Lista 1 de Arquitetura e Organização de Computadores

Respostas

1) Modelo de Von Neumann tem um conceito de programa armazenado e separação da unidade aritmética e unidade de controle, dividida entre os seguintes títulos:

   Memória:

   - A unidade de memória central serve para guardar programas e dados, sob a forma de uma representação binária;
   - Cada instrução de maquina é codificada como uma sequencia de bits;
   - Cada valor de certo tipo é codificado por uma determinada sequencia de bits.

   Unidade Central de Processamento (UCP):

   - Trata do controle global das operações e da execução das instruções;
   - Contem as seguintes unidades internas;
   - Unidade lógica e aritmética (ULA): executa as principais operações lógicas e aritméticas do computador;
   - Unidade de Controle (UC): busca, decodifica e executa.

   Entrada e Saída:

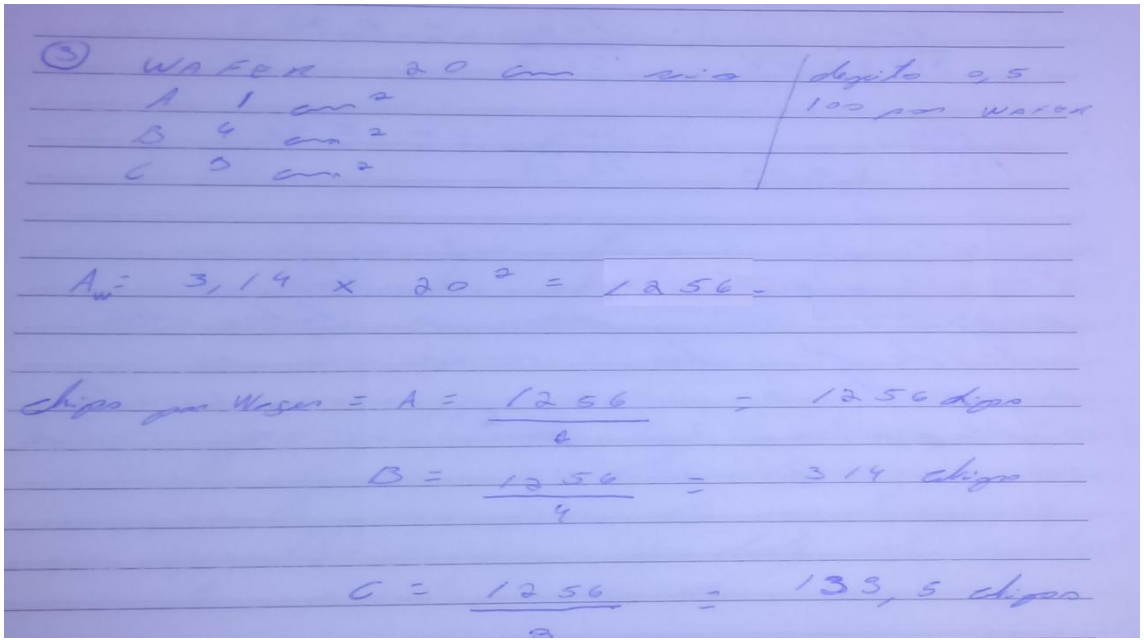
   - As unidades periféricas destinam-se a suportar as ações de comunicação da CPU e memória com o exterior;
   - Também há unidades pacificas destinadas ao armazenamento de dados, que são depois apresentados ao usuário, sob a forma de arquivos, gerados pelos programas do Sistema Operacional.

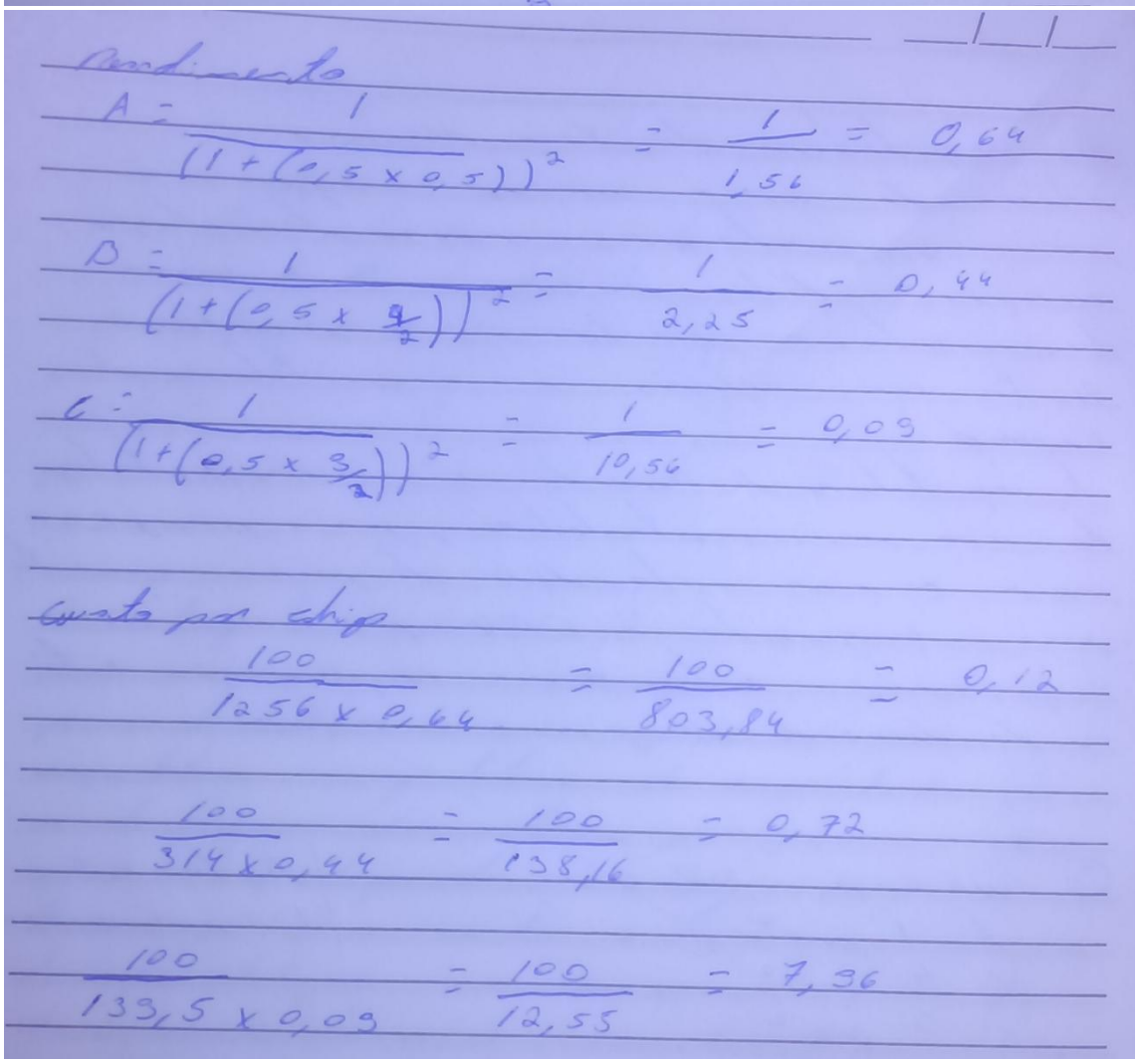
   Processadores:

   - Seu funcionamento é coordenados pelos programas
   - Executa cálculos muitos simples de forma bastante rápida;
   - Trabalha diretamente com a memória principal;
   - Utiliza o ciclo busca,execução regulados pelo clock.
2) A diferença é que a arquitetura de Harvard separa o armazenamento e o comportamento das instruções do CPU e os dados, enquanto a outra utiliza o mesmo espaço de memória para ambos.
   Na arquitetura de Von Neumann (Pricenton) é processada uma única informação por vez, visto que, nessa tecnologia, execução e dados percorrem o mesmo barramento, o que torna o processo lento em relação a arquitetura de Harvard.
   A arquitetura de Harvard é mais utilizada nos microcontroladores, pois proporcionam maior velocidade de processamento, pois, enquanto a CPU

processa uma informação outra nova informação esta sendo buscado, de forma sucessiva.

3)





4) **Compilador 1**

7 instrucoes
(5 x 1) + (1 x 2) + (1 x 3) = 10 x 10^9 ciclos
Tempo = N de instruções/frequência => Tempo = 10 x10^9 /500 x 10^6 = 20s
Mips = N de instruções/Tempo x 10^9 = Mips = 7 x10^9 / 20x 10^6 => Mips = 350

Compilador 2

12 instruções
(10 x 1) + (1 x 2) + (1 x 3) = 15 x 10^9 ciclos
Tempo = N de instruções/frequência => Tempo = = 15 x10^9 /500 x 10^6= 30s
Mips = N de instruções/Tempo x 10^9 = Mips = 12 x 10^9/30 x 10^6 => Mips = 400
Logo para o cálculo do mips podemos concluir que o Compilador 2 pode processar um número maior de instruções

5) A maquina 2 executa o programa 1 com o dobro de velocidade da maquina 1, enquanto a maquina 1 executa o programa 2 1,33 vezes mais rápido que a maquina 2.
6) RISC (Reduced Instruction Set Computer):
    - Parte do pressuposto de que um conjunto simples de instruções vai resultar numa unidade de controle simples, barata e rápida.
    - Numero reduzido de instruções
    - Instruções de mesmo tamanho
    - Muitos registradores
    - Operações somente entre registradores
    - Instruções executadas diretamente em Hardware.

    CISC (Complex Instruction Set Computer):

    - Visa facilitar a construção dos compiladores, assim, programas complexos são compilados em programas de maquina mais curtos;
    - Grande variedade de instruções;
    - Instruções de tamanho variado;
    - Poucos registradores;
    - Operações em memória;
    - Utiliza microcódigo.

7)

```
switch # a partir da posição 1000 de memoria
 beq $s0,$t1,c1 #k=0
 beq $s0,$t2,c2 #k=1
 beq $s0,$t3,c3 #k=2
 beq $s0,$t4,c4 #k=4
 c1:
        add $t0,$s3,$s4
        j EXIT
 c2:
        add $t0,$s1,$s2
        j EXIT
 c3:
        sub $t0,$s1,$s2
        j EXIT
 c4:
        sub $t0,$s3,$s4
        j EXIT

EXIT
```

Memória	OP=6	RS=5	RT=5	RD=5	SHAMT=5	FUNC=6	Tipo
1000	4	16	9	1016			I
1004	4	16	10	1024			I
1008	4	16	11	1032			I
1012	4	16	12	1040			I
1016	0	19	20	8	0	32	R
1020	2	1048					J
1024	0	17	18	8	0	32	R
1028	2	1048					J
1032	0	17	18	8	0	34	R
1036	2	1048					J
1040	0	19	20	8	0	34	
1044							

8)

```
################################
### INSERTION SORT ASSEMBLY
### ############################

.data
arr: .word  5,9,3,6,7,10,8,2,4,1
.text
.globl main
################################insertion_sort###############
insertion_sort:                 #i:$s0 j:$s1 temp:$s2
addi $sp, $sp, -8               #ajusta a pilha para receber 3 ítens
sw $s0, 0($sp)                  #salva o registrador $s0
sw $s1, 4($sp)                  #salva o registrador $s1
add $s0,$0,$0                   #i=0
for:
slt $t0,$s0,$a1                 #i<lenght? $t0=1:$t0=0
beq $t0,$0, exit
add $s1,$s0,$0                  #j=i
while:
beq $s1,$0,exit_while           #j==0?
addi $t1,$s1,-1                 #$t1 = j-1
sll $t2,$s1,2                   #$t2=j*4
sll $t3,$t1,2                   #$t5=(j-1)*4
add $t2,$t2,$a0                 #$t2=&a[j]
add $t3,$t3,$a0                 #t3=&a[j-1]
lw $t4,0($t2)                   #t4=a[j]
lw $t5,0($t3)                   #t5=a[j-1]
slt $t6,$t4,$t5                 #a[j]<a[j-1]?
beq $t6,$0,exit_while
add $s2,$t4,$0                  #temp =arr[j]
sw  $t5,0($t2)                  #arr[j]=arr[j-1]
```

```
lw $t5,0($t3)                    #t5=a[j-1]
slt $t6,$t4,$t5                  #a[j]<a[j-1]?
beq $t6,$0,exit_while
add $s2,$t4,$0                   #temp =arr[j]
sw  $t5,0($t2)                   #arr[j]=arr[j-1]
sw  $t4,0($t3)                   #arr[j-1]=arr[j]
addi $s1,$s1,-1                  #j--
j while
exit_while:
addi $s0,$s0,1                   #i++
j for
exit:
lw $s0, 0($sp)                   #restaura o registrador $s0
lw $s1, 4($sp)                   #restaura o registrador $s1
addi $sp, $sp, 8                 #restaura a pilha
jr $ra
#######################main#############################################
main:
la $a0, arr                      # coloca o endereço de arr em $a0
addi $a1, $0,10                  # lenght= 10
jal insertion_sort
li $v0,1 1                       #código de retorno  1 1 para terminar o programa
syscall
```

9)

10)