! 

**AD1 - Organização de Computadores 2018.1 Data de entrega 13/03/2018**

1. (2,5) Considere uma máquina com arquitetura semelhante àquela apresentada em aula. Pode-se endereçar no máximo 256 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 236 códigos, e dois operandos, que indicam dois endereços de memória.
   1. Qual o tamanho mínimo do REM ? (0,3)

*REM = Barramento de endereços, este terá a capacidade de endereçar 256Mcélulas = N*

*N = 256Mcélulas => N = 228 => e = 28 bits*

*REM = barramento de endereços =* ***28 bits***

* 1. Qual o tamanho mínimo do CI ? (0,3)

*CI terá o tamanho mínimo necessário para endereçar toda a memória = REM =* ***28* bits**

* 1. Qual o tamanho do barramento de endereços ? (0,3)

*REM = barramento de endereços* ***= 28 bits***

* 1. Qual o tamanho mínimo do RI ? (0,5)

*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 236 códigos diferentes (instruções).*

*CodOper = 8 bits (256 códigos diferentes)*

*1º.operando corresponde a 1 endereço de memória = 28 bits*

*2º.operando corresponde a 1 endereço de memória = 28 bits*

*Instrução = CodOper + Operando1 + Operando2 => Instrução = 8 + 28 +28 = 64 bits*

*O tamanho mínimo para RI deverá ser* ***64 bits***

* 1. Qual a capacidade máxima da memória em bits ? (0,5)

*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) = 256M células*

*T = 64 bits/célula x 228 células => T = 16 G bits (ou 2Gbytes)*

* 1. Se a largura do barramento de dados desta máquina for igual à metade do tamanho de uma instrução, como funcionará o ciclo de busca ? (0.6)

*Seriam necessários 2 ciclos de busca para transferir uma instrução completa*

1. (1,0) Considere uma máquina cujo relógio possui uma frequência de 1500 MHZ. Calcule o tempo de UCP utilizado para executar 100 instruções nesta máquina, para cada caso abaixo:

*1.500MHz = 1.500.000.000 Hz*

*Tempo de um ciclo de relógio = 1/1.500.000.000 = 0,000 000 000 6667 seg ou 0,6667ns (nanosegundos)*

* 1. Execução em uma máquina que realiza a execução das instruções de forma sequencial, sendo que cada instrução é executada em um ciclo de relógio.

*Tempo de execução de 1 instrução = 1 ciclo de relógio = 0,6667ns*

*100 instruções* ***executadas sequencialmente*** *= 100* × *0,6667ns =* ***66,67ns (para executar 100 instruções)***

* 1. Execução em uma máquina que utiliza um pipeline de 4 estágios, considerando que o tempo de execução de um estágio é 25% do ciclo de relógio da máquina e não há atrasos na execução do pipeline, devido a desvios condicionais.

*Tempo para execução de 1 estágio = 25% do ciclo de relógio = 25% de 0,6667ns = 0,1667ns*

*Tempo para execução de uma instrução = 4 estágios × t = 4 x 0,1667 = 0,6667ns*

*Tempo para execução das demais em* ***pipeline*** *= 99 × t = 99 x 0,1667 = 16,50ns*

*Tempo para execução das 100 instruções = 0,6667 (primeira instrução) + 16,50 (para as demais) = 17,17ns*

* 1. Execução em uma máquina que utiliza um pipeline de 5 estágios, considerando que o tempo de execução de um estágio é 20% do ciclo de relógio da máquina e não há atrasos na execução do pipeline, devido a desvios condicionais

*Tempo para execução de 1 estágio = 20% do ciclo de relógio = 20% de 0,6667ns = 0,1333ns*

*Tempo para execução de uma instrução = 5 estágios* × *t = 5 x 0,1333ns = 0,6667ns*

*Tempo para execução das demais em* ***pipeline*** *= 99* × *t = 99 x 0,1333ns = 13,2ns*

*Tempo para execução das 100 instruções = 0,6667 (primeira instrução) + 13,2 (para as demais) = 13,87ns*

1. (1,0) Um computador possui uma capacidade máxima de memória principal com 4G células, cada uma capaz de armazenar uma palavra de 8 bits.
   1. Qual é o maior endereço em decimal desta memória ?

*N = quantidade de células = 4G células = 4 x 230 = 4.294.967.296 células*

*Maior endereço em decimal = N – 1 = 4.294.967.296 – 1 = 4.294.967.295*

* 1. Qual é o tamanho do barramento de endereços deste sistema ?

*Para atender a especificação do computador hipotético, a de ter uma memória*

*máxima de 4G células, deveremos ter um barramento de endereços de no mínimo* ***32 bits.***

* 1. Quantos bits podem ser armazenados no RDM e no REM ?

*Barramento de endereços = REM = 32 bits*

*RDM = barramento de dados, como o barramento de dados = tamanho da palavra que será transferida durante um processo de leitura/escrita => RDM = 8 bits*

* 1. Qual é o número máximo de bits que pode existir na memória ?

*Como M = quantidade de bits em uma célula, neste computador, cada célula = palavra*

*então M = 8bits e N = 232*

*Total de bits da memória = T = N x M*

*T = N x M = 232 x 8= 32 G bits*

1. (1,5) Considere uma máquina que possa endereçar 2 Gbytes de memória física, utilizando endereço referenciando byte, e que tenha a sua memória organizada em blocos de 4 Kbytes. Ela possui uma memória cache que pode armazenar 2M 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:

OBS: Informado na plataforma e por email que os 2Mblocos da cache deveriam ser corrigidos para 2Kblocos

* 1. Mapeamento direto.

**M*emória Principal***

* *Tamanho da memória (em bytes) = 2Gbytes, como cada célula contém 1 byte, temos, então, N = 2G células*
* *A MP está organizada em blocos de 4K bytes, como cada célula = 1 byte ( K = 4K células/bloco )*

*N = 2G células e K = 4K células/bloco, o total de blocos da MP ( B ) será:*

*Total de blocos: B = N / K => B = 2G células / 4K células/bloco => B = 512K blocos*

**Memória Cache**

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

* *Tamanho da memória cache (em blocos ou linhas) => Q = 2K blocos*
* *Tamanho da memória cache em células = Q x K = 2K blocos x 4K células/bloco* = 8 M células (8 Mbytes)

**Organização da cache**

**Memória principal**

2G células: N

*512K blocos*: B

|  |  |  |  |
| --- | --- | --- | --- |
| linha | válido | tag | Conteúdo (bloco) |
| 0 | 1 bit | 8 bits | *4K células de 8 bits cada = 32Kbits* |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

…...

|  |  |  |  |
| --- | --- | --- | --- |
| Q - 2 |  |  |  |
| Q - 1 |  |  |  |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

*Para endereçarmos toda a MP precisamos da seguinte quantidade de bits ( E )*

*sendo N = 2E => N = 2G células => N = 231 => E = 31 bits*

Tamanho do endereço da MP = 31 bits

8

11

12

*Corresponde a tag:*

*tag = B / Q = 512K / 2K = 256 = 28*

*tag = 8 bits*

*Corresponde ao endereço da célula no bloco:*

*4K células por bloco = 24*

*palavra = 12 bits*

*Corresponde ao nº da linha:*

*Q = 2K linhas ou quadros (máximo) => 211*

*linha = 11 bits*

* 1. Mapeamento totalmente associativo.

***Memória Principal***

*=> N = 4G células*

*=> K = 4K blocos*

*=> B = 512K blocos*

***Memória Cache***

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

*=> Q = 2K blocos*

*=> Tamanho da memória cache = 8 M células (8 Mbytes)*

**Memória principal**

2G células: N

*512K blocos*: B

**Organização da cache**

|  |  |  |  |
| --- | --- | --- | --- |
| linha | válido | tag | Conteúdo (bloco) |
| 0 | 1 bit | 19 bits | *4K células de 8 bits cada = 32Kbits* |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

…...

|  |  |  |  |
| --- | --- | --- | --- |
| Q - 2 |  |  |  |
| Q - 1 |  |  |  |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

*Para endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 31 bits*

*Como o bloco pode ser alocado em qualquer posição da memória cache. A tag indicará qual dos blocos da MP está alocado naquela posição da memória cache*

Tamanho do endereço da MP = 31 bits

19

12

*Corresponde ao endereço da célula no bloco:*

*4K células por bloco = 24*

*palavra = 12 bits*

*Corresponde ao bloco da MP:*

*tag = B = 512K = 219*

*tag = 19 bits*

* 1. Mapeamento associativo por conjunto, onde cada conjunto possui quatro linhas, cada uma de um bloco.

***Memória Principal***

*=> N = 4G células*

*=> K = 4K blocos*

*=> B = 512K blocos*

**Memória Cache**

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

*=> Q = 2K blocos*

*=> Tamanho da memória cache = 8 M células (8 Mbytes)*

*=> 1 conjunto = 4 linhas (ou quadros) => Total de conjuntos ( C ) = 2K blocos / 4 => C = 512 conjuntos*

**Memória principal**

2G células: N

*512K blocos*: B

**Organização da cache**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Conjunto | linha | válido | tag | Conteúdo (bloco) |
|  | 0 | 1 bit | 10 bits | *4K células de 8 bits cada = 32Kbits* |
|  | 1 |  |  |  |
| 0 | 2 |  |  |  |
|  | 3 |  |  |  |
|  | 4 |  |  |  |
| 1 | 5 |  |  |  |

…...

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| C - 1 | Q - 2 |  |  |  |
|  | Q - 1 |  |  |  |

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

P*ara endereçarmos toda a MP precisamos da seguinte quantidade de bits: E = 31 bits*

*Tamanho do endereço da MP = 31 bits*

10

9

12

*Corresponde a tag:*

*B / C = 512K / 512 = 1K = 210*

*tag = 10 bits*

*Corresponde ao endereço da célula no bloco:*

*4K células por bloco = 24*

*palavra = 12 bits*

*Corresponde ao nº do conjunto da memória cache:*

*C = 512 conjuntos (máximo) => 29*

*conjuntos = 9 bits*

1. (1,5) 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 que pode ser representado na forma de 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.*

1. (1,5) 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:
   1. SUB 20

*1) RI ← (CI) , ou seja, RI ← recebe a instrução contida no endereço contido no CI*

*2) CI ← CI + 1*

*3) Decodificação do código de operação*

*- recebe os bits do código de operação*

*- produz sinais para a execução da operação de subtração*

*4) Execução da operação*

*- A UC emite sinais para que o valor do campo operando (20) seja transferido para a REM*

*- A UC emite sinais para que o valor contido no REM seja transferido para o barram*ento de endereços

*- A UC ativa a linha READ do barramento de controle*

*- Conteúdo da posição da memória, contido no barramento de endereços ( 20 ) , é transferido através*

*do barramento de dados para o RDM*

*- UC emite sinais para transferir conteúdo do acumulador para UAL(1), (UAL ← ACC) - 1o. termo*

*O conteúdo do RDM é transferido para o registrador acumulador (ACC ← RDM)*

*- UC emite sinais para transferir conteúdo do acumulador para UAL(2). (UAL ← ACC) - 2o. termo*

*- A UC emite sinais para a UAL executar da operação se subtração*

*- A UC emite sinais para a UAL liberar resultado e armazenar no acumulador ( ACC ← UAL)*

* 1. JNZ 226

*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*

*( na correção também será considerada a interpretação de ACC<= 0 )*

*CI <- Operando (CI <- 226)*

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

* 1. JP 300

*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 <- 300)*

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

1. (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 100 é lido e verifica-se se o seu valor é 0. Caso seu valor seja 0, o conteúdo de memória cujo endereço é 450 é somado ao conteúdo de memória cujo endereço é 250 e o resultado é armazenado no endereço 300. Caso contrário, o conteúdo de memória cujo endereço é 450 é subtraído do conteúdo de memória cujo endereço é 250 e o resultado é armazenado no endereço 300. Além de apresentar seu programa escrito em linguagem de montagem, apresente também o programa traduzido para linguagem de máquina.

1ª. Opção considerando que os endereços estejam em hexadecimal

|  |  |  |  |
| --- | --- | --- | --- |
| ***Endereço (hexa)*** | ***Instrução*** | ***Descrição*** | ***Linguagem Máquina (bin / hexa)*** |
| *000* | *LDA 100* | *ACC <- (100)* | *( 0001 0001 0000 0000 / 1100 )* |
| *001* | *JZ 006* | *se ACC = 0, CI <- 006* | *( 0110 0000 0000 0110 / 6006 )* |
| *002* | *LDA 250* | *ACC <- (250)* | *( 0001 0020 0101 0000 / 1250 )* |
| *003* | *SUB 450* | *ACC <- ACC - (450)* | *( 0100 0200 0101 0000 / 4450 )* |
| *004* | *STR 300* | *(300) <- ACC* | *( 0010 0011 0000 0000 / 2300 )* |
| *005* | *JMP 009* | *CI <- 009* | *( 1000 0000 0000 1001 / 8009 )* |
| *006* | *LDA 450* | *ACC <- (450)* | *( 0001 0100 0101 0000 / 1450 )* |
| *007* | *ADD 250* | *ACC <- ACC + (250)* | *( 0011 0010 0101 0000 / 3250 )* |
| *008* | *STR 300* | *(300) <- ACC* | *( 0010 0011 0000 0000 / 2300 )* |
| *009* | *HLT* | *Encerra Procedimento* | *( 0000 0000 0000 0000 / 0000 )* |

2ª. Opção considerando que os endereços estejam em decimal

|  |  |  |  |
| --- | --- | --- | --- |
| ***Endereço (hexa)*** | ***Instrução*** | ***Descrição*** | ***Linguagem Máquina (bin / hexa)*** |
| *000* | *LDA 100* | *ACC <- (100)* | *( 0001 0000 0110 0100 / 1064 )* |
| *001* | *JZ 006* | *se ACC = 0, CI <- 006* | *( 0110 0000 0000 0110 / 6006 )* |
| *002* | *LDA 250* | *ACC <- (250)* | *( 0001 0000 1111 1010 / 10FA )* |
| *003* | *SUB 450* | *ACC <- ACC - (450)* | *( 0100 0001 1100 0010 / 41C2 )* |
| *004* | *STR 300* | *(300) <- ACC* | *( 0010 0001 0010 1100 / 212C )* |
| *005* | *JMP 009* | *CI <- 009* | *( 1000 0000 0000 1001 / 8009 )* |
| *006* | *LDA 450* | *ACC <- (450)* | *( 0001 0001 1100 0010 / 11C2 )* |
| *007* | *ADD 250* | *ACC <- ACC + (250)* | *( 0011 0000 1111 1010 / 30FA )* |
| *008* | *STR 300* | *(300) <- ACC* | *( 0010 0001 0010 1100 / 212C )* |
| *009* | *HLT* | *Encerra Procedimento* | *( 0000 0000 0000 0000 / 0000 )* |