GABARITO DA AD1 – ORGANIZAÇÃO DE COMPUTADORES

1) (0,5) 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 300

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: Busca do operando na memória
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 contido no RDM seja transferido para o barramento de dados
Passo 4.3: A UC emite sinais para que o valor do campo operando (Op. = 300) 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 (300) contido no barramento de endereços

b) JNZ 100

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 = 100.
Será executada a instrução do endereço 100.
Passo 1.2: Caso seja falso: CI <- CI + 1.
Será executada a instrução do endereço seguinte

2) (1,0) 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 100 é lido e verifica-se se o seu valor é 0. Caso seu valor seja 0, o conteúdo de memória cujo endereço é 200 é somado ao conteúdo de memória cujo endereço é 350 e o resultado é armazenado no endereço 400. Caso contrário, o conteúdo de memória cujo endereço é 200 é subtraído do conteúdo de memória cujo endereço é 350 e o resultado é armazenado no endereço 400.
Além de apresentar seu programa escrito em linguagem de montagem, apresente também o programa traduzido para linguagem de máquina.

OBS: As instruções da aula 4 são compostas de 2 campos, 1 código de operação com 4 bits e 1 operando com 8 bits, desta forma cada instrução possui 12 bits. Nesta questão houve a necessidade do operando assumir o tamanho de 12 bits para poder receber os endereços contidos no título da questão. As instruções continuam a seguir os códigos de operação do slide 5 da aula 4, agora com 12 bits no campo de operando.

Endereço (hexa)	Instrução	Descrição	Linguagem Máquina (bin / hexa)
010	LDA 100	ACC ← (100)	0001 0001 0000 0000 / 1100
011	JZ 016	se ACC == 0, CI ← 016	0101 0000 0001 0110 / 5016
012	LDA 350	ACC ← (350)	0001 0011 0101 0000 / 1350
013	SUB 200	ACC ← ACC -(200)	0100 0001 0000 0000 / 4200
014	STR 400	(400) ← ACC	0010 0100 0000 0000 / 2400
015	HLT	Encerra	0000 0000 0000 0000 / 0000
016	LDA 200	ACC ← (200)	0001 0010 0000 0000 / 1200
017	ADD 350	ACC ← ACC + (350)	0011 0011 0101 0000 / 3350

016	STR 400	(400) ← ACC	0010 0100 0000 0000 / 2400
018	HLT	Encerra	0000 0000 0000 0000 / 0000

3) (1,2) Considere uma máquina com arquitetura semelhante àquela apresentada em aula. Pode-se endereçar no máximo 512 M células de memória, sendo que cada célula tem tamanho igual a 16 bits. Em cada acesso á memória, obtém-se o conteúdo de uma célula. Todas as instruções desta máquina possuem três campos: o primeiro indica o código de operação e o segundo e terceiro indicam endereços de célula de memória onde se encontram os operandos. Esta máquina possui 56 códigos de operação diferentes.

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 = E = tamanho em bits necessários para acessar toda a memória (N)
N = 2^E = 512M células = 2^{29} células => E = 29 => REM = '29 bits

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

Barramento de endereços = REM = 29 bits

c) Calcule o tamanho do RI (Registrador de Instruções).

Tamanho mínimo de RI = tamanho da instrução
Tamanho da instrução = cod.oper. + 2 operandos (operando = 1 endereço de memória)
código de operação = tamanho que permite no mínimo 56 instruções = 6 bits, pois $2^6 = 64$
tamanho da instrução = 6 bits + 2 x 29 bits = 64 bits
Tamanho mínimo de RI = 64 bits

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

Tamanho mínimo de 1 instrução = 64 bits
Tamanho de 1 célula = 16 bits
Serão necessárias <u>pelo menos</u> 4 células (64bits) para armazenar a instrução

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

N = 2^{29} células = 512 M células
M = 16 bits/célula
T = M x N => T = 2^{29} células x 16 bits/célula = 2^{33} bits (ou 2^{30} bytes ou 1 Gbyte)

f) Calcule a capacidade mínima de endereçamento 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 necessário para acessar toda a memória = 29 bits

4) (0,8) Considere uma máquina cujo relógio possui uma freqüência de 2 GHZ e um programa no qual são executadas 150 instruções desta máquina.

a) Calcule o tempo para executar este programa, considerando que cada instrução é executada em 15 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 = 15 ciclo de relógio = 15 x 0,5ns = 7,5ns
*150 instruções executadas sequencialmente = 150 x 7,5ns = **1.125ns ou 1,125µs***

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 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 150 instruções = 7,5ns + 149 x 1,5ns = **231ns***

c) Uma segunda nova implementação dessa máquina utiliza um pipeline de 6 estágios, todos de duração igual a 2 ciclo 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 0,5ns = 1,0ns
Para execução da 1ª instrução = 6 estágios x 1,0ns = 6ns
Para execução das instruções posteriores = tempo de 1 estágio devido ao pipeline = 1,0ns
*Tempo total para execução das 150 instruções = 6ns + 149 x 1,0ns = **155ns***

5) (0,5) Calcule o tempo MÍNIMO para transmitir um arquivo de 16 Megabytes armazenado em um servidor WEB para a sua máquina, considerando os seguintes modos de transmissão:

OBS: as taxas de transmissão utilizam a base 10 (não base 2), diante disso 1Kbits/seg = 1.000bits/seg

a) Modo síncrono com taxa de 56 Kbits por segundo.

Tempo = total de bits a enviar / taxa em bits/seg
Tempo = (16 x 2^{30} x 8 bits) / 56.000 bits/seg
Tempo = 134.217.728 / 56.000 seg => Tempo = 2.397 segundos

b) Modo assíncrono com taxa de 19.2 Kbits por segundo. Considere que neste caso para cada byte a ser enviado necessita-se do envio de um bit para indicar inicio de byte e outro bit para indicar fim de byte.

Tempo = total de bits a enviar / taxa em bits/seg
Para cada byte a ser enviado são adicionados 2 bits (inicio e fim)
Tempo = (16 x 2^{30} x (8 bits + 2 bits)) / 19.200 bits/seg
Tempo = 167.772.160 / 19.200 seg => Tempo = 8.738 segundos

6) (1,0) Algumas placas mãe de computadores atuais utilizam os chipsets. Explique o que são e como funcionam os chipsets (fontes de consulta: Guia do Hardware http://www.guiadohardware.net/) e Clube do Hardware http://www.clubedohardware.com.br/).

(texto baseado no site www.clubedohardware.com.br)

Chipset é o nome dado ao conjunto de chips (set significa "conjunto", daí o seu nome) usado na placa-mãe. Nos primeiros PCs, a placa-mãe usava circuitos integrados discretos. Com isso, vários chips eram necessários para criar todos os circuitos necessários para fazer um computador funcionar..Após algum tempo os fabricantes de chips começaram a integrar vários chips dentro de chips maiores. Como isso, em vez de usar uma dúzia de pequenos chips, uma placa-mãe poderia ser construída usando apenas meia dúzia de chips maiores.O processo de integração continuou e em meados dos anos 90 as placas-mãe eram construídas usando apenas dois ou até mesmo um único chip grande.

Com o lançamento do barramento PCI, um novo conceito, que ainda hoje em dia é utilizado, pode ser empregado pela primeira vez: a utilização de pontes. Geralmente as placa-maes possuem dois chips grandes: um chamado ponte norte e outro chamado ponte sul. Às vezes, alguns fabricantes de chip podiam integrar a ponte norte e a ponte sul em um único chip; neste caso a placa-mãe terá apenas um circuito integrado grande. Com o uso da arquitetura em pontes os chipsets puderam ser padronizados.

Os fabricantes de placas-mãe compram dos fabricantes de chipsets os chipsets para serem integrados em suas placas. Na verdade, existe um aspecto muito interessante nessa relação. Para construir uma placa-mãe, o fabricante da placa pode seguir o projeto padrão do fabricante do chipset, também conhecido como "modelo de referência", ou pode criar seu próprio projeto, fazendo modificações no circuito para oferecer maior desempenho e mais funcionalidades.

Ponte Norte

O chip ponte norte, também chamado de MCH (Memory Controller Hub, Hub Controlador de Memória) é conectado diretamente ao processador e possui basicamente as seguintes funções:

- *Controlador de Memória*
- *Controlador do barramento AGP (se disponível)*
- *Controlador do barramento PCI Express x16 (se disponível)*
- *Interface para transferência de dados com a ponte sul*

Alguns chips ponte norte também controlam o barramento PCI Express x1. Em alguns outros é a ponte sul quem controla o barramento PCI Express x1. Em nossas explicações assumiremos que a ponte sul é o responsável por controlar as pistas PCI Express x1, mas tenha em mente que isso pode variar de acordo com o modelo do chipset.

O processador não acessa diretamente a memória RAM (isso antes dos Intel I3, I5 e I7, pois estes passaram a acessar diretamente o banco de memória principal) ou a placa de vídeo. É a ponte norte que funciona como intermediário no acesso do processador a estes dispositivos. Por causa disso, a ponte norte tem influência direta no desempenho do micro. Se um chip de ponte norte tem um controlador de memória melhor do que outro, o desempenho geral do micro será melhor. Isto explica o motivo pelo qual você pode ter duas placa-mãe voltadas para a mesma classe de processadores e que obtêm desempenhos diferentes.

Como o controlador de memória está na ponte norte, é este chip que limita o tipo e a quantidade máxima de memória que você pode instalar no micro. A conexão entre a ponte norte e a ponte sul é feita através de um barramento. No início, o barramento utilizado para conectar a ponte norte à ponte sul era o barramento PCI. Atualmente, o barramento PCI não é mais usado para esse tipo de conexão e foi substituído por um barramento dedicado.

Ponte Sul

O chip ponte sul, também chamado ICH (I/O Controller Hub, Hub Controlador de Entrada e Saída) é conectado à ponte norte e sua função é basicamente controlar os dispositivos on-board e de entrada e saída tais como:

- *Discos Rígidos (Paralelo e Serial ATA)*
- *Portas USB*
- *Som on-board (*)*
- *Rede on-board (**)*
- *Barramento PCI*
- *Barramento PCI Express (se disponível)*
- *Barramento ISA (se disponível)*
- *Relógio de Tempo Real (RTC)*
- *Memória de configuração (CMOS)*
- *Dispositivos antigos, como controladores de interrupção e de DMA*

(*) Se a ponte sul tiver controlador de som on-board, será necessário a utilização de um chip externo chamado de codec (abreviação de codificador/decodificador) para funcionar.

(**) Se a ponte sul tiver controlador de rede on-board, será necessário a utilização de um chip chamado phy (pronuncia-se "fái", abreviação de physical, camada física, em português) para funcionar.

A ponte sul é também conectada a dois outros chips disponíveis na placa-mãe: o chip de memória ROM, mais conhecido como BIOS, e o chip Super I/O, que é o responsável por controlar dispositivos antigos como portas seriais, porta paralela e unidade de disquete.

Enquanto que a ponte sul pode ter alguma influência no desempenho do disco rígido, este componente não é tão crucial no que se refere ao desempenho geral do micro quanto à ponte norte. Na verdade, a ponte sul tem mais a ver com as funcionalidades da sua placa-mãe do que com o desempenho. É a ponte sul que determina a quantidade (e velocidade) das portas USB e a quantidade e tipo (ATA ou Serial ATA) das portas do disco rígido que sua placa-mãe possui, por exemplo

7) (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
Concluíndo, para atender a especificação do computador hipotético, a de ter uma memória

*máxima de 3G células, deveremos ter um barramento de endereços de **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?

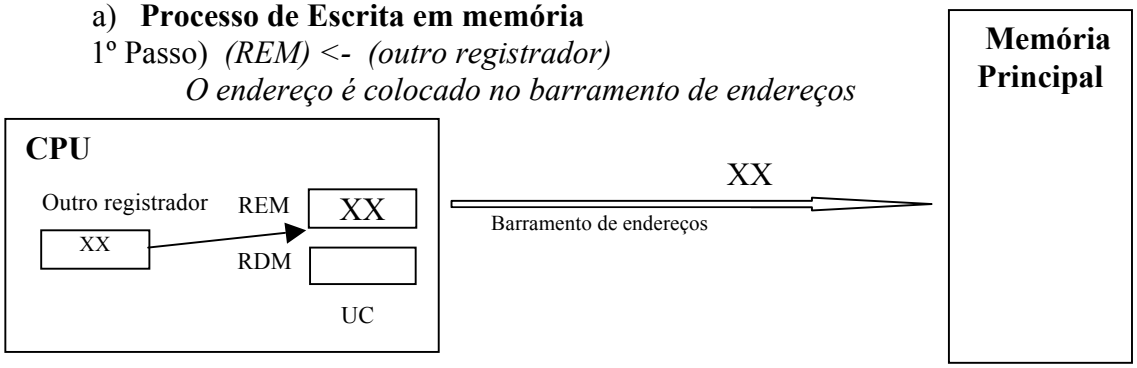
Como M = quantidade de bits em uma célula, neste computador, cada célula = palavra
então M = 12bits e $N = 3 \times 2^{30}$
Total de bits da memória = T = N x M
$T = N \times M = 3 \times 2^{30} \quad x \quad 12 \quad = 36$ G bits

8) (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**

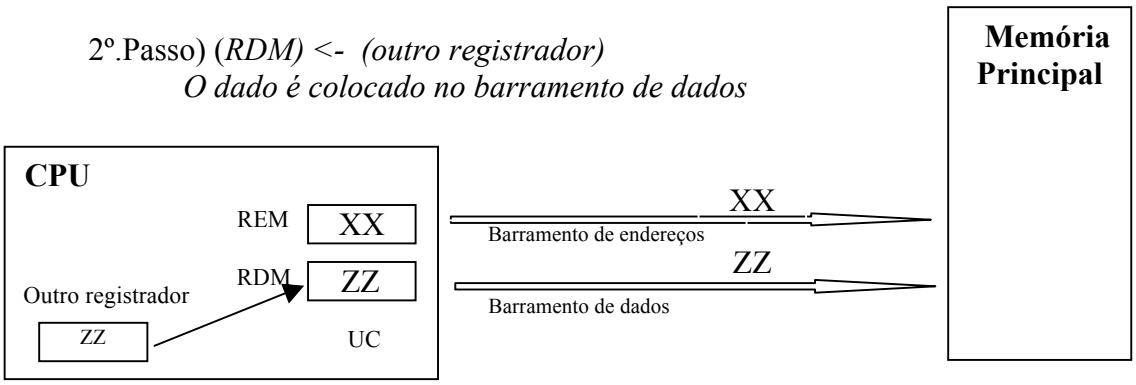
1º Passo) *(REM) <- (outro registrador)*

O endereço é colocado no barramento de endereços

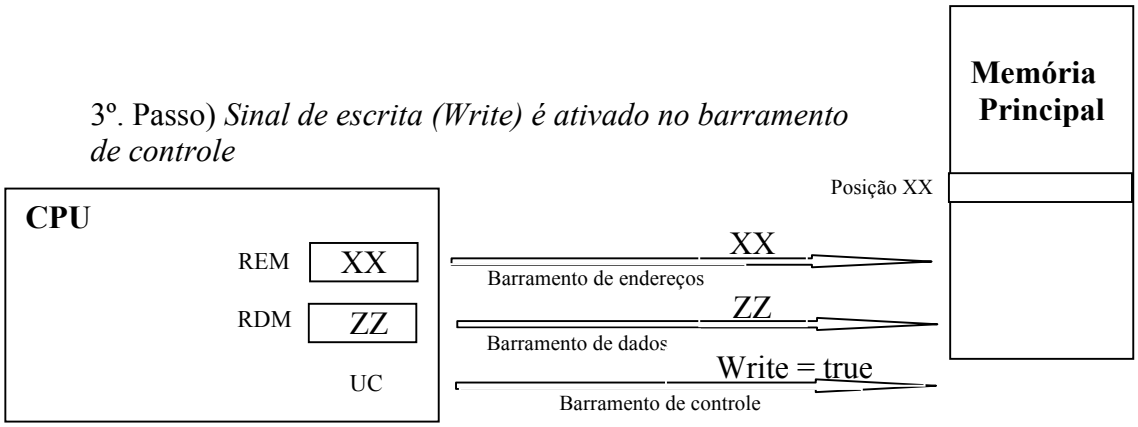


2º.Passo) (*RDM) <- (outro registrador)*

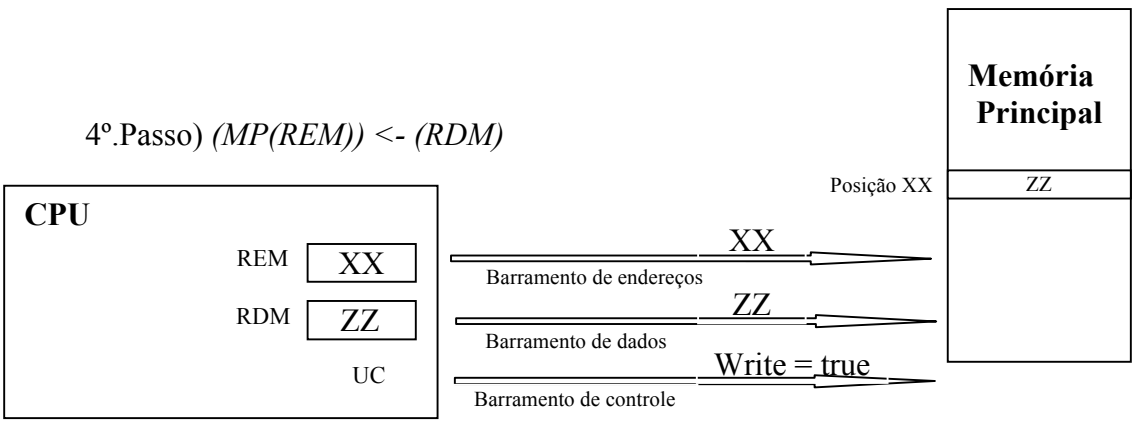
O dado é colocado no barramento de dados



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

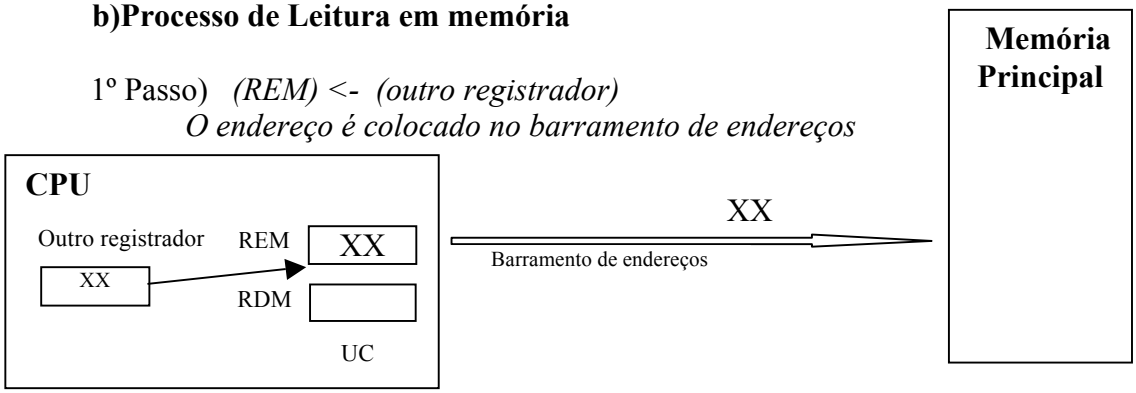


4º.Passo) *(MP(REM)) <- (RDM)*

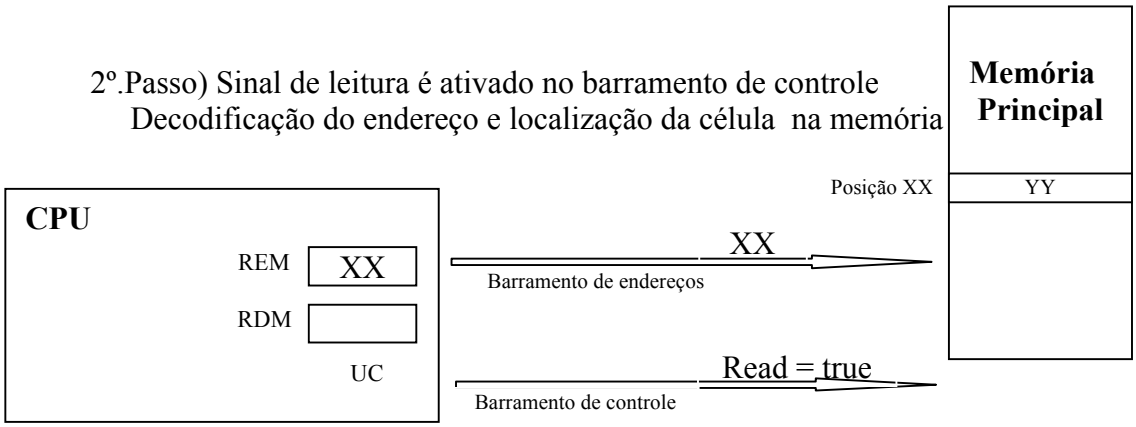


b)Processo de Leitura em memória

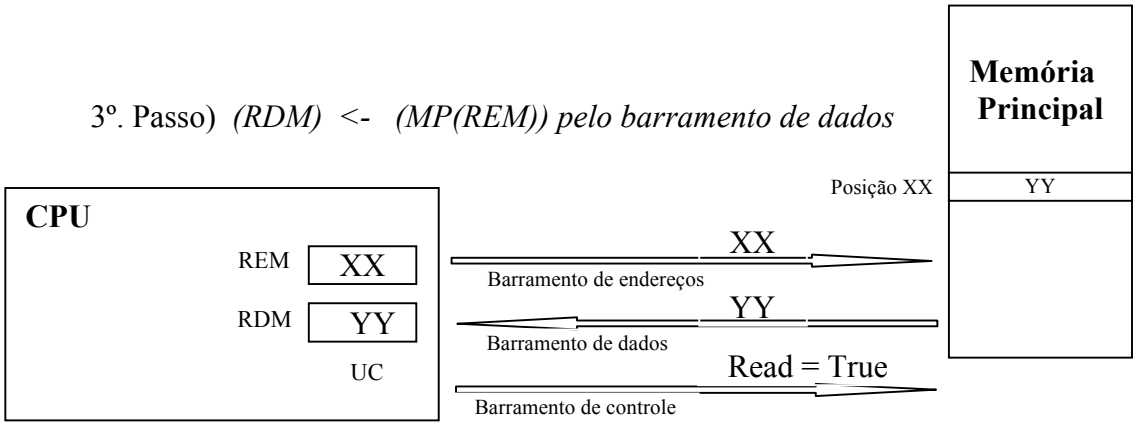
1º Passo) *(REM) <- (outro registrador)*
O endereço é colocado no barramento de endereços



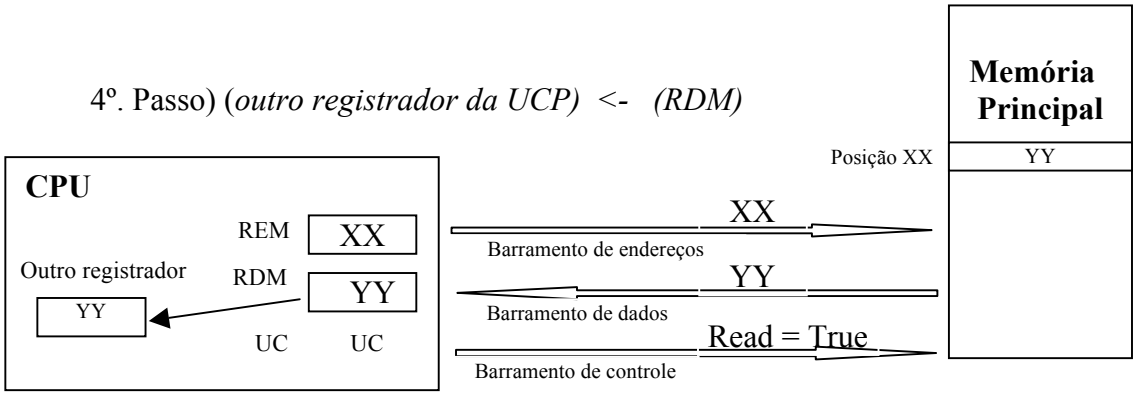
2º.Passo) Sinal de leitura é ativado no barramento de controle
Decodificação do endereço e localização da célula na memória



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



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



9) (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.

M*emória Principal*

⇒ *Tamanho da memória (em bytes) = 4Gbytes, como 1 célula referencia a 1 byte, temos N = 4G 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 = 4G células e K = 512 células / blocos o total de blocos da MP (B) será:

Total de blocos: B = N / K => B = 4G células / 512 células/bloco => B = 8 M blocos

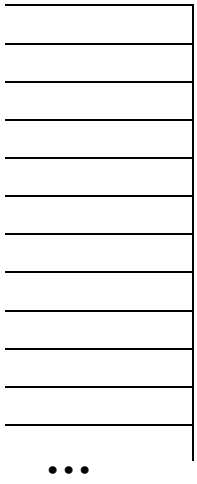
Memória Cache

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

⇒ *Tamanho da memória cache (em blocos ou linhas) => Q =256K blocos*

⇒ *Tamanho da memória cachê em células = Q x K = 256K blocos x 512 células/blocos = 128Mcélulas*

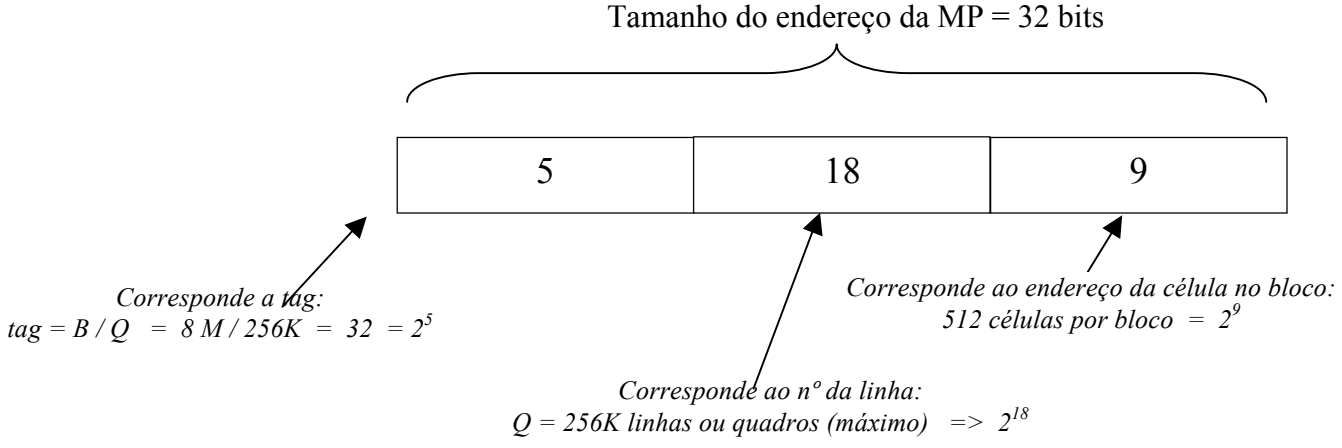
Memória principal
4G células: N
8M blocos: B



Organização da cache

linha	válido	tag	Conteúdo (bloco)
0	1 bit	5 bits	512 células de 8 bits cada = 4096 bits
1			
2			
3			
4			
5			
.....			
Q - 2			
Q - 1			

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

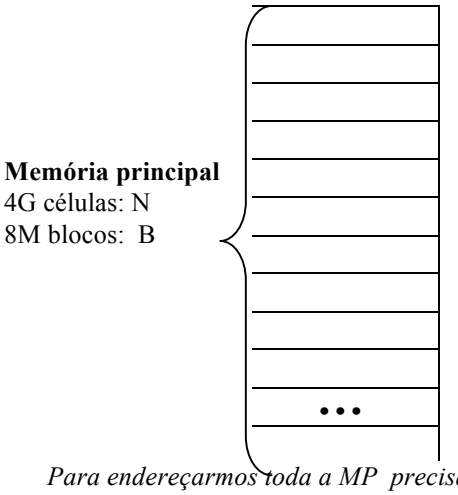


b) Mapeamento totalmente associativo.

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

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

Organização da cache

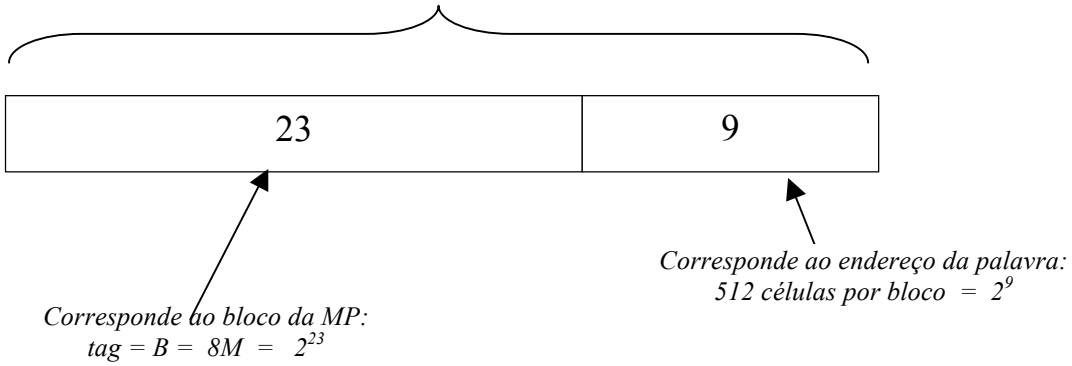


linha	válido	tag	Conteúdo (bloco)
0	1 bit	23 bits	512 células de 8 bits cada = 4096 bits
1			
2			
3			
4			
5			
.....			
Q - 2			
Q - 1			

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

Tamanho do endereço da MP = 32 bits

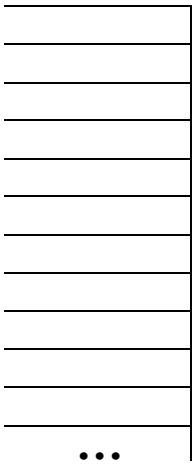


c) Mapeamento associativo por conjunto, onde cada conjunto possui quatro linhas, cada uma de um bloco.

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

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

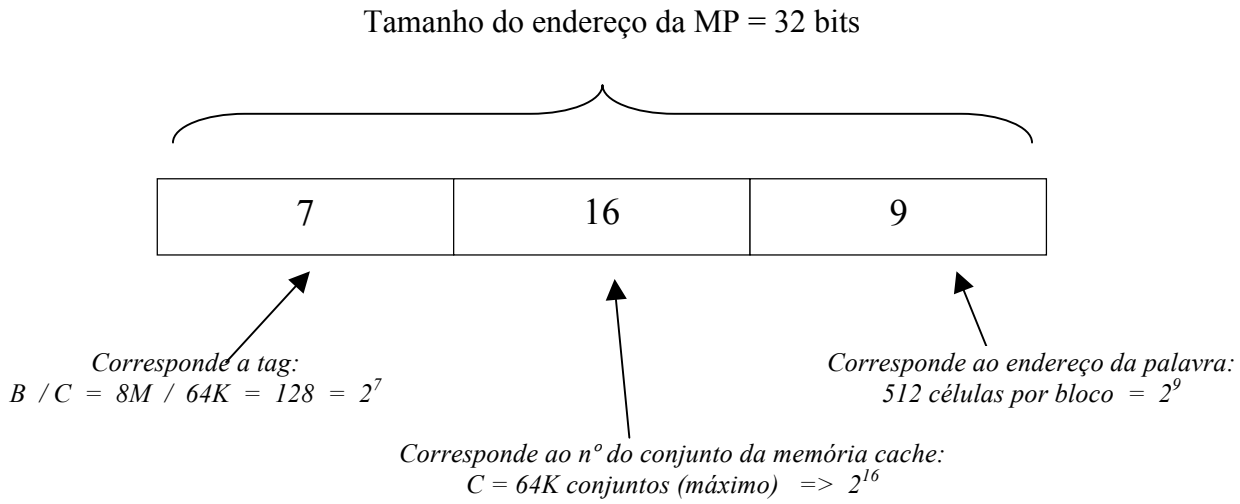
Memória principal
4G células: N
8M blocos: B



Organização da cache

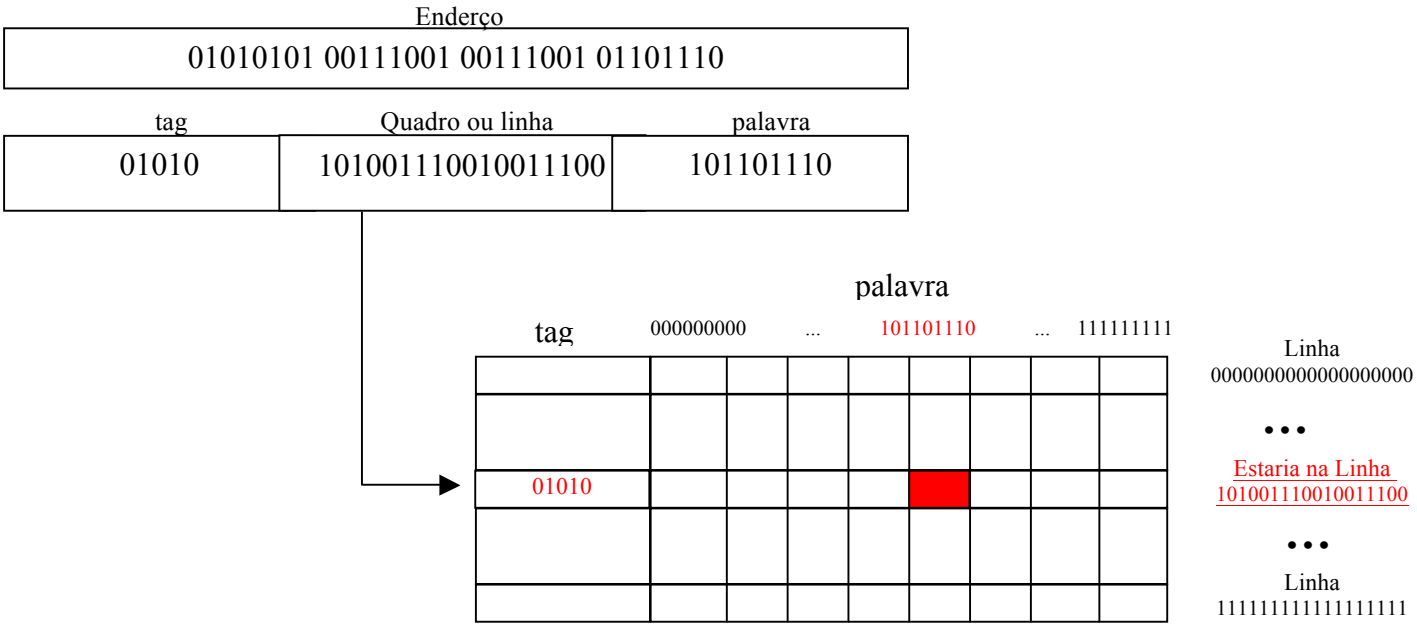
Conjunto	linha	válido	tag	Conteúdo (bloco)
0	0	1 bit	7 bits	512 células de 8 bits cada = 4096 bits
	1			
1	2			
	3			
2	4			
	5			
.....				
C - 1	Q - 2			
	Q - 1			

P*ara endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 32 bits*

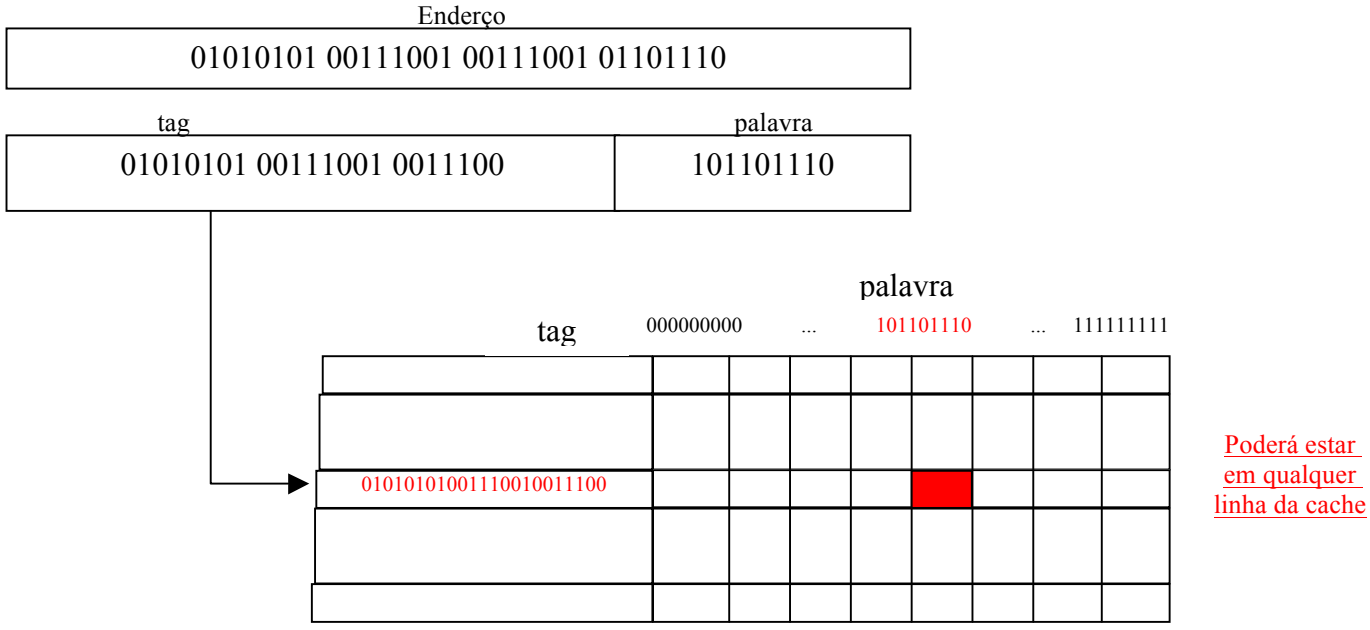


d) Em que linha, para cada um dos mapeamento dos itens anteriores, estaria contido o byte armazenado no seguinte endereço da MP: 01010101 00111001 00111001 01101110.

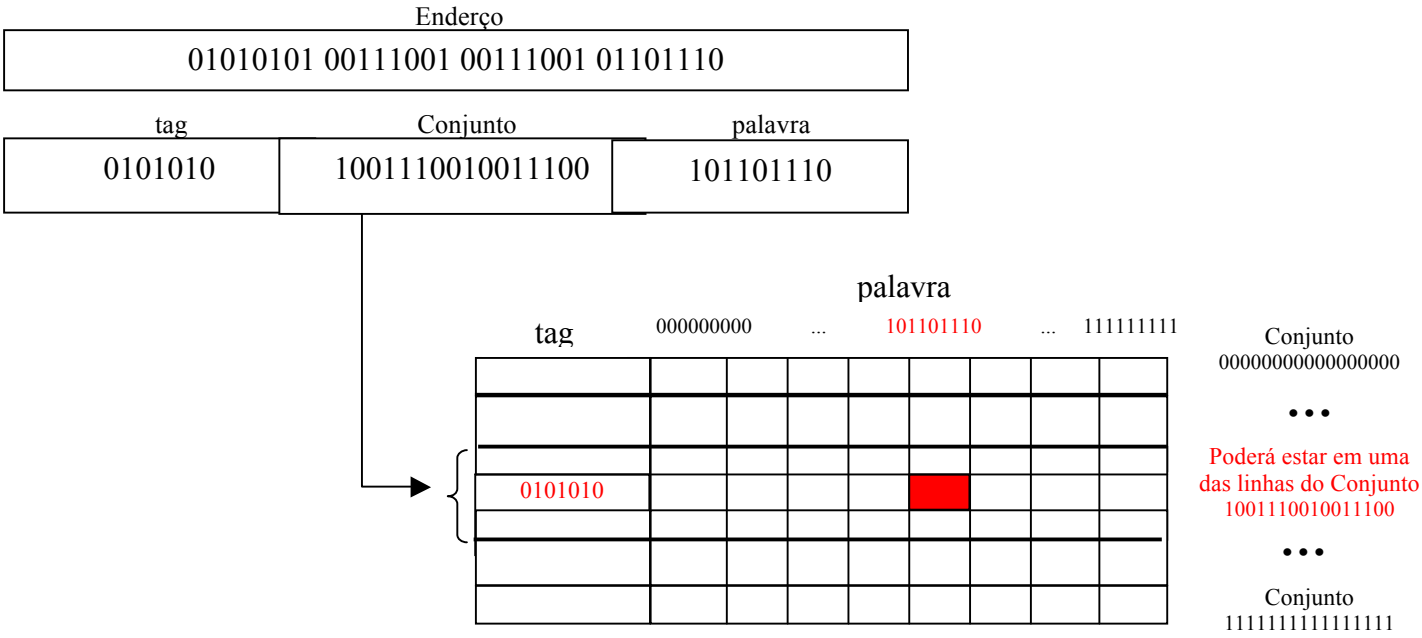
No mapeamento direto



No mapeamento totalmente associativo



No mapeamento direto



10) (1,0) Explique em detalhes a organização hierárquica do subsistema de memória nos computadores atuais. Descreva como é organizada a memória cache nos processadores 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. 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.

Processador IX (I3, I5, I7) 2ª. geração:
Fonte: http://www.clubedohardware.com.br/artigos/Por-Dentro-da-Microarquitetura-Intel-Sandy-Bridge/2146/1

O processador I3, I5 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 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, 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 compõem-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.*
- *por canal*

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*

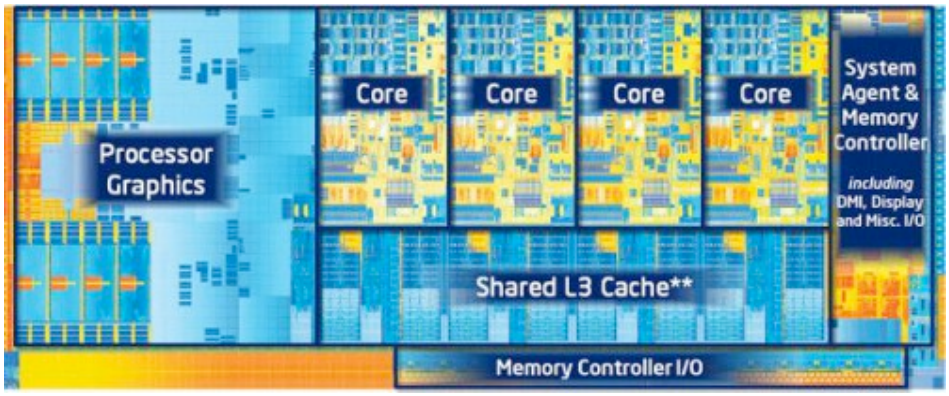
2) Organização da memória cache:

=> Série Ix (I3, I5, I7)

A tecnologia Intel de processadores Intel Sandy Bridge foi chamada de segunda geração da família 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 I3 são disponiveis em modelos de dois núcleos, possuem em torno de 4 MB de memória cache (nível L3) compartilhada,
Os processadores I5 são disponiveis em modelos de dois ou quatro núcleos, possuem em torno de 8 MB de memória cache (nível L3) compartilhada,
Os processadores I7 são disponiveis em modelos de quatro ou seis núcleos, memória cache L3 de 12 MB até 30MB,



(fonte: www.clubedohardawre.com.br)

A tecnologia Intel de processadores Ivy Bridge, lançada no último trimestre de 2011, substituiu a Sandy Bridge, e possui como diferencial a expessura de 22nm. A distribuição da cache segue o modelo da arquitetura anterior

A arquitetura Haswell, lançada em junho de 2013, substitui a Ivy Bridge. Possui um desempenho de 6% a 20% superior a anterior. A cache também segue a organização das arquiteturas anteriores.

A arquitetura Broadwell, a com lançamento em 2014/2015 tem como diferencial a expessura de 14nm.