Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Organização de Computadores
AP1 2° semestre de 2008 - GABARITO

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

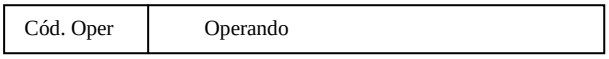
1. (1,0) Explique como funciona um barramento síncrono e um barramento assíncrono.

Nos barramentos que possuem operação síncrona, a ocorrência e duração de todos os eventos que ocorrem nas diversas linhas do barramento são guiados por pulsos de um relógio. Existe uma linha no barramento por onde circulam os pulsos gerados pelo relógio e todos os acontecimentos nas linhas de barramento, como, por exemplo, envio de endereço e envio de sinal de leitura, tem sua inicialização e duração de ocorrência determinadas por estes pulsos. Por exemplo, em uma operação de leitura realizada em um barramento síncrono, os passos serão tomados por base em intervalos de tempo iguais a T, baseados nos ciclos do relógio. O processo de leitura inicia no intervalo T1, onde a linha de início é ativada indicando que o endereço foi colocado no barramento de endereços; no próximo intervalo de tempo T2, os dados serão colocados no barramento de dados pelo dispositivo solicitado e no intervalo T3 é ativada a linha de confirmação.

Nos barramentos que operam de forma assíncrona, não existe um relógio sincronizador. Os eventos ocorrem no barramento de acordo com um protocolo de aperto de mão (handshaking). Cada evento no barramento não depende dos pulsos do relógio, mas sim de algum evento que deve ocorrer anteriormente a ele e que pode ter qualquer duração de tempo. Podemos exemplificar da seguinte forma. Quando uma CPU deseja realizar uma operação de leitura, a unidade de controle coloca o endereço da célula de memória (ou ativação do dispositivo) no barramento e ativa o sinal de leitura no barramento de controle (READ). Logo após estes estarem disponíveis no barramento é ativado o sinal de execução da tarefa (MSYN – ativação do "mestre"), e assim que o dispositivo "escravo" detectar este sinal, ele inicia de forma imediata a operação, com base no endereço, colocando os dados no barramento de dados.Concluindo esta etapa, o dispositivo "escravo" informa que os dados estão disponíveis ativando a linha SSYN. Logo após, o barramento fica disponível para início de outra operação da CPU.

2. (2,0) Suponha que você tenha que projetar uma máquina com as seguintes especificações:

- Capaz de endereçar 2 M células de memória principal, sendo que cada célula armazena 4 bytes.
- Deve possuir os registradores RDM (utilizado para enviar e receber dados para/de uma célula de memória pelo barramento de dados), REM (utilizado para enviar endereço de uma célula no barramento de endereços), CI (utilizado para indicar o endereço da célula da instrução a ser lida da memória) e RI (utilizado para armazenar uma instrução).
- Cada instrução tem o tamanho de 32 bits e deve conter um código de operação e um operando como mostrado abaixo:



onde Operando é um endereço de uma célula da memória principal.

N = 2M células

a) (0,2) Indique qual deve ser o tamanho mínimo em bits do REM.

REM = E = tamanho em bits necessários para acessar toda a memória (N)
N = 2^E = 2M células = 2^{21} células => E = 21 => REM = ***21 bits***

b) (0,2) Indique qual deve ser o tamanho mínimo em bits do barramento de endereços.

Barramento de endereços = REM = ***21 bits***

c) (0,4) Calcule a quantidade máxima possível de códigos de operação diferentes.

Tamanho da instrução = 4 bytes (32 bits)
Tamanho da instrução = código de operação + operando (endereço de memória)
Código de operação = Tamanho da instrução – operando = 32bits – 21 bits = 11bits
Quantidade máxima de cod.oper. = 2^{11} = ***2048 códigos de operações (instruções)***

d) (0,6) Indique **o tamanho do RDM e do barramento de dados** de modo que a Unidade Central de Processamento obtenha uma instrução da memória principal realizando somente um acesso à memória principal.

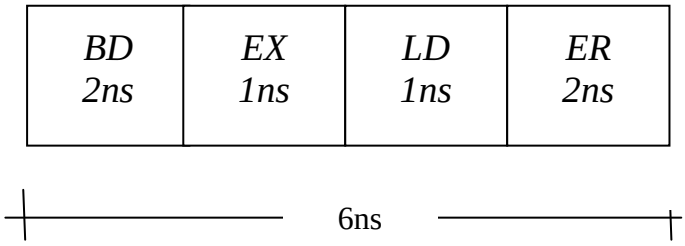
Tamanho do barramento = tamanho da instrução transferida em uma só operação de leitura => ***tamanho do barramento de dados = 32 bits***
RDM = tamanho do barramento de dados => ***RDM = 32bits***

e) (0,6) Calcule a capacidade de armazenamento em bits dos registradores RI e CI, utilizando-se os valores calculados nos itens anteriores.

RI terá que ter no mínimo o tamanho de uma instrução => ***RI = 32bits***
CI terá que ter tamanho que permita endereçar toda a memória => ***CI = 21bits***

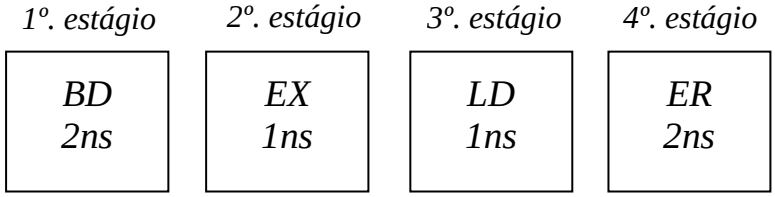
3. (2,0) Considere uma máquina que pode ter seu ciclo de busca e execução de uma instrução dividido em 4 estágios totalmente independentes: Busca e decodificação (BD), Execução (EX), Leitura de Dados (LD) e Escrita de Resultado (ER). Cada um dos estágios BD e ER possui a duração de 2 ns e cada estágio EX e LD tem duração de 1 ns. Cada instrução desta máquina precisa executar os 4 estágios que serão sempre executados na seqüência BD, EX, LD e ER.

   a) (0,5) Uma implementação desta máquina foi realizada de modo que cada instrução deve ser completamente realizada em um único ciclo de relógio e uma instrução só começa a ser realizada após o término da anterior. Calcule a duração do ciclo de relógio que esta implementação deve possuir. Lembre-se que todas as instruções necessitam dos 4 estágios.



*Ciclo de relógio para execução de uma instrução (sem pipeline) = 2ns + 1ns + 1ns + 2ns = **6ns***

   b) (0,5) Como cada estágio é independente um do outro, implementou-se uma **nova** arquitetura utilizando-se um pipeline de 4 estágios. Calcule a duração do ciclo de relógio que a implementação pipeline deve ter. Considere que qualquer estágio do pipeline deve poder ser realizado em um único ciclo de relógio.



*Ciclo de relógio será igual ao tempo para execução do estágio de maior tempo de execução = **2ns.***

   c) (1,0) Mostre o tempo em que um programa que contenha 10 instruções será executado pela **implementação do item a e do item b.** Considere que estas 10 instruções possam ser executadas em fluxo constante.

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 6ns para execução de ‘ instrução = 6ns
*Ttotal = 10 instruções x Tex = **60ns***

Item b (pipeline: 4 estágios)
Tex (primeira instrução) = 4 estágios x 2ns = 8ns
Ttotal = Tex + 9 x (tempo do ciclo do relógio)
*Ttotal = 8ns + 9 x 2ns = **26ns***

4. (1,0) Descreva 2 maneiras possíveis de se implementar uma unidade de controle.

   O *objetivo da unidade de controle é transformar seus sinais lógicos de entrada em um conjunto de sinais lógicos de saída, que constituem os sinais de controle, os quais irão coordenar as atividades da CPU, tais como transferência de dados entre registradores e entre registrador e memória, operações da ULA etc. Duas maneiras possíveis de implementá-la são: por hardware e por microprograma.*

   Para implementar um unidade de controle por hardware *geram-se as equações booleanas necessárias para produzir os sinais de controle de saída como função dos sinais de entrada e cria-se um circuito combinatório que satisfaça estas equações.*

   Em uma unidade de controle microprogramada*, a lógica da unidade de controle é especificada por um microprograma. A unidade é projetada de modo a executar uma seqüência de microinstruções (ou conjunto de microoperações) que ao serem executadas geram sinais de controle para os componentes da UCP para a execução de cada microinstrução.*

5. (2,0) Um computador possui uma capacidade máxima de memória principal de 128 Mcélulas, cada uma capaz de armazenar uma palavra de 8 bits.

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

   Maior endereço (em decimal) = N - 1 = 128M – 1 = 2^{27} - 1 = ***134.217.727***

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

   Barramento de endereço = E = tamanho em bits necessários para acessar toda a memória (N)
   N = 2^{E} = 128M células = 2^{27} células => E = 27 => ***Barramento de endereço = 27 bits***

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

   REM = E = tamanho em bits necessários para acessar toda a memória (N) => ***REM = 27 bits***
   RDM = tamanho da palavra => ***RDM = 8 bits***

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

   T = M x N => T = 8bits/célula x 128Mcélulas => ***T = 1Gbits***

6. (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 16 bytes. Ela possui uma memória cache que pode armazenar 4K blocos, sendo um bloco por linha (ou quadro). 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) = 512Mbytes, como 1 célula referencia a 1 byte, temos* <u>*N = 512M células*</u>

⇒ *Será organizada em blocos de 16 bytes, como 1 célula = 1 byte, temos cada bloco = 16 células, K = 16*

⇒ *Sendo N o tamanho endereçável da memória e K que é a quantidade de células por blocos temos: N = 512M células e K = 16 células / blocos o total de blocos da MP (B) será: Total de blocos: B = N / K => B = 512M células / 16 células/bloco => B = 32 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 = 4K blocos*

⇒ *Tamanho da memória cachê em células = Q x K = 4K blocos x 16 células/blocos = 64K células*

***Endereço da MP**: 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*

Composição do endereço em função da memória cache

=> tag = B / Q = 32 M / 4K = 8K = 2^{13} => tag = 13bits

=> nº da linha: Q = 4K linhas ou quadros (máximo) => 2^{12} => 12bits

=> células por bloco: 16 células por bloco = 2^4 => 4bits

29bits

Tag = 13bits	*No. Linha =12bits*	*Célula no bloco=4bits*

b) Mapeamento totalmente associativo.

Memória Principal

=> N = 512M células

=> K = 16

=> B = 32 M blocos

Memória Cache

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

=> Q =4K blocos

=> Tamanho da memória cache = 64K células

***Endereço da MP** = 29 bits*

Composição do endereço em função da memória cache

=> tag = B = 32M = 2^{25} => tag = 25bits

=> células por bloco: 16 células por bloco = 2^4 => 4bits

29bits

Tag = 25bits	*Célula no bloco=4bits*

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

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

Memória Cache
OBS: O K (quantidade de células/bloco) tem de ser igual a MP.
=> Q = 4K blocos
=> Tamanho da memória cache = 64K células
=> 1 conjunto = 4 linhas (ou quadros) =>
Total de conjuntos => C = 4K blocos / 2 => ***C = 2K conjuntos***

Endereço da MP *= 29 bits*

Composição do endereço em função da memória cache
=> tag = B / C = 32 M / 2K = 16K = 2^{14} => tag = 14bits
=> nº do Conjunto: Q = 2K linhas ou quadros (máximo) => 2^{11} => 11bits
=> células por bloco: 16 células por bloco = 2^{4} => 4bits

29bits

Tag = 14bits	*No. Conjunto =11bits*	*Célula no bloco=4bits*