

Parabéns!

52,0

a prova + completa que já vi!

Avaliação 1 - Arquitetura de computadores.

• Discente - Paulo Henrique Diniz de Lima Alencar.

Questão 1 -

2,0

HAB

resolução:

a) Computadores descartáveis - definiria como: pequenos chips que buscam realizar uma tarefa específica. Um exemplo seria o Chip RFID (Radio Frequency identification) possui menos de 0,5 mm de espessura e é capaz de transmitir seu número de volta à uma determinada antena quando pulsado por uma antena externa. Atualmente são utilizados como chips identificadores, buscando substituir o código de barras. Isso acaba evitando filas em estabelecimentos, pois os clientes não iriam precisar passar cada produto pelo caixa registradora. Com um chip RFID em cada produto não haveria essa necessidade, isso porque na saída da loja um leitor que possui um antena iria enviar um sinal, registrando cada produto que encontra-se no carrinho, realizando em seguida um somatório e enviando a fatura automaticamente para o cliente. Além disso, furtos seriam evitados, afinal cada produto retirado de loja nem a realização do pagamento iria enviar um sinal por rádio frequência a um determinado aparelho que poderia disparar um alarme. Com o tempo esses computadores descartáveis evoluíram bastante, sendo utilizados hoje no rastreamento de veículos, animais, bagagens aéreas e até mesmo nôo incorporados em moedas de dinheiro.

resolução:

b) Microcontroladores: são computadores que são embutidos em dispositivos que não são vendidos como computadores. Alguns exemplos são: geladeiras, micro-ondas ou alarmes. De forma simplória, eles gerenciam os dispositivos onde são embutidos executando determinados subsistemas e controlando determinadas funcionalidades. Cada microcontrolador possui um processador, memória e capacidade de entrada e saída. Uma aplicação bastante conhecida dos microcontroladores é na plataforma de controle embutido Arduino muito utilizada no ensino de eletrônica por professores e facilmente acessível para pessoas autodidatas, isso porque o Sistema Arduino é um projeto de hardware de fonte aberta. Como os microcontroladores são computadores de propósito específico e além disso estão embutidos em outros dispositivos, normalmente não percebemos o seu uso frequentemente. No entanto, boa parte dos eletrônicos em nossas casas possui um microcontrolador, quando você utiliza o micro-ondas de sua casa para esquentar algum alimento, tem um visor LCD que mostra os segundos que faltam para o seu alimento esquentar, isso é possível graças a um microcontrolador. Por fim, acredito que sua criação e evolução ocorreram devido a grande necessidade e automação de produtos que interagem diretamente com o usuário, como os eletrodomésticos.

resolução:

C) Computadores móveis e de jogos:

↳ Móveis - Embora essas máquinas não sejam tão poderosas quanto os computadores pessoais, eles possuem a grande vantagem de serem computadores de "bolso". Além disso, computadores móveis têm o requisito adicional de que utilizam o mínimo de energia possível para realizar suas tarefas.

↳ Computadores de Jogos - são considerados computadores normais, porém possuem um software limitado e com pouquíssima capacidade de extensão. Por outro lado, exigem grande poder de processamento relacionado ao recursos gráficos e capacidade de som. Por fim, são classificados como máquinas que possuem o sistema fechado, visto que, os hardware são fechados, impossibilitando assim algumas atualizações.

d) Computadores Pessoais: O termo "computadores pessoais" vai abranger os modelos de desktop e notebook. Somando-se a isso, é um computador de pequeno porte e normalmente baixo custo, que pode variar a depender das especificações da cada máquina.

Um PC desktop é composto basicamente de um gabinete (que possui a função de guardar/proteger os componentes que estão dentro dele), um monitor, teclado, mouse e estabilizador. Na parte interior do gabinete podemos encontrar:

- * Placa mãe;
- * Placa de vídeo;
- * Placa de Som.
- * Memória Ram;
- * Hd;
- * Placa de Rede
- * Processador;

O notebook é basicamente um PC-desktop mais compactado, menor, mas com os mesmos componentes de hardware que um desktop. Por fim, costumam executar os mesmos softwares que os PC's de desktop.

e) Servidores - são os famosos computadores "bombardeiros". Em termos de arquitetura, um servidor não é muito diferente de um computador pessoal. Ele é apenas mais rápido, focado em processamento de rede, mais espaço em disco e desvantajoso no quesito gráfico.

✓

f) Main Frames - famosos por ocuparem um sala inteira devido o seu tamanho. São mais rápidos que os servidores, possuem mais capacidade de entrada e saída e costumam ser equipados com discos que contém milhares de gigabytes de dados. Apesar de serem caros, é comum serem mantidos em funcionamento por causa dos Sistemas Iguais, dados e procedimentos operacionais que possuem. Assim é mais barato para alguns milhões de dólares de vez em quando na manutenção, do que reprogramar todos as suas aplicações para máquinas menores.

2 - Resolução:

2.0

Na programação costuma-se dar o nome "contador" a uma variável responsável por armazenar um determinado valor que será incrementado no decorrer de algum algoritmo. Um fato que podemos observar é que a evolução dos computadores, assim como a variável contador, é cumulativa, ou seja, o seu número costuma aumentar com o tempo. É foi isso que resumidamente aconteceu no cenário da computação no desenrolar da história, a computação veio sofrendo um incremento gigantesco na evolução do computadores, desde os primórdios até o nosso recente âmbito social.

Para se ter um ideia, a gênese do computadores data de séculos atrás, com uma ferramenta bastante simples, utilizada por diversos povos na realização de cálculos no cotidiano, esta, recebe o nome de ábaco. Tratava-se basicamente de uma ferramenta muito útil na hora de realizar pequenas operações matemáticas, como adição e subtração, sendo usado na época demasiadamente por diversos povos, como: Grécia, Babilônia, Egito, Roma, etc.

O tempo foi passando e o homem deu continuidade ao que costuma fazer de melhor: construir ferramentas que, reduza o esforço físico e deixe sua vida muito mais fácil. Assim, em 1942, o grande matemático Blaise Pascal elaborou o que os historiadores chamam da primeira calculadora mecânica da história, a chamada Máquina Pascaliana. A priori, Pascal desejava que a máquina realizasse 4 operações, lamentavelmente a ferramenta de calcular, só foi capaz de realizar somas e subtrações. No entanto, anos mais tarde o alemão Gottfried Leibnitz conseguiu desenvolver uma calculadora capaz de efetuar as quatro operações, além de raízes quadradas.

Dando continuidade as diversas contribuições na computação, chegamos no considerado avô do computador, isso baseado no ponto de vista do hardware. O chamado Charles Babbage, foi responsável por contribuir consideravelmente com suas ideias, desenvolvendo diversas máquinas capazes de realizar cálculos logarítmicos e resolver equações trigonométricas. Infelizmente boa parte dos seus inventos não foram implementados, seja pela falta de recursos financeiros ou limitações técnicas do seu tempo.

A história começou a tomar novos caminhos durante a Segunda Guerra Mundial e novos computadores foram surgindo, durante, e após o maior conflito bélico da história. Nessa época, diversas máquinas foram construídas, Mark I desenvolvida em 1944 pela Universidade de Harvard e o Colossus criado em 1946 por Alan Turing, são os dois grandes destaque da época.

A partir de 1946, damos início à computação moderna, que costuma ser dividida em várias gerações. A primeira geração data de 1946 - 1959 na qual a principal característica era o uso de válvulas eletrônicas, grande quantidade de fios e máquinas com tamanhos gigantescos. A principal máquina dessa época foi o ENIAC, com aproximadamente 30 metros de comprimento, 6 metros de largura e 30 toneladas.

Já a segunda geração datada de 1959 - 1964, tem como principal característica o uso de transistores, máquinas com menores tamanhos e um novo patamar de velocidade. Depois de alguns anos, em 1964 inicia-se a terceira geração, onde os computadores começaram a fazer uso de circuitos integrados, diminuindo consideravelmente o tamanho e o preço dos computadores no mercado.

Chegamos a famosa quarta geração, onde nomes como o de Bill Gates e Steve Jobs surgiu de maneira estrondosa, graças a realização dos seus feitos no âmbito computacional e empresarial. Essa geração é conhecida basicamente pelo aparecimento do famoso cérebro do computador, ou melhor, microprocessador, capaz de realizar bilhões de operações em milésimos de segundo. Somando-se a isso, nessa geração os circuitos também acabaram se tornando muito mais integrados, e por consequência os computadores acabaram ficando menores e mais baratos, adentrando assim, boa parte das residências. Hoje andamos na rua com aparelhos em nossos bolsos que realizam milhões de processamento por segundo. E o mais interessante é que eles são dezenas, centenas ou até milhares de vezes menores e mais leves que os primeiros computadores.

3- Resolução:

2,10

Atualmente estamos tão próximos dos computadores, que até esquecemos que no fundo eles "preferem" resolver os problemas utilizando simplesmente 0s e 1s. Já nós humanos, preferimos lidar e resolver boa parte dos problemas de uma maneira diferente e normalmente evitamos números. Se um computador tivesse vida e alguém perguntasse a ele qual seria a melhor forma de realizar uma comunicação, ele provavelmente responderia "Utilizando 0s e 1s". Por outro lado, se a mesma pergunta fosse feita a um brasileiro ele responderia, "utilizando o bom e velho português". Se em seguida perguntassemos ao computador quanto é $5 + 4$, ele iria converter

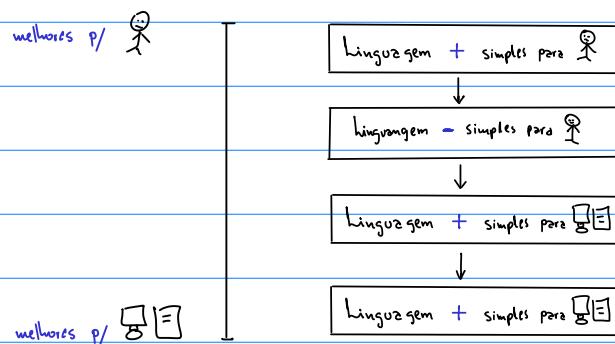
0 5 e 0 4 em suas representações binárias (decimal \rightarrow 5, binário \rightarrow 101, decimal \rightarrow 4, binário \rightarrow 100), realizaria a soma dos números binários $\begin{array}{r} 101 \\ 100 \\ \hline 1001 \end{array}$ pegaria o 1001, corresponderia a um determinado código Unicode, e posteriormente buscava o glifo (desenho/representação em símbolo) do algarismo 9. Só depois mostraria uma representação do número 9 na tela. Já nós humanos, realizariamos a soma utilizando a base decimal e dariamos nossa resposta falando por extenso "Nove".

Com esses exemplos citados acima, podemos notar que existe uma grande lacuna entre o que é conveniente para as pessoas e o que é conveniente para os computadores. Se para o computador é mais fácil e vantajoso no quesito desempenho realizar somas utilizando a base 2, para os humanos é mais eficiente utilizar a base 10. No entanto, é sabido que precisamos escrever uma sequência de instruções (programa) e essas instruções precisam ser entendidas tanto por humanos, quanto por máquinas. Então como vamos fazer isso, já que as duas constumam resolver seus problemas de uma maneira totalmente diferente? Uma solução seria utilizar a abordagem de abstrações por comandos.

A abordagem de abstrações por comandos funciona da seguinte forma: estamos cansados de saber que computadores preferem manipular os bits 0 e 1, porém, isso é ótimo para as máquinas e péssimo para os humanos. Por outro lado, os humanos preferem manipular palavras ou abreviações. Então como conciliar comandos/instruções de natureza numérica com comandos passados por meio de abreviações ou palavras? Podemos usar a abstração por comandos, onde um programador pode escrever programas utilizando determinadas abreviações, como por exemplo, usar a abreviação LOAD, do que utilizar o valor binário dessa operação, isto é, 00101101. Assim, em vez de ficar escrevendo 0s e 1s, basta abstrair e digitar a palavra LOAD (comando utilizado na linguagem Assembly). Claro que em seguida, aquele comando LOAD será convertido pelo Montador para valores binários que representam essa operação para que assim possa ser executado diretamente por circuitos eletrônicos.

- Montador é responsável por converter instruções Assembly para o equivalente em linguagem de máquina.

* Simplificações do modelo de abstrações por comandos:



Tentando explicar a abordagem por comandos de maneira mais formal, imagine que temos um programa escrito em linguagem que chamarímos de **L0** (apelido carinhoso p/ linguagem 0). Além disso, outra informação importante é que programas escritos em **L0**, podem ser executados diretamente pelos circuitos eletrônicos pois são simplesmente cadeias de bytes representando instruções e comandos. No entanto, **L0** vai ser difícil para humanos, então uma solução seria desenvolver um linguagem **L1** que não seja tão diferente de **L0**, porém **L1** deve ser um pouco mais próximo de uma linguagem que alivie o trabalho do programador da hora de escrever programas. Voltarei nesse assunto neste, mas antes precisamos entender Tradução e interpretação.

Como dito anteriormente **L1** não pode ser tão diferente de **L0**, pois a tradução ou a interpretação devem ser práticas.

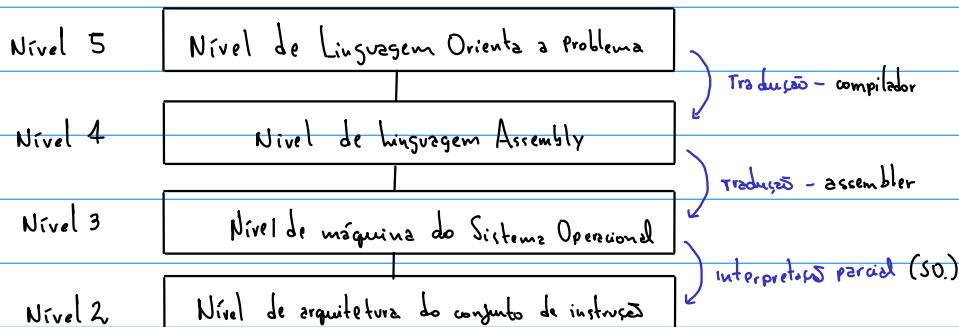
Tradução - o programa **L1** inteiro é convertido para um **L0**, o programa **L1** é descartado e depois **L0** é carregado na **[]** memória do computador e executado.

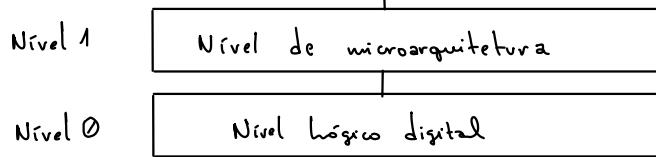
Interpretação - depois que cada instrução **L1** é examinada e decodificada, ela é executada de imediato. Nenhum programa traduzido é gerado. Aqui o computador está no controle do computador. Para ele o programa **L1** é apenas dados.

Voltando a questão de buscar ALIVIAR o trabalho do programador, por mais que programar em **L1** seja melhor que programar utilizando **L0**, a linguagem **L1** ainda está longe de aliviar o trabalho do programador. Então a abordagem óbvia seria inventar outro conjunto de instruções que seja mais orientado a pessoas e menos orientado a máquinas que a **L1**. Esse terceiro conjunto também forma um linguagem, que podemos chamar de **L2**. Assim, os programadores podem escrever programas em **L2** exatamente como se existisse uma máquina real com Linguagem de Máquina **L2**. Esse programas podem ser Traduzidos para **L1** ou **L0**.

Dessa forma, a invenção de toda uma série de linguagens, cada uma mais conveniente que suas antecessoras, pode seguir indefinidamente, até chegar a uma linguagem adequada. Cada linguagem usa a sua antecessora como base, portanto, podemos considerar um computador que use essa técnica como uma série de Camadas ou Níveis, um sobre o outro.

* Multinível atualmente





- Nível 0 - é o hardware verdadeiro da máquina, cujos circuitos executam os programas em linguagem de máquina de nível 1. Esse nível é formado basicamente por portas lógicas ou digitais.
 - Nível 1 - nível de microprogramação, é o verdadeiro nível de máquina, havendo um programa denominado microprograma, cuja função é interpretar as instruções de nível 2. A instrução neste nível é denominada microinstrução.
 - Nível 2 - é chamado de arquitetura do conjunto de instruções. Os manuais de referência para o uso de máquinas, fornecidos pelos fabricantes, referem-se a este nível.
 - Nível 3 - é o nível de máquina do S.O. É um nível híbrido, com maior parte das instruções executadas no nível ISA. As instruções de nível 3 são executadas por um interpretador rodando no nível 2, chamado Sistema Operacional.
 - Nível 4 - esse nível é voltado ao usuário, que tem um problema a solucionar. **OBS:** os níveis 2 e 3 não sempre interpretador, o nível 4 e acima são geralmente traduzidos.
OBS: As linguagens de nível 1, 2 e 3 são de natureza numérica - boa para máquinas e péssimas para humanos. A partir do nível 4, as linguagens contêm palavras ou abreviações.
- O nível 4 é o de linguagem Assembly. Esse nível permite que as pessoas escrevam programas p/ níveis 1, 2 e 3 de forma amigável. Programas escritos em Assembly são primeiro traduzidos para linguagens de nível 1, 2 e 3 por meio de um programa chamado Assembler, que é um montador.
- Perfeito!**
- Nível 5 - encontramos as linguagens de alto-nível: Java, C, C++, C#, PHP...
- Programas escritos em linguagens de alto-nível são traduzidas para os níveis 3 ou 4 por compiladores.

Em conclusão, o conceito de microinstrução se encaixa no Nível 1, ou seja, o verdadeiro nível de máquina, onde existe um programa denominado microprograma, cuja função é interpretar as instruções de nível 2. Essa arquitetura de microprogramação é constituída por um conjunto de Microinstruções que manipulam, fundamentalmente, memória de rascunho, além disso as microinstruções que compõem o microprograma são instruções primitivas que codificam um único ciclo de máquina a ser executado no caminho de dados da CPU, ou seja, as operações do caminho de dados é basicamente controlada por o microprograma.

Por fim, o conceito de microinstruções impactou no desenvolvimento dos computadores, ao passo que tornou os computadores mais confiáveis, um menor número de circuitos eletrônicos, tornando os computadores mais fáceis de montar e entender.

4 - Resolução:

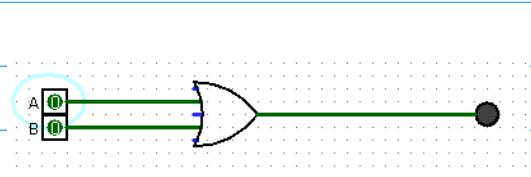
$Z = \bar{A}B$

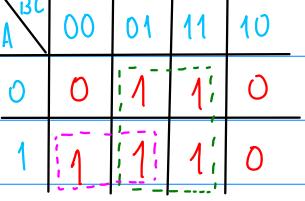
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

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

- $A = \cancel{0/1}$ e $B = 1 \quad \therefore B \quad \left. \begin{array}{l} \\ \end{array} \right\} A + B$
- $A = 1$ e $B = \cancel{0/1} \quad \therefore A$

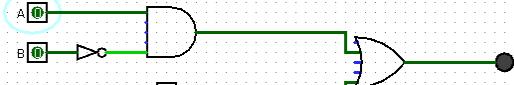
Imagem da simulação:



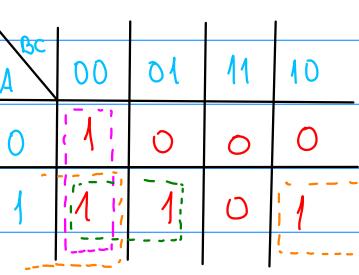
b) 

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

Imagens : simulador



$\bullet A = 1 \quad B = 0 \quad c = 0/1 \quad \therefore A \cdot \bar{B} + C \quad \checkmark$
 $\bullet A = 0/1 \quad B = 0/1 \quad c = 1 \quad \therefore C \quad \checkmark$

c) 

A	B	C	S ¹		S ²	
			0	1	0	1
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	1	1	1	0
1	1	0	1	0	1	1
1	1	1	0	0	0	0

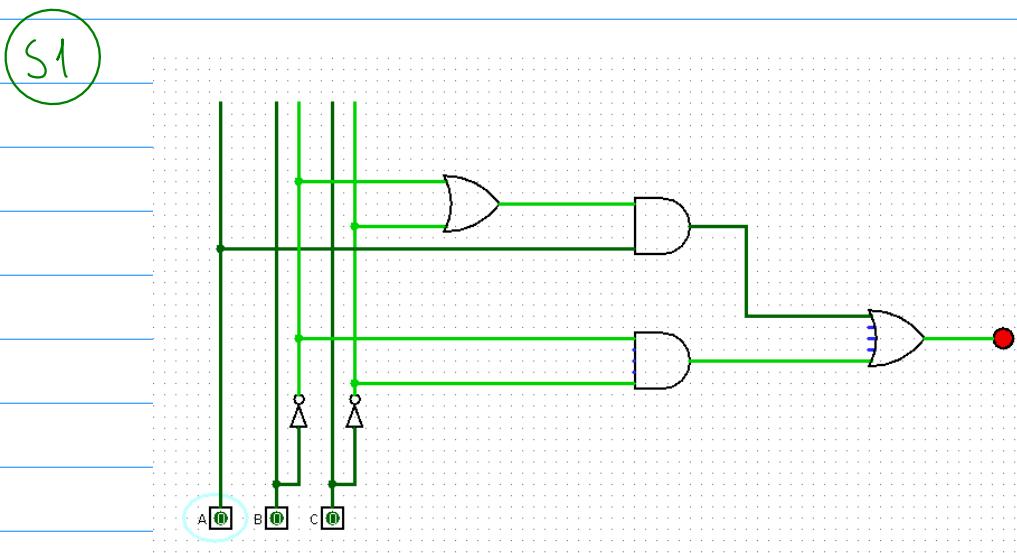
(S1)

$\bullet A = 0/1 \quad B = 0 \quad c = 0 \quad \therefore \bar{B} \cdot \bar{C} \quad \checkmark$
 $\bullet A = 1 \quad B = 0 \quad c = 0/1 \quad \therefore A \cdot \bar{B} \quad \checkmark$
 $\bullet A = 1 \quad B = 0/1 \quad c = 0 \quad \therefore A \cdot \bar{C} \quad \checkmark$

$\bar{B} \cdot \bar{C} + A \cdot \bar{B} + A \cdot \bar{C}$

$A \cdot (\bar{B} + \bar{C}) + (\bar{B} \cdot \bar{C})$

Imagem : simulador

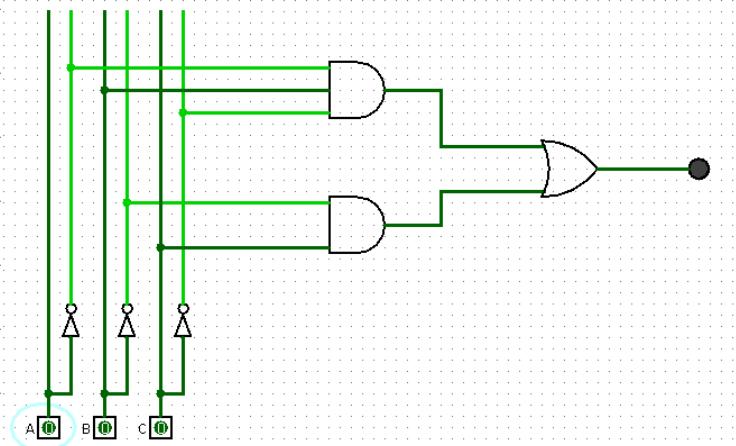


A	B	C	S ¹	S ²
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	0
1	0	1	1	
1	1	0	1	0
1	1	1	0	0

A	BC	00	01	11	10
		0	0	1	0
B	C	0	1	0	1
		1	0	1	0

$\bullet A = 0 \quad B = 1 \quad C = 0 \quad \therefore \bar{A} \cdot B \cdot \bar{C}$ } $\bar{A} \cdot B \cdot \bar{C} + \bar{B} \cdot C$
 $\bullet A = 0/1 \quad B = 0 \quad C = 1 \quad \therefore \bar{B} \cdot C$

(S2)



$$2^4 = 2 \cdot 2 \cdot 2 \cdot 2 = 16$$

d)

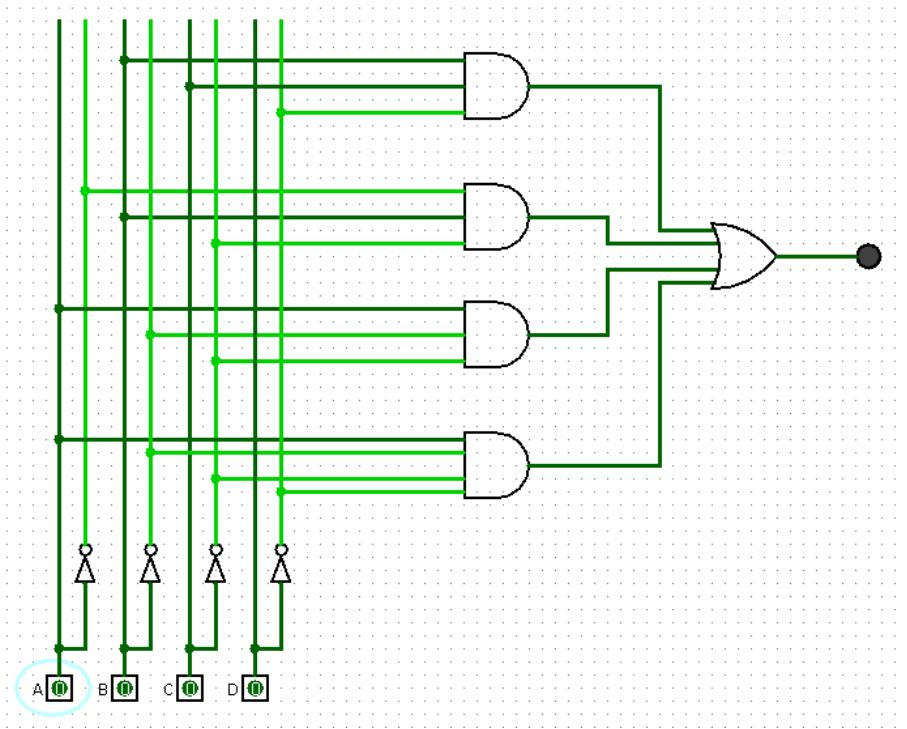
A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

AB	00	01	11	10
	00	0	0	0
BC	01	1	1	0
	01	1	0	1

$\bullet A = 0/1 \quad B = 1 \quad C = 1 \quad D = 0 \quad \therefore B \cdot C \cdot \bar{D}$
 $\bullet A = 0 \quad B = 1 \quad C = 0 \quad D = 0/1 \quad \therefore \bar{A} \cdot B \cdot \bar{C}$
 $\bullet A = 0/1 \quad B = 1 \quad C = 0 \quad D = 1 \quad \therefore B \cdot \bar{C} \cdot D$
 $\bullet A = 1 \quad B = 0 \quad C = 0 \quad D = 0 \quad \therefore A \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$

$$B \cdot C \cdot \bar{D} + \bar{A} \cdot B \cdot \bar{C} + B \cdot \bar{C} \cdot D + A \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} //$$

Imagem: simulador.



5) Resolução:

a) Multiplexador: pelo que entendi, um multiplexador é um circuito com 2^n ENTRADAS DE DADOS, 1 SAÍDA e n ENTRADAS DE CONTROLE, que vão ter o papel de **SELECIONAR** um das ENTRADAS. Essa ENTRADA selecionada é dirigida até a SAÍDA. Vou tentar dar um exemplo:

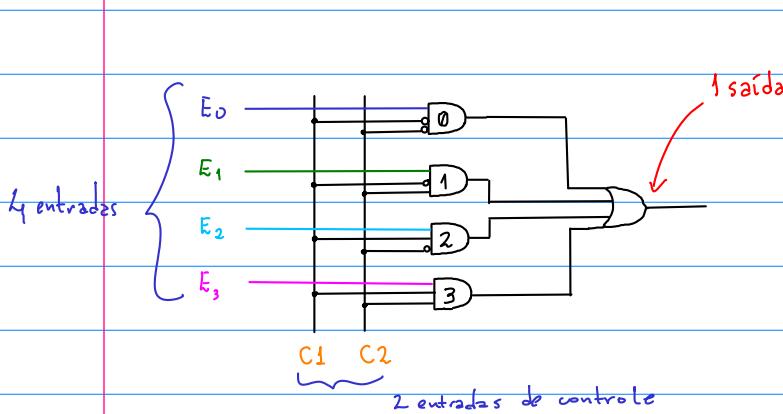
* Suponha que temos 2 entradas de controle, isto é $n = 2$

* Por consequência teremos $2^2 = 4$ entradas.

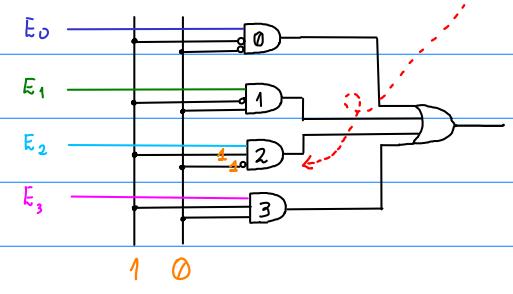
* Por padrão temos 1 saída.

* Lembrando que:

00	01	10	11	— binário
0	1	2	3	— decimal

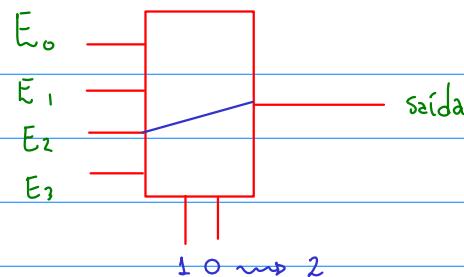


• Exemplo prático: quero selecionar porta 2



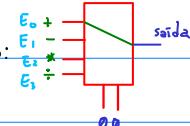
Como 2 em binário é 10

chip multiplexador de 4 entradas e 2 controladores



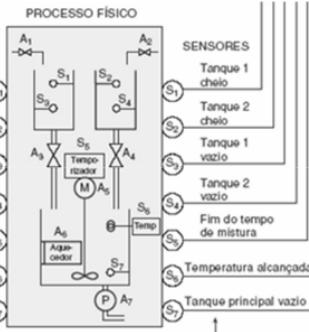
* Aplicações :

- O multiplexador poderia ser utilizado para escolher qual operação matemática será realizada, como por exemplo:



- Sequenciamento de operações - podemos controlar um processo industrial por meio de um sequenciamento de operações, porém para fazer isso é necessário combinar com outros circuitos digitais.

000 Parado
001 Abrir válvula A1 até S1 atuar
010 Abrir válvula A2 até S2 atuar
011 Abrir válvula A3 até S3 atuar
100 Abrir válvula A4 até S4 atuar
101 Ligar misturador A5 até S5
110 Ligar aquecedor A6 até S6
111 Ligar bomba A7 até S7



Fonte: slide-fábrica campos.

- Conversão Paralelo-Série - uma utilização típica da conversão paralela para serial é um teclado, onde cada acionamento de um tecla define implicitamente um número de 7 ou 8 bits que deve ser enviado por um enlace serial, como USB.

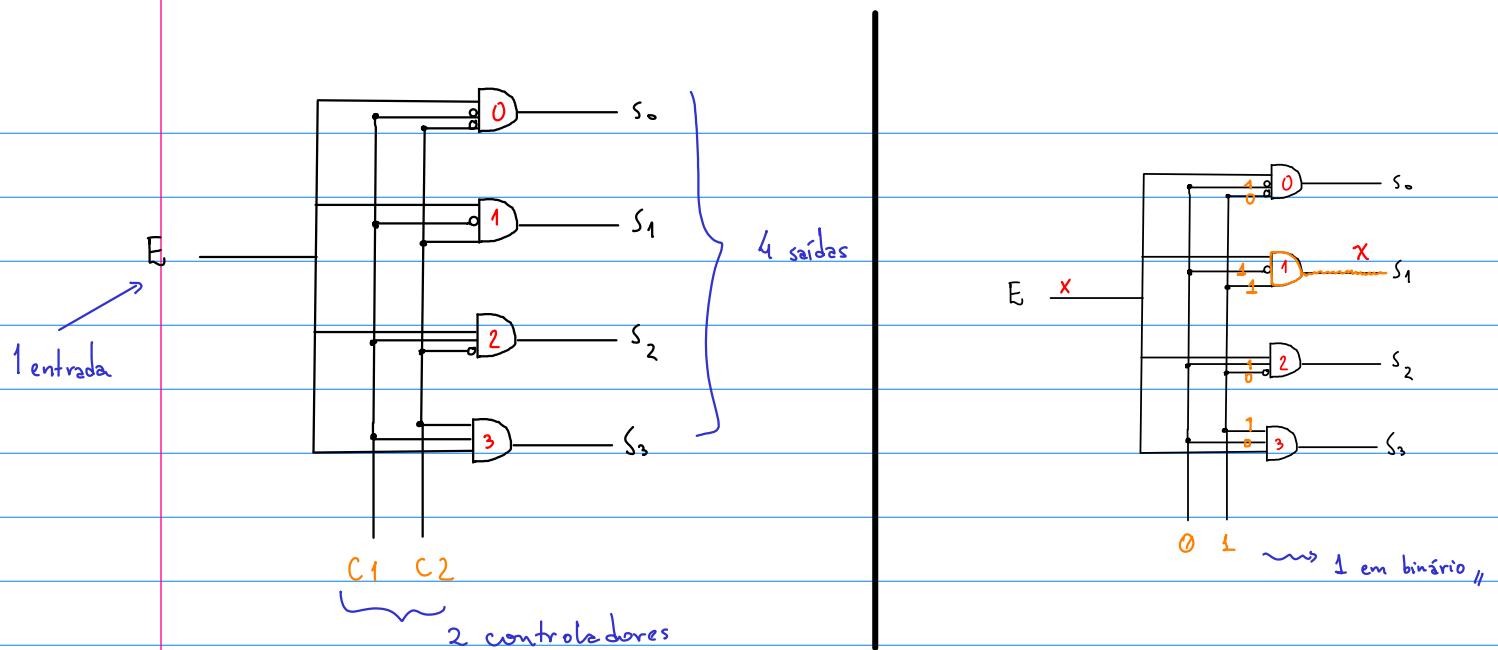
Demultiplexador - é o inverso de um multiplexador. O demultiplexador dirige sua **ÚNICA ENTRADA** até **UMA DAS 2^n SAÍDAS**, dependendo assim dos números das n linhas de controle.

* Suponha que temos **2 entradas de controle**, isto é $n = 2$

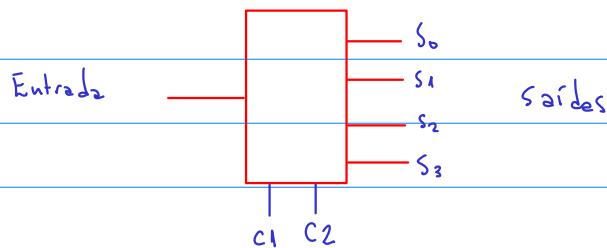
* Por consequência teremos $2^2 = 4$ saídas

* Por padrão temos **1 Entrada**

* Lembrando que: **00 01 10 11** — binário
 0 1 2 3 — decimal

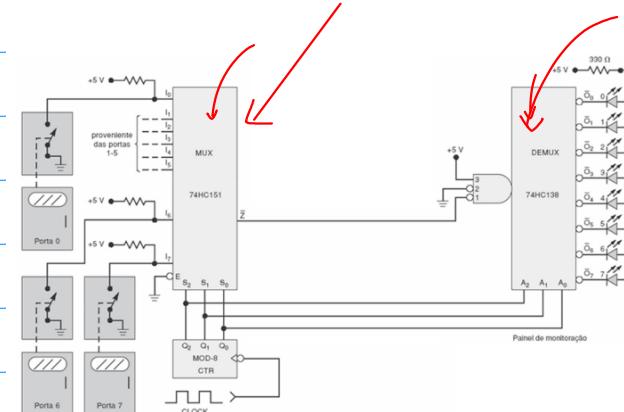


Chip de demultiplex



* Aplicações :

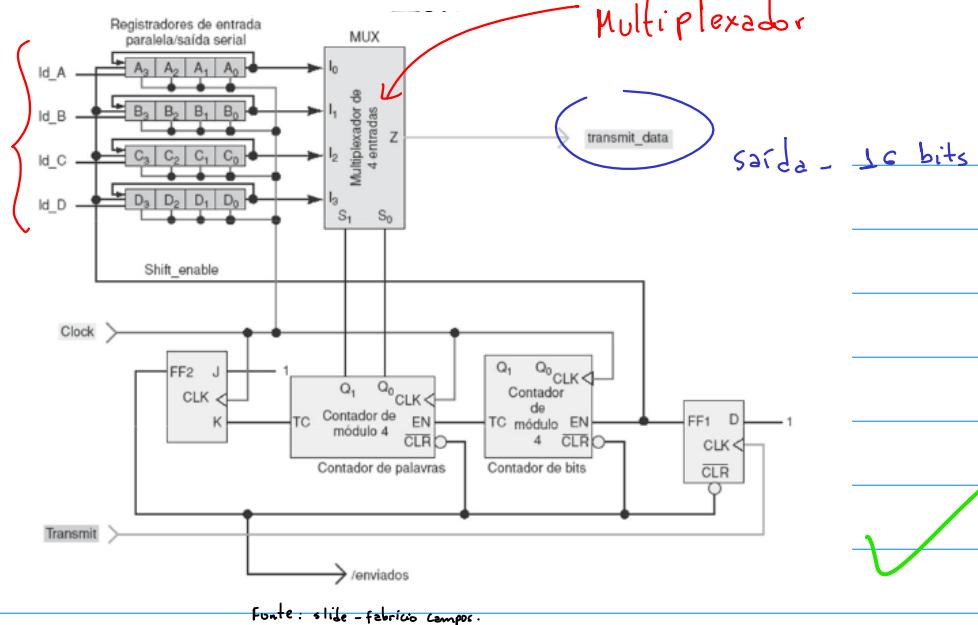
- Sistema de monitoramento de segurança - **Multiplexador** **demultiplexador.**



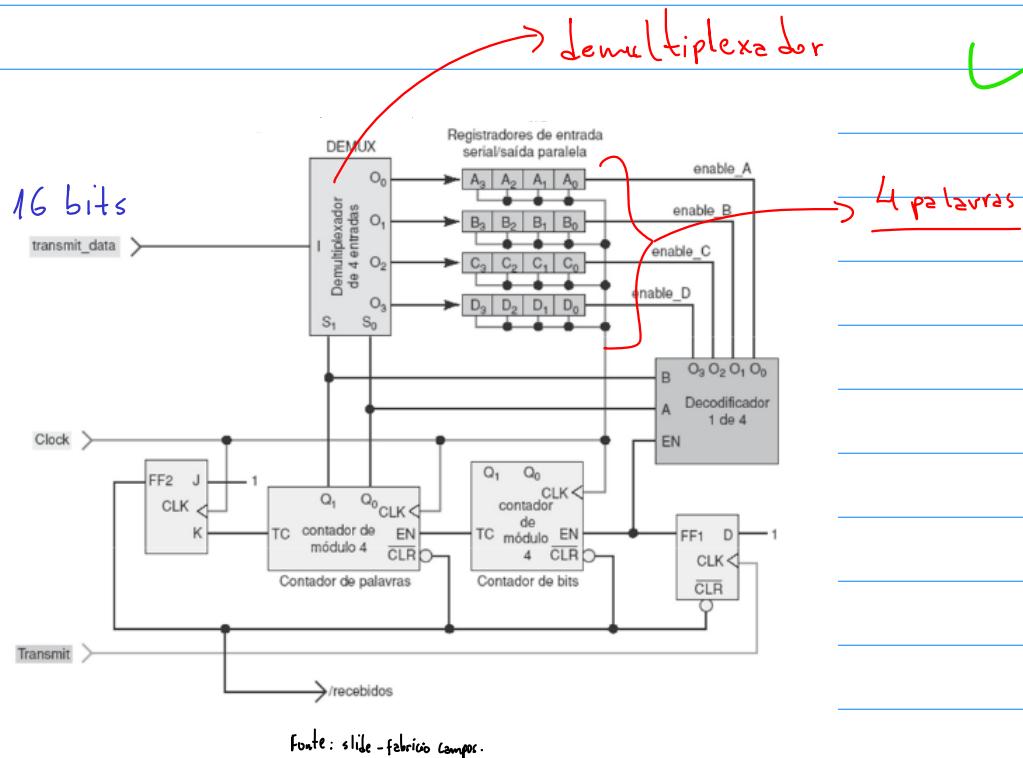
Fonte: slide - fabricio campos.

- Sistema síncrono de transmissão de dados - suponha que precisamos transmitir serialmente palavras de dados, onde cada palavra é 4 bits. Quatro palavras serão transmitidas totalizando 16 bits. Para reduzir o número de fios posso fazer uso de um multiplexador e demultiplexador entre o transmissor e receptor.

4 palavras -
cada palavra 4 bits
 $4 \cdot 4 = 16$ bits



Fonte: slide - febrero campo.



Fonte: slide - febrero campo.

b) Decodificador - é um circuito que toma um número de n bits como entrada e o usa para **SELECIONAR** (isto é, definir em 1) exatamente 2^n linhas de **SÁIDA**.

Uma grande utilização de um decodificador é no armazenamento de dados em uma memória RAM



Imagine uma pequena memória com 8 chips, cada um contendo 256 MB. O chip 0 tem endereços de 0 a 256 MB. O chip 1 tem endereços de 256 MB a 512 MB e assim por diante. Quando o endereço é apresentado à memória, os 3 bits de ordem alta são usados para selecionar um dos 8 chips. Usando a figura abaixo, esses 3 bits são as três entradas A, B, C. Assim, dependendo das entradas, exatamente UMA DAS

8 linhas de saída, D_0, D_1, \dots, D_7 , é 1; o resto é 0. Dessa forma cada linha habilita um dos 8 chips de memória. Como só uma linha de saída é colocada em 1, apenas 1 chip é habilitado.

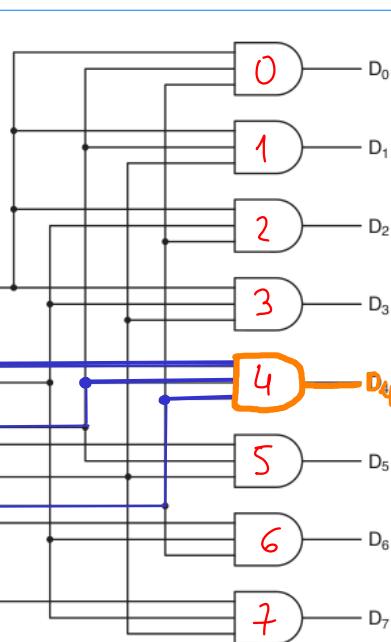
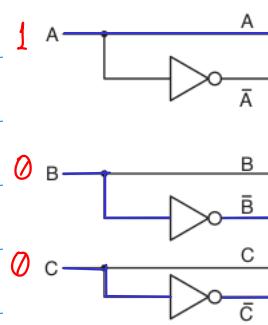
* decodificador 3 para 8

Exemplo:

3 bits de ordem alta

$\begin{smallmatrix} 1 & 0 & 0 \\ 4 & 2 & 1 \end{smallmatrix} = 4$

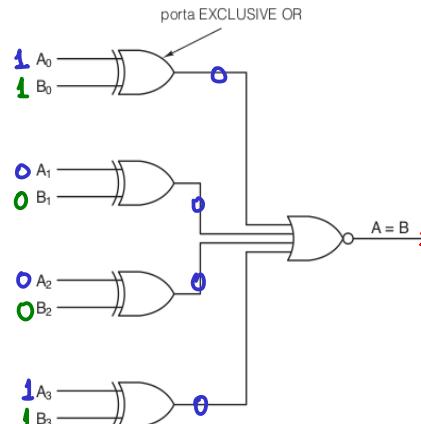
04 em decimal



• Memória RAM

Fonte: Livro-Organizações Estruturadas de Computadores.

C) Comparador - responsável por comparar duas palavras de entrada. O comparador simples mostrado na figura abaixo, pega duas entradas, A e B, cada uma de 4 bits de comprimento, e produz 1 se elas forem iguais e 0 se forem diferentes. O circuito é baseado na porta XOR, que produz 0 se suas entradas forem iguais e 1 se elas forem diferentes. Assim, se as duas palavras forem iguais, todas as quatro portas XOR devem produzir 0. Então, pode-se efetuar uma operação OR nesses quatro sinais; se o resultado for 0, as palavras de entrada são iguais; caso contrário, não. No exemplo abaixo, foi usado uma porta NOR como estágio final para revertir o sentido do teste: 1 significa igual, 0 diferente.



		XOR
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

		Or	Nor
A	B		
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Comparar os números:

9 e 9 } decimal

$1001 = 1001$

$$1001 \\ 8+1=9 \\ 9+1=10$$

d) Deslocador - é um tipo de circuito que recebe um CONJUNTO DE BITS e os desloca para a DIREITA ou ESQUERDA.

Ex₁:

0 0 0 0 1 1 1 deslocando 2 bits para direita temos 0 0 0 0 0 1 1

Ex₂:

0 0 0 0 1 1 1 deslocando 4 bits para esquerda temos 1 1 1 1 0 0 0 0

Ex₃:

1 0 0 0 1 1 1 deslocando 3 bits para direita temos 1 1 1 1 0 0 0 1

* Aplicações - utilizado em conjunto com a ULA, realizando algumas multiplicações e divisões. Exemplos:

→ Cada bit deslocado para a DIREITA, corresponde a uma divisão por 2.

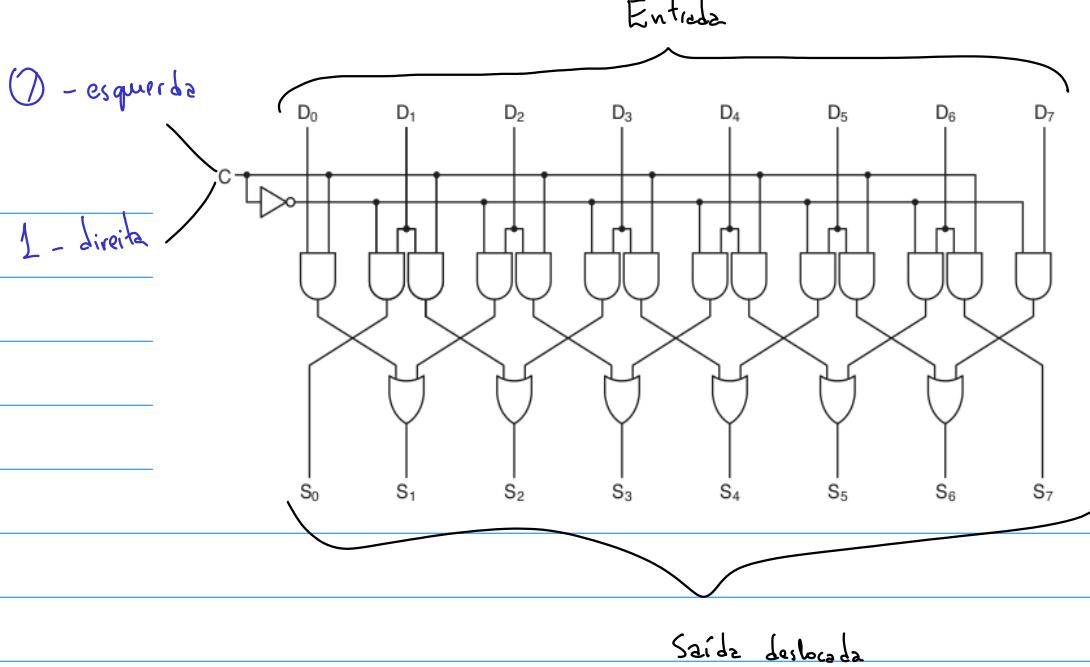
→ Cada bit deslocado para a ESQUERDA, corresponde a uma multiplicação por 2.

Ex: 0 0 0 0 0 1 0 0 = 4 , deslocando 2 bits para direita, temos 1 = 1 , corresponde em dividir por $2^2 = 4$

Ex₂: 0 0 0 1 0 0 0 0 = 16 , deslocando 3 bits para direita, temos 0 0 0 0 0 0 1 0 = 2 , corresponde em dividir por $2^3 = 8$

Ex₁: 0 0 0 0 1 0 0 0 = 8 , deslocando 2 bits para a direita, temos 0 0 1 0 0 0 0 0 = 32 , corresponde a multiplicar por $2^2 = 4$

Ex₂: 0 0 0 0 0 1 1 1 = 7 , deslocando 1 bit para a esquerda, temos 0 0 0 0 1 1 0 = 14 , corresponde a multiplicar por $2^1 = 2$.



e) Somador - circuito responsável por efetuar adição.

1º Meio - Somador

vamos tentar somar $6 + 7$, ou melhor $\overset{6}{110} + \overset{7}{111}$ em binário

$$\begin{array}{r} 1 \ 1 \\ 1 \ 1 \ 0 \\ + \ 1 \ 1 \ 1 \\ \hline 1 \ 1 \ 0 \ 1 \ \text{Soma} \end{array}$$

tabela verdade para adições de 1 bit

A	B	Soma	carry out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

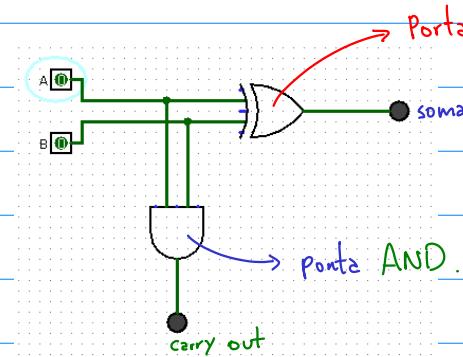
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

- Se observarmos o resultado da Soma é idêntico a saída na tabela a) $A \oplus B$ utilizando a porta XOR.

A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

- O mesmo ocorre no resultado da tabela b), que possui o $A \cdot B$ igual ao Carry out.

Portanto, para fazer o meio Somador vamos precisar:

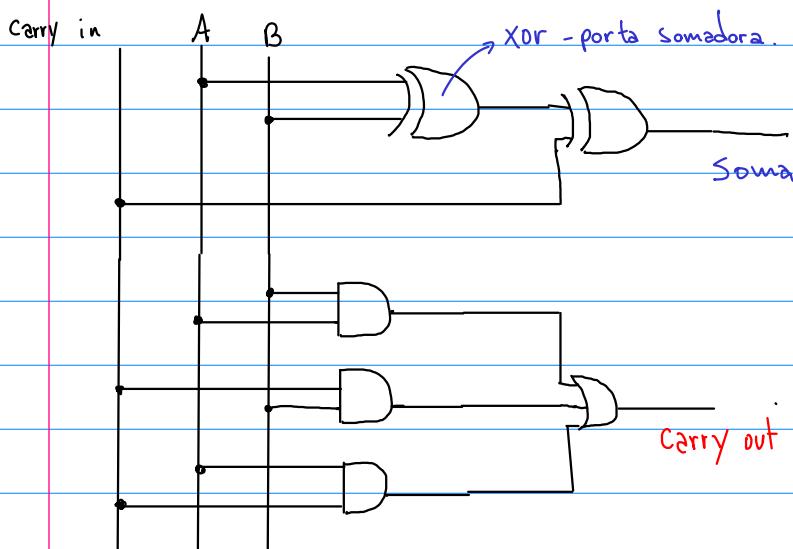
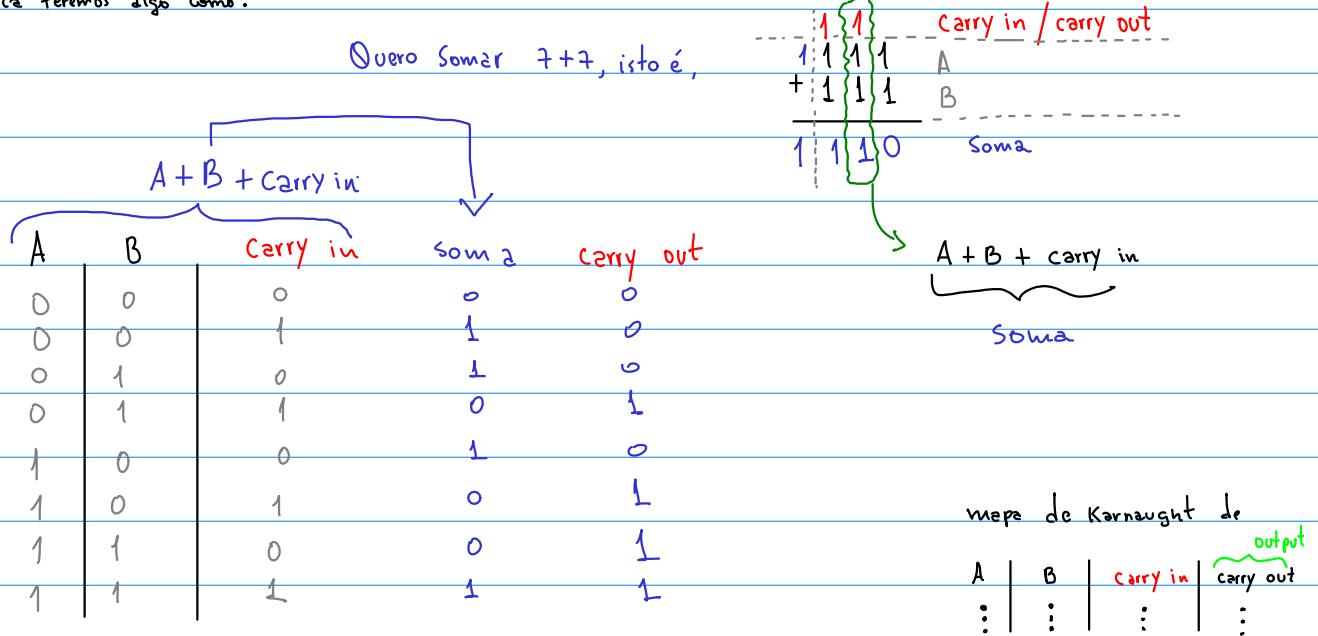


2º Somador Completo: Embora um meio somador seja adequado para somar os bits de ordem baixa, ele não é adequado para uma posição de bit no meio da palavra, pois ele NÃO TRATA o TRANSPORTE DE BIT da posição à direita (vem-um). Ex:

$$\begin{array}{r}
 \text{1:1} \\
 + \text{111} \\
 \hline
 \text{110}
 \end{array}$$

Na prática não conseguiria acessar o (Carry-in) para realizar a soma corretamente.

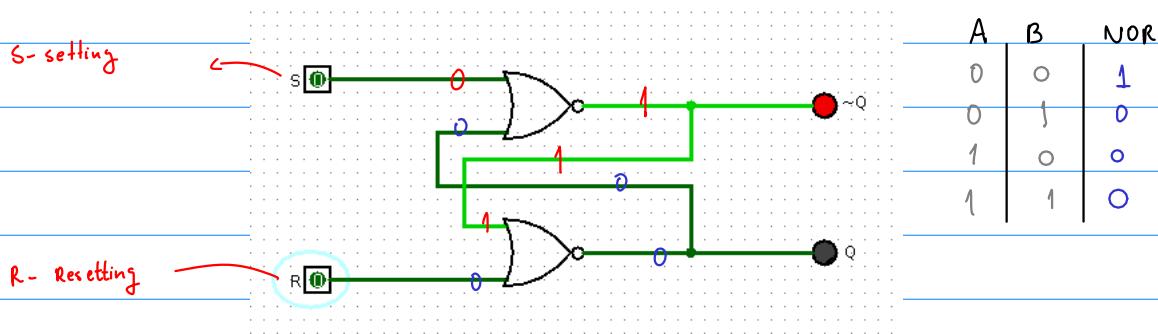
Por isso a necessidade de um somador completo. Um somador completo é a função de dois meio somadores. Na prática teremos algo como:



A	BCI			
	00	01	11	10
0	0	0	1	0
1	0	1	1	1

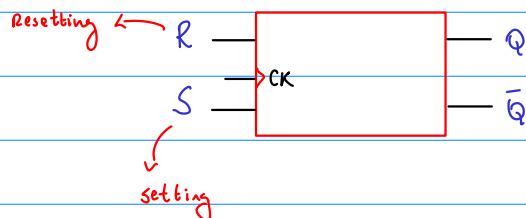
- $A = 1, B = 1, ci = 0/1 \therefore A \cdot B$
 - $A = 0/1, B = 1, ci = 1 \therefore B \cdot ci$
 - $A = 1, B = 0/1, ci = 1 \therefore A \cdot ci$
- $\therefore A \cdot B + B \cdot ci + A \cdot ci$

f) latch ou memória de 1 bit - Memória utilizada para armazenar somente 1 bit, então ela vai armazenar 1 ou 0. Para criar uma memória de 1 bit ("latch"), precisamos de um circuito que se LEMBRE, de algum modo de VALORES de entrada anteriores. Um circuito que faça isso pode ser construído da seguinte forma:



Por mais simples que pareça, pelo fato de armazenar somente 1 único bit, o simples fato do circuito conseguir lembrar de entradas anteriores torna essa memória poderosa e base para as memórias atuais.

g) Flip-flops: o flip flop é parecido com uma memória, sendo responsável por ler o valor em determinada linha em todo instante, e armazená-lo.



h) Registradores - conjunto de flip-flops combinados em grupos, que vai ser responsável por manter dados com comprimentos maiores do que 1 bit. Os Registradores estão no topo da hierarquia de memória, sendo uma memória de acesso bem rápido e financeiramente custosa. No entanto, apesar do alto custo por bit armazenado, sua velocidade de acesso é essencial para os computadores.