AD1 - Organização de Computadores 2013.1
Gabarito

1) (1,0) Considere uma máquina com arquitetura semelhante àquela apresentada em aula. Pode-se endereçar no máximo 64K 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 256 códigos, e dois operandos, que indicam endereços de memória.

a) Qual o tamanho mínimo do CI ?

REM = E = tamanho em bits necessários para acessar toda a memória (N)
N = 2^E = 64K células = 2^{16} células => E = 16 => REM = 16 ***bits***
O CI deverá ter o tamanho (em bits) necessário para acessar toda a memória:
CI = REM= 16 ***bits***

b) Qual a capacidade máxima da memória em bits ?

N = 64K células = 2^{16} células
Célula deverá ter o tamanho necessário para armazenar uma única instrução
instrução = cod.oper. + 2 operandos
cod.oper. terá o tamanho necessário para 256 instruções, cod.oper. = 8 bits (2^8 = 256),
Operando deverá ter o tamanho de um endereço de uma célula, operando = 16 bits
instrução = 8 bits + 2 x 16 bits = 40 bits
M = mesmo tamanho da instrução = 40 bits
T = M x N = 40 bits/célula x 2^{16} células = 2.621.440 bits

c) Qual o tamanho mínimo do REM ?

REM = E = tamanho em bits necessários para acessar toda a memória (N)
N = 2^E = 64K células = 2^{16} células => E = 16 => REM = ***16 bits***

d) Qual o tamanho mínimo do RI ?

RI deverá ter o tamanho mínimo para armazenar uma única instrução
O RI deverá ter o tamanho mínimo de 40 bits

e) Qual o tamanho do barramento de endereços ?

Barramento de endereços = E
N = 2^E = 2^{16}, portanto E = 16, Barramento de endereços = ***16 bits***

d) 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 leitura para a transferência de uma instrução.

2. (1,0) Considere uma máquina cujo relógio possui uma freqüência de 4 GHZ e um programa no qual são executadas 500 instruções desta máquina.

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

4GHz = 4.000.000.000 Hz
Tempo de um ciclo de relógio = 1/4.000.000.000 = 0,000 000 000 25 seg ou 0,25ns (nanosegundos)
Tempo de execução de 1 instrução = 2 ciclos de relógio = 0,5ns

*500 instruções **executadas sequencialmente** = 500 x 0,5ns = **250ns** **(para executar 500 instruções)***

b) Considere que essa máquina utilize um pipeline de 4 estágios, todos de igual duração. Calcule o tempo máximo que o estágio deve durar para que o tempo de execução do programa seja menor do que o tempo calculado no item anterior

Tempo total para execução das 500 instruções deverá < que 250ns que foi o tempo do 1º.caso
Consideremos t como sendo o tempo para execução de um estágio
Tempo para execução de uma instrução = 4 estágios x t = 4t
*Tempo para execução das demais em **pipeline** = 499 x t = 499t*
Tempo para execução das 500 instruções = 4t (primeira instrução) + 499t (para as demais) = 503t

O tempo total para excução do 2º. Caso < tempo total execução do 1º. Caso =>
503t < 250ns => t < 250/503 => t < 0,497ns ou t_{max} = 0,497ns (tempo para um estágio)

3. (0,5) 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.

Passos de uma operação de leitura
1) (REM) <- (outro registrador da UCP)
1.1) O endereço é colocado no barramento de endereços
2) Sinal de leitura é colocado no barramento de controle
2.1) Decodificação do endereço e localização da célula na memória
3) (RDM) <- (MP(REM)) pelo barramento de dados
4) (outro registrador da UCP) <- (RDM)

Passos de uma operação de escrita
1) (REM) <- (outro registrador)
1.1)O endereço é colocado no barramento de endereços
2) (RDM) <- (outro registrador)
2.1)O dado é colocado no barramento de dados
3) Sinal de escrita é colocado no barramento de controle
4) (MP(REM)) <- (RDM)

4. (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) LDA 100

a) RI <- Instrução lida
b) CI <- CI + 1
c) Decodificação do código de operação
- recebe os bits do código de operação
- produz sinais internos para a execução da operação de leitura em memória
d) Busca do operando na memória
- A UC emite sinais para que o valor do campo operando = 100 seja transferido para a REM
- A UC emite sinais para que o valor contido no REM seja transferido para o

barramento de endereços

- A UC ativa a linha READ do barramento de controle

- Conteúdo da posição da memória com endereço contido no barramento de

endereço (100) é transferido através do barramento de dados para o RDM
.
- O conteúdo do RDM é transferido para o registrador acumulador (ACC <- RDM)

b) JN 50

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 <- 50)
d) Inicia o procedimento de leitura da instrução contida no endereço que consta em CI

5. (1,0) Explique detalhadamente como funciona uma unidade de controle microprogramada e explique as diferenças existentes entre microinstruções verticais e horizontais.

Em uma arquitetura microprogramada, a unidade de controle é especificada por um microprograma que consiste de uma seqüência de instruções de uma linguagem de microprogramação. Estas instruções são muito simples e especificam microoperações. Uma unidade de controle microprogramada é implementada com circuitos lógicos e é capaz de seguir uma seqüência de microinstruções gerando sinais de controle para que cada uma delas seja executada. Os sinais de controle gerados por uma microinstrução são usados para causar transferências de dados entre registradores e memória e execução de operações pela ULA.

As microinstruções horizontais tem como característica ter funções distintas para cada bit que a compõe, como por exemplo, controlar uma linha de controle interna da UCP, controlar uma linha de barramento externo de controle, definir condição de desviro e endereço de desvio entre outras. Tem a vantagem de ser simples e direto possível, podendo controlar várias microoperações em paralelo, além de uma eficiente utilização do hardware. E possui a desvantagem de maior ocupação de espaço de memória de controle em relação à microinstrução vertical.

As microintruções verticais se caracterizam por possuir um decodificador extra para identificar quais as linhas que serão efetivamente ativadas. Sua principal vantagem é reduzir o custo da Unidade de controle em função do menor tamanho da instrução, o que poderá ser necessário uma maior quantidade de instruções. Tem como principal desvantagem a redução do tempo devido à necessidade da decodificação dos campos de cada microinstrução.

6. (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 60 é lido e verifica-se se o seu valor é menor que 0. Caso seu valor seja menor que 0, o conteúdo de memória cujo endereço é 80 é adicionado ao conteúdo de memória cujo endereço é 60 e o resultado é armazenado no endereço 60. Caso contrário, o conteúdo de memória cujo endereço é 50 é multiplicado por 3 e o resultado é armazenado no endereço 80. Além de apresentar seu programa escrito em linguagem de montagem, apresente também o programa traduzido para linguagem de máquina.

OBS: Considerando endereços em hexadecimal = tamanho da célula = 12 bits e códigos de operação com 4bits.

Endereço	***Instrução***	***Descrição***	***Linguagem Máquina***
Em hexad			*Considerando Endereços em hexa*
10	*LDA 60*	*ACC <- (60)*	*(0001 01100000 / 160)*
11	*JN 17*	*se ACC<0 CI <- 17*	*(0111 00010111 / 717)*
12	*LDA 50*	*ACC <- (50)*	*(0001 01010000 / 150)*
13	*ADD 50*	*ACC <- ACC + (50)*	*(0011 01010000 / 350)*
14	*ADD 50*	*ACC <- ACC + (50)*	*(0011 01010000 / 350)*
15	*STR 80*	*(80) <- ACC*	*(0010 10000000 / 280)*
16	*HLT*	*Encerra*	*(0000 00000000 / 000)*
17	*ADD 80*	*ACC <- ACC + (80)*	*(0101 10000000 / 380)*
18	*STR 60*	*(60) <- ACC*	*(0010 01100000 / 260)*
19	*HLT*	*Encerra*	*(0000 00000000 / 000)*

7. (1,0) Faça uma pesquisa no livro “Arquitetura e Organização de Computadores” de William Stallings e descreva os métodos utilizados para lidar com desvios condicionais em arquiteturas que utilizam pipeline.

O Desvio condicional consiste no principal impedimento para um fluxo constante de instruções em um pipeline de instruções. Diversas abordagens são adotadas para lidar com o desvio condicional, entre elas:

Múltiplos fluxos: *Consiste em duplicar os estágios iniciais da pipeline para permitir a busca de ambas as instruções, usando assim dois fluxos de instruções. Esta abordagem apresenta problemas como o atraso à contenção de acesso a registradores e à memória, além da entrada de instruções adicionais na pipeline antes que seja tomada a decisão sobre o desvio original.*

Busca antecipada da instrução-alvo de desvio*. Consiste em buscar antecipadamente tanto a instrução-alvo de desvio quanto a instrução consecutiva ao desvio, no instante em que uma instrução de desvio-condicional é reconhecida. A instrução-alvo é armazenada em um registrador, até que a instrução de desvio seja executada. Seja o desvio tomado ou não, a próxima instrução a ser executada terá sido buscada antecipadamente.*

Memória de laço de repetição: *Consiste em usar uma pequena memória de alta velocidade, mantida pelo estágio de busca de instrução da pipeline, que é usada para manter as* ***n*** *instruções buscadas mais recentemente, em seqüência. Na ocorrência do desvio, será verificado se a instrução-alvo está presente nesta memória.*

Previsão de desvio*: Consiste em várias técnicas que podem ser usadas para prever se um desvio será tomado ou não, entre as mais comuns estão as seguintes.*

⇒ *Prever que o desvio nunca será tomado: esta previsão não depende do histórico de execução de instruções até o momento, não supõe que o desvio não seja tomado e continua a buscar instruções na seqüência em que ocorrem no programa.*

⇒ *Prever que o desvio sempre será tomado: conforme a anterior diferindo que o desvio será tomado, buscando sempre as próximas instruções a partir do endereço-alvo do desvio.*

⇒ *Prever se o desvio será tomado ou não conforme o código de operação: em casos de determinados códigos de operação, o processador supõe que o desvio sempre é tomado; em outros, que o desvio nunca é tomado, estudo feito com esta técnica apresenta taxas de sucesso superiores a 75%.*

⇒ *Prever o desvio com base em chaves de desvio tomado e de desvio não tomado: consiste em uma estratégia dinâmica onde um ou mais bits podem estar associados a cada instrução de desvio condicional, usados para refletir a história recente da execução da instrução. Bits estes chamados de chaves de desvio tomado ou de desvio tomado ou de desvio não tomado.*

⇒ *Prever o desvio com base em uma tabela de histórico de desvios: Esta técnica mantém uma pequena tabela de histórico de instruções de desvio executadas mais recentemente, incluindo um ou mais bits de entrada. O processador pode endereçar essa tabela de maneira associativa, tal como uma memória cache, ou usar bits de mais baixa ordem do endereço das instruções de desvio. A tabela de histórico de desvios é uma pequena memória cache associada ao estágio de busca de instrução da pipeline.*

Atraso de desvio*: técnica para reordenar automaticamente as instruções de um programa, de modo que instruções de desvio ocorram mais tarde do que ocorrem de fato na seqüência especificada.*

8. (1,5) 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) = 512Mbytes, como 1 célula referencia a 1 byte, temos N = 512M células*

⇒ *Será organizada em blocos de 32bytes, 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 = 512Mbytes e K = 32 células / blocos o total de blocos da MP (B) será:

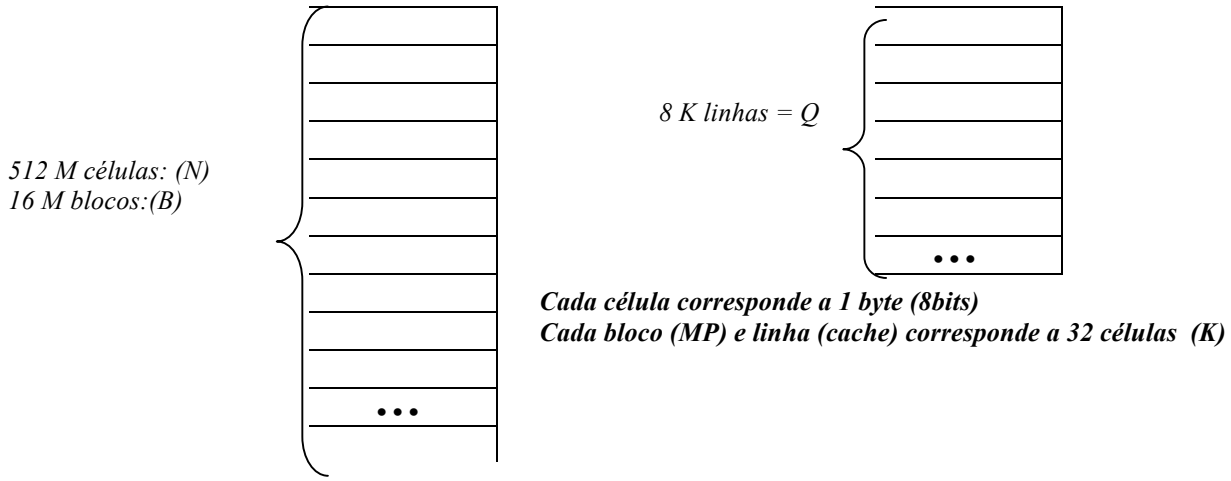
Total de blocos: B = N / K => B = 512Mbytes / 32 células/bloco => B = 16 M blocos

Memória Cache

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

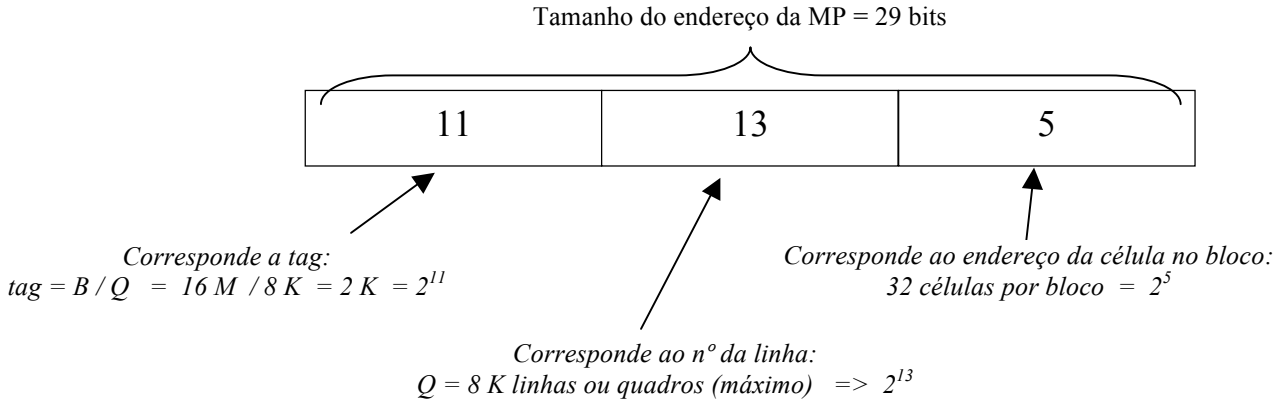
1. Tamanho da memória cache em blocos = 8K blocos

2. Tamanho da memória cache em células = 8K blocos x 32 células/bloco = 256 K células



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

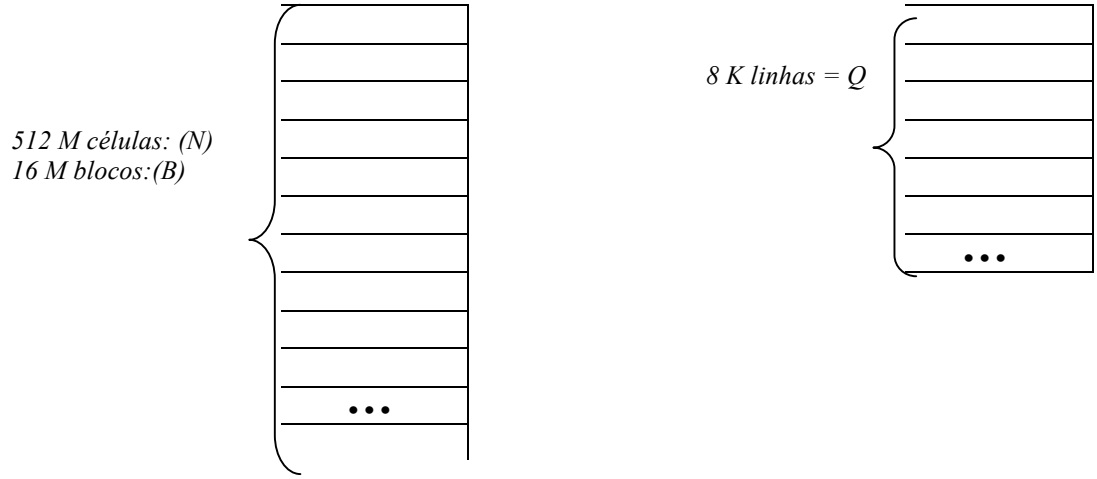
Para endereçarmos toda a MP precisamos da seguinte quantidade de bits (E)
sendo $N = 2^E$ => N = 512M células => $N = 2^{29}$ => E = 29 bits



b) Mapeamento totalmente associativo.

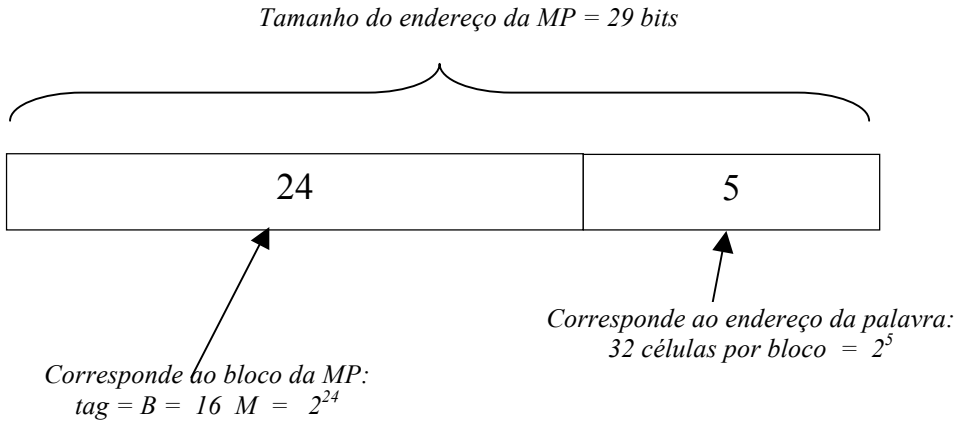
Memória Principal
=> N = 512M células
=> K = 32 células por bloco
=> B = 16 M blocos

Memória Cache
OBS: O K (quantidade de células/bloco) tem de ser igual a MP.
=> Q =8 K blocos
=> Tamanho da memória cache = 256 K 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



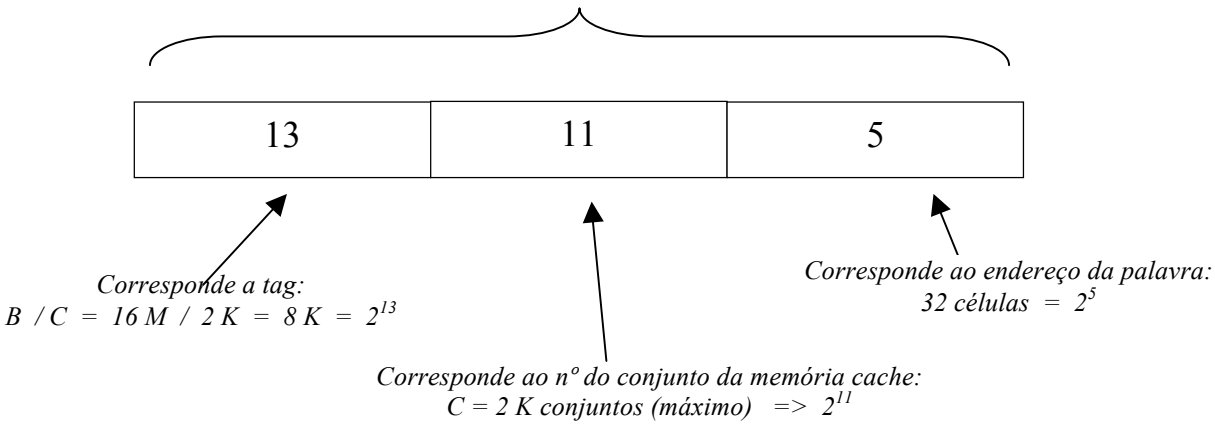
c) Mapeamento associativo por conjunto, onde cada conjunto possui quatro linhas, cada uma de um bloco.

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

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

=> Tamanho da memória cache = 256K células
=> 1 conjunto = 4 linhas (ou quadros) =>
Total de conjuntos => C = 8K células / 4 => ***C = 2 K conjuntos***

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



9. (1,0) Explique em detalhes a organização hierárquica do subsistema de memória nos computadores atuais.

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.

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: disco rígido, outros são conectados quando necessário: disquetes, fitas de armazenamento, CD-ROM. São os mais lentos em comparação com os outros níveis de memória, mas possuem a maior capacidade de armazenamento e armazenam os dados de forma permanente.

10. (1,0) Considere um computador que possua uma UCP com CI de 16 bits e RI de 40 bits. Suas instruções possuem dois operandos do mesmo tamanho (cada um com 16 bits) e um código de operação. Cada célula de memória tem o tamanho igual ao de uma instrução.

a) Qual o tamanho da instrução ?

Tamanho mínimo de RI = tamanho da instrução
tamanho da instrução = 40 bits

b) Qual o tamanho do código de operação ?

Tamanho da instrução = código de operação + 2 operandos
40 = cod.oper. + 2 x 16
código de operação = 8 bits

c) Considerando que a configuração básica dessa máquina é de 16 Kbytes de memória, mostre se é possível aumentar a quantidade de memória desta máquina. Caso seja possível, calcule a quantidade máxima que pode ser adicionada a este sistema.

CI = 16 bits
$N_{max} = 2^{16}$ = *64Kcélulas*
T_{max} = *N x M = 64Kcélulas x 40 bits (5 bytes) = 320Kbytes*
Memória que pode ser adicionada = 320Kbytes - 16 Kbytes = 304Kbytes