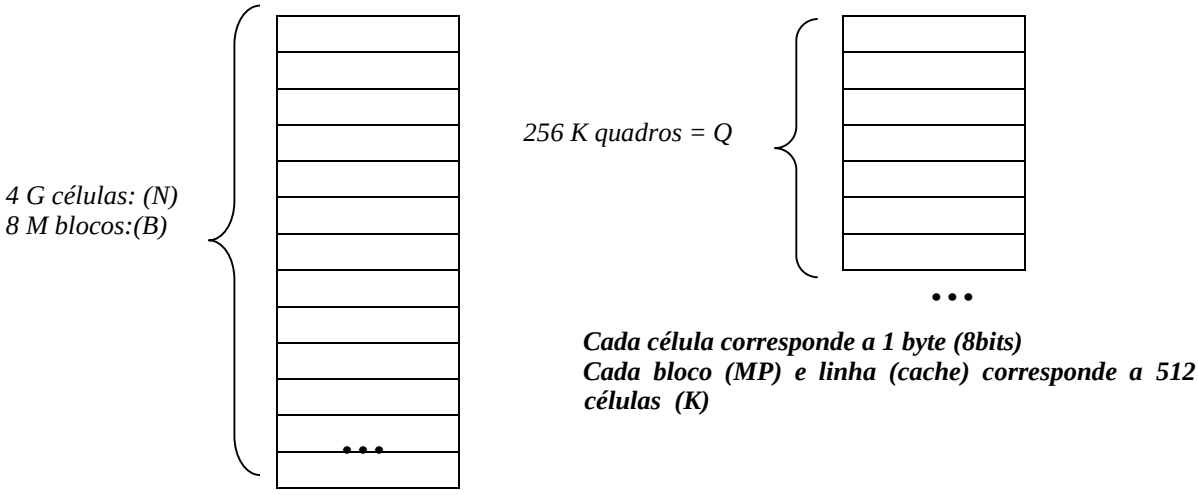
⇒ *Sendo N o tamanho endereçável da memória e K que é a quantidade de células por blocos temos: N = 4 G células e K = 512 células / blocos o total de blocos da MP (B) será: Total de blocos: B = N / K => B = 4 G células / 512 células/bloco => B = 8M blocos*

Memória Cache

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

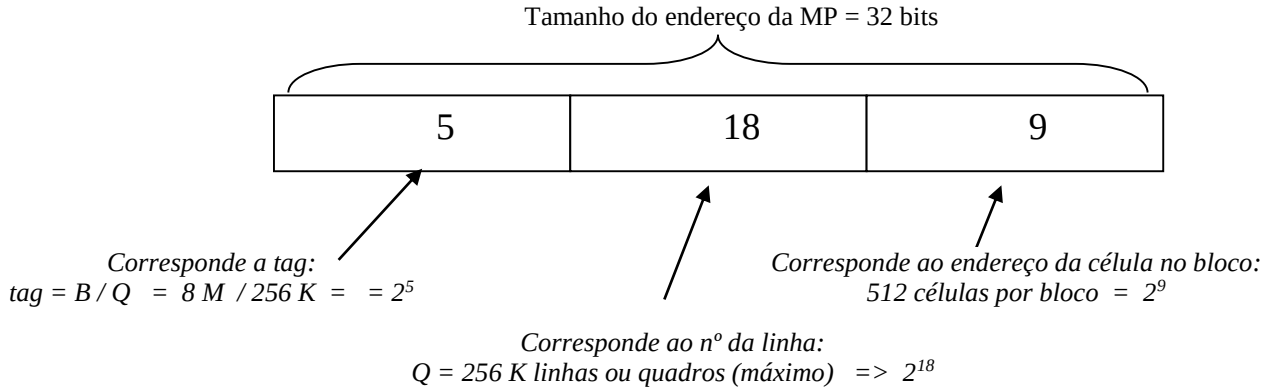
⇒ *Tamanho da memória cache em blocos = 256K linhas que podem armazenar 256 K blocos*

⇒ *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 512 células (K)

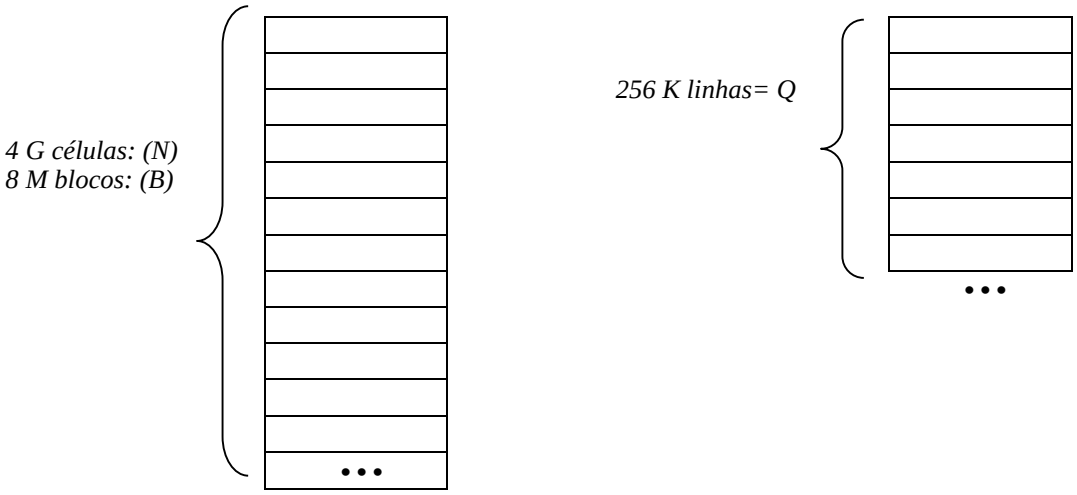
Para endereçarmos toda a MP precisamos da seguinte quantidade de bits (E)
sendo $N = 2^E$ => N = 4G células => $N = 2^{32}$ => E = 32 bits



b) Mapeamento totalmente associativo.

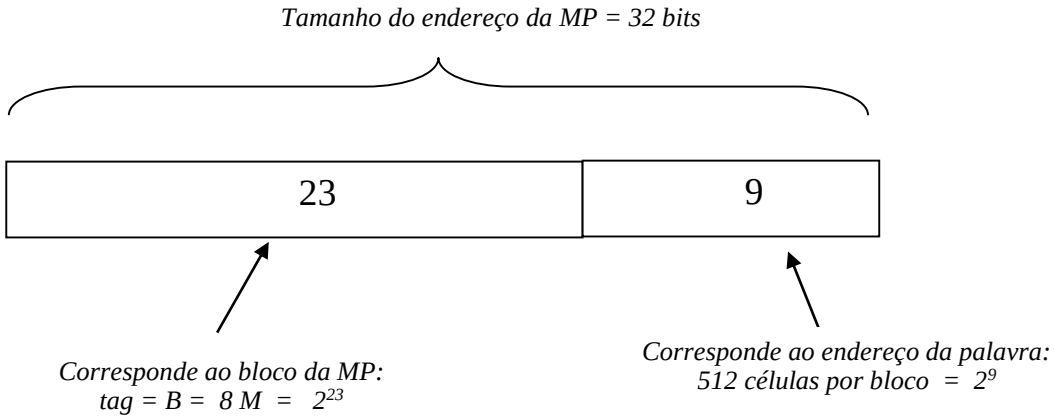
Memória Principal
=> N = 4G células
=> K = 512 células por bloco
=> B = 8 M blocos

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

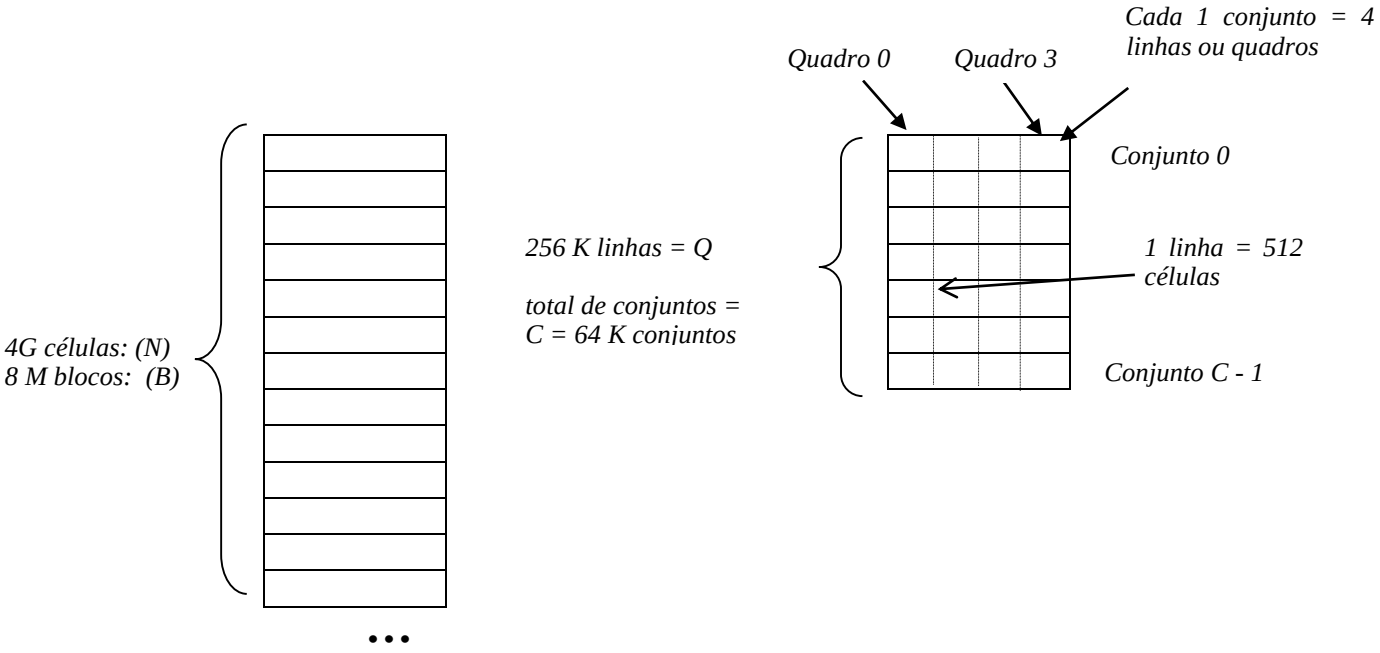
Como o bloco pode ser alocado em qualquer posição da memória cacha a tag indicará qual dos blocos da MP está alocado naquela posição da memória cachê



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

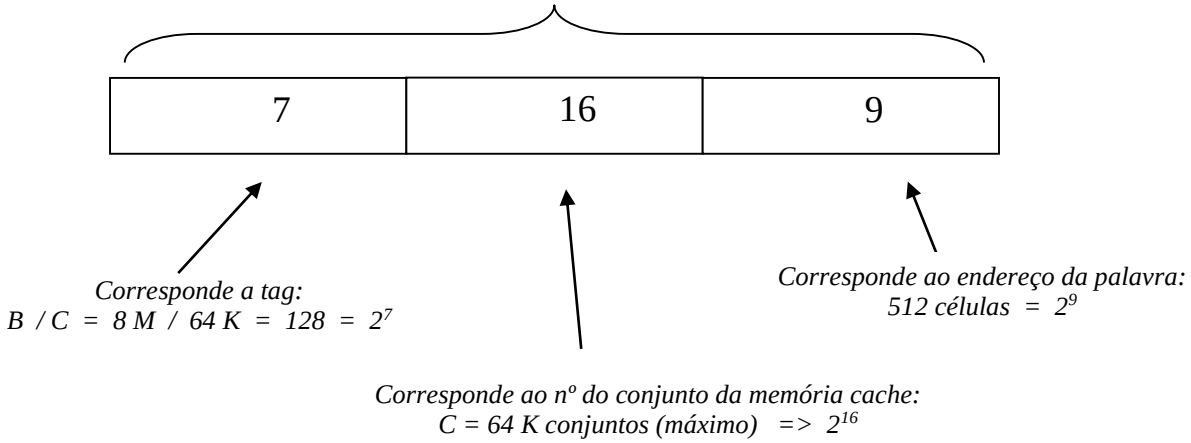
Memória Principal
=> N = 4 G células
=> K = 512 células
=> B = 8 M blocos

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***



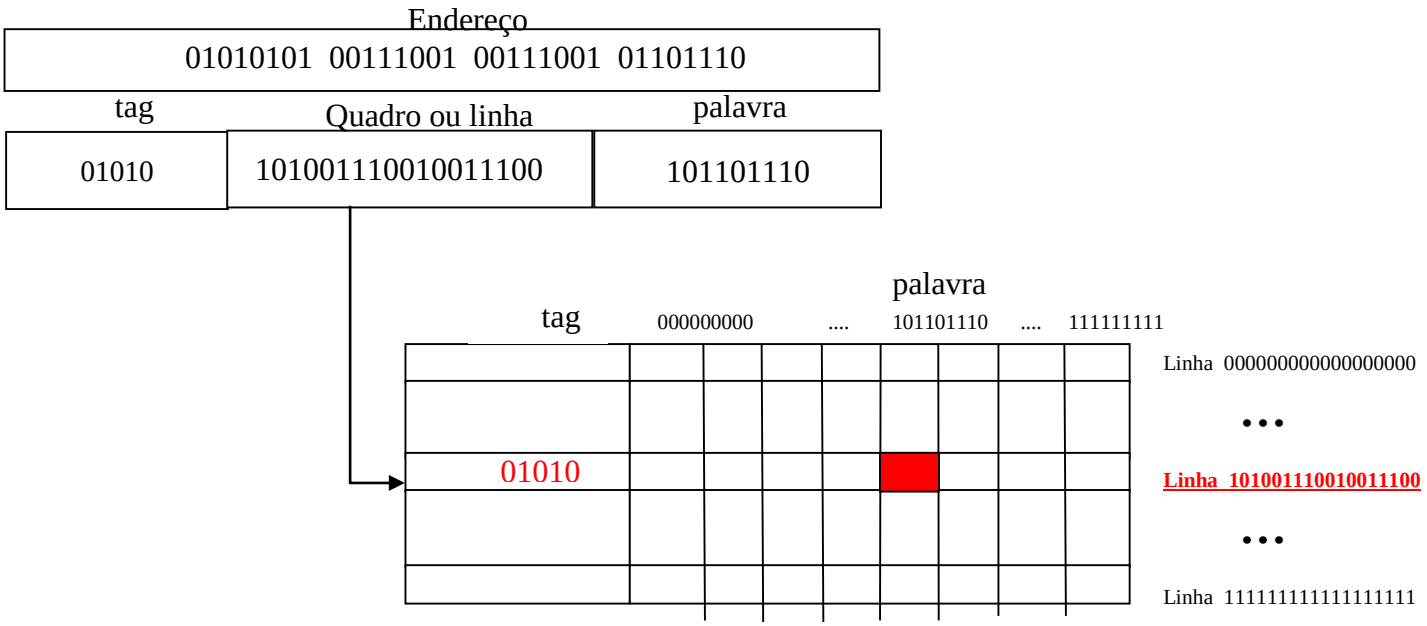
Para endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 32 bits

Tamanho do endereço da MP = 32 bits



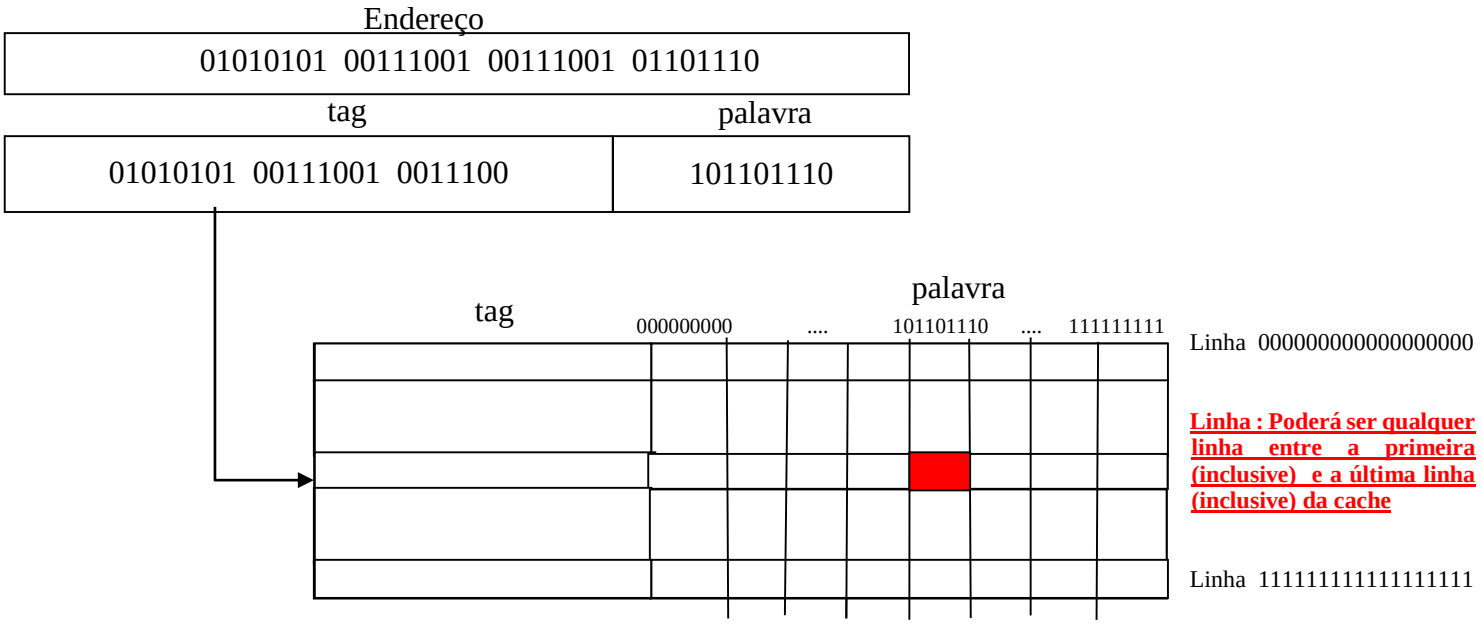
d) Em que linha, para cada um dos mapeamento dos itens anteriores, estaria contido o byte armazenado no seguinte endereço da MP: 01010101 00111001 00111001 01101110.

i) Para o mapeamento direto



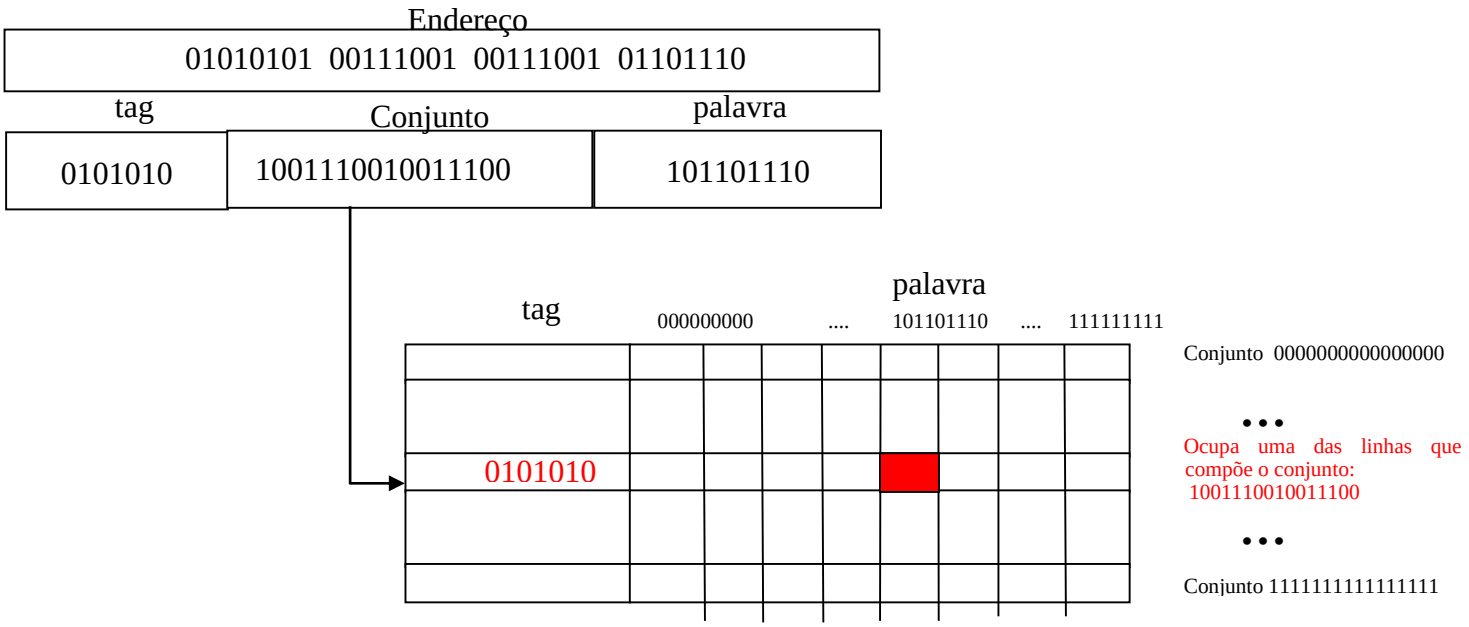
OBS: Com o endereço fornecido, será conferido o campo tag do endereço com a tag da linh caso não sejam iguais, teremos um cache miss, ou seja a palavra não está na memória cache.

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.

,

6) (1,0) Descreva como é organizada a hierarquia de memória, detalhando-a, no processador Intel Core I7

Hierárquia de memória

O subsistema de memória é interligado de forma bem estruturada e organizado hierarquicamente que pode ser representado na forma de 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.

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 das memórias no I7 : (fonte: www.clubedohardawre.com.br)

O processador I7 como o I3 e I5 utilizam a microarquitetura Sandy Bridge. Com relação a memória esta arquitetura tem como principais características:

- *Possuir uma cache de instruções decodificadas denominada cache L0, capaz de armazenar cerca de 1536 instruções, em torno de 6KB.*
- *A cache L1 é dividida em 2: cache de instruções com cerca de 32KB, e cache de dados de igual tamanho. A cache L2 foi renomeado para cache intermediário 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 um dos 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*
- *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 compõem-se de cerca de 12 novas instruções.*

Detalhes dos níveis de memória do I7

1) Registradores:

=> registradores da arquitetura 32 bits

- *Os registradores de uso geral (16 bits) são AX, BX, CX, DX*
- *O Intel x86 possui ainda registradores de 16 bits para segmentos de memória CS, DS, SS, ES, FS, GS.*
- *Os registradores de uso geral (32 bits) são EAX, EBX, ECX, EDX*
- *Os registradores apontadores de memória (32 bits) são ESI, EDI , EBP e ESP (este ponteiro de pilha)*
- *IPR - (Instruction Pointer Register) é o registrador de 32 bits como apontador de posição de memória (CI)*
- *RFLAGS (32 bits) flags*
- *Registradores para ponto flutuante ST0–ST7 (32 bits),*
- *vetores SIMD (Single Instruction, Multiple Data) MM0–MM7 (64 bits) e XMM0-XMM7 (128 bits).*
- *Registradores MMX0-7 de 64 bits*

=> registradores da arquitetura 64 bits

Além dos registradores dos 32 bits (manter compatibilidade), são acrescidos:

- *Instruções AVX, implementadas a partir da arquitetura Sandy Bridge, utilizam registradores XMM de 256 bits com o objetivo de armazenar vários dados menores e processá-los em uma única instrução (conceito SIMD)*
- *Registradores de 64 bits (RAX, RBX, RCX, RDX, RSI, RDI , RBP e RSP)*
- *Registradores de 80 bits de ponto flutuante (FPR0-7)*
- *IPR (Instruction Pointer Register) passa a ser de 64 bits (apontador de posição de memória)*
- *RFLAGS (64 bits) flags*

2) Organização da memória cache:

=> Série Ix (I3, I5, I7)

A tecnologia Intel de processadores Intel Sandy Bridge foi chamada de segunda geração da familía Intel Core, tem como característica a expessura de 32nm. A cache dessa arquitetura segue a seguinte organização:

- *Cache de microinstruções decodificadas (cache L0, capaz de armazenar 1.536 microinstruções, o que equivale a mais ou menos 6 kB);*
- *Cache L1 de instruções de 32 kB e cache L1 de dados de 32 kB por núcleo (nenhuma mudança em relação à arquitetura Nehalem – arquitetura anterior);*
- *O cache de memória L2 - cache intermediário com 256 kB por núcleo;*
- *O cache L3 é chamado é compartilhado entre os núcleos do processador e o processador gráfico, e variam a capacidade conforme de acordo com o processador:*
    - *Os processadores I7 estão disponíveis nas versões de quatro, seis núcleos e oito núcleos, memória cache L3 chegando a 40 MB,*

3) Memória principal:

O tamanho máximo da memória principal depende do registrador apontador de endereço de memória (IPR - Instruction Pointer Register).
=> arquitetura 32 bits, o IPR é de 32 bits, máximo de memória acessível é de 2^32 endereços
=> arquitetura 64 bits, o registrador IPR é de 64 bits, máximo de memória acessível é de 2^64 endereços.