Unidade IV

7 PORTAS LÓGICAS

As portas lógicas trabalham muito com a base binária, que é a unidade básica para qualquer sistema digital, podendo ser encontrada em processadores e microprocessadores.

Nas portas lógicas existem duas variáveis importantes, que são a entrada e a saída, em que a função recebe uma quantidade de entrada que será calculada pelas portas lógicas para resultar nas saídas do problema.

Lembrete

Quando se trabalha com portas lógicas, automaticamente estamos manipulando números binários, então é bom saber cálculos binários.

As portas lógicas mais usadas são: AND (E), OR (OU), NOT (NÃO).

7.1 Porta AND

A porta AND, básica, recebe duas entradas e produz uma saída, conforme a Figura 7.1.

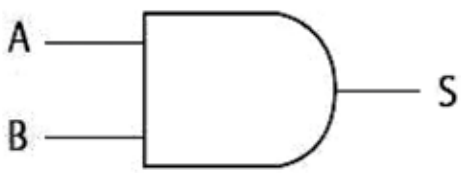


Figura 59 – Porta lógica AND

Nas entradas A e B, são inseridos *bits* e é realizado um cálculo que resulta na saída S. A porta AND está diretamente atrelada à função AND, a qual trabalha com sinais de 0 e 1.

Na tabela 05 é ilustrada a regra da função AND para duas entradas.

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 05 – Função AND

A função AND pode ser comparada à função aritmética de multiplicação, então, na tabela-verdade há as duas entradas (A e B), na qual é realizada a função (similar à multiplicação). Então, teremos na primeira linha da tabela: A=0, B=0; o S será A.B, ou seja, 0.0, que resulta 0. Levando isso em conta, sempre que houver uma das entradas igual a 0, o resultado será sempre 0; o resultado só será 1 quando todas as entradas forem 1.

Caso existam mais entradas, teremos mais linhas na tabela-verdade, conforme é mostrado a seguir na figura 60.

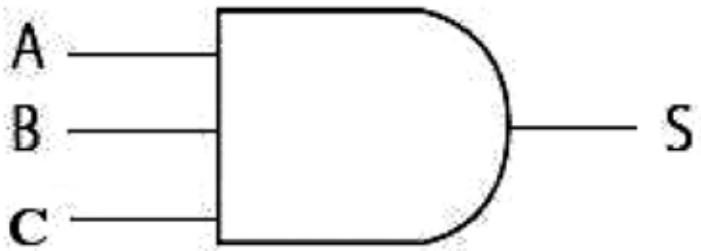


Figura 60 – Porta AND com 3 entradas.

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tabela 06 – Função AND com 3 entradas.

Note que sempre haverá todas as combinações entre as entradas.

Quando se trata de lógica digital, o valor de entrada e saída igual a 0 significa que não há pulso elétrico ou falso e é representado por nível de tensão 0 *V*. Porém, se o valor for 1, isso representa que há sinal elétrico ou verdadeiro, sendo representado pelo nível de tensão +*Vcc*.

Então, pode-se construir a tabela-verdade com os sinais de falso (F) e verdadeiro (V).

A	B	S
F	F	F
F	V	F
V	F	F
V	V	V

Tabela 07 – Função AND

Mas, onde é usado isso?

Pode-se mostrar isso no diagrama de tempo, que é visualizado com a utilização de osciloscópio, onde são demonstrados, por exemplo, os sinais de duas entradas e da saída obtida, e isso obedece a regra da porta AND.

A figura 61 mostra dois sinais de entrada (A, B) e o sinal de saída S.

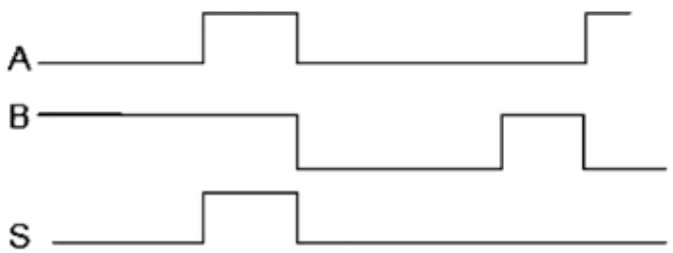


Figura 61 – Sinal de saída usando porta AND

Como saber se está sendo usada a função AND, separa-se em intervalo de tempo (Figura 62).

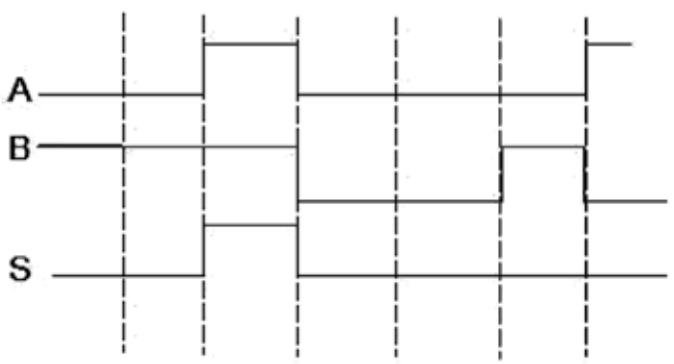


Figura 62 – Sinal de saída usando porta AND

Observa-se que A começa embaixo e B em cima, ou seja, A = 0, B = 1, o que resulta S = 0, então monta-se a tabela envolvendo todos os intervalos (Figura 63).

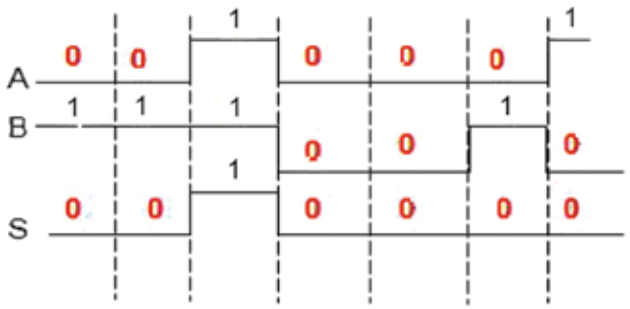


Figura 63 – Sinal de saída usando porta AND

A	B	S (A.B)
0	1	0
0	1	0
1	1	1
0	0	0
0	0	0
0	1	0
1	0	0

Tabela 08 – Função AND

Repare que, mesmo colocando todos os valores existentes nos intervalos dos sinais, bastaria a tabela simples que foi ilustrada na tabela 05 para atingir todas as possibilidades.

Tabela exibindo exemplos reais:

A Elevador parado no andar	B Botão para abrir a porta pressionado	S Elevador tem a porta aberta
F	F	F
F	V	F
V	F	F
V	V	V

Tabela 09 – Função AND para exemplos reais

A Processador livre	B Existe aplicativo requisitando o processador	S Processador processa a informação
F	F	F
F	V	F
V	F	F
V	V	V

Tabela 10 – Função AND para exemplos reais

7.1.1 Porta OR

A porta OR, básica, recebe duas entradas e produz uma saída, conforme a figura 64.

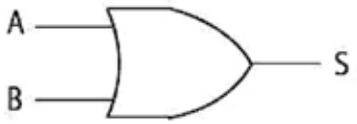


Figura 64 – Porta OR

Nas entradas A e B, são inseridos *bits* e é realizado um cálculo que resulta na saída S. A porta OR está diretamente atrelada à função OR.

Na tabela 11 é ilustrada a regra da função OR para duas entradas.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 11 – Função OR

A função OR pode ser simplificada da seguinte maneira: sempre que houver algum *bit* 1 de entrada, teremos 1 como saída, ou seja, só teremos 0 na saída quando todas as entradas forem 0.

Caso tenha mais entradas, teremos mais linhas na tabela-verdade, conforme é mostrado na figura 65.

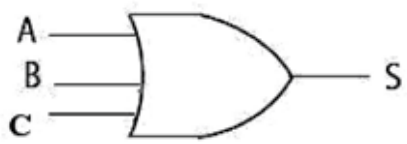


Figura 65 – Porta OR com 3 entradas

A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Tabela 12 – Função OR com 3 entradas

Note que sempre haverá todas as combinações entre as entradas.

Tabela-verdade com os sinais de falso (F) e verdadeiro (V).

A	B	S
F	F	F
F	V	V
V	F	V
V	V	V

Tabela 13 – Função OR

A figura a seguir mostra o diagrama de tempo de dois sinais de entrada (A, B) e o sinal de saída S.

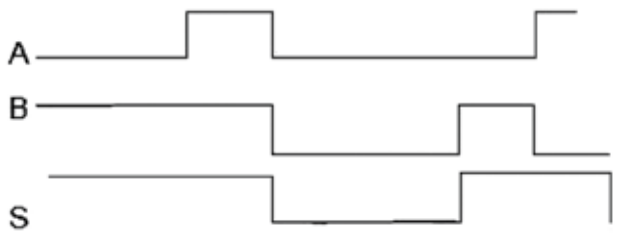


Figura 66 – Sinal de saída usando porta OR

Para saber se a função OR está sendo usada, separa-se em intervalo de tempo:

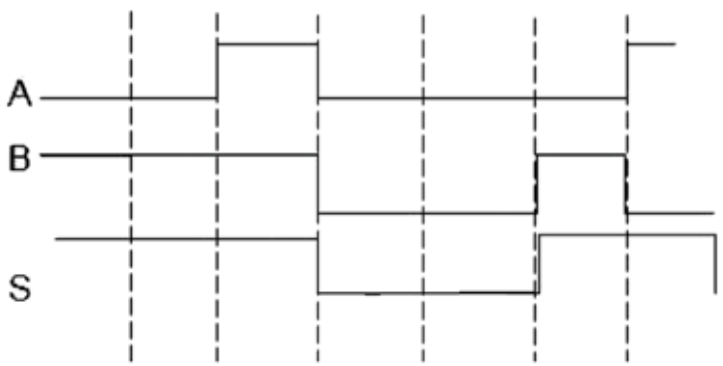


Figura 67 – Sinal de saída usando porta OR

Observa-se que A começa embaixo e B em cima, ou seja, A = 0, B = 1, o que resulta S = 1, então monta-se a tabela envolvendo todos os intervalos.

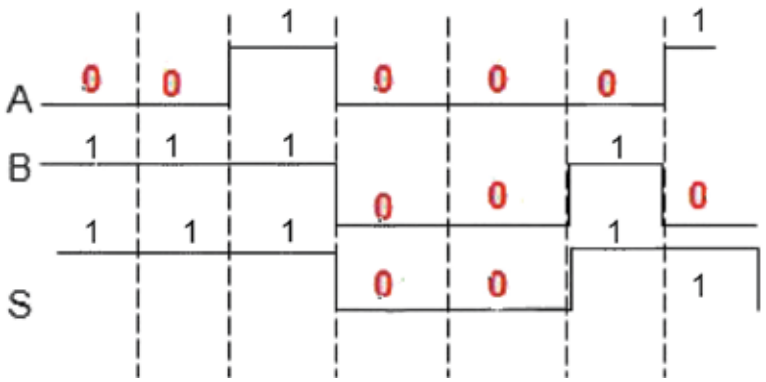


Figura 68 – Sinal de saída usando porta OR

A	B	S
0	1	1
0	1	1
1	1	1
0	0	0
0	0	0
0	1	1
1	0	1

Tabela 14 – Função OR

Repare que, mesmo colocando todos os valores existentes nos intervalos dos sinais, bastaria a tabela simples que foi ilustrada na Tabela 14 para atingir todas as possibilidades.

Tabela exibindo exemplos reais:

A Pessoa com fome	B Hora do almoço	S Pessoa come a comida
F	F	F
F	V	V
V	F	V
V	V	V

Tabela 15 – Função OR para exemplos reais

A pessoa irá comer a comida se ela estiver com fome OU se estiver na hora do almoço.

A Memória requisitando o processador	B HD requisitando o processador	S Processador processa a informação
F	F	F
F	V	V
V	F	V
V	V	V

Tabela 16 – Função OR para exemplos reais

7.1.2 Porta NOT

A porta NOT recebe somente uma entrada e produz uma saída, conforme a Figura 69.

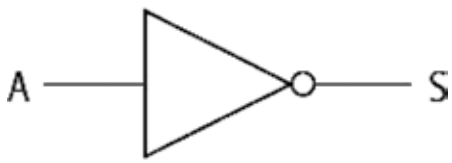


Figura 69 – Porta NOT

Na entrada A é inserido o *bit* e então é realizado um cálculo que resulta na saída S. A porta NOT está diretamente atrelada à função NOT.

Na Tabela 17 é ilustrada a regra da função NOT.

A	S
0	1
1	0

Tabela 17 – Função NOT

A função NOT pode ser simplificada da seguinte maneira: sempre irá inverter o valor de entrada, se entrar 0, sairá 1, e se entrar 1 o resultado será 0.

Tabela verdade com os sinais de Falso (F) e Verdadeiro (V)

A	S
F	V
V	F

Tabela 18 – Função NOT

A Figura 70 mostra o diagrama de tempo.

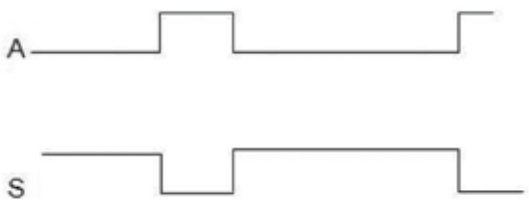


Figura 70 – Sinal de saída usando porta NOT

Para saber se a função NOT está sendo usada, separa-se em intervalo de tempo.

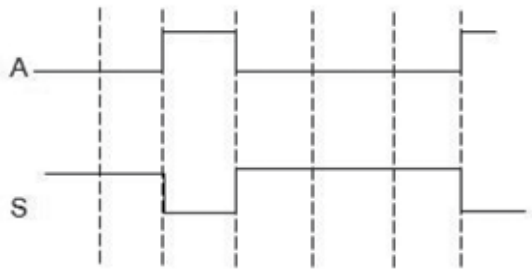


Figura 71 – Sinal de saída usando porta NOT

Observa-se que sempre que A estiver em cima, S estará embaixo, e vice-versa.

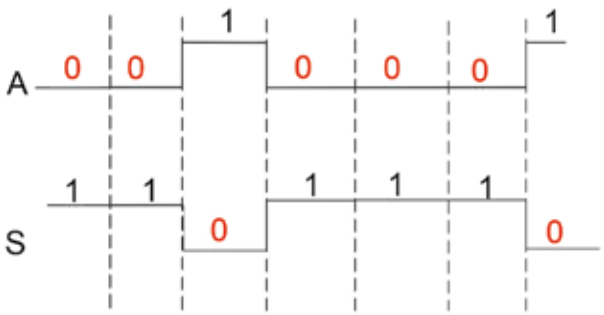


Figura 72 – Sinal de saída usando porta NOT

A	S
0	1
0	1
1	0
0	1
0	1
0	1
1	0

Tabela 19 – Função NOT

Repare que mesmo colocando todos os valores existentes nos intervalos dos sinais, bastaria a tabela simples que foi ilustrada na Tabela 19 para atingir todas as possibilidades.

7.1.3 Portas NAND

A porta NAND, básica, recebe duas entradas e produz uma saída, conforme a Figura 73.

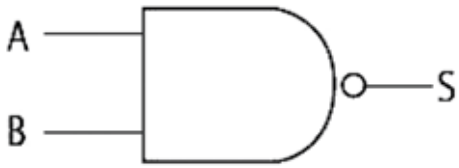


Figura 73 – Porta NAND

É uma mistura da porta AND com a NOT, ou seja, realiza-se primeiro a função AND e o resultado equivale à entrada da função NOT, conforme a Figura 74.

Figura 74 – Sinal de saída usando porta NAND

O círculo após a Figura 73 e Figura 74 significa que estão sendo negadas, ou seja, será invertido o valor.

Na Tabela 20 é ilustrada a regra da função AND para duas entradas.

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

Tabela 20 – Função NAND

Repare que a NAND é o inverso da função AND, pois após se realizar a função AND, inverter-se-á o *bit* com o NOT.

Caso tenha mais entradas, teremos mais linhas na tabela verdade, conforme é mostrado na figura 75.

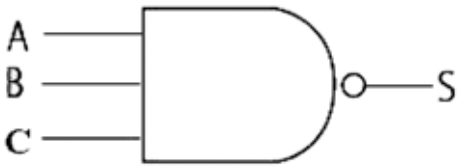


Figura 75 – Sinal de saída usando porta NAND com 3 portas.

A	B	C	S
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Tabela 21 – Função NAND com 3 entradas

Tabela-verdade com os sinais de Falso (F) e Verdadeiro (V)

A	B	S
F	F	V
F	V	V
V	F	V
V	V	F

Tabela 22 – Função NAND

7.1.4 Porta NOR

A porta NOR, básica, recebe duas entradas e produz uma saída, conforme a Figura 76.

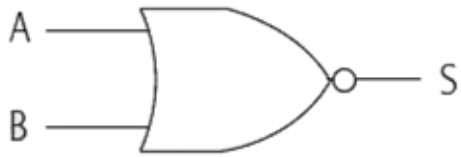


Figura 76: Porta NOR

É uma mistura da porta OR com a NOT, ou seja, realiza-se primeiro a função OR e o resultado equivale à entrada da função NOT, conforme a Figura 77.

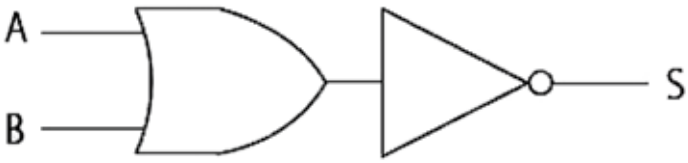


Figura 77: Porta NOR

O círculo após a Figura 76 e Figura 77 significa que estão sendo negadas, ou seja, será invertido o valor.

Na tabela 23 é ilustrada a regra da função AND para duas entradas.

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

Tabela 23: Função NOR

Repare que a NOR é o inverso da função OR, pois após se realizar a função OR, inverte-se o *bit* com o NOT.

Caso existam mais entradas, teremos mais linhas na tabela verdade, conforme é mostrado na figura 78.

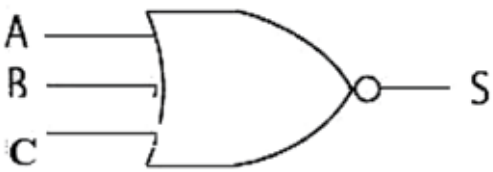


Figura 78 – Porta NOR com 3 entradas

A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Tabela 24 – Função NOR com 3 entradas

Tabela verdade com os sinais de Falso (F) e Verdadeiro (V):

A	B	S
F	F	V
F	V	F
V	F	F
V	V	F

Tabela 25: Função NOR

7.1.5 XOR

O XOR, também chamado de OU EXCLUSIVO, é uma mistura de NOTs, ANDs e OR, onde a saída só valerá 1, quando uma das entradas for igual a 1, e valerá 0 quando ambas entradas forem iguais a 1 ou 0.

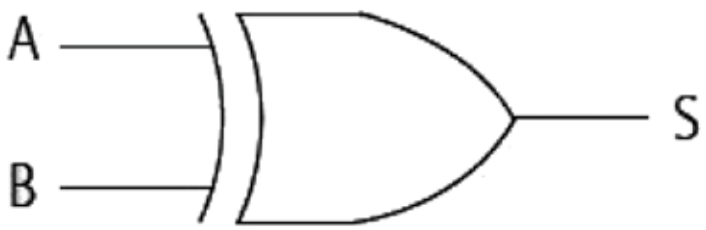


Figura 79 – Porta XOR

A mistura mencionada anteriormente e ilustrada na Figura 80.

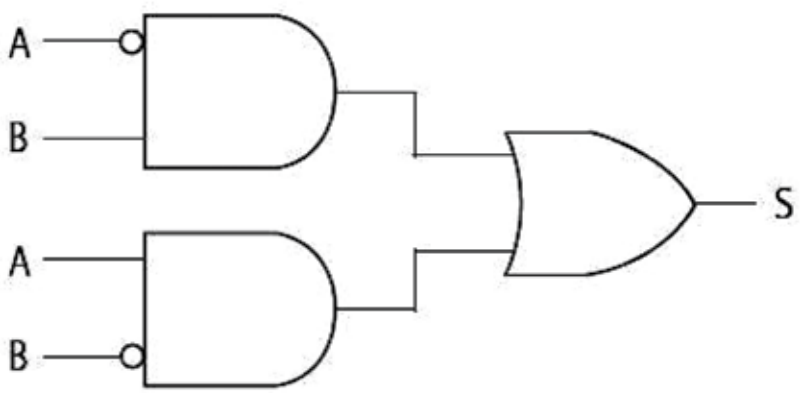


Figura 80 – Porta XOR

Na primeira entrada A, o *bit* já recebe o NOT (o círculo), e no segundo B também há um NOT (círculo), e então são realizados dois AND, onde cada saída resulta em uma entrada para um OR.

Na Tabela 26 é ilustrada a regra da função XOR para duas entradas.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 26 – Função XOR

Caso existam mais entradas, teremos mais linhas na tabela verdade, conforme é mostrado na figura 81.

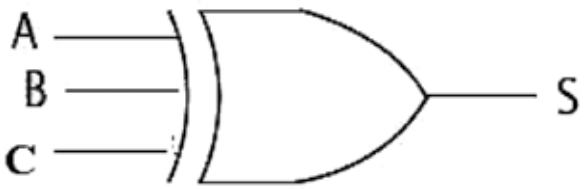


Figura 81 – Porta XOR com 3 entradas

A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Tabela 27 – Função XOR com 3 entradas

Tabela-verdade com os sinais de Falso (F) e Verdadeiro (V):

A	B	S
F	F	F
F	V	V
V	F	V
V	V	F

Tabela 28 – Função XOR

Quando se fala em sistemas digitais, é necessário que as instruções sigam uma ordem e um ciclo; para que isso seja possível é utilizado o *clock*, que é um circuito que emite pulsos iguais – de mesmo tamanho – em intervalos iguais.

O intervalo de tempo entre dois pulsos consecutivos é denominado período de *clock*, sendo que a frequência desses pulsos ficam entre 1 e 500 MHz, o que corresponde a períodos de 1000 ns a 2 ns. Mas como é possível essa precisão?

O *clock* é controlado por um oscilador de cristal, por isso é possível obter a precisão de intervalo entre os pulsos. Embora o período de *clock* seja rápido, é possível executar várias instruções ou eventos no período, então o *clock* é dividido em subperíodos que são conhecidos como subciclos e existem para que ocorra a ordenação na execução.

A Figura 82 ilustra um sinal de clock

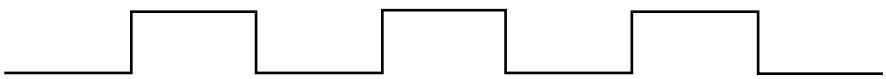
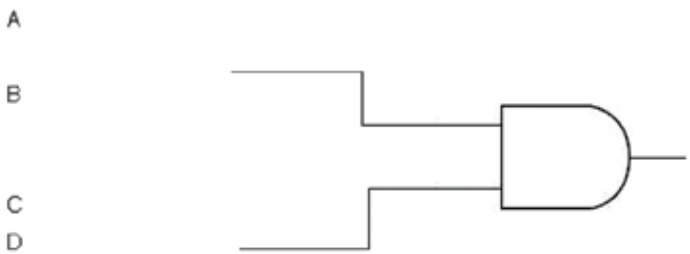


Figura 82 – Sinal de *clock*

Exemplos de portas lógicas

Três portas AND



Sendo:

A = 0
B = 1
C = 1
D = 1
S = Saída

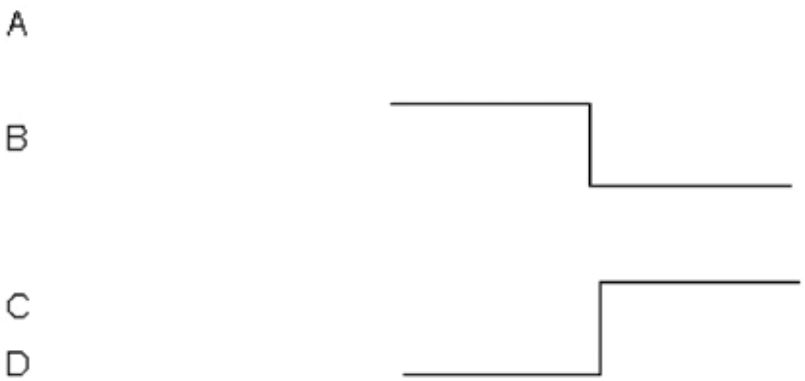
A função AND pode ser escrita como:

Sab = A.B
Scd = C.D

Então temos:

S = (A.B). (C.D)
S = (0.1). (1.1)
S = 0. 1
S = 0

Duas portas AND e uma OR



Sendo:

A = 0
B = 1
C = 1
D = 1
S = Saída

A função AND pode ser escrita como:

Sab = A.B

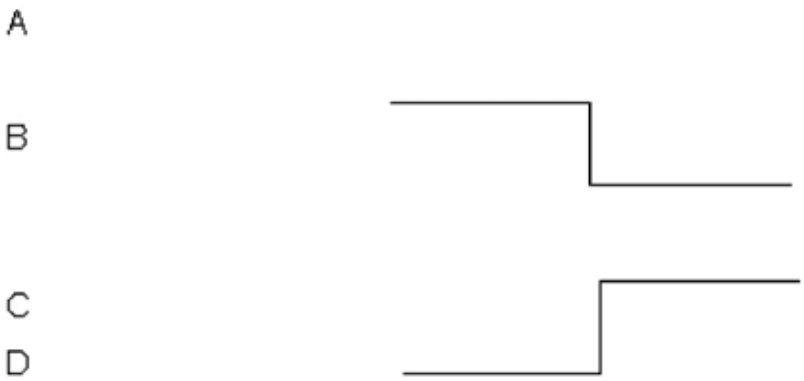
A função OR pode ser escrita como:

Scd = C+D

Então temos:

S = (A.B). (C+D)
S = (0.1). (1.1)
S = 0. 1
S = 0

Duas portas OR e uma AND



Sendo:

A = 0
B = 0
C = 0
D = 1
S = Saída

A função OR pode ser escrita como:

Sab = A+B
Scd = C+D

Então temos:

S = (A+B). (C+D)
S = (0+0). (0+1)
S = 0. 1
S = 0

Três portas OR

Sendo:

A = 0
B = 0
C = 0
D = 1
S = Saída

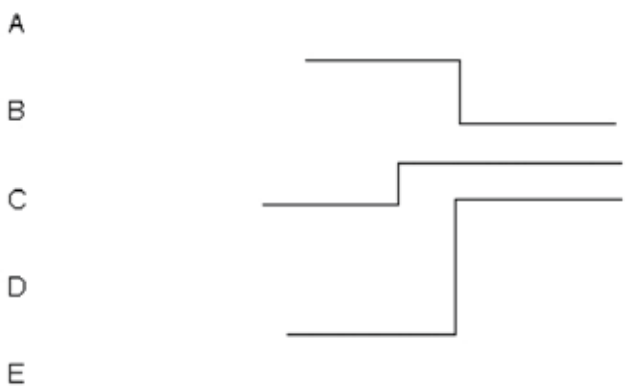
A função OR pode ser escrita como:

Sab = A+B
Scd = C+D

Então temos:

S = (A+B) + (C+D)
S = (0+0) + (0+1)
S = 0 + 1
S = 1

Duas portas OR, uma NOT e uma AND



Sendo:

A = 0
B = 1
C = 0
D = 1
E = 1
S = Saída

A função OR pode ser escrita como:
Sab = A+B

A função AND pode ser escrita como:

Scd = D.E

A função NOT pode ser escrita como:

Scd = $\overline{C}$

Então temos:

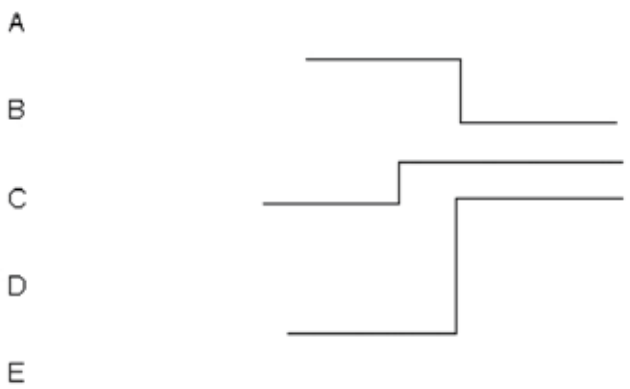
S = (A+B) + $\overline{C}$ + (D.E)

S = (0+1) + $\overline{0}$ + (1.1)

S = 1 + 1 + 1

S = 1

Duas portas OR, uma NOT e uma NAND



Sendo:

A = 0
B = 1
C = 0
D = 1
E = 1
S = Saída

A função OR pode ser escrita como:

Sab = A+B

A função NAND pode ser escrita como:

$Scd = \overline{D \cdot E}$

A função NOT pode ser escrita como:

$Scd = \overline{C}$

Então temos:

$S = (A+B) + \overline{C} + (\overline{D.E})$

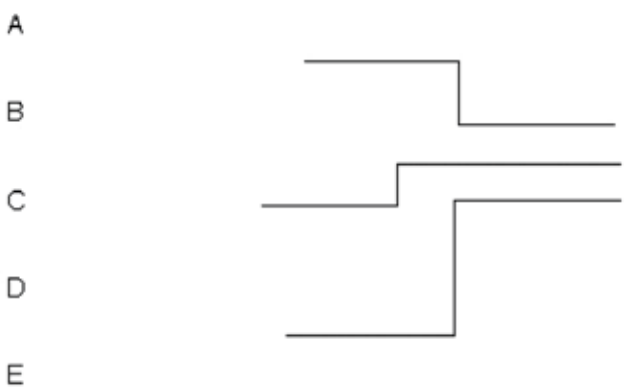
$S = (0+1) + \overline{0} + (\overline{1.1})$

$S = (0+1) + 1 + (0 + 0)$

$S = 1 + 1 + 0$

$S = 1$

Uma porta XOR, uma OR, uma NOT e uma NAND



Sendo:

A = 0
B = 1
C = 0
D = 1
E = 1
S = Saída

A função XOR pode ser escrita como:

$Sab = \overline{A} \cdot B + A \cdot \overline{B}$

A função NAND pode ser escrita como:

$Scd = \overline{D.E}$

A função NOT pode ser escrita como:

$Scd = \overline{C}$

Então temos:

$S = (\overline{A} . B + A . \overline{B}) + \overline{C} + (\overline{D . E})$

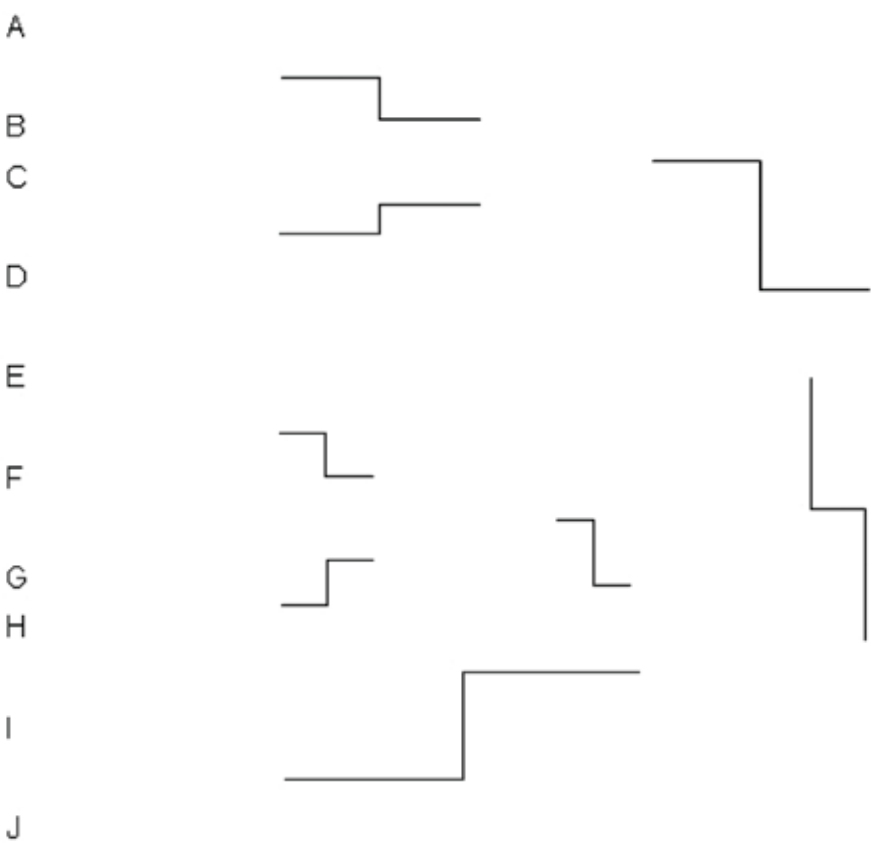
$S = (\overline{0} . 1 + 0 . \overline{1}) + \overline{0} + (\overline{1 . 1})$

$S = (1.1 + 0.0) + 1 + (0 + 0)$

$S = (1 + 0) + 1 + 0$

$S = 1 + 1 + 0$

Portas OR e AND



Sendo:

A = 0
B = 0
C = 0
D = 1
E = 1
F = 0
G = 1
H = 1
I = 0
J = 0
S = Saída

Então temos:

S = [(A+B) + (C.D)] + {[(E+F). (G.H)].(I.J)}
S = [(0+0) + (0.1)] + {[(1+0). (1.1)].(00)}
S = [(0) + (0)] + {[(1) (1)].(0)}
S = [0] + {[1]. 0 }
S = [0] + { 0 }
S = 0

Saiba mais

<http://www.inf.ufsc.br/ine5365/portlog.html>

8 MEMÓRIA, INTERFACES E PROCESSADORES

8.1 Memória

As memórias são itens indispensáveis nos computadores, pois elas servem para armazenar tanto as instruções que serão executadas como os dados que serão utilizados por essas instruções.

Quando se fala de memória, logo também se fala de *latches* e *Flip-Flop*. Os *latches* são uma das maneiras pelas quais as memórias podem ser criadas. Os *latches* são circuitos que agem quando o sinal de *clock* está em 1 ou 0. Já os *flip-flops* são utilizados quando se precisa visualizar os dados e depois armazená-los. Os *flip-flops* são ativados nas transições do *clock*, ou seja, quando o *clock* passa de 0 para 1 (transição positiva) e quando passa de 1 para 0 (transição negativa).

Quando existe um agrupamento de *flip-flop* podemos formar o que chamamos de registrador, o qual serve para manipulação de armazenamento de dados. Os registradores padrão podem ser no modo serial e paralelo.

No modo serial, os dados são transmitidos (recebidos e enviados) em um *bit* por vez. Já no modo paralelo é possível transmitir mais de um *bit* por vez.

8.2 Registrador Serial

Bits a serem transmitidos: 11001001

Recebendo	1							

Bits restantes: 1001001

Recebendo	1	1						

Bits restantes: 001001

Recebendo	1	1	0					

Bits restantes: 01001

Recebendo	1	1	0	0				

Bits restantes: 1001

Recebendo	1	1	0	0	1			

Bits restantes: 001

Recebendo	1	1	0	0	1	0		

Bits restantes: 01

Recebendo	1	1	0	0	1	0	0	

Bits restantes: 1

Recebendo	1	1	0	0	1	0	0	1

Bits restantes:

Enviando	1	1	0	0	1	0	0	

Bits enviados: 1

Enviando	1	1	0	0	1	0		

Bits enviados: 01

Enviando	1	1	0	0	1			

Bits enviados: 001

Enviando	1	1	0	0				

Bits enviados: 1001

Enviando	1	1	0					

Bits enviados: 01001

Enviando	1	1						

Bits enviados: 001001

Enviando	1							

Bits enviados: 1001001

Enviando								

Bits enviados: 11001001

Lembrete

No modo serial, os *bits* são recebidos ou enviados um a um, conforme foi ilustrado anteriormente.

Registrador Paralelo:

Bits a serem transmitidos: 11001001

Recebendo	1	1	0	0	1	0	0	1

Bits restantes:

Enviando								

Bits enviados: 11001001

Lembrete

No modo paralelo, foi possível visualizar que os *bits* são armazenados em um único passo.

Em ambos os modos foram utilizados 8 *bits*, porém, a quantidade de *bits* que é possível armazenar é igual à quantidade de *Flip-Flop* que compõe o registrador.

8.3 Barramentos

O barramento é dito como o caminho elétrico que faz a ligação de dois ou mais dispositivos. Na história da computação existiram diversos tipos de barramentos; alguns já estão em desuso enquanto outros continuam sendo utilizados por diversas placas.

No computador existem diversos barramentos. Por exemplo, existe barramento que transmite dados do processador para uma memória RAM, assim como existe barramento dentro do processador, sendo que ele transmite dados da ULA para os registradores, o barramento dos dispositivos etc.

Os computadores atuais costumam utilizar dois barramentos principais, sendo um para conectar o processador com os dispositivos de E/S (Entrada e Saída), e o segundo para ligar a memória ao processador.

Quando os dispositivos utilizam o barramento, sempre há o dispositivo principal e o secundário; por exemplo, se o processador faz uma requisição para o disco rígido, nesse caso o processador será o dispositivo principal e o disco rígido será o secundário. O único dispositivo que não pode ser o principal é a memória, pois ela sempre recebe requisições.

Então teremos dispositivos principais, outros secundários e outros que podem ser tanto principais como secundários. Porém, em alguns casos, os sinais elétricos (*bits* binários) que os dispositivos emitem não são fortes o bastante para chegar ao destino; então os dispositivos principais utilizam um *chip* denominado alimentador do barramento. Este é um amplificador de sinal que é emitido ao barramento, podendo assim o sinal chegar ao destino sem problema. Já os dispositivos secundários utilizam um receptor de barramento para serem capazes de receber o sinal emitido pelo alimentador de barramento. E os dispositivos que atuam tanto como principal quanto secundário utilizam um *chip* denominado transceptor de barramento, que atua como alimentador e receptor.

Exemplos:

Atividade	Principal	Secundário
Leitura de informação	Processador	Memória
Gravação de Dados	Processador	Disco Rígido
Armazenamento de Informação	Dispositivos de Entrada e Saída (USB)	Memória

Tabela 29 – Exemplo de dispositivos principais e secundários

Mas como enviar mais dados pelo barramento?

Existem duas formas de se transmitir os dados mais rapidamente: a primeira forma é aumentando a largura do barramento. Imagine o barramento como se fosse uma estrada; quanto mais faixas existirem, mais carros poderão passar ao mesmo tempo. Porém, essa estrada irá ocupar mais espaço e serão necessários mais funcionários, câmeras e equipamentos para gerenciá-la. O mesmo ocorre com os barramentos: esses são formados por linhas e, quanto mais linhas existirem, mais dados poderão trafegar, porém, esse barramento ocupará mais espaço na placa em que está impresso, sem contar que os *chips* que controlam também tendem a ser maiores, e tudo isso elevará o preço do produto final.

Então, se o processador puder trabalhar com um barramento com uma largura grande, poderá ter acesso a mais endereços de memória, agilizando o processo de leitura e escrita na memória; por isso, não adianta ter 4 GB de memória se o barramento da placa-mãe que faz o acesso não tiver vazão para toda a potencialidade da memória.

A segunda forma é aumentando a velocidade do barramento, que trafegaria mais rapidamente e deixaria que outros dados utilizassem o espaço ocupado anteriormente. Porém, esse procedimento é de difícil construção e há grande possibilidade de problemas com os dados, pois os sinais que vêm dos diferentes barramentos provavelmente não estarão na mesma velocidade, sem contar que, caso existam placas antigas, essas não irão funcionar com um barramento que trafegue em uma velocidade maior. Por isso as fábricas tendem a utilizar o aumento de linha (largura do barramento) para melhorar os barramentos.

Então os barramentos tendem a aumentar e aumentar. Sendo assim, pensou-se em uma forma de manter um determinado tamanho e utilizar a mesma linha que trafega endereço para trafegar dados. Isso ocorre da seguinte maneira: primeiro são enviados os endereços que se deseja acessar, depois os dados são enviados pelas mesmas linhas do barramento. Isso faz com que o barramento não fique tão grande e seu custo não aumente, porém, a transmissão fica mais lenta, pois em vez de endereços e dados serem transmitidos ao mesmo tempo, primeiro se transmite o endereço para depois se transmitir os dados.

Ainda em barramento existe a temporização, que pode ser de dois tipos: síncrono e assíncrono. No modo síncrono, há um oscilador de cristal (*clock*) que alimenta o barramento, e todas as atividades do barramento são executadas em um número inteiro de ciclos do sinal que são denominados ciclos de barramento. Já no modo assíncrono não há um sinal de *clock*, então não existe um padrão de tamanho para todas as execuções, podendo cada uma ter um ciclo de tamanho diferenciado.

Alguns tipos de barramentos são:

- ISA (*Industry Standard Architecture*). Esse barramento não é mais utilizado; surgiu com o IBM PC, inicialmente trabalhava com 8 *bits* e depois, com 16 *bits*.

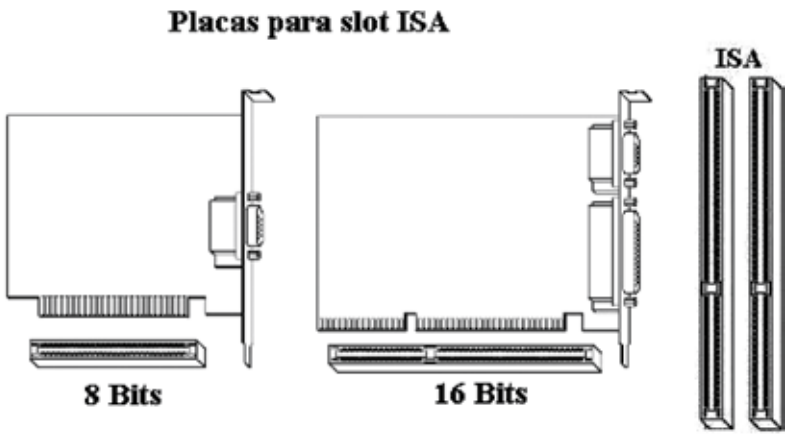


Figura 83 – Placas e *Slots* ISA

- PCI (*Peripheral Component Interconnect*). O barramento PCI foi introduzido no mercado na década de 90 e substituiu o barramento ISA, pois contava com 32 *bits* (o dobro do barramento ISA) que, juntamente com a frequência do *clock* com que trabalhava, era possível transmitir até 132 MB por segundo, o que deu um salto enorme comparado com os 16 MB que era capaz de transmitir o barramento ISA. Outra vantagem do *Slot* PCI foi o tamanho do *slot* e das placas, que eram bem menores das que eram utilizadas pelos *Slots* ISA, então houve ganho na fabricação não só das placas de vídeo, som etc., mas também da placa-mãe.

Entretanto, o barramento PCI veio com uma inovação que fez a diferença, que foi o BUS Mastering. Esse recurso permite a comunicação direta do dispositivo que esteja utilizando o barramento PCI com a memória RAM. Mas qual o ganho disso? O processador não precisa ser o mediador entre o dispositivo e a memória, então o processador pode ficar preocupado com outras tarefas.

Mas o PCI não parou por aí. Junto a ele chegou o recurso PnP (*Plug and Play*), ou seja, o sistema operacional juntamente com os *hardwares* eram capazes de reconhecer e instalar automaticamente os dispositivos que estavam plugados no computador.

O PCI foi evoluindo e passou a ter 64 *bits*, passando a transmitir 512 MBs.

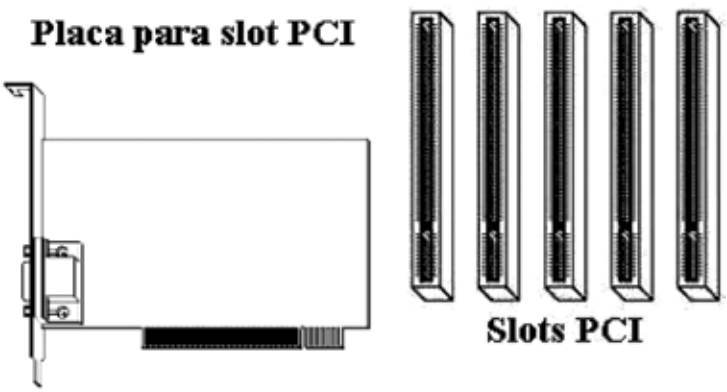


Figura 84 – Placa e *Slots* de PCI

- PCI-X (*Peripheral Component Interconnect Extended*). O PCI-X é uma evolução do PCI de 64 *bits*, podendo este ser compatível com o antigo padrão PCI.

- AGP (*Accelerated Graphics Port*). O AGP é um barramento voltado para a área gráfica, ou seja, os *slots* são para a utilização de placa de vídeo. Esse barramento foi criado porque, com a evolução da computação, a parte gráfica começou a ser exigida e a exigir *hardwares* que ilustrassem com mais qualidade as cores e as quantidades de *pixels* na tela. O barramento AGP começou a trabalhar com 32 *bits*, podendo transferir até 266 MBs, porém, esse mesmo barramento poderia ser configurado para utilizar uma frequência maior, duas vezes o *clock* padrão, conseguindo assim atingir até 532 MBs.

Com o tempo foi lançada uma nova versão do AGP, que passou a atingir 1.066 MBs, e logo após passou a transmitir 2.133 MBs, tendo um ganho muito bom.

Esse barramento também passou a permitir que as placas de vídeo utilizassem um "pedaço" da memória RAM como memória incremental para a da própria placa; esse recurso foi denominado *Direct Memory Execute*. E também foi ótimo para programas que exigiam muito recurso visual, porém acabou perdendo espaço e sendo substituído PCI-EX.

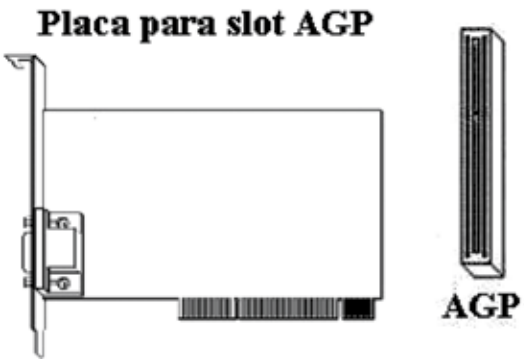


Figura 85 – Placa e *Slot* AGP

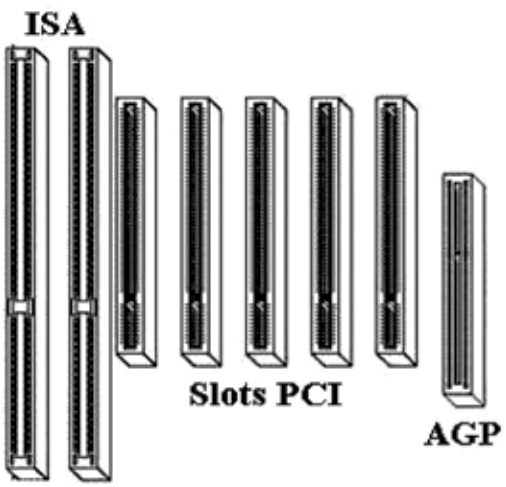


Figura 86 – *Slots* ISA, PCI e AGP

PCI				PCI-X				AGP			
MHz	Bits	Pulso	Taxa*	MHz	Bits	Pulso	Taxa	MHz	Bits	Pulso	Taxa
33	32	1	133	66	64	1	533	66	32	1	266
66	32	1	266	133	64	1	1.066	66	32	2	533
33	64	1	266	133	64	2	2.132	66	32	4	1.066
66	64	1	533	133	64	4	4.266	66	32	8	2.133

Tabela 30 – Tabela de comparação dos barramentos PCI, PCI-X e AGP.

* Taxa = MB/s

- PCIe ou PCI-EX (PCI *Express*). Essa tecnologia foi criada em meados de 2004 e veio para substituir os barramentos PCI e AGP.

O PCI-EX é diferente do PCI-X, pois o segundo é uma evolução do PCI de 64 *bits*. Já o PCI-EX tem outra tecnologia e possui várias velocidades, podendo ir de 250 MBs até 8 GBs. Isso fez com que o barramento PCI-EX superasse o AGP, pois a taxa de transmissão era superior, e a placa de vídeo é um dos recursos que vêm sendo mais exigidos no computador.

O PCI-EX tem uma particularidade: quando uma placa utiliza o PCI, essa placa troca informação por meio do barramento PCI, sendo que todos os *slots* PCI utilizam o mesmo barramento. Já no PCI-EX, cada *slot* tem sua própria ligação, então tem um caminho privado sem a divisão com outros *slots*.

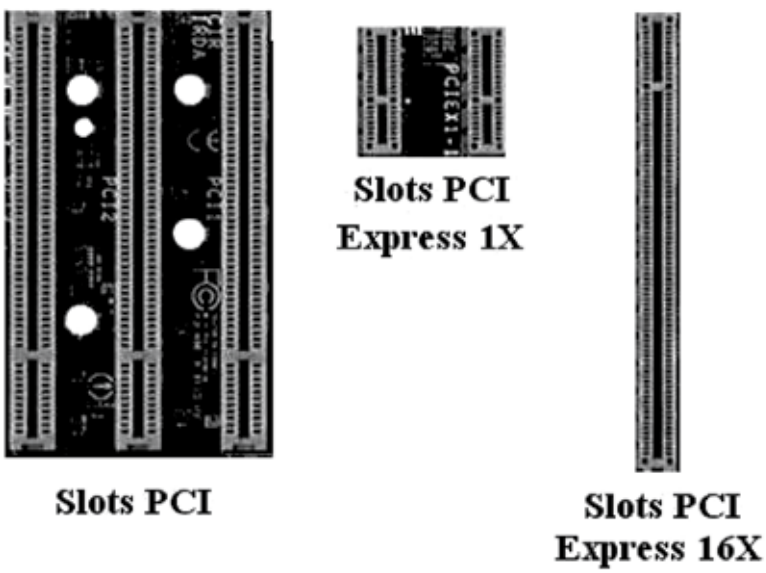
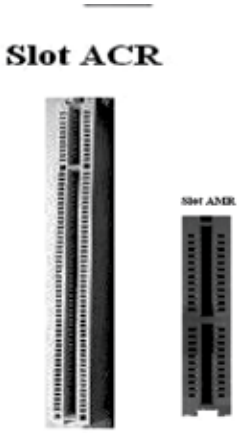


Figura 87 – Comparação dos *Slots* PCI e PCI Express.

AMR (*Audio Modem Riser*). Esse *slot* e barramento foi desenvolvido para ser usado por *modem* e áudio, porém, para ser utilizado, o *chipset* da placa-mãe necessitava aceitar esse tipo de conexão.

-CNR (*Communications and Network Riser*). Esse barramento é bem similar ao AMR e veio para substituí-lo, sendo que o *slot* é igual, tendo como vantagem o suporte à placa de rede, além de áudio.

-ACR (*Advanced Communications Riser*). Foi desenvolvido pela AMD, enquanto o AMR e o CNR foram desenvolvidos pela Intel. As comunicações principais desse barramento são as comunicações de redes e USB. O *slot* ACR é muito similar ao *slot* PCI.



Slot ACR Slot AMR

Figura 88 – *Slots* ACR e AMR

- SATA (*Serial Advanced Technology Attachment*). Essa tecnologia chegou para substituir o padrão IDE (*Integrated Development Environment*) e é usada pelos dispositivos como Disco Rígido, CD, DVD ou Blu-ray.

Mas qual a diferença entre o SATA e o IDE?

O IDE, também conhecido para ATA ou PATA, é uma forma de comunicação paralela, ou seja, transmite vários *bits* por vez, enquanto no SATA a transmissão é de forma serial, um *bit* por vez, há fila de *bits*. Então, se o IDE envia vários dados juntos e o SATA envia um por vez, o IDE é mais rápido?

Não, pois a transmissão paralela tem um padrão de 16 *bits* por vez, o que ocasiona problema por interferência. Então, se formos lembrar, os cabos IDEs padrão eram os de 40 vias, ou seja, 40 linhas de comunicação, porém, por causa das interferências, os cabos IDEs passaram para 80 vias, sendo que havia vias que atuavam contra as interferências. Por causa das interferências, a velocidade de transmissão era limitada a 133 MBs.

E na comunicação serial não há problema com interferência, pois o cabo passou de 80 para 4 vias e a velocidade de comunicação é bem maior do que na comunicação paralela, sendo que o SATA começou com uma velocidade de 150 MBs depois passou para 300 MBs e chegou a 760 MBs.

Outra diferença entre essas tecnologias é o modo de configuração. Quando utiliza-se o modo IDE, é possível colocar até dois discos rígidos no mesmo canal, sendo que eles têm que ser configurados por

meio de *jumper* para identificar qual é o disco primário e o secundário. Já no SATA, o canal é único para o dispositivo, então é um canal particular, sem perda de rendimento.

A primeira versão do SATA trabalha com taxa máxima de transferência de dados de 150 MB por segundo (MB/s). Essa versão recebeu os seguintes nomes: SATA 150, SATA 1.0, SATA 1,5 Gbps (1,5 *gigabits* por segundo) ou, simplesmente, SATA I.

Não demorou muito para surgir uma versão denominada SATA II (ou SATA 3 Gbps – na verdade, SATA 2,4 Gbps – ou SATA 2.0, ou SATA 300), cuja principal característica é a velocidade de transmissão de dados a 300 MB/s, o dobro do SATA I. Alguns discos rígidos que utilizam essa especificação contam com um *jumper* que limita a velocidade do dispositivo a 150 MB/s, uma medida aplicada para fazer com que esses HDs funcionem em placas-mãe que suportam apenas o SATA I.

Figura 89 – *Slots* SATA e IDE

- SCSI (*Small Computer System Interface*). O barramento pode transmitir de duas formas: transmissão assíncrona e síncrona.

Na transmissão assíncrona é enviada uma informação pelo barramento e aguarda-se a confirmação do recebimento. Já na transmissão síncrona há a confirmação de resposta e é possível enviar várias informações ao mesmo tempo, sendo que há um período para enviar e outro para receber os dados.

Tipos de SCSI:

SCSI – o barramento SCSI foi iniciado com 5 MHz de frequência, com 8 *bits* para transmissão de dados.

- *Wide* SCSI: esse tipo de SCSI passou para 16 *bits* com a mesma frequência, aumentando a velocidade de transmissão no barramento de 5 MB/s para 10 MB/s.
- Fast SCSI: o barramento funciona em 10 MHz em barramento de 8 *bits*, operando ainda com 10 MB/s.
- *Fast Wide* SCSI: o barramento funciona em 10 MHz em barramento de 16 *bits*, operando com 20 MB/s.
- Ultra SCSI: passou para 20 MHz de frequência no barramento, com 8 *bits* e transferindo 20 MB/s.
- *Wide* Ultra SCSI: manteve os 20 MHz de frequência no barramento, com 16 *bits* e transferindo 40 MB/s.
- Ultra2 SCSI: como o nome diz, a frequência é dobrada de 20 MHz para 40 MHz, com 8 *bits* e resultando em 40 MB/s.

- *Wide* Ultra2 SCSI: passou para 16 *bits* com os mesmos 40 MHz, resultando em 80 MB/s.

- Ultra160 SCSI: passou a transmitir 160 MB/s, fazendo duas transferências por ciclo de *clock*.

- Ultra320 SCSI: passou a transmitir 320 MB/s, fazendo duas transferências por ciclo de *clock*, e aumentou a frequência de 40 MHz para 80 MHz.

Durante a evolução, além da velocidade, também foi influenciada a quantidade de dispositivos que poderiam ser ligados na controladora SCSI. Quando a mesma era de 8 *bits*, era possível conectar até sete dispositivos, e quando chegou em 16 *bits* foi possível conectar até 15 dispositivos.

Em comparação com o padrão IDE, o SCSI ganhava em praticamente todos os quesitos (velocidade, qualidade, quantidade de dispositivos), mas o custo era muito mais elevado.

Por isso, o IDE ganhou espaço nos computadores pessoais, pois o custo era bem mais atrativo, enquanto o padrão SCSI pegou o mercado dos servidores, pelo melhor desempenho, onde passou-se a utilizar os RAIDs, pois era possível colocar vários discos rígidos na mesma controladora. Em relação à velocidade, a vantagem do SCSI é que os dispositivos nele conectados conseguiam aproveitar melhor a velocidade do processador, sendo assim melhoravam o desempenho do computador em geral.

Com a evolução foi criado o SAS (*Serial Attached* SCSI), que é compatível com o padrão SATA, sendo que passou para 300 MB/s e pode chegar até 600 MB/s.

- USB (*Universal Serial Bus*): é uma tecnologia que possibilitou o avanço da comunicação do computador com memórias secundárias.

A USB permitiu uma comunicação mais rápida, simples que permitia a qualquer usuário colocar um dispositivo no computador e utilizá-lo. Assim como o barramento PCI, o USB também utiliza PnP, sendo que qualquer dispositivo é instalado quando é plugado no computador. A alimentação elétrica dos dispositivos é realizada pela conexão USB, sendo que a grande maioria dos dispositivos não necessita de outra fonte de energia como, por exemplo, celulares, MP3; porém, existem dispositivos que requerem mais energia do que a porta USB é capaz de transmitir, então esses precisam estar ligados na tomada, como a impressora, por exemplo.

Essa comunicação foi iniciada no mercado com uma velocidade de transmissão de 190 KB por segundo, sendo que depois chegou a 1,5 MB por segundo. Já a versão USB 2.0 evoluiu para 60 MB por segundo e a USB 3.0, que alcança velocidades da ordem de 625 MB/s. Ainda em 2013 surgiu a USB 3.1 com o dobro da velocidade da versão anterior.

A seguir serão ilustrados dois tipos de barramentos:

Na Figura 90 há uma ponte que faz o intermédio do processador, memória e dispositivos, sendo que nessa arquitetura ainda se utilizava o barramento ISA.

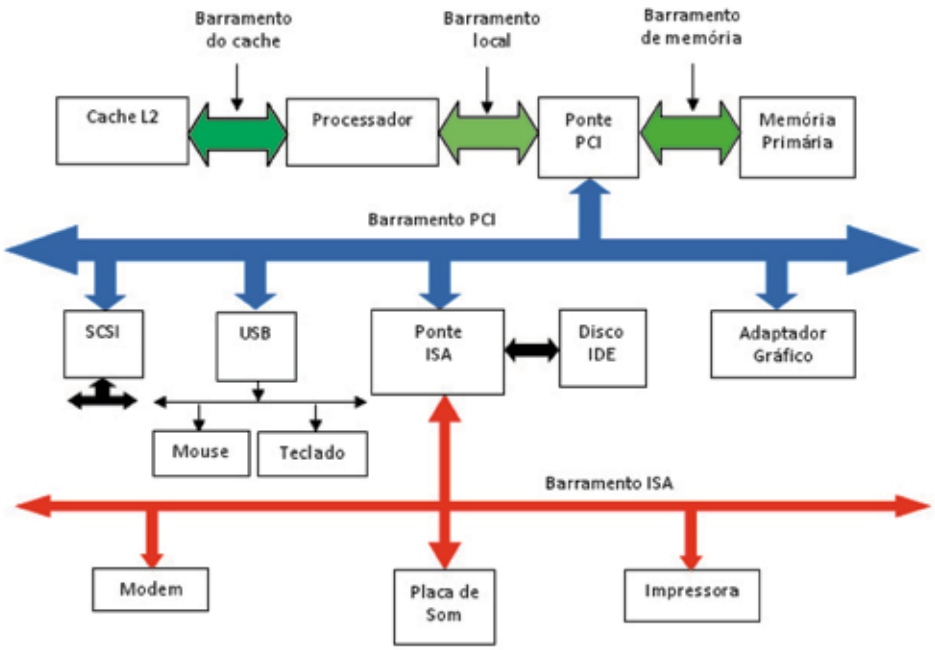


Figura 90: Barramento do Pentium II.

Fonte: TANENBAUM, 2007a.

Com a evolução dos computadores, a placa de vídeo ganhou um barramento próprio para ela, melhorando sua comunicação, conforme pode ser observado na Figura 91.

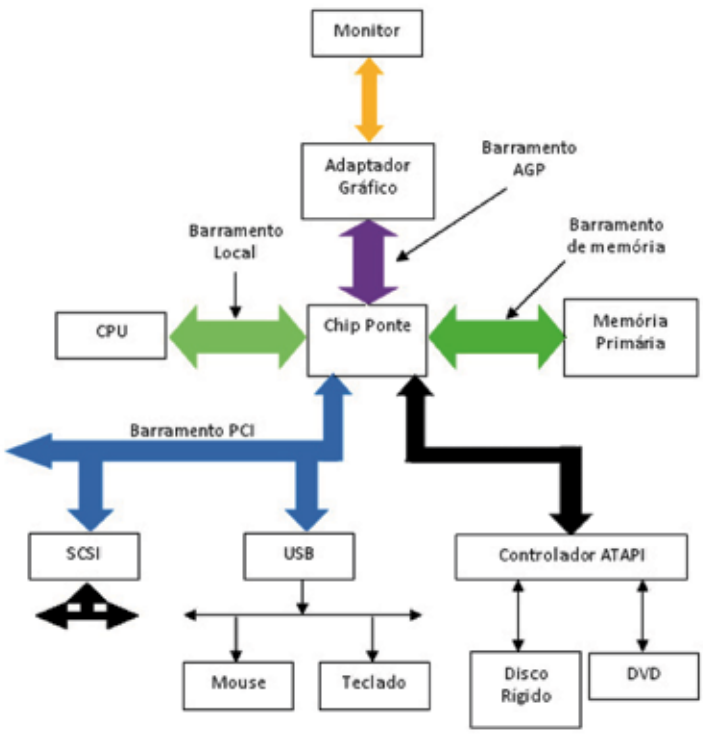


Figura 91 – Barramento do Pentium IV.

Fonte: TANENBAUM, 2007a.

Já na Figura 91 o adaptador gráfico, que era via PCI, passou a operar pelo barramento AGP, otimizando e melhorando a comunicação gráfica de dos dispositivos PCI.

Com a criação do PCI Express, cada dispositivo passou a ter um acesso dedicado, não tendo que dividir o barramento com os demais dispositivos, como ilustra a Figura 92.

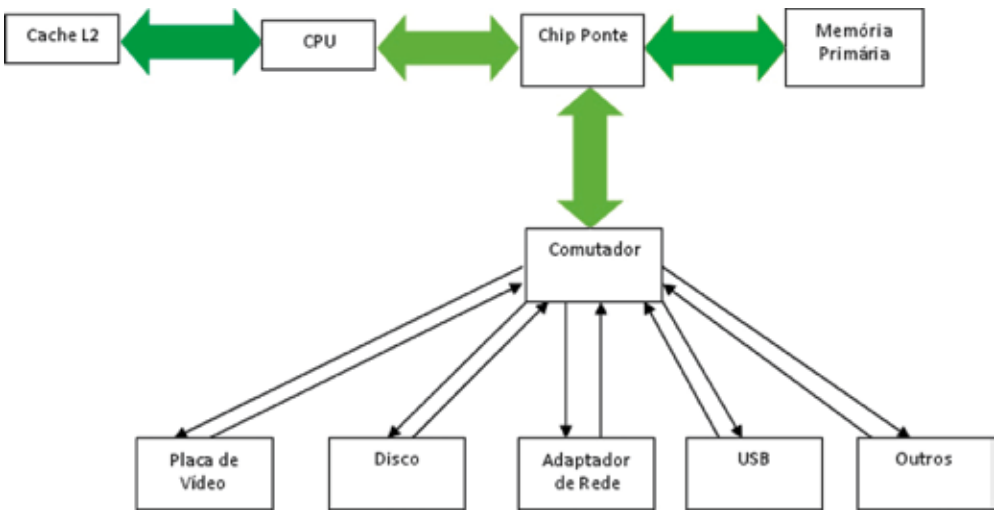


Figura 92: Barramento PCI Express.

Fonte: TANENBAUM, 2007a.

8.4 *Chips* de processadores

Os *chips* estão em constante evolução, sendo que, a cada novo processador, algum item na arquitetura é modificado, como, por exemplo, a memória cache, o barramento interno, a quantidade de núcleo etc.

Um exemplo é o Pentium IV, que tem como base a geração da Intel denominada Netburst. Algumas de suas melhorias foram:

- Dados por pulso de *clock*: o Pentium IV começou a enviar quatro dados por pulso de *clock*, fazendo com que seu desempenho fosse quatro vezes melhor.
- O barramento interno que comunica o cache L1 e L2 passou para 256 *bits*, sendo que, nos processadores anteriores, chegou a 64 *bits*.
- Existência de aproximadamente 128 registradores internos, enquanto nos anteriores havia no máximo 40 registradores.

Podemos considerar como exemplo também a arquitetura Core, que apresenta mais de um núcleo de processamento e um *cache* L2 sendo único para os dois núcleos, evitando assim que o cache L2 de um núcleo acabe e o do outro não.

Outra característica que costuma ser modificada é o *pipeline*, que foi implantado pela Intel no 486 e dividia o processamento da informação em cinco níveis. Quando uma informação chegava para ser

processada, ela passava pelo primeiro nível, ficava um ciclo de *clock*, ia para o segundo nível, esperava um ciclo de *clock*, ia para o terceiro nível e assim por diante. Com essa característica, o nível 1 poderia ser ocupado assim que ficasse livre, não precisando esperar que a informação acabasse de passar pelos cinco níveis. Com a evolução dos processadores, o *pipeline* foi modificado para auxiliar os processadores na melhora do desempenho.

8.5 Nível de microarquitetura

A microaqruitetura é um nível da lógica digital e sua função é implementar a camada denominada *Instruction Set Architecture* (ISA), conforme já mencionado anteriormente. Verifique a figura a seguir:

Nível 5	Nível das linguagens orientadas para solução de problemas	
		Compilador
Nível 4	Nível da linguagem do montador	
		Montador
Nível 3	Nível do sistema operacional	
		Sistema Operacional
Nível 2	Nível da arquitetura do conjunto de instruções	
		Microprograma
Nível 1	Nível da microarquitetura	
		Hardware
Nível 0	Nível da lógica digital	

Figura 93 – Níveis de arquitetura

Um projeto de nível de microarquitetura dependerá do tipo de projeto a ser implementado, sempre levando em conta a *performance* e o custo desejado para o computador para o qual a microarquitetura será montada.

Um microprograma tem um conjunto de variáveis denominadas de *estado do computador*, as quais podem ser acessadas por todas as funções. Um exemplo de estado do computador é o *Program Counter* (PC), o qual indica o endereço de memória que contém a próxima instrução a ser executada.

Observação

As instruções podem ser simples ou complexas, depende do programa que será construído.

O caminho de dados é onde está a Unidade Lógica Aritmética (ULA) e todas as suas entradas e saídas. Assim como as Instruções, o caminho de dados também irá variar dependendo do microprograma a ser executado.

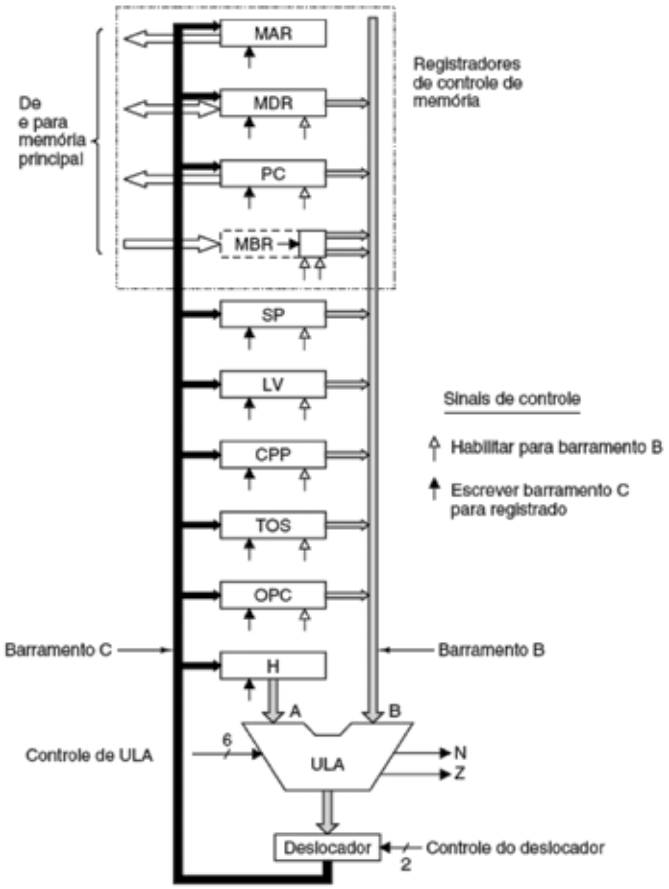


Figura 94 – Caminho da microarquitetura

A ULA possui basicamente três entradas, uma responsável pelo código binário referente à instrução a ser executada e as outras duas entradas referentes aos dados a serem processados (Barramentos A e B).

Há duas entradas de dados porque muitas operações utilizam dois deles, por exemplo: adição, subtração, *and*, *or* etc. Quando operações que não necessitam de dois dados são executadas, como é caso da instrução *not*, um dos dados é mantido para que o resultado não seja influenciado.

Nesse caminho, há um conjunto de 32 registradores que enviam o conteúdo para o barramento B. Esse barramento alimenta a ULA juntamente com o registrador H. A saída da ULA irá alimentar o Barramento C e os dados que irão para o Barramento C podem ser enviados a um ou mais registradores de acordo com a necessidade do programa.

Dentre os registradores, há quatro de controle de memória: MAR, MDR, PC e MBR:

- MAR (*Memory Address Register*): é o registrador de endereço de palavras para leitura ou escrita na memória.
- MDR (*Memory Data Register*): são os dados retirados da memória e que serão escritos.
- PC (*Program Counter*): indica o endereço de memória que contém a próxima instrução a ser executada.
- MBR (*Memory Buffer Register*): armazena a próxima instrução a ser executada.

A temporização desse caminho está ilustrada na figura a seguir:

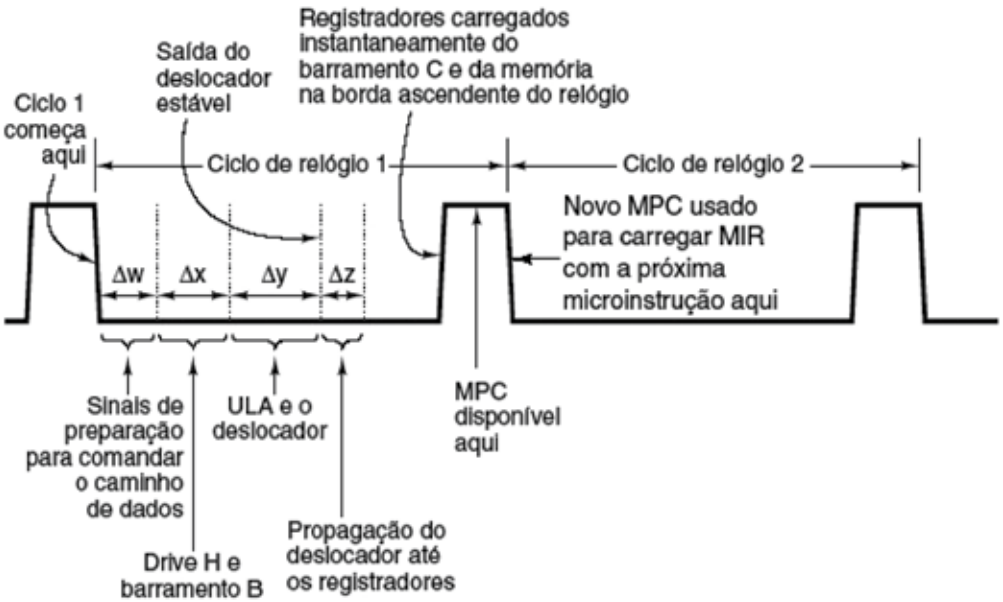


Figura 95 – Diagrama do ciclo de caminho da informação

Nessa figura, observa-se que é dado um pulso de *clock* e os *bits* que serão utilizados pelos registradores e pela ULA serão ativados no período negativo do ciclo, período esse denominado de Δw. Em seguida, o barramento A receberá os dados do registrador H e o barramento B receberá os dados de um dos registradores. Para que esses dados sejam inseridos no barramento, é necessário um período no ciclo de *clock* denominado Δx. Após esse período, a ULA fará todo o procedimento de acordo com a instrução recebida e enviará os dados para o deslocador. O intervalo de tempo necessário para essa operação recebe o nome de Δy. O deslocador irá então colocar o resultado no Barramento C para que este vá para o(s) registrador(es) endereçado(s). O período gasto nessa operação é denominado de Δz.

A comunicação com a memória pode ser executada de duas maneiras:

- Por meio dos registradores já mencionados MAR e MDR e essa comunicação envolveria 32 *bits*;
- Pelo registrador PC a comunicação seria por 8 *bits*. O PC lê o *byte* que está na memória e insere os 8 *bits* que foram lidos por ele no MBR.

Mas, qual é a diferença entre o MAR e o PC?

O MAR guarda o endereço de palavra e o PC guarda endereço de *byte*. Por exemplo: se for armazenado o valor 1 no MAR, ele irá ler os *bytes* 4 a 7, ou seja, os *bytes* que são referentes à palavra 1 – aqui, os 32 *bits* serão guardados no MDR. Se o valor 1 for armazenado no PC, este irá ler o *byte* 1 e armazenará os 8 *bits* no MBR.

Observe que na figura 95 há alguns registradores que possuem embaixo deles uma seta preta, o que indica um sinal que habilita a escrita e, mediante isso, nota-se que o registrador MBR, por sua vez, não possui essa seta, pois ele não recebe dados do barramento C.

Lembrete

O barramento B não pode receber dados de vários registradores ao mesmo tempo, caso isso ocorra, o resultado será um erro lógico e até mesmo físico.

O formato de uma microinstrução pode ser definido como:

- 9 *bits* para endereço: contém o endereço da próxima microinstrução que será utilizada.
- 3 *bits* para desvio: o desvio indica como a próxima microinstrução será selecionada.
- 8 *bits* para a ULA: nesses *bits*, as funções da ULA e do deslocador são setadas.
- 9 *bits* para o barramento C: nesses *bits*, serão selecionados os registradores que receberão os dados fornecidos ao barramento C pela ULA e pelo deslocador.
- 3 *bits* para a memória: os *bits* serão usados para setar as funções que usadas pela memória.
- 4 *bits* para o barramento B: servem para selecionar qual será o registrador que enviará os dados para o barramento B.

Se necessário, esse formato pode ser observado mais atentamente na figura 96.

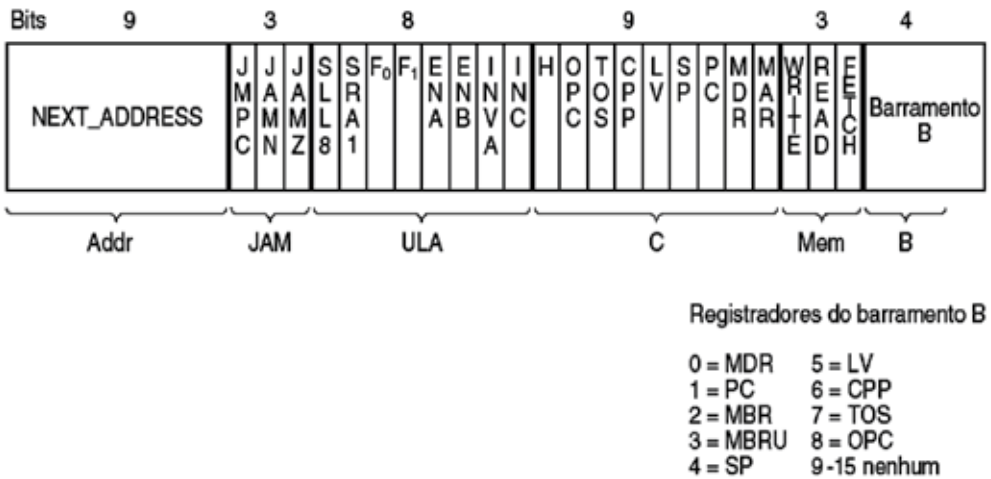


Figura 96 – Formato da microinstrução

Pilhas

As pilhas são recursos utilizados pelos programadores para armazenar itens na memória sem que estes sejam perdidos ou fiquem inacessíveis.

Existem dois tipos de pilhas: as pilhas de variáveis e as pilhas de operandos. A mais utilizada é primeira, reservada para armazenar variáveis.

A pilha de variáveis funciona da seguinte maneira: quando uma instrução chamar um procedimento, o registrador *LV* apontará para o endereço inicial das variáveis locais do procedimento em questão. Por exemplo, foi chamado um procedimento *X*, formado por *x1*, *x2* e *x3*, sendo que o registrador *LV* apontará para o endereço que está a primeira variável, *x1*. Nesse exemplo, o endereço será *100* e o registrador *SP* apontará para a última variável, *x3*, que será o endereço *108*, conforme demonstrado no diagrama a seguir:

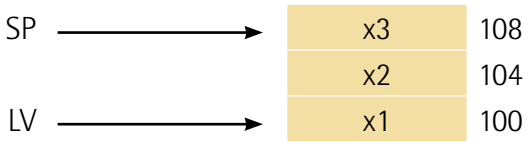


Figura 97 – Pilha.

Fonte: TANENBAUM, 2007a.

Caso o procedimento *X* chame um outro procedimento, o qual denominaremos de *Y* (*y1*, *y2*), então o *LV* passará a apontar para a variável inicial de *Y* (*y1*) e *SP* para a última variável de *Y* (*y2*), conforme aponta a ilustração a seguir:

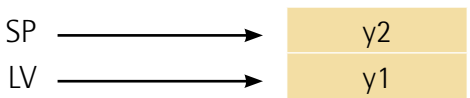


Figura 98 – Pilha.

Fonte: TANENBAUM, 2007a.

Se o procedimento *Y* chamar mais um procedimento, *Z* (*z1*, *z2*, *z3*, *z4*), então *LV* apontará para *z1* e *SP* para *z4*.

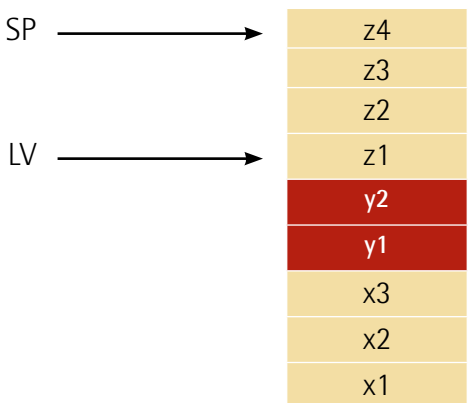


Figura 99 – Pilha.

Fonte: TANENBAUM, 2007a.

Quando o procedimento *Z* é executado e sai da pilha, *LV* e *SP* voltam a apontar para o procedimento *Y*.

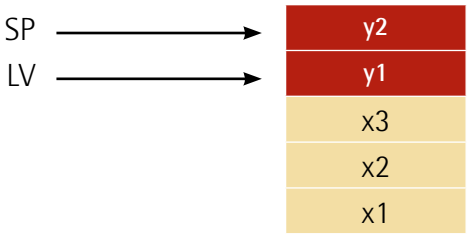


Figura 100 – Pilha.

Fonte: TANENBAUM, 2007a.

Sendo executado o procedimento *Y*, os registradores apontarão novamente para o procedimento *X*:

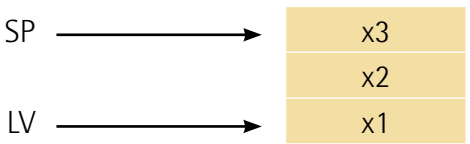


Figura 101 – Pilha.

Fonte: TANENBAUM, 2007a.

Já a pilha de operandos armazenará os próprios operandos, por exemplo: se for executado um cálculo aritmético, onde se somará *x2* com *x3* e o resultado será *x1*, ou seja, *x1* = *x2* + *x3*, lê-se então o procedimento *X* onde *LV* e *SP* apontarão para *x1* e *x3*.

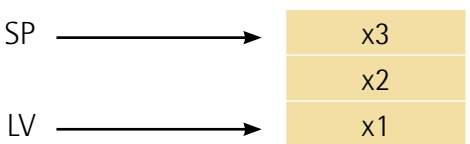


Figura 102 – Pilha.

Fonte: TANENBAUM, 2007a.

Em seguida, tomando-se novamente *x2*, *SP* passará a apontar para o novo *x2*:

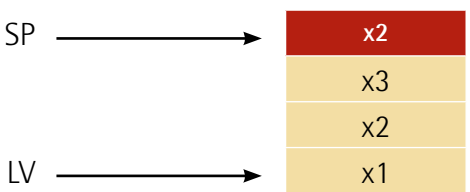


Figura 103 – Pilha.

Fonte: TANENBAUM, 2007a.

Do mesmo modo, tomando-se *x3*, SP apontará para o novo *x3*:

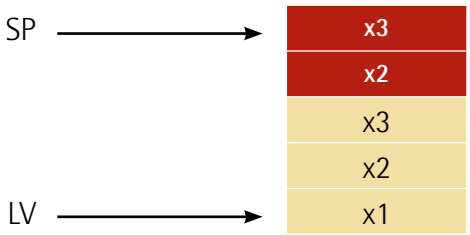


Figura 104 – Pilha.

Fonte: TANENBAUM, 2007a.

Assim, os dados vão para a ULA e a operação é executada, retornado em seguida a conta para a pilha:

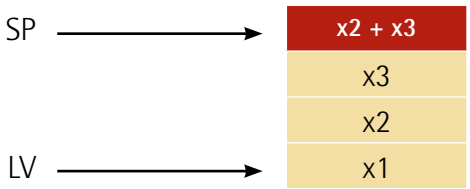


Figura 105 – Pilha.

Fonte: TANENBAUM, 2007a.

Por fim, o valor obtido vai para o Barramento, passa pela ULA e substitui o valor de *x1*:

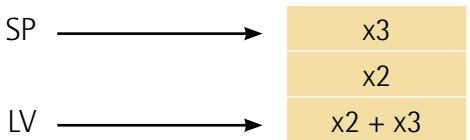


Figura 106 – Pilha.

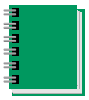
Fonte: TANENBAUM, 2007a.

Resumo

Vimos como funcionam portas lógicas e seus sinais e entendemos o funcionamento dessa lógica na memória do computador. Reforçamos a ideia da microarquitetura dos computadores, abordando o assunto a fim de expressar de maneira mais simples possível seu conceito de funcionamento. Além disso, procuramos compreender ao longo do conteúdo aqui exposto como também se dá o funcionamento da ULA (Unidade Lógica Aritmética) e das pilhas.

Saiba mais

http://www.di.ufpb.br/raimundo/ArqDI/Arq5.htm



Exercícios

1. Os computadores são construídos com o objetivo definido de processar dados obtendo algum tipo de resultado. Entre as tarefas comuns de um sistema computacional podem ser citadas: operações aritméticas (somar, subtrair, multiplicar, dividir,...), operações lógicas (and, or, xor, not,...), movimentação de dados (memória – CPU, CPU – memória, registrador – registrador,...), desvios (alteração da sequência de execução das instruções) e operações de entrada e saída. O principal dispositivo da CPU para a realização das atividades de processamento é _____.

Qual das alternativas preenche a lacuna acima?

A) A memória RAM.

B) A memória Cache.

C) O barramento.

D) A ULA.

E) A MBR.

Resposta correta: alternativa D.

Análise das alternativas

A) Alternativa incorreta.

Justificativa. É a memória principal do computador. Permite ler e escrever dados. Para que um processo possa ser executado pela CPU, ele deve estar na memória RAM.

B) Alternativa incorreta.

Justificativa. A memória cache é um dispositivo de acesso rápido, que serve de intermediário entre um operador de um processo e o dispositivo de armazenamento ao qual esse operador acede.

C) Alternativa incorreta.

Justificativa. O barramento é dito como o caminho elétrico que faz a ligação de dois ou mais dispositivos.

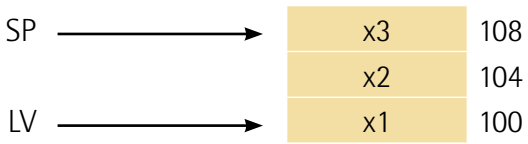
D) Alternativa correta.

Justificativa. Ela utiliza os registradores de propósito gerais (ACC) como auxiliares à função de processamento. A ULA é um aglomerado de circuitos lógicos e componentes eletrônicos simples que, integrados, realizam as funções acima citadas.

E) Alternativa incorreta.

Justificativa. MBR (Memory Buffer Register), armazena a próxima instrução que será executada.

2. Nos cursos de computação a disciplina de Estrutura de Dados é frequentemente oferecida. Para os alunos, a disciplina apresenta um grau de dificuldade elevado por apresentar o estudo de uma serie de algoritmos para a manipulação de dados. Os algoritmos estudados nesta disciplina são utilizados na implementação de várias estruturas de outras disciplinas tais como: Banco de Dados, Sistemas Operacionais, Compiladores etc. Uma das estruturas mais utilizadas é a pilha. As pilhas são utilizadas pelos programadores para armazenarem itens na memória sem que os mesmos sejam perdidos, e que possam ser acessados quando for necessário. A mais utilizada é a pilha de variáveis, que funciona da seguinte maneira, quando uma instrução chamar um procedimento, o registrador LV apontará para o endereço inicial das variáveis locais do procedimento em questão. Por exemplo, foi chamado um procedimento X, formado por x1, x2 e x3; sendo que o registrador LV apontará para o endereço que está a primeira variável, x1, no exemplo o endereço será o 100, e o registrador SP apontará para a última variável, x3, que será o endereço 108; conforme a figura



Pilha

Fonte: TANENBAUM, 2007a.

I. O elemento X1 poderá ser removido antes do elemento X3.

II. O elemento X2 somente poderá retirado da pilha após o elemento x3 ser retirado.

III. O primeiro elemento a entrar na pilha será o ultimo a sair.

IV. O primeiro elemento a entrar na pilha será o primeiro a sair.

Quais das afirmativas são verdadeiras?

A) I e IV apenas.

B) II e III apenas.

C) I, II e IV apenas.

D) II e IV apenas.

E) I, II, III e IV.

Resolução desta questão na Plataforma.

FIGURAS E ILUSTRAÇÕES

Figura 01

ENTRAPROSSAIDA2.JPG. Largura: 354 pixels. Altura: 275 pixels. Formato: JPEG. Disponível em: < http://rdrblog.com.br/blog/>. Acesso em: 09 jun. 2011.

Figura 03

IMAGE36.JPG. Largura: 640 pixels. Altura: 480 pixels. Formato: JPEG. Disponível em: <http://matematica.com.sapo.pt/abaco.htm>. Acesso em: 26 nov. 2010.

Figura 04

SLIDERULE.JPG. Largura: 320 pixels. Altura: 263 pixels. Formato: JPEG. Disponível em: <http://piano.dsi.uminho.pt/museuv/imagens/sliderule.jpg>. Acesso em: 26 nov. 2010.

Figura 05

CIRCSLIDE.JPG. Largura: 247 pixels. Altura: 92 pixels. Formato: JPEG. Disponível em: <http://piano.dsi.uminho.pt/museuv/imagens/circslide.jpg>. Acesso em: 26 nov. 2010.

Figura 06

PASCALINA.JPG. Largura: 640 pixels. Altura: 421 pixels. Formato: JPEG. Disponível em: <http://upload.wikimedia.org/wikipedia/commons/archive/1/12/20100809225404!Pascalina.jpg >. Acesso em: 26 nov. 2010.

Figura 07

LEIBNITZRECHENMASCHINE.JPG. Largura: 800 pixels. Altura: 600 pixels. Formato: JPEG. Disponível em: <http://commons.wikimedia.org/wiki/File:Leibnitzrechenmaschine.jpg>. Acesso em: 26 nov. 2010.

Figura 08

050114_2529_DIFFERENCE.JPG. Largura: 460 pixels. Altura: 460 pixels. Formato: JPEG. Disponível em: <http://agaudi.files.wordpress.com/2008/04/050114_2529_difference.jpg>. Acesso em: 26 nov. 2010.

Figura 09

10.JPG. Largura: 460 pixels. Altura: 306 pixels. Formato: JPEG. Disponível em: <http://agaudi.files.wordpress.com/2008/04/10.jpg?w=460&th=306>. Acesso em: 26 nov. 2010.

Figura 10

JACQUARD_LOOM_P1040320.JPG. Largura: 1920 pixels. Altura: 2560 pixels. Formato: JPEG. Disponível em: <http://upload.wikimedia.org/wikipedia/commons/b/b6/Jacquard_loom_p1040320.jpg>. Acesso em: 26 nov. 2010.

Figura 11

COI53.JPG. Largura: 443 pixels. Altura: 352 pixels. Formato: JPEG. Disponível em: <http://www-03.ibm.com/ibm/history/exhibits/markI/images/coi53.jpg>. Acesso em: 26 nov. 2010.

Figura 12

ENIAC_PENN1.JPG. Largura: 796 pixels. Altura: 599 pixels. Formato: JPEG. Disponível em: <http://commons.wikimedia.org/wiki/File:ENIAC_Penn1.jpg>. Acesso em: 26 nov. 2010.

Figura 14

VALVULA_PB_G_1_EDITADO.GIF. Largura: 258 pixels. Altura: 358. Formato: GIF. Disponível em: <http://www.lsi.usp.br/~chip/como_funcionam.html>. Acesso em: 1º dez. 2010.

Figura 15

133.JPG. Largura: 118 pixels. Altura: 98. Formato: JPEG. Disponível em: <http://www.centralcomponentes.com.br/produto_view.php?id=4485>. Acesso em: 1º dez. 2010.

Figura 16

TX-0.COMPONENTS.C1950.102631238.LG.JPG. Largura: 392 pixels. Altura: 595. Formato: JPEG. Disponível em: <http://www.computerhistory.org/collections/accession/102631238>. Acesso em: 1º dez. 2010.

Figura 17

180PX-PDP-1.JPG. Largura: 180 pixels. Altura: 135. Formato: JPEG. Disponível em: <http://pt.wikilingue.com/es/Programmed_Data_Processor>. Acesso em: 1º dez. 2010.

Figura 18

PDP-8.JPG. Largura: 477 pixels. Altura: 493. Formato: JPEG. Disponível em: <http://newmedia.umaine.edu/images/Craig%20Dietrich/pdp-8.jpg>. Acesso em: 1º dez. 2010.

Figura 21

CDC_6600_CONSOLE.JPG. Largura: 400 pixels. Altura: 300. Formato: JPEG. Disponível em: <http://vinachip.com/images/CDC_6600_console.jpg>. Acesso em: 1º dez. 2010.

Figura 22

COI93.JPG. Largura: 443 pixels. Altura: 359. Formato: JPEG. Disponível em: <http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_intro2.html>. Acesso em: 1º dez. 2010.

Figura 23

PDP-11-40.JPG. Largura: 370 pixels. Altura: 493. Formato: JPEG. Disponível em: <http://www.esacademic.com/pictures/eswiki/80/Pdp-11-40.jpg>. Acesso em: 02 dez. 2010.

Figura 24

ALTAIR-8800.JPG. Largura: 427 pixels. Altura: 341. Formato: JPEG. Disponível em: <http://radames.manosso.nom.br/bitabit/files/altair-8800.jpg>. Acesso em: 03 dez. 2010.

Figura 25

APPLE1_1244596I.JPG. Largura: 571 pixels. Altura: 400. Formato: JPEG. Disponível em: <http://i.telegraph.co.uk/telegraph/multimedia/archive/01244/apple1_1244596i.jpg>. Acesso em: 03 dez. 2010.

Figura 26

APPLE2.JPG. Largura: 328 pixels. Altura: 432. Formato: JPEG. Disponível em: < http://www.verso.co.nz/wp-content/uploads/2008/02/apple2.jpg>. Acesso em: 03 dez. 2010.

Figura 27

6705PH01.JPG. Largura: 443 pixels. Altura: 342. Formato: JPEG. Disponível em: <http://www-03.ibm.com/ibm/history/exhibits/pc25/images/6705PH01.jpg>. Acesso em: 06 dez. 2010.

Figura 28

MACINTOSH.JPG. Largura: 500 pixels. Altura: 408. Formato: JPEG. Disponível em: <http://www.hephesto.com/agrega/wp-content/uploads/2008/12/macintosh.jpg>. Acesso em: 03 dez. 2010.

Figura 30

CANSTOCK4940654.PNG. Largura: 325 pixels. Altura: 216 pixels. Formato: PNG. <http://www.canstockphoto.com.br/processador-4940654.html>. Acesso em: 09 jun. 2011.

Figura 37

MEMORIA1.JPG. Largura: 300 pixels. Altura: 300 pixels. Formato: JPEG. Disponível em: <http://tecnologialivre.com.br/blog/wp-content/uploads/2010/08/memoria1.jpg>. Acesso em: 30 nov. 2010.

Figura 43

IMAGEVIEW.JPG. Largura: 300 pixels. Altura: 300 pixels. Formato: JPEG. Disponível em: <http://www.clubedohardware.com.br/imageview.php?image=26099>. Acesso em: 09 jun. 2011.

Figura 49

TECLADO-PARA-COMPUTADOR.JPG. Largura: 414 pixels. Altura: 276 pixels. Formato: JPEG. Disponível em: <http://www.techclube.com.br/blog/wp-content/uploads/2010/03/teclado-para-computador.jpg>. Acesso em: 10 jan. 2011.

Figura 50

811COMPUTER_MOUSE.JPG. Largura: 700 pixels. Altura: 467 pixels. Formato: JPEG. Disponível em: <http://www.faqs.org/photo-dict/photofiles/list/439/811computer_mouse.jpg>. Acesso em: 10 jan. 2011.

Figura 51

MONITOR-15.GIF. Largura: 299 pixels. Altura: 299 pixels. Formato: GIF. Disponível em: <http://3.bp.blogspot.com/_MLUmLiODDmE/SeNEayWOiZI/AAAAAAAAACA/8SUfoAH-XtU/s1600-h/monitor-15.gif/>. Acesso em: 10 jan. 2011.

Figura 52

MONITORES-LCD-15. JPG. Largura: 396 pixels. Altura: 396 pixels. Formato: JPEG. Disponível em: <http://blogmaneiro.com/www/blogmaneiro.com/wp-content/gallery/monitores-lcd/monitores-lcd-15.jpg>. Acesso em: 10 jan. 2011.

Figura 54

C11C558011_IMAGENFEATUREV4.GIF. Largura: 260 pixels. Altura: 200 pixels. Formato: GIF. Disponível em: <http://www.epson.com.br/pais/GL/images/C11C558011_ImagenFeatureV4.gif>. Acesso em: 10 jan. 2011.

Figura 55

Z1320.JPG. Largura: 267 pixels. Altura: 292 pixels. Formato: JPEG. Disponível em: <http://images.lexmark.com/vgn/files/portal/z1320.jpg>. Acesso em: 10 jan. 2011.

Figura 56

ARQUIVOEXIBIR.ASPX.PNG. Largura: 350 pixels. Altura: 350 pixels. Formato: PNG. Disponível em: <http://imagens.lojahp.com.br/Control/ArquivoExibir.aspx?IdArquivo=3780>. Acesso em: 10 jan. 2011.

Figura 58

CM765A_190X170.JPG. Largura: 170 pixels. Altura: 190 pixels. Formato: JPG. Disponível em: <http://h10010.www1.hp.com/wwpc/images/emea/CM765A_190x170.jpg>. Acesso em: 10 jan. 2011.

Figura 83

TUT02010.GIF. Largura: 400 pixels. Altura: 206 pixels. Formato: GIF. Disponível em: <http://www.prof2000.pt/users/afaria2004/imagens/tut02010.gif>. Acesso em: 09 jun. 2011.

REFERÊNCIAS

Audiovisuais

OS FILÓSOFOS no cinema. Dir. Roberto Rossellini. Versátil *Home Video* sobre licença do Instituto Luce, coletânea, 2009.

Textuais

BOSCO, J. Portas lógicas. s. d. Disponível em <http://www.inf.ufsc.br/ine5365/portlog.html>. Acesso em: 09 jun. 2011.

CARROL, P. *Big blues: a derrocada da IBM***.** Rio de Janeiro: Ediouro, 1994.

COLOSSETTI, A. P. Construção de uma rede bayesiana para diagnóstico da doença de alzheimer a partir de neuroimagem, histórico e sintomas**.** São Paulo: 2009. Originalmente apresentada como dissertação de mestrado, apresentada na FEI, 2009. Disponível em: <http://www.fei.edu.br/Download%20de%20Pesquisas/FEI-IAAA-011-AdrianeColossetti.pdf>. Acesso em: 30 nov. 2010.

COMPUTADOR e Internet: 1900-1939. s. d. Disponível em: <http://www.cultura.ufpa.br/dicas/net1/int-h190.htm>. Acesso em: 09 jun. 2011.

FONSECA FILHO, C. *Historia da computação: o caminho do pensamento e da tecnologia.* Porto Alegre: EDIPUCRS, 2007. Disponível em <http://www.pucrs.br/edipucrs/online/historiadacomputacao.pdf>. Acesso em: 30 nov. 2010.

GREGG, J. *Ones and zeroes: understanding boolean algebra, digital circuits and the logic of sets.* New Jersey: John Wiley and Sons, 1998.

HISTÓRIA da computação. 2001. Disponível em <http://pt.wikipedia.org/wiki/Hist%C3%B3ria_do_computador#Gera.C3.A7.C3.B5es_de_computadores>. Acesso em: 09 jun. 2011.

INTRODUÇÃO à informática. Rio Grande do Sul: Universidade Federal de Pelotas, s. d. Disponível em <http://ci.ufpel.edu.br/treinamento/apostilas/nocoes_de_informatica/computador.pdf>. Acesso em: 09 jun. 2011.

LIVIO, M. *Razão áurea: a história de Fi, um número surpreendente*. Rio de Janeiro: Record, 2006.

NOBREGA FILHO, R. G. Fundamentos de *Hardware*. s. d. Disponível em: <http://www.di.ufpb.br/raimundo/ArqDI/Arq5.htm>. Acesso em: 09 jun. 2011.

SINGH, S. *O livro dos códigos*. Rio de Janeiro: Record, 2001.

STALLINGS, Willian. *Arquitetura e organização de computadores*. 5 ed. São Paulo: Pearson Prentice Hall, 2002.

TANENBAUM, A. S. *Organização estruturada de computadores*. 5ª edição. São Paulo. Ed. Pearson Prentice Hall, 2007a.

_____. *Sistemas operacionais modernos*. 5 ed. São Paulo: Pearson Prentice Hall, 2007b.

VON, Z *et al*. Capítulo 2. *In*: *Introdução à arquitetura de computadores*. s. d. Disponível em <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ea869_03/turma_U/cap2_parteII.pdf>. Acesso em: 09 jun. 2011.

Exercícios

Unidade I – Questão 1 (adaptada de): COMPANHIA DO METROPOLITANO DE SÃO PAULO (Metrô). *Concurso para provimento de cargo de analista* trainee *em Ciência da Computação*. Disponível em: <http://www.pciconcursos .com.br/provas/metro>. Acesso em: 08 abr. 2011.

Unidade II – Questão 2 (adaptada de): INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TOCANTINS – IFTO. *Concurso Público 2009*. Disponível em <http://www.pciconcursos.com.br/provas>. Acesso em: 05 abr. 2011.

Unidade III – Questão 1: INSTITUTO NACIONAL DE ESTUDOS E PESQUISAS EDUCACIONAIS ANÍSIO TEIXEIRA (INEP). *Exame Nacional de Desempenho dos Estudantes (ENADE) 2008*: Computação. Questão 38. Disponível em: <http://download.inep.gov.br/download/Enade2008_RNP/COMPUTACAO.pdf>. Acesso em: 09 jun. 2011.

Sites

<http://www.dca.ufrn.br/~pablo/FTP/arq_de_comp/apostilha/capitulo2.pdf>.

<http://rossano.pro.br/fatec/cursos/sistcomp/apostilas/organizacao-computadores-processadores.pdf>.

Interativa

Informações:
www.sepi.unip.br ou 0800 010 9000