GABARITO AD1 - Organização de Computadores 2007.1

1. Descreva 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.

   Resposta:

   Na leitura:

   1º. Passo: REM <- de outro registrador da UCP
   REM é atualizado com um endereço contido em um registrador qualquer da UCP.
   O endereço armazenado no REM é colocado no barramento de endereços.

   2º.Passo: A unidade de controle gera um sinal de leitura no barramento de controle.
   É feita a decodificação do endereço e localização da célula

   3º.Passo: RDM <- MP(REM)
   O RDM é carregado com o valor da memória principal relativo o endereço contido no REM, através do barramento de dados.

   4º.Passo: Um outro registrados da UCP é atualizado com o conteúdo do RDM.

   Na escrita:

   1º.Passo: (REM) <- (outro registrador)
   Iinicialmente o REM é atualizado com um endereço contido em um registrador qualquer da UCP. E o endereço armazenado no REM é colocado no barramento de endereços.

   2º.Passo: (RDM) <- outro registrador
   Da mesma forma o RDM é atualizado com um valor contido em um registrador qualquer da UCP.

   3º.Passo: A unidade de controle gera um sinal de escrita no barramento de controle.

   4º.Passo: (MP(REM)) <- (RDM)
   A memória é atualizada com o valor do RDM no endereço dado pelo REM.

2. Um computador possui uma capacidade máxima de memória principal com 64K células, cada uma capaz de armazenar uma palavra de 8 bits.
   a) Qual é o maior endereço em decimal desta memória ?
   Resposta:
   Os endereços variam de 0 até 2^{16} - 1, ou seja, de 0 até 65535
   O maior endereço é portanto 65535

   b) Qual é o tamanho do barramento de endereços deste sistema ?
   Resposta:
   Como a máquina endereça 64K células= 2^{16} células, são necessários 16 bits para endereçá-las. Logo o barramento de endereços tem que ter tamanho igual a 16 bits

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

Resposta:

RDM tem 8 bits, tamanho da palavra da memória principal;

REM tem a mesma quantidade de bits do barramento de endereços, 16 bits.

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

Resposta:

A capacidade da memória é igual ao número de células multiplicado pelo tamanho da célula, ou seja, $64K \times 8 = 512K$ bits

3. 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 8K blocos, sendo um bloco por linha. Mostre o formato da memória cache, indicando os campos necessários (válido, 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.
   b) Mapeamento totalmente associativo.
   c) Mapeamento associativo por conjunto, onde cada conjunto possui quatro linhas, cada uma de um bloco.

Resposta:

Mapeamento direto.

Memória Principal

=> Tamanho da memória (em bytes) = 512Mbytes, como 1 célula referencia a 1 byte, temos N = 512M células

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

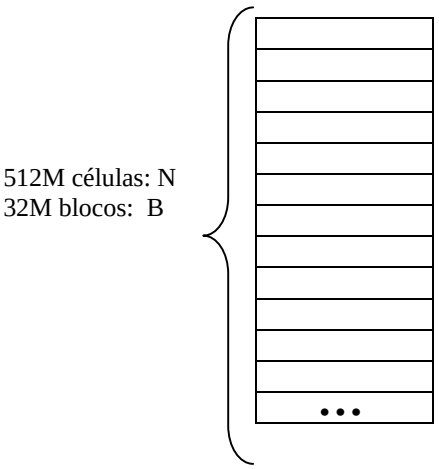
=> Sendo N o tamanho da memória e K a quantidade de células por blocos temos:
N = 512M células e K = 16 células / blocos o total de blocos da MP (B) será:
Total de blocos: B = N / K => B = 512M células / 16 células/bloco => B = 32 M blocos

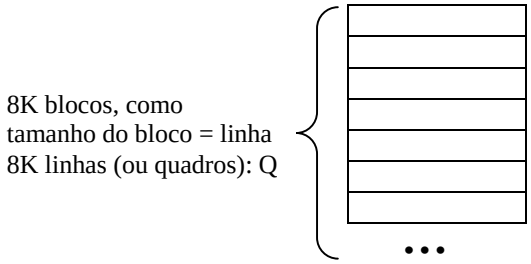
Memória Cache

quantidade de células/bloco tem de ser igual a MP.

=> Tamanho da memória cache (em blocos ou linhas) => Q = 8K blocos

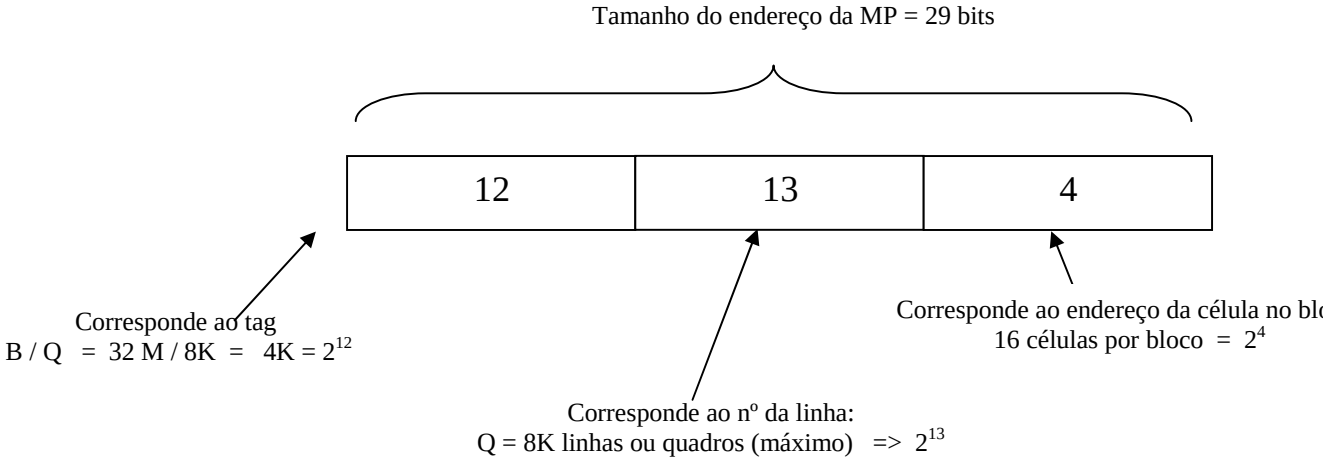
=> Tamanho da memória cache em células => Q x K = 8K blocos x 16 células/blocos = 128K células





Cada célula corresponde a 1 byte (8bits)
Cada bloco (MP) e linha (cache) corresponde a 16 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

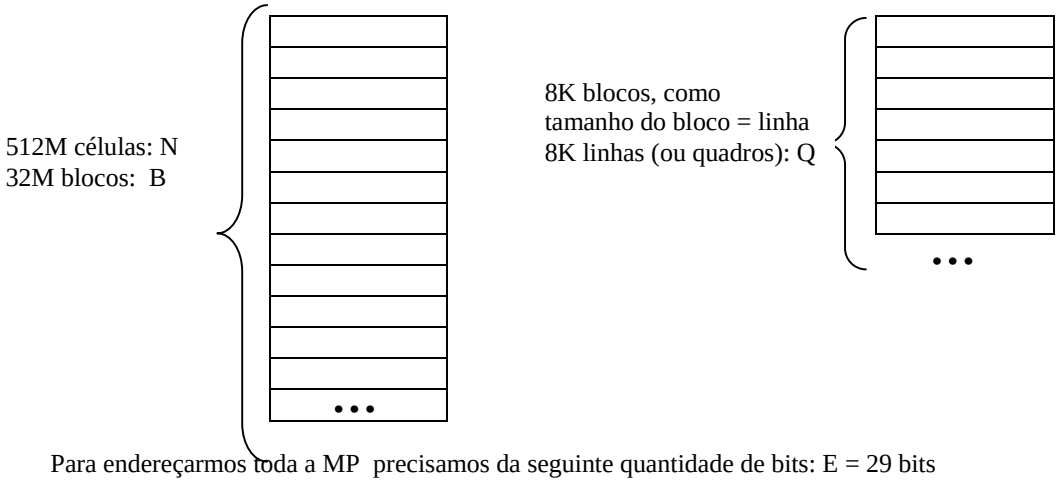


b) Mapeamento totalmente associativo.

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

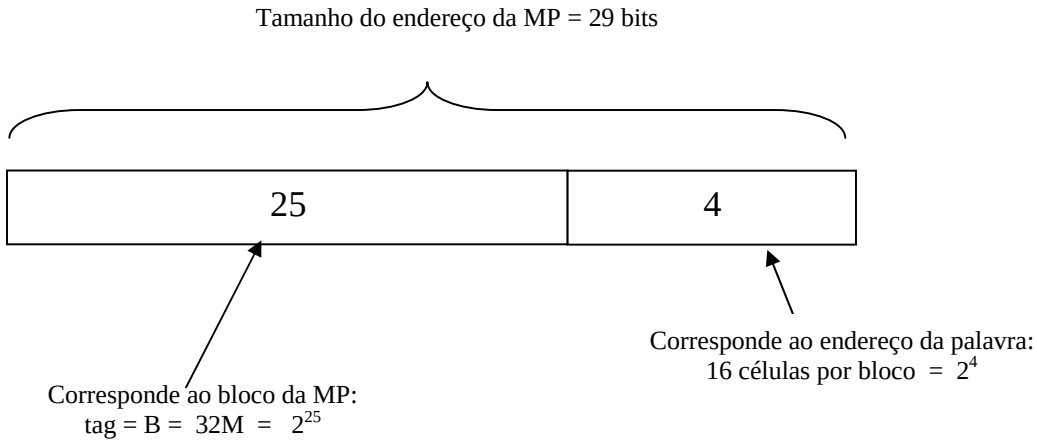
Memória Cache
OBS: quantidade de células/bloco tem de ser igual a MP.

=> Q = 8K blocos
=> Tamanho da memória = 128K 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 cache e tag indicará qual dos blocos da MP está alocado naquela posição da memória cache



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

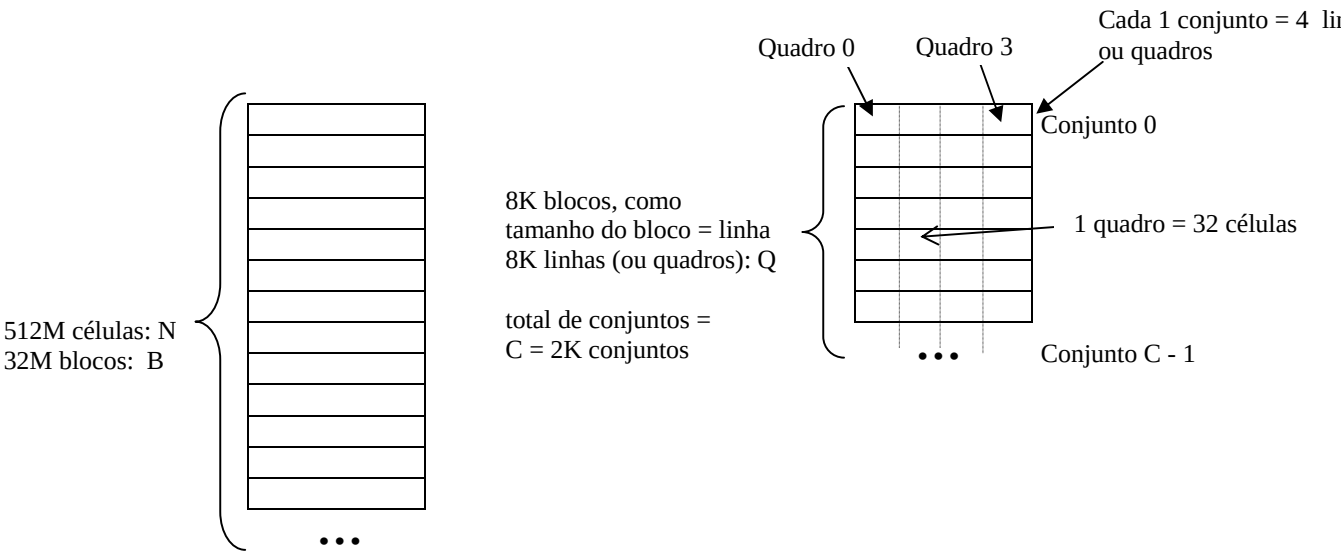
Memória Principal

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

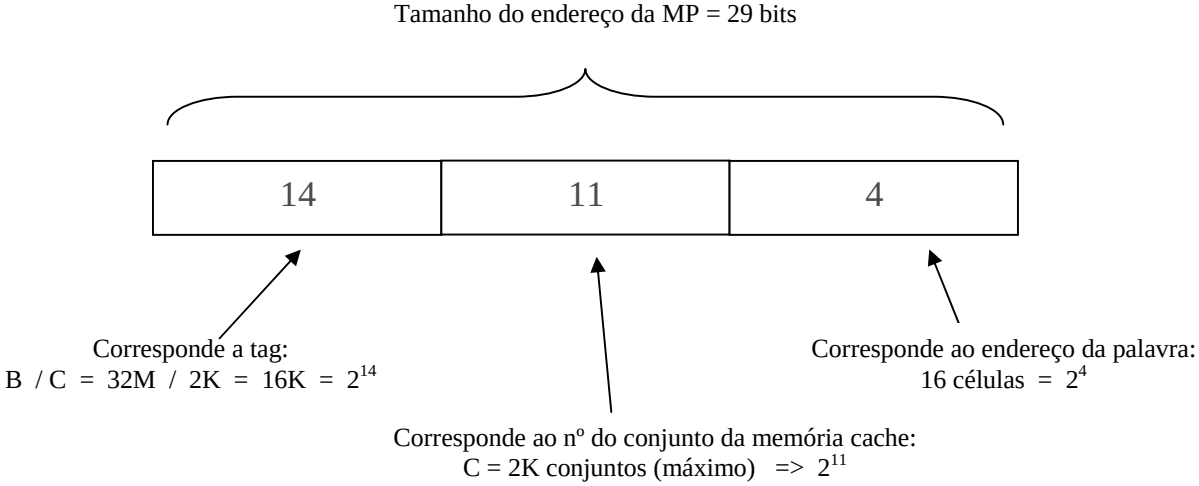
Memória Cache

OBS: quantidade de células/bloco tem de ser igual a MP.

=> Q = 8K blocos
=> Tamanho da memória cache = 128K células
=>1 conjunto = 4 linhas (ou quadros) =>
Total de conjuntos => C = 8K blocos / 4 => **C = 2K conjuntos**



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



4. Explique em detalhes a organização hierárquica do subsistema de memória nos computadores atuais.

Resposta:
Há muitas memórias no computador: 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 de transferência, menor capacidade de armazenamento e maior custo.
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.
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.
Finalmente, na base da pirâmide teríamos a memória secundária, memória auxiliar ou memória de massa, que fornece garantia de armazenamento mais permanente aos dados e programas do usuário.
Alguns dispositivos são diretamente ligados: discos rígido, outros são conectados quando necessário: disquetes, fitas de armazenamento, CD-ROM. Possuem menor velocidade de transferência, maior capacidade de armazenamento e menor custo.

5. Considere uma máquina com 128 K 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 32 códigos de operação distintos, sendo cada uma delas composta do código de operação e um único operando, que indica o endereço de memória.
   a) Qual o tamanho mínimo do REM ?
      Resposta:
      REM = E => 2^E = 128K células = 2^{17} células => E = 17 => REM = 17 bits

   b) Qual o tamanho mínimo do RI ?
      Resposta:
      Para termos 32 códigos de operação. Serão necessários 5 bits ($2^5 = 32$)
      Tamanho da instrução = código de operação + 1 operando = 5 bits + 17 bits (endereço) = 22 bits

   c) Qual o tamanho mínimo do RDM ?
      Resposta:
      O RDM será igual ao tamanho da palavra, portanto RDM = 22bits

   d) Qual o tamanho da memória em bits ?
      Resposta:
      Quantidade de bits = 128K células
      Cada célula armazena uma palavra e cada palavra corresponde ao tamanho de uma instrução, conforme o enunciado, temos então:
      Quantidade de bits (T) = 128K células x tamanho de cada célula (ou tamanho da instrução neste caso) =>
      128K células x 22 bits/célula = 2^{17} x 22 = 2883584 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 ?.
      Resposta:
      Serão necessários 2 ciclos de busca para compor uma instrução que neste problema é o tamanho de uma palavra

6. Considere o sistema visto na aula 4 que possui uma memória com 256 células, sendo que cada célula pode armazenar 12 bits. Cada instrução possui 12 bits, sendo que 4 bits indicam o código de operação e 8 bits indicam o operando. Suponha que, em um determinado momento, alguns endereços da memória contenham os seguintes conteúdos (todos os valores estão em hexadecimal):

Endereço	Conteúdo
20	160
21	470
22	528
23	260
24	180
25	380
26	280
27	820
28	000

Considere que nos endereços 20 a 28 estão armazenadas instruções de um programa.

a) Traduza as instruções para as siglas e operandos correspondentes. Por exemplo, a instrução contida no endereço 20 é LDA 60.
Resposta:

20	LDA 60
21	SUB 70
22	JZ 28
23	STR 60
24	LDA 80
25	ADD 80
26	STR 80
27	JMP 20
28	HLT

b) Supondo que o CI possua o endereço 20, indique como será realizada a execução de cada instrução deste programa e mostre o conteúdo dos registradores RDM, REM, RI, CI e ACC e das células de memória cujos endereços são 60, 70 e 80 ao final da execução deste programa, para cada um dos casos abaixo:

b.1) Inicialmente os conteúdos das células 60, 70 e 80 são 2, 1 e 4 (decimal)
Resposta:

Instrução	RI	ACC	RDM	REM	CI	(60)		(70)		(80)	
						Dec	Hex	Dec	Hex	Dec	Hex
					20	2	2	1	1	4	4
LDA 60	160	2	2	60	21	2	2	1	1	4	4
SUB 70	470	1	1	70	22	2	2	1	1	4	4
JZ 28	528	1	528	22	23	2	2	1	1	4	4
STR 60	260	1	1	60	24	1	1	1	1	4	4
LDA 80	180	4	4	80	25	1	1	1	1	4	4
ADD 80	380	8	4	80	26	1	1	1	1	4	4
STR 80	280	8	8	80	27	1	1	1	1	8	8
JMP 20	820	8	820	27	20	1	1	1	1	8	8
LDA 60	160	1	1	60	21	1	1	1	1	8	8
SUB 70	470	0	1	70	22	1	1	1	1	8	8
JZ 28	528	0	528	22	28	1	1	1	1	8	8
HLT	000	0	000	28	29	1	1	1	1	8	8

b.2) Inicialmente os conteúdos das células 60, 70 e 80 são 4, 1 e 2 (decimal)
Resposta:

Instrução	RI	ACC	RDM	REM	CI	(60)		(70)		(80)	
		Hex	Hex	Hex	Hex	Dec	Hex	Dec	Hex	Dec	Hex
					20	4	4	1	1	2	2
LDA 60	160	4	4	60	21	4	4	1	1	2	2
SUB 70	470	3	1	70	22	4	4	1	1	2	2
JZ 28	528	3	528	22	23	4	4	1	1	2	2
STR 60	260	3	3	60	24	3	3	1	1	2	2
LDA 80	180	2	2	80	25	3	3	1	1	2	2
ADD 80	380	4	2	80	26	3	3	1	1	2	2
STR 80	280	4	4	80	27	3	3	1	1	4	4
JMP 20	820	4	820	27	20	3	3	1	1	4	4
LDA 60	160	3	3	60	21	3	3	1	1	4	4
SUB 70	470	2	1	70	22	3	3	1	1	4	4
JZ 28	528	2	528	22	23	3	3	1	1	4	4
STR 60	260	2	2	60	24	2	2	1	1	4	4
LDA 80	180	4	4	80	25	2	2	1	1	4	4
ADD 80	380	8	4	80	26	2	2	1	1	4	4
STR 80	280	8	8	80	27	2	2	1	1	8	8
JMP 20	820	8	820	27	20	2	2	1	1	8	8
LDA 60	160	2	2	60	21	2	2	1	1	8	8
SUB 70	470	1	1	70	22	2	2	1	1	8	8
JZ 28	528	1	528	22	23	2	2	1	1	8	8
STR 60	260	1	1	60	24	1	1	1	1	8	8
LDA 80	180	8	8	80	25	1	1	1	1	8	8
ADD 80	380	10	8	80	26	1	1	1	1	8	8
STR 80	280	10	10	80	27	1	1	1	1	16	10
JMP 20	820	10	820	27	20	1	1	1	1	16	10
LDA 60	160	1	1	60	21	1	1	1	1	16	10
SUB 70	470	0	1	70	22	1	1	1	1	16	10
JZ 28	528	0	528	22	23	1	1	1	1	16	10
HLT	000	0	0	28	29	1	1	1	1	16	10

7. Faça uma pesquisa no livro “Arquitetura e Organização de Computadores” de William Stallings e descreva como funcionam as arquiteturas que utilizam pipeline.
Resposta:
Nas arquiteturas que utilizam pipeline de instruções, o processamento da instrução é decomposto em estágios que possuem duração aproximadamente igual e podem ser executados de forma independente. Desta forma, uma instrução tem sua execução iniciada antes que a instrução anterior a ela seja finalizada. Nesta arquitetura, o ganho de desempenho da máquina se deve ao aumento do número de instruções que podem ser executadas em uma unidade de tempo.
Um fator importante para o funcionamento eficiente de uma máquina com pipeline é assegurar um fluxo constante de instruções nos estágios iniciais da pipeline. O principal impedimento a isso é a existência de desvios condicionais. Existem várias soluções implementadas para resolver este problema.
A seguir, são descritas alguma implementações pipeline existentes:

Na Arquitetura CISC

Intel 486 implementa uma pipeline de 5 estágios:
- Estágio de Busca: que compreende a busca da instrução
- Decodificação 1: decodificação do código de operação e do modo de endereçamento
- Decodificação 2: expande cada código de operação em sinais de controle para ULA e cálculo de endereços, no caso de modos de endereçamentos mais complexos
- Execução de instrução: execuções de operações pela ULA
- Escrita de resultados: se necessário, atualiza registradores e códigos de condição.

Na arquitetura RISC

Os primeiros processadores RISC tinham uma taxa de execução de aproximadamente uma instrução por ciclo de clock. Foram organizados em duas classes de processadores: com arquitetura superescalar e com arquitetura superpipeline. Uma arquitetura superescalar essencialmente replica cada um dos estágios da pipeline, possibilitando que duas ou mais instruções em um mesmo estágio da pipeline possam ser processadas simultaneamente. Uma arquitetura superpipeline é um refinamento da estrutura da pipeline, que usa maior número de estágios. São exemplos de superescalares os PowerPC e de superpipelines os RS4000.

Em um MIPS R3000, a pipeline avança uma vez por ciclo de relógio. Todas as instruções seguem a mesma seqüência de cinco estágios na pipeline:
- Busca da instrução
- Busca do operando fonte no banco de registradores
- Operação da ULA ou geração de endereço de operando
- Referência ao dado na memória
- Armazenamento do resultado no banco de registradores.

A técnica de superpipeline explora o fato de que muitos dos estágios de uma pipeline desempenham tarefas que requerem um tempo menor que a metade de um ciclo de relógio.
A pipeline do MIPS R4000 tem oito estágios, podendo estar executando até 8 instruções ao mesmo tempo. A pipeline avança a uma taxa de dois estágios por ciclo de relógio.

Os ciclos são:
- Primeira metade da busca de instrução
- Segunda metade da busca de instrução

} 1º.ciclo de clock

- Busca de operandos em registradores
- Execução da instrução

} 2º.ciclo de clock

- Primeira metade de cache de dados
- Segunda metade de cache de dados

} 3º.ciclo de clock

- Verificação de rótulos
- Escrita de resultados

} 4º.ciclo de clock

No Power PC, a pipeline consiste nas seguintes etapas:
1º. ciclo de busca: comum a todas as instruções e ocorre antes que a instrução seja despachada para uma unidade particular.
2º. ciclo de despacho: começa com o despacho de uma instrução para uma unidade particular.

a) Caso seja uma instrução de desvio, este 2º. ciclo envolve decodificação e execução
b) Caso de carga/armazenamento: existe um ciclo de geração de endereço (3º.ciclo), seguido do endereço resultante para a cache (4º.ciclo) e, caso necessário, de um ciclo de escrita de resultado (5º.ciclo).
c) Caso de números inteiros: Neste caso a cache não é envolvida, existindo apenas um ciclo de execução (3º. ciclo) seguido por uma escrita em registrador (4º.ciclo).
d) Caso de ponto flutuante : Terá o ciclo de decodificação (3º. Ciclo) e segue uma pipeline semelhante, mas existem dois ciclos de execução (4º. e 5º.ciclo), refletindo a complexidade das operações de ponto flutuante e o de resposta (6º.ciclo)

O UltraSPARC II emprega uma pipeline de instruções de nove estágios que se divide em duas partes: instruções de número inteiro e instruções de ponto flutuante e gráficas. São estas as etapas:

- Busca: onde até quatro instruções são buscadas na cache de instruções de cada vez
- Decodificação: decodificação das instruções
- Agrupamento: seleciona até quatro instruções da área de armazenamento. Duas podem ser de número inteiro e duas de ponto flutuante
- Execução: duas ULAs de número inteiro efetuam acesso ao banco de dados de registradores e executam as instruções de número inteiro
- Cache: estágio em que se determina se o endereço virtual do estágio anterior resulta em acerto ou falta na cache
- N1: quando ocorre falta na cachê, a instrução entra na fila de carga ou na fila de armazenamento
- N2: esse estágio é simplesmente um estágio de espera para a pipeline de ponto flutuante
- N3: as exceções de programa são tratadas durante este estágio
- Escrita: resultados de operações são escritos nos bancos de registradores nesse estágio

8. Explique como funcionam os barramentos síncrono e assíncrono. Faça uma pesquisa sobre o barramento PCI e indique se ele é síncrono ou assíncrono.
Resposta:
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 de 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.

O barramento PCI é síncrono.

9. Explique como funciona uma arquitetura com unidade de controle microprogramada.
   Resposta:
   A unidade de controle microprogramada é aquela em que a lógica da unidade de controle é especificada por um microprograma. O projeto desta unidade visa executar uma seqüência de microinstruções e gerar sinais de controle para os componentes da UCP para a execução de cada microinstrução.
   A unidade de controle microprogramada possui quatro componentes básicos: memória de controle, registrador de endereço de controle, registrador de microinstrução e unidade de seqüenciamento. A unidade de seqüenciamento envia um comando de leitura para a memória de controle e a microinstrução, cujo endereço é especificado no registrador de endereço de controle, é lida para o registrador de microinstrução. Parte do conteúdo deste registrador (microinstrução lida) gera os sinais de controle para os outros componentes da UCP para executar a atividade especificada na microinstrução e outra parte é utilizada para gerar informação para a unidade de seqüenciamento sobre o próximo endereço a ser enviado para a memória de controle. Ao concluir cada microinstrução, a unidade de seqüenciamento carrega um novo endereço no registrador de endereço de controle, que dependendo dos bits de condição da ULA, pode ser tomada a decisão de buscar a próxima instrução ou desviar para uma nova rotina com base em uma microinstrução de desvio ou ainda, desviar para uma rotina de instrução de máquina.