Pontifícia Universidade Católica de Minas Gerais
Sistemas de Informação - 2° Período
Arquitetura de Computadores
Nicolas Matheus Ferreira

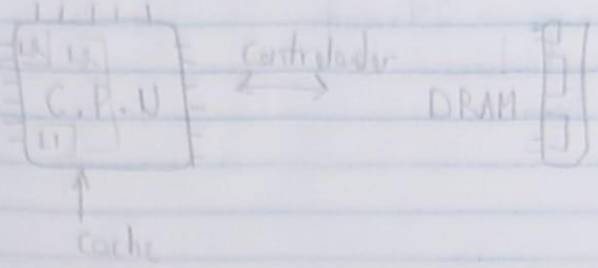
1. Conceitue
a. Memória cache
b. Memória principal
c. Memória secundária
d. Barramento (defina de cada tipo)

a.) A memória cache foi desenvolvida no momento em que a memória RAM não estava mais acompanhando o desenvolvimento dos processadores.

A memória cache é um sub-componente em que opera internamente no processador, realizando a função de guardar dados temporários em que poderão ser solicitados pelo usuário novamente.

É necessário a utilização da memória cache, pois como a um tempo de espera entre a comunicação da memória RAM, e o processador, demora um tempo para recuperar os dados e processar, a memória cache que é mais rápida que a RAM, intermedia entre a CPU e o DRAM guardando informações a medida em que as arquivos vão sendo solicitados e não sendo preciso comunicar novamente com o

assim aumentando a velocidade de processamento.



b.) Memória principal também conhecida como DRAM (Dynamic Random access memory) Armazena (programas em execução, objetos, dados de entrada e saída do S.O) etc.

É uma memória volátil. É memória rápida comparada com a memória secundária (HD, SSD ...) mas ainda lenta se comparada com o registrador e a memória cache.

c.) Memória secundária ou também conhecida como memória externa. É aquela construída por meios magnéticos e diferentemente da memória principal, não é uma memória volátil pois não armazena dados quando o componente está desligado.

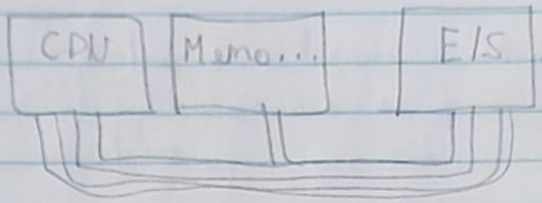
ex.: HD, SSD, fitas magnéticas, pendrive, disquete, CD-ROM, DVD...

MÁXIMA

1.) Barramento é um conjunto elétrico que interliga componentes, permitindo a comunicação e troca de dados entre os dispositivos através do barramento. como (processador, memórias, periféricos etc)

São classificados em três tipos de barramentos

- Barramento de dados - Data Bus
- Barramento de endereços - Addres Bus
- Barramento de controle - Control Bus



A largura é determinada pela quantidade de dispositivos.

Barramento - Conectores para barramentos

SATA, PCI, PCI Expr. USB, FireWire...

2. Explique as diferenças entre as arquiteturas Von Neumen e Harvard.

A diferença principal entre a arquitetura de Neuman e Harvard. se dá que na arquitetura de Neuman.

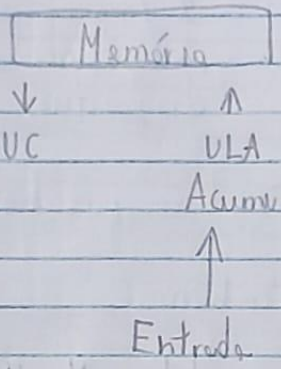
A CPU e a memória principal são ligados diretamente pelos barramentos (BUS) e a arquite-

tura de Harvard, é que a ULA e UC tem um barramento próprio dedicado a ele, e o acesso de ambos são simultâneos.

3.) Conceitue CISC e RISC, Vantagens e desvantagens de cada tipo.

Arquitetura ...

(CISC) - Complex Instruction Set Computer



Vantagem: São embutidos na arquitetura instruções mais simples, simplificado a programação e ocupando menos espaço

Desvantagem: Por ser instruções mais simples impossibilitando de alterar as funções já que são embutidas, são necessário mais acessos e causando perda de desempenho.

(RISC) - Reduced Instruction Set Computer

Vantagens: As funções são compostas assim evitando maiores quantidades de acessos que na arquitetura CISC, ganhando em desempenho.

Desvantagem: Apesar de ganhar no desempenho por ter menos acessos perde no espaço e complexidade pois os programas são maiores e mais complexos.

4. Explique a figura a seguir, referente à arquitetura MIPS. Quais são as principais características dessa arquitetura.

O mips é uma arquitetura baseada em registradores, ou seja, a CPU usa apenas registradores para realizar suas operações aritméticas e lógicas

Existem outros tipos de processadores, tais como processadores baseados em pilha e processadores baseados em acumuladores.

Microprocessor without interlocked pipeline stages.

5. Explique a diferença entre as instruções add, addi e addu

A instrução add, é uma instrução de adição ou soma com 3

Onde você passa a instrução add, passa dois endereços ou um endereço e uma constante ou três endereços de memória

ex.: endu.

li $t0, 75 # Carrega o valor 75 pro registrador $t0

load Imediato

li $t1, 25

add ($t2, $t0, $t1) # Soma dois valores.

registrador armazena a soma dos valor dois end.

equivalente T2 = T0 + T1

ou

addi

Também como o add, soma porém

addi $T0, $T1, (Valor imediato) ← constante

ex.: addi $T0, $T1, 100

equivalente

T0 = T1 + 100

addu

A instrução addu, opera como qualquer outra função de adição add ou addi porém opera apenas com valores using int apenas valores positivos

ex: addu $T0, $T1, +25 ← using int T0

6. Descreva as instruções mult, div e rem.

A instrução mult multiplica os valores contidos nos registradores e armazena o resultado no registrador especial lo

ex: mult $T0, $T1 ← lo

Ao passar a instrução div ele divide o valor que está armazenado em ambos registradores e armazena o registrador de destino

```
.text

li $t0, 4
li $t1, 8

div $t2, $t1, $t0 <- equiv. T2= T1 / T0

syscall
li $v0, 10

.data
```

ou

É possível armazenar o resultado no registrador lo...

rem.

O operador rem, calcula o resto da divisão dos registradores

ex.: rem $t1, $t2, $t3

2, 10 4

resto

MÁXIMA

7.) Qual é o resultado numérico em cada registrador após a execução de cada instrução. Considere os valores iniciais como zero.

```
ori $t0, $zero, 4
addiu $t1, $t1, 4
sub $t2, $t1, $t0
```

ori $t0, $zero, 4 ← Sob em $t0 o valor 4
addiu $t1, $t1, 4 ← Soma em $t1 o 4
sub $t2, $t1, $t0 ← Subtrai $t1 e $t0

8) Qual é o resultado numérico em cada registrador após a execução de cada instrução.

```
addi $t0, $t0, 2
sll  $t0, $t0, 2
rol  $t0, $t0, 2
slr  $t0, $t0, 3
```

Adiciona em $t0, 0 valor 2.
Realiza um deslocamento lógico em 2, 2 e 3.

9) Escreva um programa que coloca o valor 764 em um registrador e divida esse valor por 37. Coloque o quociente em $s0. e o resto em $s1.

```
.text
```

MAXIMA

```
.text

li $T0, 764
li $T1, 37
div $T2, $T0, $T1 # 20
div $T3, $T0, $T1
syscall
```

10.) Escreva um programa que carregue o valor 100000 em um registrador e multiplique esse valor por 1200. Apresente os resultados nos registradores $t2 e $t3

```
.text

li $t0, 100000 # Carrega o valor no registrador
li $s1, 1200 # Carrega o valor no registrador

mul $t2, $s0, $s1 # Multiplica o valor do
mul $t3, $s0, $s1 # resultado em $t2, $t3


Syscall
li $v0, 10

.data
```