GABARITO AD1 - Organização de Computadores 2014.1

1) (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 200

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 (200) 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 200 (conteúdo do barramento de endereços) o conteúdo recebido através do barramento de dados*

b) JZ 16

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 <- 16)
f) Inicia o procedimento de leitura da instrução contida no endereço que consta em CI

c) JMP 19

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 incondicional*

d) UC emite sinal para transferir o operando para o CI
CI <- Operando (CI <- 19)
e) Inicia o procedimento de leitura da instrução contida no endereço que consta em CI

2) (1,5) Considere uma máquina com 256 Mega células de memória onde cada célula armazena uma palavra e cada instrução tem o tamanho de uma palavra. Esta máquina possui um conjunto de instruções com 256 instruções distintas, sendo cada uma delas composta de um código de operação e dois operandos, que indicam o endereço de memória.

a) Qual o tamanho mínimo do REM?

REM = E = tamanho em bits necessários para acessar toda a memória (N)
$N = 2^E$ = 256 M células = 2^{28} células => E = 28 => REM = 28 bits

b) 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
Código de operação deve permitir 256 instruções distintas, assim o código de operação deverá ter 8 bits
O operando deverá indicar um endereço de memória, assim cada operando deverá ter 28 bits
Instrução = cod.oper + 2 operandos => instrução = 8 + 2 x 28 = 64 bits
RI = tamanho de uma instrução = 64 bits

c) Qual o tamanho mínimo do RDM?

O RDM deverá ser capaz de permitir a transferência de pelo menos 1 célula
como 1 célula = 1 palavra = 1 instrução, então RDM deverá ter no mínimo 64 bits

d) Qual o tamanho da memória em bits?

N = 256 M células = 2^{28} células , e cada célula (M) tem 64 bits
T = N x M => T = 2^{28} x 64 bits => T = 2^{3}□ bits = 17.179.869.184 bits

e) Se a largura do barramento de dados desta máquina for igual à metade do tamanho de uma instrução, como funcionará o ciclo de busca?

Serão necessários 2 ciclos para a transferência de 1 instrução.

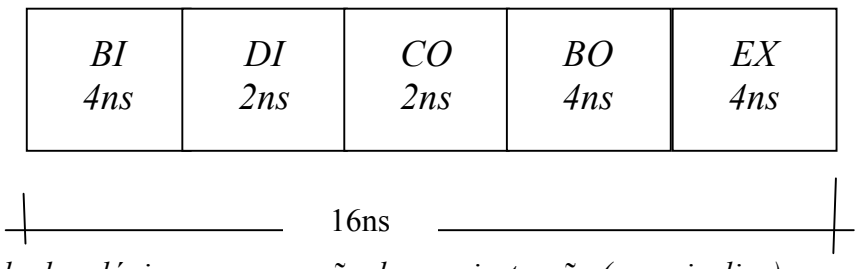
3) (1,5) Escreva um programa que utilize as instruções de linguagem de montagem apresentadas na aula 4 para executar o seguinte procedimento. O conteúdo da memória cujo endereço é 15 é lido e verifica-se se o seu valor é 0. Caso seu valor seja 0, o conteúdo de memória cujo endereço é 20 é subtraído do conteúdo de memória cujo endereço é 30 e o resultado é armazenado no endereço 40. Caso contrário, o conteúdo de memória cujo endereço é 20 é somado ao conteúdo de memória cujo endereço é 30 e o resultado é armazenado no endereço 40. Além de apresentar seu programa escrito em linguagem de montagem, apresente também o programa traduzido para linguagem de máquina.

Obs.: Endereços foram considerados em hexadecimal, mas podem ser utilizados em decimal, lembrando que na linguagem de máquina devem ser convertidos para binário. O tamanho das instruções é igual ao tamanho da célula (12bits) e igual ao modelo hipotético da aula 4.

Endereço (hexa)	***Instrução***	***Descrição***	***Linguagem Máquina (bin / hexa)***
00	*LDA 15*	*ACC <- (15)*	*(000100010101 / 115)*
01	*JZ 07*	*se ACC=0, CI <- 06*	*(010100000110 / 506)*
02	*LDA 20*	*ACC <- (20)*	*(000100100000 / 120)*
03	*ADD 30*	*ACC <- ACC + (30)*	*(001100110000 / 330)*
04	*STR 40*	*(40) <- ACC*	*(001001000000 / 240)*
05	*JMP 09*	*CI <- 09*	*(100000001001 / 809)*
06	*LDA 30*	*ACC <- (30)*	*(000100110000 / 130)*
07	*SUB 20*	*ACC <- ACC - (20)*	*(010000100000 / 420)*
08	*STR 40*	*(40) <- ACC*	*(001001000000 / 240)*
09	*HLT*	*Encerra Procedimento*	*(000000000000 / 000)*

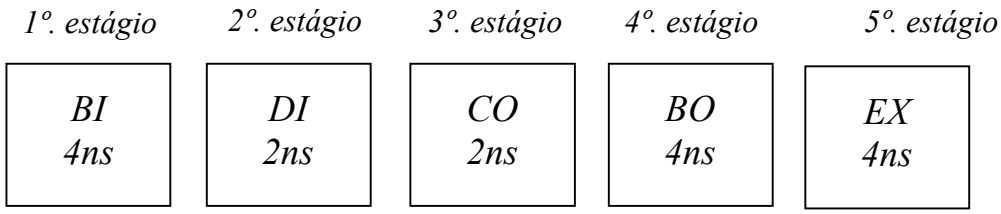
4) (1,0) Considere uma máquina que pode ter seu ciclo de busca e execução de uma instrução dividido em 5 estágios totalmente independentes: Busca de Instrução (BI), Decodificação (DI), Cálculo de Endereços de Operandos (CO), Busca dos Operandos (BO) e Execução (EX). Cada um dos estágios BI, BO e EX possui a duração de 4 ns e cada estágio DI e CO tem duração de 2 ns. Cada instrução desta máquina precisa executar os 5 estágios que serão sempre executados na sequência BI, DI, CO, BO e EX.

a) Uma implementação desta máquina foi realizada de modo que cada instrução deve ser completamente realizada em um único ciclo de relógio. Calcule a duração do ciclo de relógio que esta implementação deve possuir. Lembre-se que todas as instruções necessitam dos 5 estágios.



*Ciclo de relógio para execução de uma instrução (sem pipeline) = 4ns + 2ns + 2ns+ 4ns + 4ns = **16ns***

b) Como cada estágio é independente um do outro, deseja-se implementar uma nova arquitetura utilizando-se um pipeline de 5 estágios. Nesta nova implementação cada estágio do pipeline deve ser executado em um ciclo de relógio. Calcule a duração do ciclo de relógio que esta implementação pipeline deve possuir.



*Ciclo de relógio será igual ao tempo para execução do estágio de maior tempo de execução = 4**ns.***

c) Considere um programa que necessita executar 100 instruções. Calcule o tempo de execução deste programa na máquina do item a e na máquina do item b. Caso as duas máquinas custem o mesmo preço, qual das duas você compraria para executar este programa?

Seja Tex = tempo de execução de uma instrução
= número de estágios x ciclo de relógio (determinado nos itens anteriores)

Item a (sem pipeline) :
Tex = 1 estágio x 16ns para execução de instrução = 16ns
*Ttotal = 100 instruções x Tex = **1600ns***

Item b (pipeline: 4 estágios)
Tex (primeira instrução) = 5 estágios x 4ns = 20ns
Ttotal = Tex + 99 x (tempo do ciclo do relógio)
*Ttotal = 20ns + 99 x 4ns = **416ns***

A máquina escolhida para a compra será aquela que alcance o melhor desempenho, pois possuem os mesmos valor de mercado. Baseado nos resultados acima fica claro que a escolha será pela 2a. Máquina (com pipeline).

5) (1,0) Faça uma pesquisa e indique um microprocessador comercial, cuja unidade de controle possa ser caracterizada como sendo por hardware e um outro, cuja unidade de controle possa ser caracterizada por microprogramada. Explique detalhadamente as suas indicações e coloque a fonte de sua pesquisa.

Fonte: livro texto da disciplina

As unidades de controle microprogramadas estão presentes nos processadores da arquiteturas CISC Esta arquiteturas tem como exemplo: processadores da série Intel Ix (I3,I5 e I7). A unidade de controle microprogramada é utilizada para se desenvolver a implementação de complexas instruções que não podem ser implementadas em forma de hardware. Isto torna mais fácil de se projetar e flexibilizar uma unidade de controle. A microprogramação tem a vantagem de permitir até a emulação de outro computador, e ainda, que uma instrução possa ser desenvolvida e ser utilizada em diferentes modelos de hardware.

As unidades de controle por hardware estão presentes nos processadores das arquiteturas RISC. Esta arquitetura tem como exemplo: a serie Power PC da Motorla/IBM. Nesta unidade de controle, as microinstruções serão executadas diretamente pelo hardware. Em geral, o número de equações booleanas pode ser muito grande, o que pode tornar difícil a implementação combinatória de um grande número de instruções. A vantagem da implementação por hardware sobre microprogramação está na velocidade de execução das instruções. Esta velocidade de execução pode compensar o baixo número de instruções, o que viabiliza este tipo de arquitetura.

6) (2,0) Para os mesmos microprocessadores escolhidos na questão anterior explique a hierarquia de memória dessas máquinas.

Organização da memória do Intel Ix (I3, I5 e I7)

Fonte base para o texto: http://www.clubedohardware.com.br/artigos/Por-Dentro-da-Microarquitetura-Intel-Sandy-Bridge/2146/1

1º. Nível: Registradores presentes apenas no interior do núcleo das CPUs. Os principais registradores da série Ix são:

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 extendido: XCR0 (64 bits)

2º. Nível: Memória Cache divididas nos subníveis L1, L2 e L3 presentes dentro do chip da CPU. Os processadores I3, I5 e I7 da 2ª. geração utilizam a microarquitetura Sandy Bridge. Esta arquitetura possui uma cache de instruções decodificadas que poderia ser denominada como uma 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 é uma cache intermediária com 256KB por núcleo do chip do processador. A cache L3 é compartilhada entre os núcleos, ou seja, não é ligada a um núcleo em particular. A cache L3 pode ser utilizadas pelo componente gráfico presente no chip do processador para armazenar dados, em especial texturas aumentando o desempenho 3D, sem a necessidade de buscar dados na RAM.

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. Os processadores Ix possuem um controlador de memória DDR3 de dois canais, suportando memórias de até DDR3-1333

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 de memória do processador Power PC.
http://pds.twi.tudelft.nl/vakken/in101/labcourse/instruction-set/
http://www.cebix.net/downloads/bebox/pem32b.pdf

1º. Nível: Os principais registradores do Power PC são:
32 registradores de uso geral (32 ou 64 bits)
32 registradores de ponto flutuante (64 bits)
8 registradores de condição (32bits)
Registrador de status do ponto-flutuante (32 bits)
Registrador contador (equivalente ao CI, 32 ou 64 bits)
Registrador de ligação (32 ou 64 bits)

2º. Nível: Memória Cache dividida em 2 ubníveis L1 e L2. Processadores Power PC (tomado o G4 como exemplo) possui cache L1 também dividida em 2 partes, cada uma com 32Kbytes. E cache L2 com 128K, 512K ou 1M.

3o. Nível e 4o. Nível são similares à série Ix, diferenciando quanto ao controlador de memória que continua presente no chipset.

OBS : O Power PC G4 é bem anterior ao Intel Ix, as arquiteturas mais atuais que evoluiram do Power PC como o

IBM PowerX (Power3,Power6...Power9) passaram a incluir microprogramação em sua arquitetura.

7) (2,0) Considere uma máquina que possa endereçar 1 Giga bytes de memória física, sendo que cada endereço referencia uma célula de 4 bytes. Ela possui uma memória cache que pode armazenar 1 Mega blocos, sendo um bloco por linha e cada bloco possui 2 células. Mostre o formato da memória cache, indicando os campos necessários (tag, bloco) e o número de bits para cada campo, o formato de um endereço da memória principal, indicando os bits que referenciam os campos da cache, e a capacidade em bits que a memória cache deve possuir (pode deixar a conta indicada) para os seguintes mapeamentos:

a) Mapeamento direto.

M*emória Principal*

⇒ *Tamanho da memória (em bytes) = 1Gbytes, como 1 célula contem4 byte, temos, então, N = 256M células*

⇒ *A MP está organizada em blocos de 2 células, (K = 2 células/bloco)*

N = 256M células e K = 2 células / bloco, o total de blocos da MP (B) será:

Total de blocos: B = N / K => B = 256M células / 2 células/bloco => B = 128 M blocos

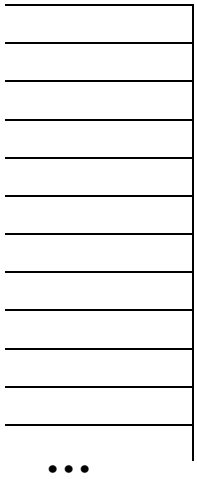
Memória Cache

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

⇒ *Tamanho da memória cache (em blocos ou linhas) => Q = 1M blocos*

⇒ *Tamanho da memória cachê em células = Q x K = 1M blocos x 2 células/blocos*
= 2M células (8 Mbytes)

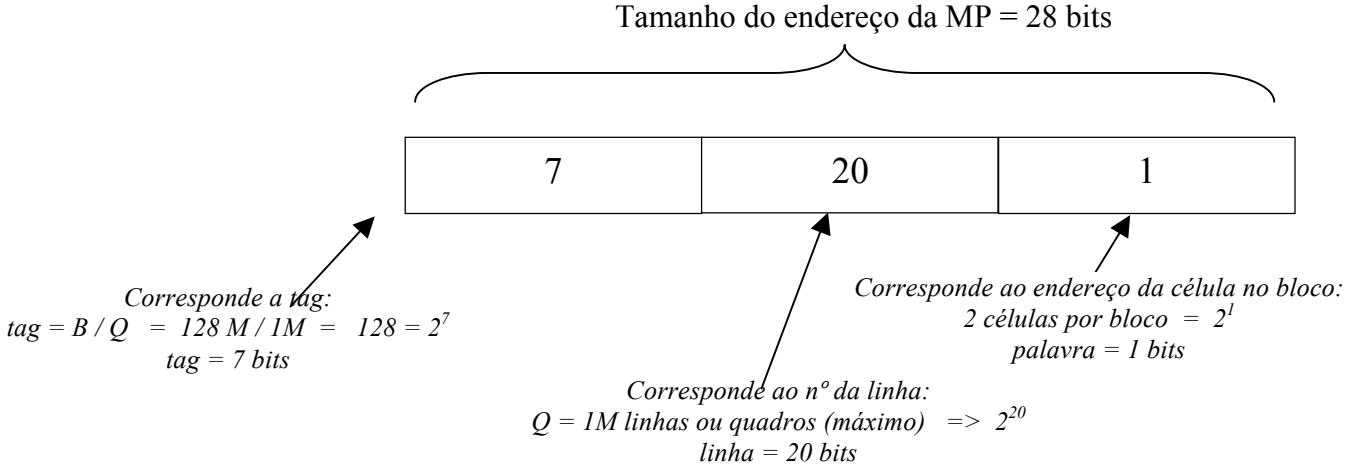
Memória principal
256M células: N
128M blocos: B



Organização da cache

linha	válido	tag	Conteúdo (bloco)
0	1 bit	7 bits	2 células de 32 bits cada = 64bits
1			
2			
3			
4			
5			
.....			
Q - 2			
Q - 1			

Para endereçarmos toda a MP precisamos da seguinte quantidade de bits (E)
sendo $N = 2^E$ => N = 256M células => $N = 2^{28}$ => E = 28 bits

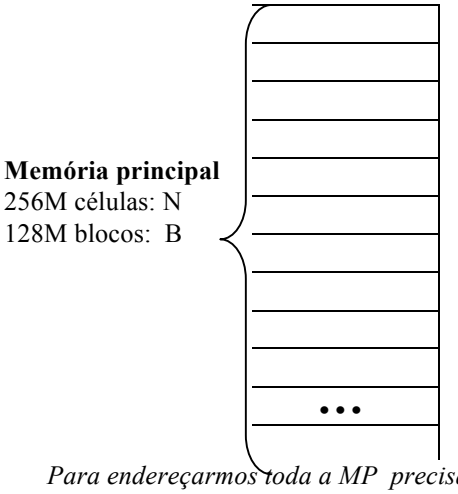


b) Mapeamento totalmente associativo.

Memória Principal
=> N = 256M células
=> K = 2
=> B = 128 M blocos

Memória Cache
OBS: O K (quantidade de células/bloco) tem de ser igual a MP.
=> Q = 1M blocos
=> Tamanho da memória cache = 8 Mbytes

Organização da cache

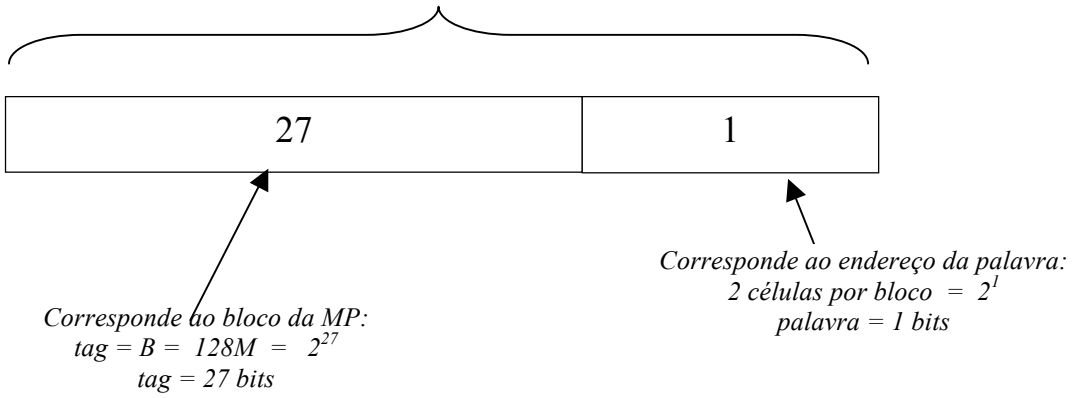


linha	válido	tag	Conteúdo (bloco)
0	1 bit	27 bits	2 células de 32 bits cada = 64bits
1			
2			
3			
4			
5			
.....			
Q - 2			
Q - 1			

Para endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 28 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 = 28 bits

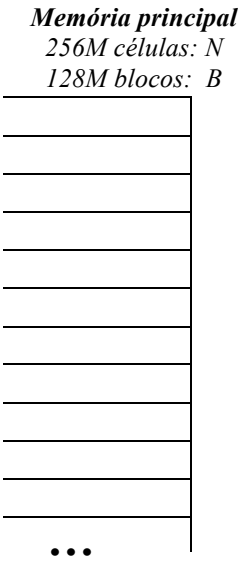


c) Mapeamento associativo por conjunto, onde cada conjunto possui duas linhas, cada uma de um bloco.

Memória Principal
=> K = 2 (quantidade de células/bloco)
=> B = 1M blocos
=> *N = 256M células*

Memória Cache
OBS: O K (quantidade de células/bloco) tem de ser igual ao da MP.

=> Q = 1M blocos
=> Tamanho da memória cache = 8 Mbytes
=> 1 conjunto = 2 linhas (ou quadros) => Total de conjuntos (C) = 1M células / 2 => C = 512K conjuntos



Conjunto	linha	válido	tag	Conteúdo (bloco)
0	0	1 bit	8 bits	2 células de 32 bits cada = 64bits
	1			
1	2			
1	3			
2	4			
	5			
.....				
C - 1	Q - 2			

Para endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 28 bits