GABARITO DA AD1 - Organização de Computadores 2018.2

1. **(1,0) Explique em detalhes como funciona a execução da instrução *STR op* na arquitetura mostrada em aula.**

STR op

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 (op) 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 op (conteúdo do barramento de endereços) a palavra recebida através do barramento de dados*

2. **(2,0) Considere uma máquina com arquitetura semelhante àquela apresentada em aula. Pode-se endereçar no máximo 128 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 600 códigos, e dois operandos, que indicam endereços de memória.**

a) Qual o tamanho mínimo do REM ?
REM = Barramento de endereços, este terá a capacidade de endereçar 128Mcélulas = N
N = 128Mcélulas => $N = 2^{27}$ => e = 27 bits
REM = barramento de endereços = ***27 bits***

b) Qual o tamanho mínimo do CI ?
CI terá o tamanho mínimo necessário para endereçar toda a memória = REM = ***27*** **bits**

c) Qual o tamanho do barramento de endereços?
REM = barramento de endereços ***= 27 bits***

d) Qual o tamanho mínimo do RI ?
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 600 códigos diferentes (instruções). CodOper = 10 bits
1º.operando e 2º. operando correspondem a 1 endereço de memória = 27 bits
Instrução = CodOper + Operando1 + Operando2 => Instrução = 10+ 27 +27 = 64 bits
O tamanho mínimo para RI deverá ser ***64 bits***

e) Qual a capacidade máxima da memória em bits ?
T = M x N => T = tamanho da célula x quantidade de células da memória principal =>
tamanho da célula (M) = 64 bits
total de endereços da memória (N) = 128M células
T = 64 bits/célula x 2^{27} células => T = 8192 M bits => T = 8Gbits (ou 1GBytes)

f) Se a largura do barramento de dados desta máquina for igual à metade do tamanho de uma instrução, como funcionará o ciclo de busca ?
Seriam necessários 2 ciclos de busca para transferir uma instrução completa

3. (1,5) Escreva um programa que utilize as instruções de linguagem de montagem apresentadas na aula para executar o seguinte procedimento. O conteúdo da memória cujo endereço é 200 é lido e verifica-se se o seu valor é 0. Caso seu valor seja 0, o conteúdo de memória cujo endereço é 100 é somado ao conteúdo de memória cujo endereço é 250 e o resultado é armazenado no endereço 350. Caso contrário, o conteúdo de memória cujo endereço é 100 é subtraído do conteúdo de memória cujo endereço é 250 e o resultado é armazenado no endereço 350. 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 200*	*ACC <- (200)*	*(0001 0010 0000 0000 / 1200)*
0 01	*JZ 006*	*se ACC == 0, CI <- 006*	*(0110 0000 0000 0110 / 6006)*
002	*LDA 250*	*ACC <- (250)*	*(0001 0010 0101 0000 / 1250)*
003	*SUB 100*	*ACC <- ACC - (100)*	*(0100 0001 0000 0000 / 4100)*
004	*STR 350*	*(350) <- ACC*	*(0010 0011 0101 0000 / 2350)*
005	*JMP 009*	*CI <- 009*	*(1000 0000 0000 1001 / 8009)*
006	*LDA 100*	*ACC <- (100)*	*(0001 0001 0000 0000 / 1100)*
007	*ADD 250*	*ACC <- ACC + (250)*	*(0100 0010 0101 0000 / 3250)*
008	*STR 350*	*(350) <- ACC*	*(0010 0011 0101 0000 / 2350)*
009	*HLT*	*Encerra Procedimento*	*(0000 0000 0000 0000 / 0000)*

4. (1,5) Considere uma máquina cujo relógio possui uma frequência de 2,4 GHZ e um programa no qual são executadas 180 instruções desta máquina.

a) Calcule o tempo para executar este programa, considerando que cada instrução é executada em 12 ciclos de relógio e a execução de uma instrução só se inicia quando a execução da instrução anterior é finalizada.

2,4GHz = 2.400.000.000 Hz
Tempo de um ciclo de relógio = 1/2.400.000.000 = 0,000 000 000 4167 seg ou 0,4167ns (nanosegundos)
Tempo de execução de 1 instrução = 12 ciclos de relógio = 5ns
*180 instruções **executadas sequencialmente** = 180 × 5ns = **900ns** **(para executar 180 instruções)***

b) Uma nova implementação dessa máquina utiliza um pipeline de 5 estágios, todos de duração igual a 4 ciclos de relógio. Calcule o tempo para executar este programa, considerando que não existem conflitos de qualquer tipo.

Tempo para um estágio = 4 ciclos de relógio = 4 x 0,4167ns = 1,667ns
Para execução da 1ª instrução = 5 estágios x 1,667ns = 8,334ns
Para execução das instruções posteriores = tempo de 1 estágio devido ao pipeline = 1,667ns
*Tempo total para execução das 180 instruções = 8,334ns + 179 x 1,667ns = **306,37ns***

c) Uma segunda nova implementação dessa máquina utiliza um pipeline de 6 estágios, todos de duração igual a 3 ciclo 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 0,4167ns = 1,25ns
Para execução da 1ª instrução = 6 estágios x 1,25ns = 7,5ns
Para execução das instruções posteriores = tempo de 1 estágio devido ao pipeline = 1,25ns
*Tempo total para execução das 180 instruções = 7,5ns + 179 x 1,25ns = **231,25ns***

5. (2,0) Considere uma máquina que possa endereçar 512 Mbytes de memória física, utilizando endereço referenciando byte, e que tenha a sua memória organizada em blocos de 32 bytes. Ela possui uma memória cache que pode armazenar 8K 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) = 512M bytes, como 1 célula referencia a 1 byte, temos N = 512M células*

⇒ *Será organizada em blocos de 32 bytes, como 1 célula = 1 byte, temos cada bloco = 32 células, K = 32*

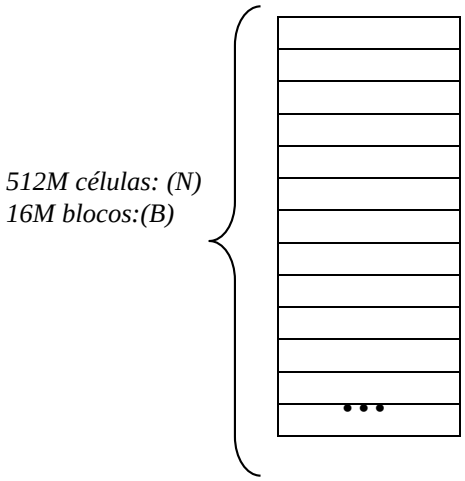
⇒ *Sendo N o tamanho endereçável da memória e K que é a quantidade de células por blocos temos: N = 512M células e K = 32 células / blocos o total de blocos da MP (B) será:*
Total de blocos: B = N / K => B = 512M células / 32 células/bloco => B = 16M blocos

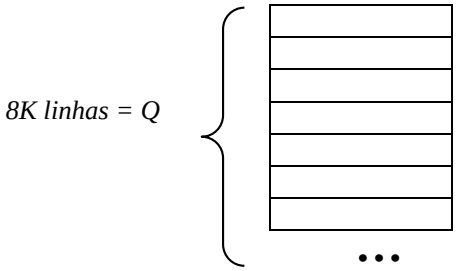
Memória Cache

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

⇒ *Tamanho da memória cache em blocos = 8K linhas que podem armazenar 8K blocos*

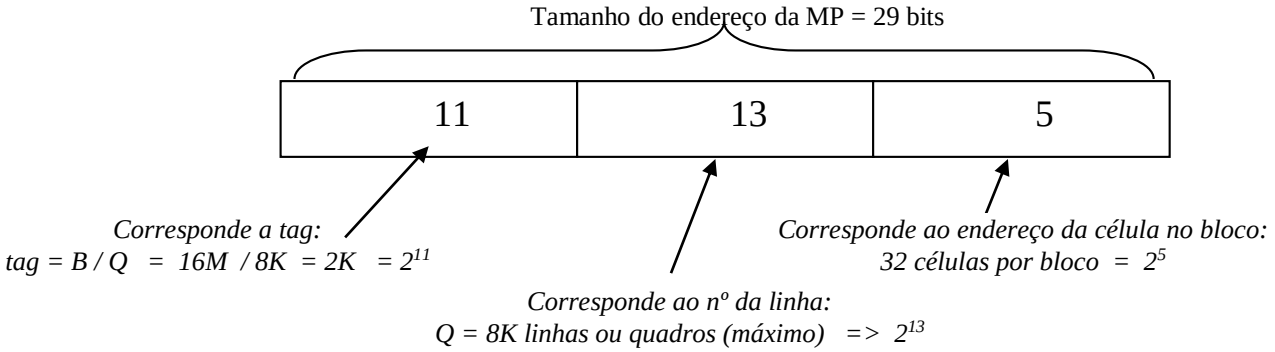
⇒ *Tamanho da memória cache em células = 8K blocos x 32 células/bloco = 256K células*





Cada célula corresponde a 1 byte (8bits)
Cada bloco (MP) e linha (cache) corresponde a 32 células (K)

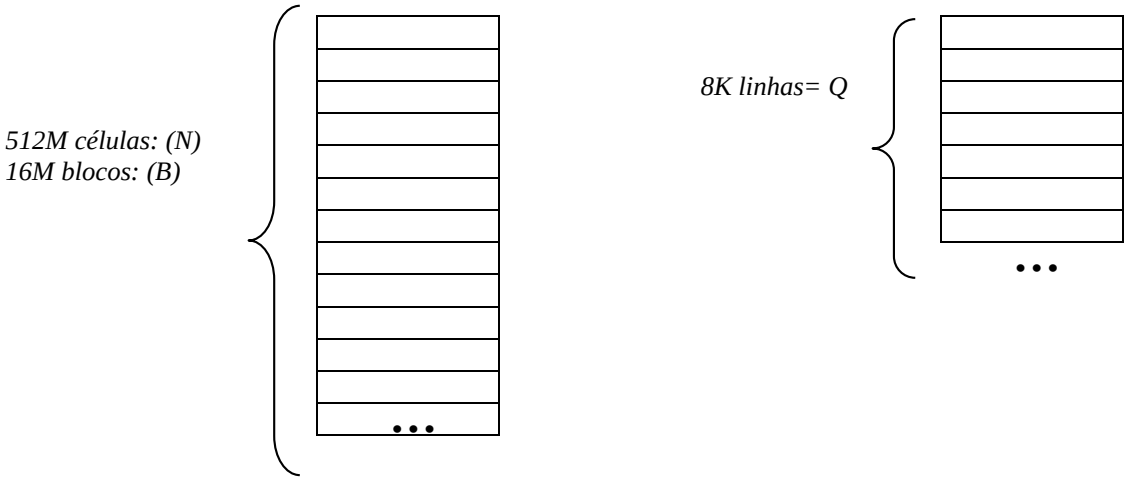
Para endereçarmos toda a MP precisamos da seguinte quantidade de bits (E)
sendo $N = 2^E$ => N = 512M células => $N = 2^{29}$ => E = 29 bits



b) Mapeamento totalmente associativo.

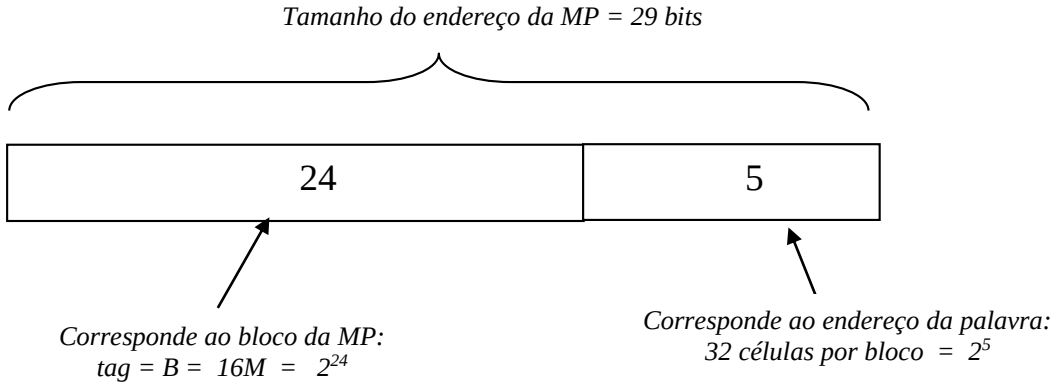
Memória Principal
=> N = 512M células
=> K = 32 células por bloco
=> B = 16M blocos

Memória Cache
OBS: O K (quantidade de células/bloco) tem de ser igual a MP.
=> Q =8K linhas ou 8K blocos
=> Tamanho da memória cache = 8K células



Para endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 29 bits

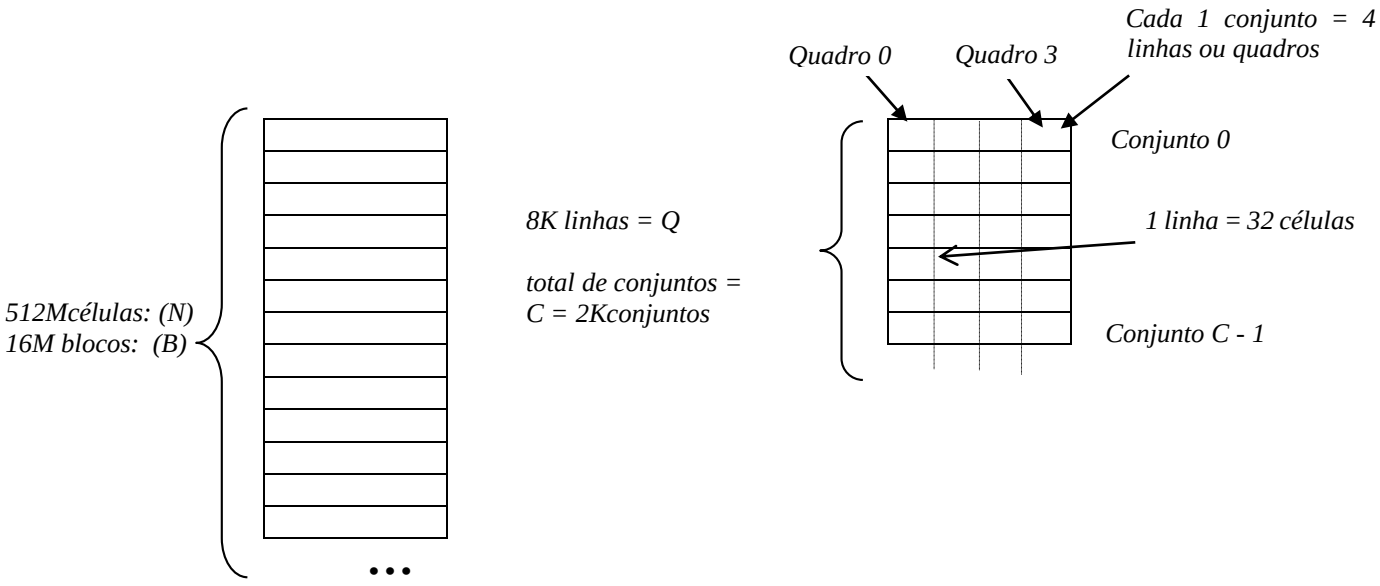
Como o bloco pode ser alocado em qualquer posição da memória cacha a tag indicará qual dos blocos da MP está alocado naquela posição da memória cachê



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

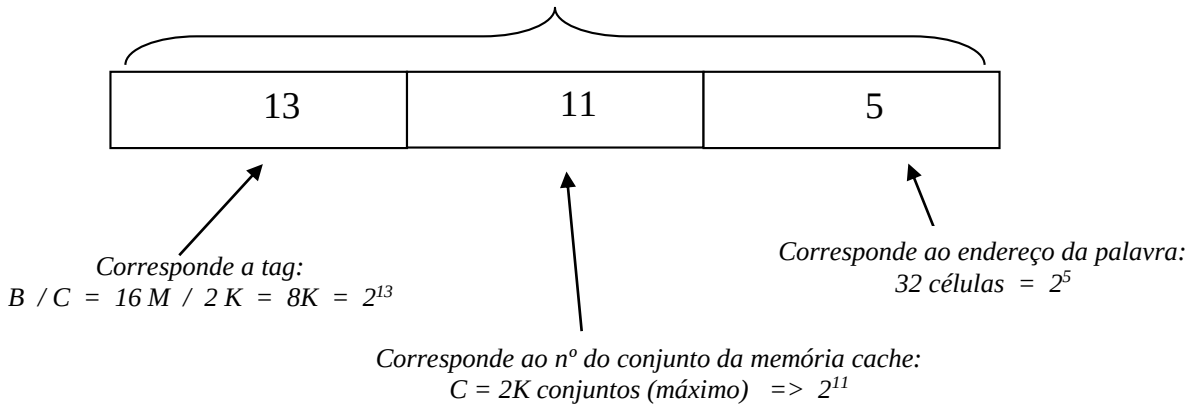
Memória Principal
=> N = 512M células
=> K = 32 células por bloco
=> B = 16M blocos

Memória Cache
OBS: O K (quantidade de células/bloco) tem de ser igual a MP.
=> Q =8 K linhas ou 8K blocos
=> Tamanho da memória cache = 8K células
=> 1 conjunto = 4 linhas (ou quadros) =>
Total de conjuntos => C = 8K células / 4 => ***C = 2K conjuntos***



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

Tamanho do endereço da MP = 32 bits



6. **(2,0) Faça uma pesquisa (pode utilizar a Wikipedia) e descreva as memórias DRAM e SRAM, especificando suas diferenças.**

A memória DRAM *(sigla em inglês para Dynamic Random Access Memory, ou Memória de Acesso Randômico Dinâmica) possui uma estrutura simples baseada em um capacitor e um transistor para cada bit de armazenamento. Como vai havendo fuga de elétrons do capacitor, a informação acaba por se perder, a não ser que a carga seja atualizada periodicamente. Essa atualização constante é chamada de REFRESH, o que faz com que essa memória gaste mais energia se comparado com a SRAM. Entretanto, a estrutura simplificada das memórias DRAM permitem baixo custo e uma maior densidade de bits e consequentemente uma maior capacidade de armazenamento.*

A memória SRAM *(singla em inglês para Static Random Access Memory, ou memória estática de acesso aleatório) é um tipo de memória de acesso aleatório que mantém os dados armazenados desde que seja mantida sua alimentação, não precisando do REFRESH para atualização das células. A estrutura da SRAM é mais complexa que a da DRAM, usando cerca de 4 transistores para cada bit armazenado o que encarece essa memória. Ainda em relação a DRAM, a SRAM é mais econômica quanto ao gasto de energia, possui menor dissipação de calor e melhor performance, tendo especial uso em cache L1 e L2.*