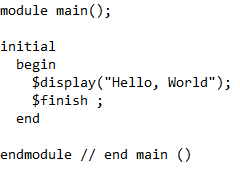
Aula dia 03/08/2022

* Programação em Verilog:
  + Module main() -> função main.
  + $ = marca da linguagem
  + $display = printf
  + Linguagem **paralela**, porém, dentro de um bloco é sequencial, possibilidade de escolha
    1. Paralela àTudo executado ao mesmo tempo
    2. Sequencial àTudo executado em uma sequencia
  + Always àrepetição infinita com condição
  + Mais de um tipo de atribuição: instantânea e permanente
  + Exemplo de programa:
  + Para compilar : iverilog -o hello.vvp hello.v
  + Para executar: vvp hello.vvp
  + System verilog é uma adaptação da linguagem que tem orientação por objetos
  + Mais informações na apostila Verilog.pdf

Estudo Individual 04/08/2022

**Guia01**

* Representação de dados
  + Função continua em um intervalo: Analogico
  + Função discreta em um intervalo : Digital
* Sistemas de Numeração
  + Conversoes entre bases:
    1. Decimal p Binario

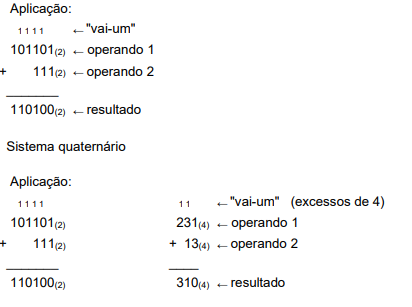
Divisoes sucessivas por dois , e tomar restos na ordem inversa.

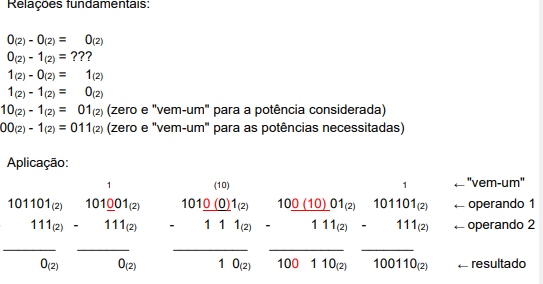
OBS : manter o mesmo processo para conversão para outras bases

* + 1. Decimal para hexadecimal

Mesmo processo, porem se resto >= 10, usa-se A , 11 = B, 12 = C ...

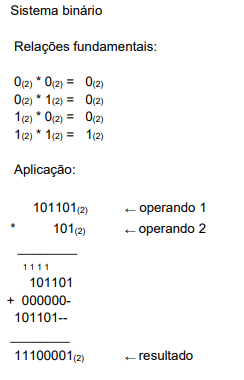
* Sistemas de numeração – Operações Aritméticas
  1. Adicao à Regra vai um no excesso da base

EX:

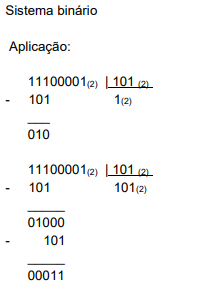
* 1. Subtração

OBS: Vai um no meio.

0-1 = 1 e vai um

* 1. Multiplicação

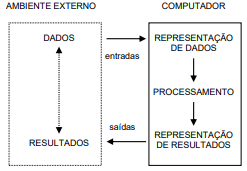
OBS: mesmo algoritmo padrão.

* 1. Divisão

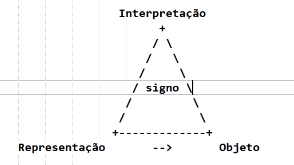
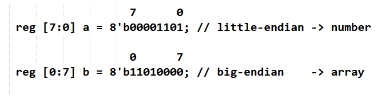
Representação de dados em java:

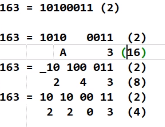
* Tipos:
  1. Boolean
  2. Byte
  3. Char
  4. Short
  5. Int
  6. Long
  7. Float
  8. Double
  9. String
* Representação binaria dependendo do número de bits
  + - Para os valores inteiros, por exemplo, pode-se utilizar o formato em que o primeiro bit, à esquerda, para o sinal e o restante para a amplitude, responsável pela magnitude (grandeza) do valor representado. Exemplo: 5(10) = 101(2) +5(10) = 0101(2) - 5(10) = 1101(2)
    - **Complemento de 1** Uma das possíveis representações para valores negativos pode ser aquela onde se invertem os valores individuais de cada bit.
    - **Complemento de 2** Outra das possíveis representações para valores negativos pode ser aquela onde se invertem os valores individuais de cada bit, e acrescenta-se mais uma unidade ao valor encontrado, buscando completar o que falta para atingir a próxima potència da base.

**Capitulo I – conceitos básicos**

* Tipos de computadores:
  + Digital à representam informações através de grandezas discretas, maior precisão e armazenamento
  + Analógicos à representam informações através de grandezas continuas, maior velocidade
  + Hibridosàreunem características dos dois tipos acima.
  + OBS: A tecnologia analógica é o processo de receber um sinal de áudio ou vídeo e traduzi-lo em pulsos eletrônicos. Já a tecnologia digital, por outro lado, converte esse mesmo sinal em um formato binário, no qual os dados de áudio e de vídeo são convertidos em uma série de zeros e uns.
* Sistema computacional
  + Hardware à conjunto de componentes eletrônicos com os quais são construídos os computadores.
  + Softwareàconjunto de programas e sua documentação.
* Processamento de dados
* Modelo para processamento de dados:

Aula dia 05/08/2022(Online)

* Programação em verilog:
  + Reg [7:0] à register é um tipo de dado, parecendo um array que vai de 7 a 0.
  + %b à bit -- %8b = 8 bit.(binário)
  + %o àoctal
  + %h à hexadecimal
  + Bi-ending, ou seja, aceita as duas representações.
* Modelos e Representação de dados
  + Existem objetos abstratos, que representam um objeto.
  + Nos rotulamos objetos com representação, que necessita da interpetação
* Unicode = Sistema que usa 16 bits ou 2 bytes.
* Big endian à maior potencia está no fim(array)
* /0 encerra a representação
* Little endian àmenor potencia está no fim(number)
* Conversão de bases múltiplas



Aula dia 10/08/2022

**Representação de dados**

1. Números com sinal
   * 13 à 1101(base 2)
   * +13à01101 ?
   * -13à10011 ? à complemento
   * OBS:
     + Representação em sinal em amplitude ( 0 = + e 1 = -) à Da ruim para fazer as operações
     + Vai 1 depois do sinal é desprezado.
     + Sistema decimal é um sistema aberto.
     + Sistema do relógio é um sistema fechado(negativo de 13 horas não é o próprio número)
     + X + Y = Z – 01101(13) + y = 00000 – Y = 10011(19) ou seja, 32 que é o maior número em binário com 5 algoritmos.
     + Método para descobrir rápido: Inverter o número e somar 1
     + 01101 + 10010 = C1 na base 2 com 5 algoritmos
     + 01101+10011 = C2 na base 2 com 5 algoritmos – **ou seja, negativo vai ser o complemento de 2** **pra base 2**
     + **Cn(x) = Cn-1(x) + 1 – n>= 2**
2. Número fracionário negativo
   * 27,25 = 11011,01 --- 011011,01 à 100100,11 – mesma coisa sem considerar a virgula e depois volta com ela no mesmo lugar.

Aula dia 12/08/2022

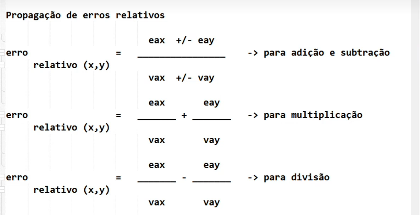
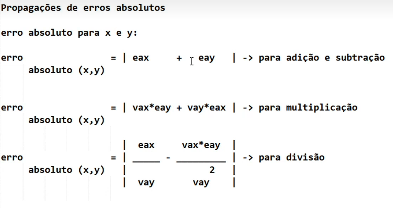
* Padding – preenchimento com zeros na direita ou esquerda.
* Operações aritméticas com congruência modular
  + Relógio
  + Trigonometria
* Problema da representação de C1 é a necessidade de concertar os valores, quando o zero é representado duas vezes (+ 0 e – 0).
* Processador àcircuitos pequenos fáceis de reproduzir

Aula dia 17/08/2022

* Problema do ponto flutuante
  + É um problema causado quando a representação interna de números de ponto flutuante, que usa um numero fixo de digito binário para representar um número decimal
  + Representação mínima à 13,25 = 1101,01 – em NC – 1,325 \* **10^1** – 1,10101 \* **2^3**(potência necessária) – necessidade de 2 bytes.
    - OBS: LEMBRAR DO BIT PARA O SINAL
    - Solução do IEEE 754: Tirar o 1 depois da virgula para economizar 1 bit. – Economia de 1/2^23
      * Expoente será o excesso de 127 = 3 + 127 = 130 – número de corte

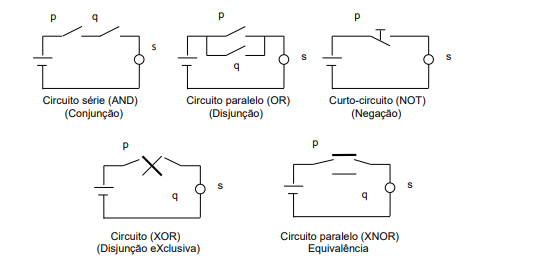
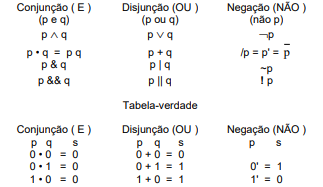
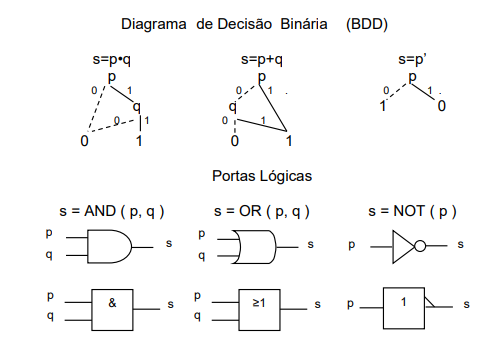
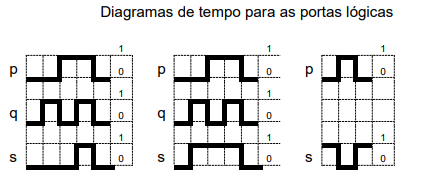
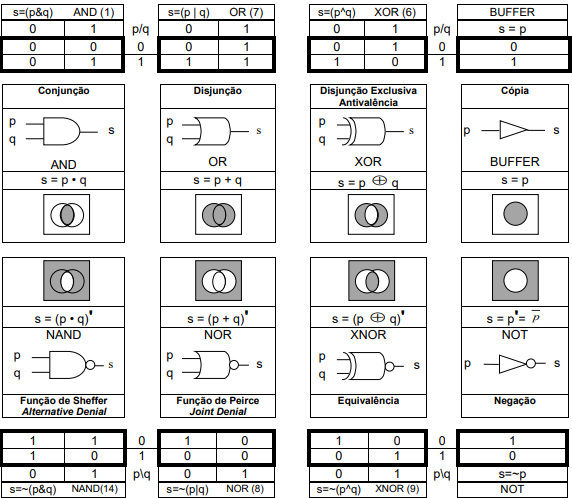
Aula 19/08/2022

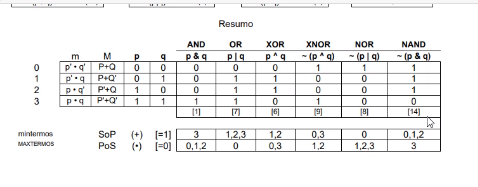
* Erro absoluto = valor real – valor representado.
* Erro relativo = erro absoluto / valor real
* Não se pode desconsiderar os erros.
* Propagação dos erros:



* Truncamento e arredondamento
  + Precisão razoável = 80bits
  + Truncamento = ignorar alguns bits
  + Arredondamento = arredondar uma casa decimal

Aula dia 24/08/2022

* Já na Revolução Industrial existia algo parecido com as portas lógicas
* Representação por circuitos:
  + Sempre irá ocorrer perda nos circuitos, mesmo que minimizado existe
  + Atuar sobre a chave: fechar e abrir circuito , porém no NOT é ao contrario, pois se atuar irá mudar o valor porém funciona sem
  + Disjunção exclusiva: contrário da disjunção – diferença(XOR)apenas diferentes serão verdadeiros
  + Equivalência: igualdade (XNOR) apenas iguais serão verdadeiro
  + Representações:
    - X = valor do meio – o qual não sabemos o que está acontecendo
    - Z = valor de mudança – o qual está quase mudando de valor(pronto para reagir)
    - Níveis da saída tem que ser mais precisas.
* Buffer = compensa perda – contrario da NOT
* NAND NOR XNOR NOT(representação pela bolinha) – contrario de AND OR XOR BUFFER
* O que tem dentro de uma porta logica?
  + Entrada/Saida
  + Gate(botão) – transistor
  + Porta de transmissão(catraca)
* Clock é necessário para funcionamento do circuito ( como se fosse o botão da câmera)

Aula dia 26/08/2022

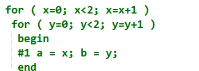
* A soma de inversos sempre dará 15, ou seja 16-1 – pois tem 3 algarismos
* SoP = 1(mintermos) + -- priorizando disjunção -- Sum of products
* PoS = 0(maxtermos) – priorizando conjunção – Products of sum
* Dualidade de comportamento – OR e NOR.

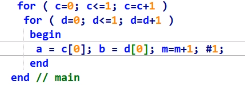
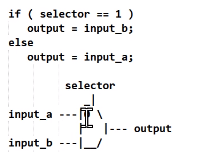
Aula dia 31/08/2022

* Relação De Morgan:
  + Inverte todos os sinais e relações
* Toda vez que a carga passa pela porta logica é necessário de um tempo para ele passar. Os tempos das portas não são iguais.
* Programacao em Verilog:
  + $monitor – será executado toda vez que o valor mudar.

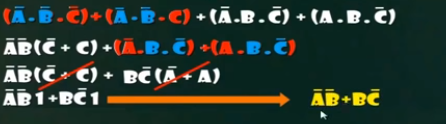
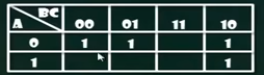
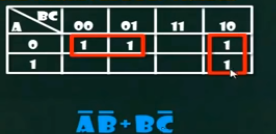
Aula dia 02/09/2022

* É muito caro usar for em verilog para mudar apenas um bit em um ineitro, pois cada for usa-se 32 bits:

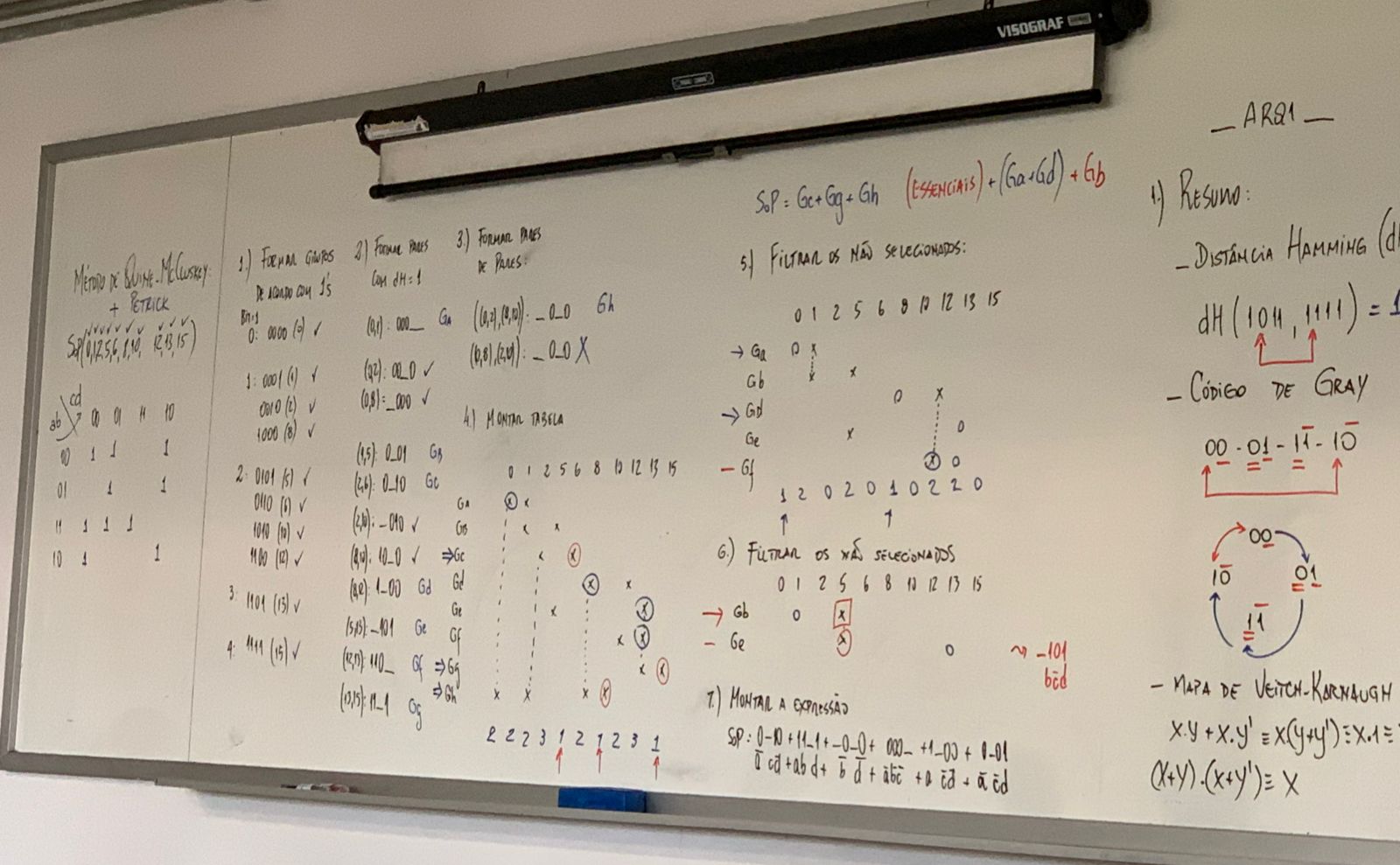


* É melhor usar variável de apenas 2 bits – usando o reg:
* Portas nand e nor usam menos transistores que and ou or, ou seja, é mais benéfico usar as primeiras para desenvolver circuitos.
* Circuitos uteis:
  + Multiplexador(MUX) – IF/ELSE
  + 

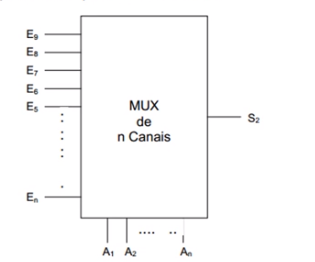
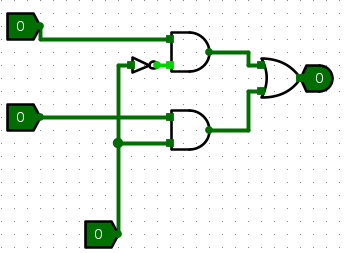
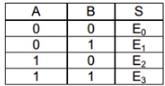
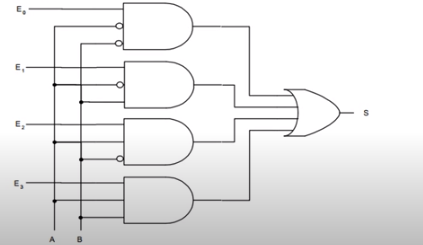
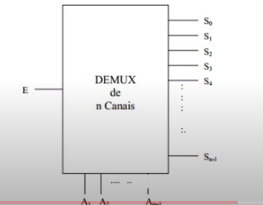
Estudo Individual 08/09/2022 –Mapa de Karnaugh(Mapa K)

* Mapa que serve para acharmos expressão booleana otimizado de forma mais fácil
* Primeiro devemos simplificar a expressão:
* Mapa K é construído a partir da tabela verdade
* Não podemos fazer – chama atenção para o número de posições em que duas ou mais combinações diferem entre si(não pode ter mais de uma diferença – sempre a menor distancia)
* Depois colocamos os resultados dos 1:
* 1 tecnica – Agrupamento de dois quadros:
  + Pegar combinações de 1s
  + Entrada útil(não se altera) ou não-util –
  + EX:

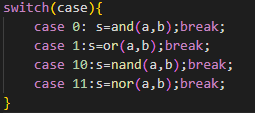
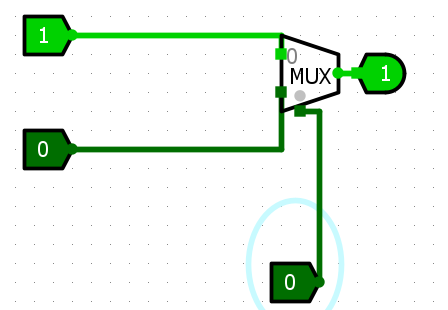
Aula dia 14/09/2022

* Distancia de Hamming(DH)
  + Aplicadas sob sequencias de bits
  + 1011,1111 à 1
* Código de gray
  + Sequencia de valores não é mais representação binaria
  + Um bit so de diferença ( 00 01 11 10)
* Mapa de karnaugh – simplificação de x.y + x.’y à problema de vizualicao com muitas variáveis
* Metodo de Quime McCluskey:
  + Formar grupos de grupos de 1’s
  + Se não combinar tudo – é necessário guarda-lo
  + EX(FOTO)
  + SOP = Gc+Gg+Gh (essenciais) primos implicantes

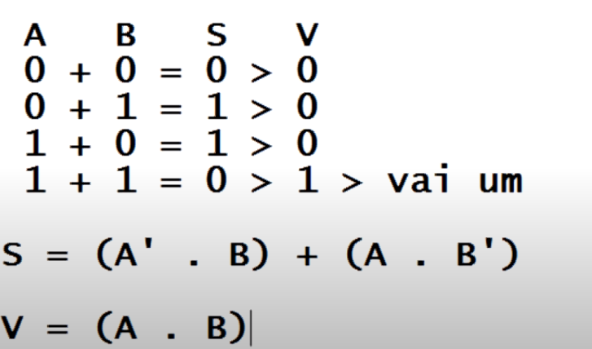
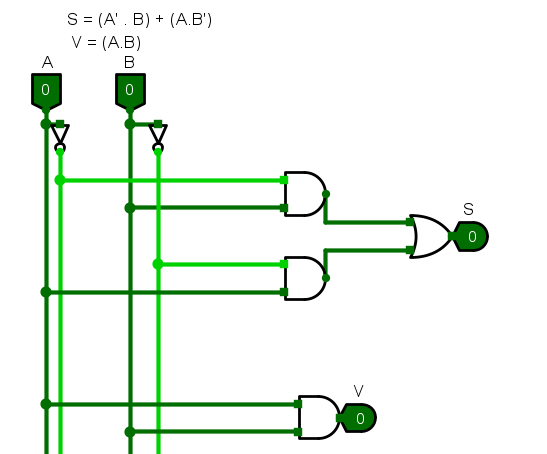
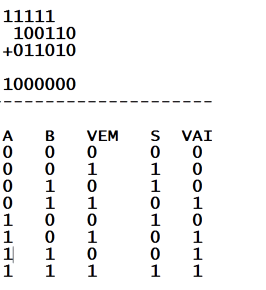
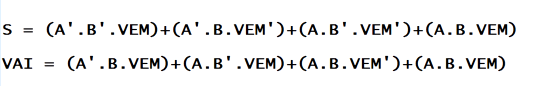
Estudo Individual 14/09/2022 Circuitos Combinatórios-multiplexador

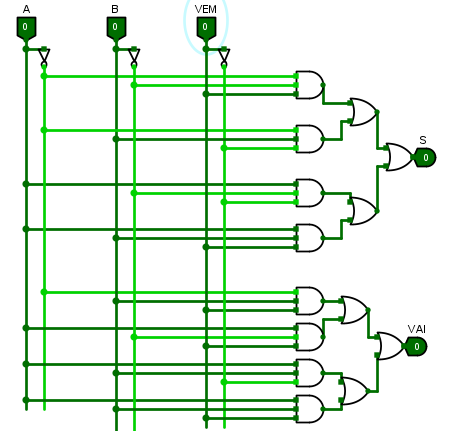
* Multiplexação:
  + Consiste na operação de transmitir varias comunicações diferentes através de um único canal físico
  + O dispositivo que realiza esse tipo de operação chama-se multiplexador
  + Aplicaçoes: telefonia fixa, celular e tv
  + Número de canais à n = 2^m
    - Nànumero de canais de entrada
    - Mànúmero de variáveis de seleção
  + MUX de 2 canais:
    - 
    - S= ‘A.E0 + A.E1
    - Circuito lógico:
    - A = variável de seleção
  + MUX de 4 canais:
    - A e B = variáveis de seleção
    - ‘A.’B.E0 + ‘A.B.E1 + A.’B.E2 + A.B.E3
* Demultiplexão:
  + Selecionar através das variáveis de seleção, qual de suas saídas deve receber a informação de uma única entrada
  + Número de canais à n = 2^m
  + S0 = E.’A
  + S1 = E.A

21/09/2022

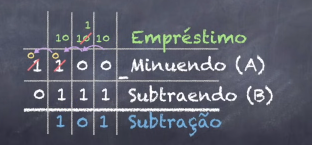
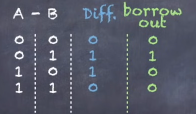
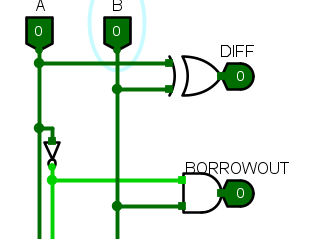
* Multiplexador(MUX) àselecionar dentre as entradas um valor que vai para saída -- switch/case
* Unidade aritmética
  + Somadores e Subtradores

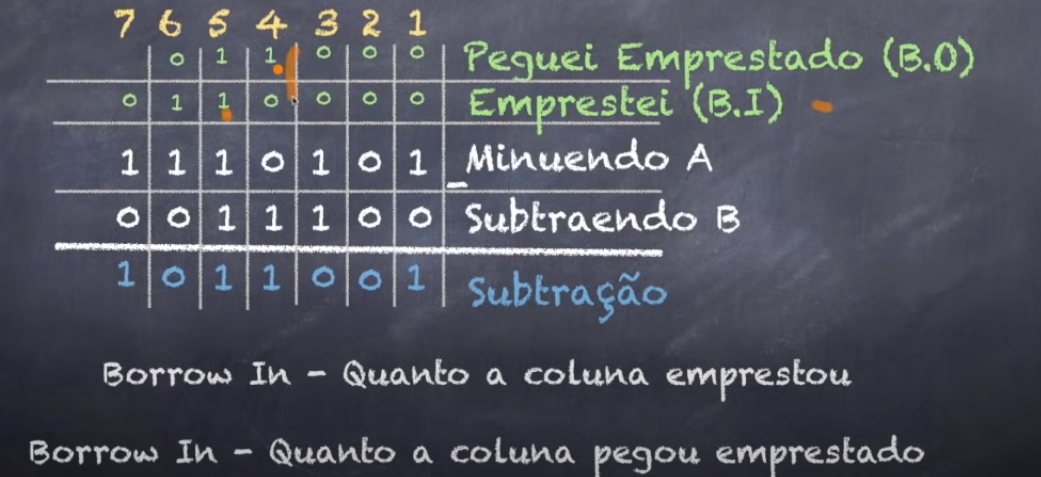
Estudo Individual 22/09/2022 – Somadores e Subtradores

* O circuito é o responsável por uma soma binária de dois dígitos binários (bits), e é chamado de circuito de meia-soma. (half-adder).
* Para se operar três ou mais bits, vários desses serão combinados em cascata para formar um circuito de soma-completa.
* Somador de 2 bit(MEIO SOMADOR):
* Somador de 4 bits (FULL ADDER)

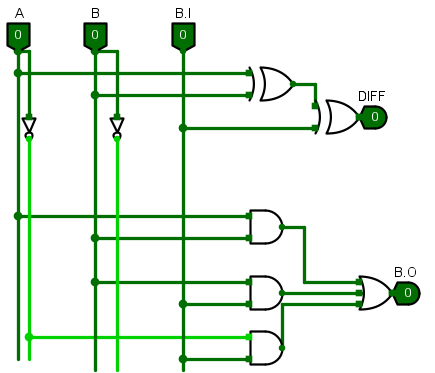
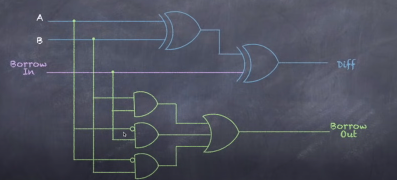
SUBTRADORES:

Meio subtrador:(2entradas e 2 saidas)

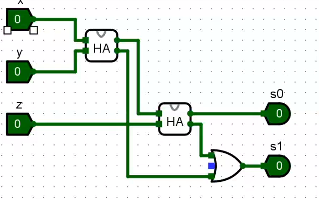
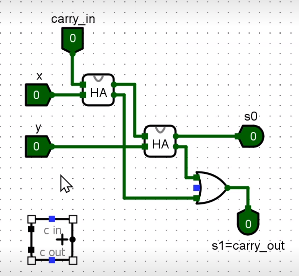
* Lembrando:
* Borrow out = AND(A’ . B)
* Diffe = XOR(A XOR B)
* Circuito: \

Resolvendo o problema:

* DIFF = (XOR(A XOR B) XOR BI)

Subtrador completo:

Aula do dia 23/09/2022

* Sinal só pode ser usado em uma das meias somas(OU)
* EX:
* Full adder = tres entradas (carry\_in x y) duas meias somas + ou + duas saidas:
* Or excluivo pode ser usado par diferença

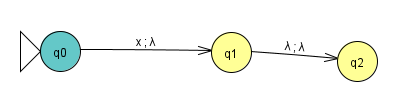
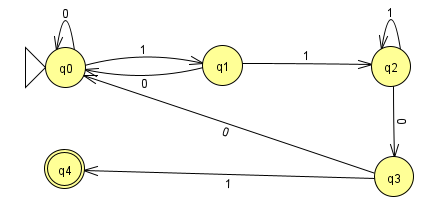
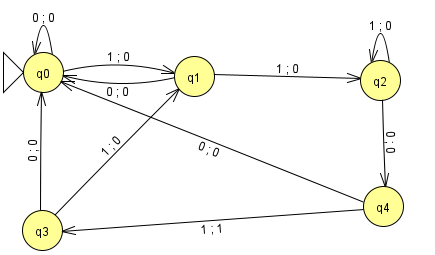
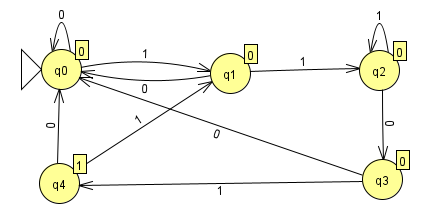
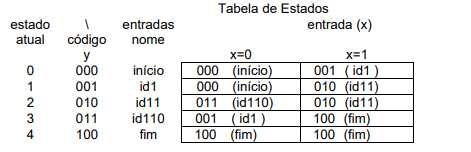
Aula 28/09/2022

* ULA
  + Logicas
  + Aritimeticas
  + Relacionais
  + Flags àBero, Crazy, Overflow, Party, Minus, Plus
  + Deslocamento
* Revisão comparadores:
  + Comparacao de dois bits + AND:
  + Or exclusivo àdiferença
  + XOR àigualdade

Estudo Individual 04/10/2022

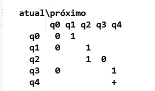
* Circuito combinacionalàsaida depende de uma combinação de entradas
* Circuito sequencialàsaida depende das entradas e de outras variaveis, como por exemplo o tempo.
  + Sistema deve ter memoria para passar ao proximo estado
  + Dois tipos:
    - Assincronos – estados podem mudar a qualquer instante \
    - Sincronos – estados mudam em instantes determinados
  + Se ocorrerem as transições ocorrerem durante uma variação de 0 para 1 ( ↑ - borda de subida), o sistema será dito de **nível alto**; caso contrário, durante uma variação de 1 para 0 ( ↓ - borda de descida), o sistema será dito de **nível baixo**.
* Mquinas de estados finitos(Finite State Machines)
  + É um modelo de comportamento composots de estados, transicoes e acoes.
  + **Estado**àaramazena informacoes sobre a historia do sistema(reflete como mudancas nas entradas trouxeram o sistema até o estado atual)
  + **Transicao**àindica uma mudança de estado e é descrita por uma condicao que a permite
  + **Ação**àdescriacao de uma atividade externa executada em um certo instante
  + Podem ser usadas para descrever cicuitos sequenciais
  + Os modelos de Mealy e Moore são usados para descrever maquinas de estados finitos
    - Saida em funcao do estado e entradas à Mealy, pois mais eficiencia a variacoes das entradas
    - Saida em funcao apenas do estado àMoore, pois garante a transição completa entre estados

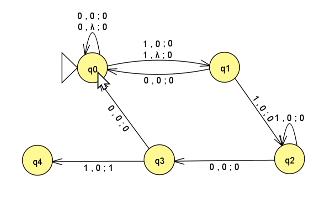
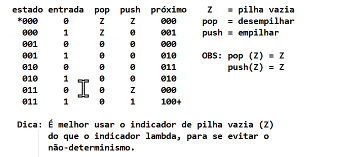
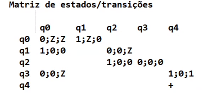
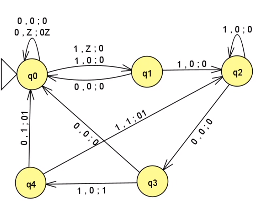
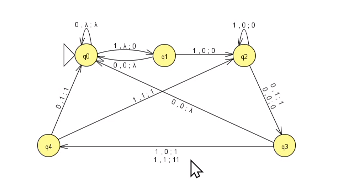
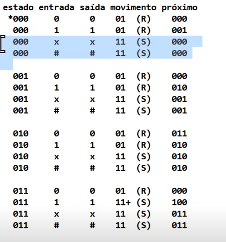
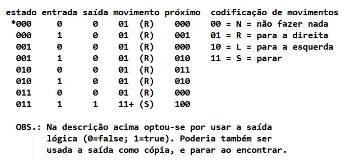
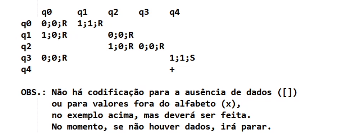
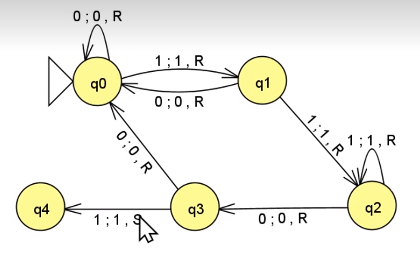
Aula dia 05/10/2022

* Para entender o controlador de tempo é necessario voltar as maquinas mais primitivas, desta forma, torna-se de extrema importancia compreende-las
* Maquina de Estados Finitos(Automatomo Finito):
  + Tipos:
    - Deterministicos(DFA) – convergem em um resultado – prever TODAS saídas
    - Não-deterministicos(NFA) – divergencia
    - Uma distinção adicional está entre autômato **determinístico** ([AFD](https://pt.wikipedia.org/wiki/Aut%C3%B4matos_finitos_determin%C3%ADsticos)) e **não-determinístico** ([AFN](https://pt.wikipedia.org/wiki/Aut%C3%B4matos_finitos_n%C3%A3o_determin%C3%ADsticos)). No autômato determinístico, para cada estado há exatamente uma transição para cada entrada possível. No autômato não determinístico, pode haver nenhuma, uma ou mais de uma transição de um determinado estado para uma entrada possível.
  + Categorias:
    - Detectores/reconhecedores
    - Transdudores: Manly e Moore
    - Classificadores
    - Sequenciadores/Geradores
  + Definicao:
    - Grande conjunto de alternativas
    - Compostas por estados, armazenando mémoria durante sua execuçao
    - Q0 = Estado Inicial
    - Q2 = Estado Final
    - É possivel ficar no mesmo Estado , voltar para o anterior ou avançar para o prximo
    - Entrada seriada = 1 BIT
    - Difença para outras mquinas: **reconhecimento do Estado Final**, ou seja, para se der erro ou chegar ao final
  + Exemplo de recohecedor de 1101:
  + Maquina de Mearly:
    - Não teremos estado final
  + Maquina de Moore:
    - Ticket à precisa concluir a transicao para dar a resposta , diferentemente do Mearly que na saída já te da resposta
  + Tabelas de Transição:
    - A partir do diagrama é possivel fazer a tabela e vice-versa
  + Pilha:
    - Memória de dados e não estado
    - Diferença que eremos um item a mais
  + Maquina de Turing:
    - Pode andar com a fita para esquerda ou direita
    - Anda sobre o programa e sobre os dados

Aula do dia 07/10/2022

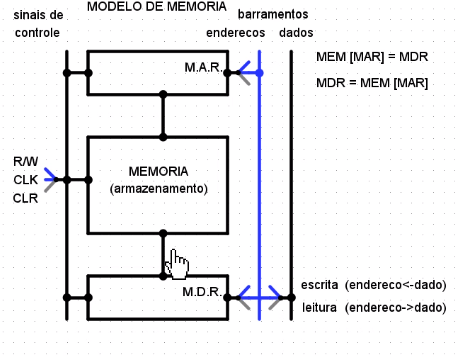
* É possivel representar as mudancas de estados por meio de metriz. EX:

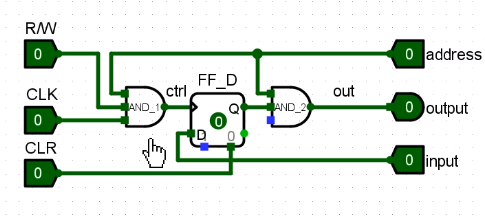


* Automato de Pilha:
  + Precisaremos consultar o topo da pilha, retirando ele.
  + Z ou lampida representa pilha vazia
  + 0,0,0 – Entrada / Pilha Vazia / Saida para o topo
  + Q4 é o estado final
  + Tabela:
  + Matriz de Estados: -- matriz esparsa, bem espalhadas
  + Identificacao instatanea de uma sequencia, o que não é muito interresante
  + Entretanto, existe a possibilidade de contar usando a pilha:
  + Existe a possibilidade de usar a pilha como memoria auxilar tambem:
* Maquina de Turing
  + Maquina bem primitiva
  + Tabela de estados: à prexisamos considerar simbolo invalido(X) e fim da fita(#)
  + Fita estavel e cabeça muda
  + Matriz de estados:
  + Simulação:
* Maquinas de Estado finito em VERILOG:
  + Reset à indicar entrada inicial

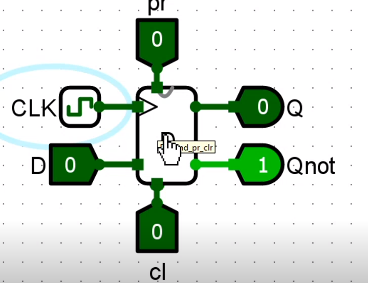
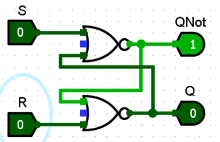
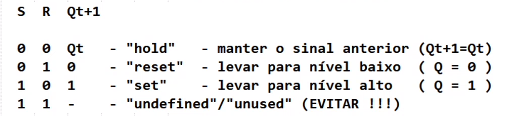
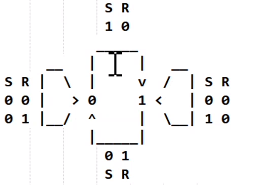
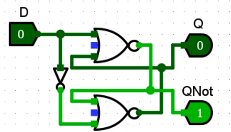
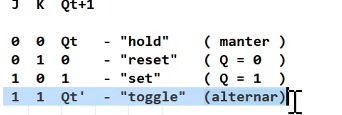
Aula dia 14/10/2022

Modelo Simpliflicado do Computador:

* Barramentoàvia preferencial/via de controle
  + Endereco
  + Controle
  + Dados à datapath
* Memoria:
  + ROM – Read Only Memory
  + RAM – Randaom Acess Memory à pode ser acessado qualuqer posicao endereçavel
    - Espaço de armazenamento se comportará como se fosse um arranjo
    - MAR e MDR – registradores
    - Controle – READ/WRITE , CLOCK e CLEAR



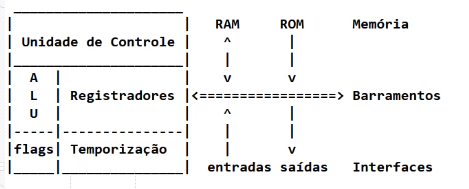
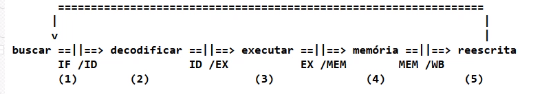
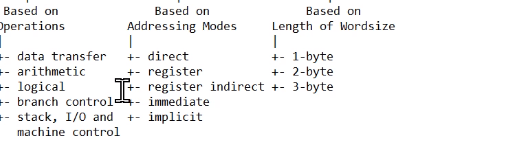
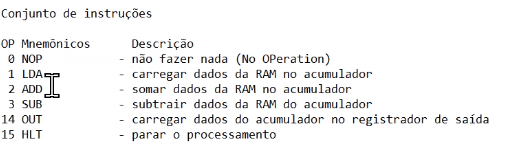
Aula dia 21/10/2022

* Circuitos para armazenar 1 bit – circuitos sequências 🡪 célula
* Flip-flops:
  + Parte das saidas irao voltar e servir como entradas(realimentacao)
  + A realimentacao pode servir como **memoria** ou como **elementos** de **correcao** ou **avanco.**
  + A realimentação esta presente até hoje, muito presente em AI
  + É necessario um certo tempo para que a saída seja produzida, e modificacao de estado – Não é instataneamente
  + Para corrigir o problema da inicializacao APENAS por clock usamos o preset e o clear
  + Flip-flop tipo D – para dados:
* Latches:
  + Não tem clock , **diferentemente do FLIP FLOP**
  + Set-Reset(SR)
    - Mesma ideia porém não precisamos do clock para inicializar
    - Pode-se fazer o circuito tambem com nand, porem ira ter uma diferenca quando S e R = 1 🡪 correcao porta NOT
    - Tabela:
    - Colocando no mapa de Karnaugh teremos 111 como indefinido
    - Equacao caracteristica: Q = S + ‘R . Q ( S.R = 0)
    - Se usa a idefinicao para chegar na equacao, porém não é interresante deixa-lo
    - Diagrama de estudos:
  + Na **celula D** so teremos a situacao de set e reset:
    - **Simplificacao** da anterior
    - **Saida depende da entrada**
    - Q = D
  + Latche tipo T (toogle = troca)
    - Se t = 0 manter estado anterior
    - Se t = 1 ai irá trocar sempre será o anterior negado
    - Q = ‘T . QT + T . ‘QT – T XOR QT
  + Latche tipo JK(Jump-Keep)
    - Juncao de todos os outros – Mais completo de todos
    - Tabela:
    - Q = J.’QT + ‘K . QT – quase um ou excluisvo

Aula dia 26/10/2022

* Contadores:
  + Sincrona – contar toda vez que clock mudar
  + Assincrona – contar a cada evento – não estao todos no mesmo clock

Aula dia 11/11/2022

* Modelo de Arquitetura de Computadores:
  + Barramentos nos permitem a comunicar com memoria e interface
  + Flags 🡪 codigos de decisoes
* Limites para processamentos:
  + Procssos submetidos a limitacoes devido a entrada/saida
    - Memoria
    - Tempo
  + O sistema operacional é importante para o controle do tempo entre as execucoes
* Evolucao da ideia de melhor uso dos recursos:
  + Execucao de instrução separada por estágios- pipeline
    - Cada parte é separada da outra. Ex: um faz a porta , o outro a janela , o ouro o teto , até que a casa esteja pronta
    - é o processo que divide tarefas sequenciais em estágios distintos que podem ser executados no modelo de linha estruturada.
* Microprocessador - 8085:
  + Classificação das instruçoes:
  + Registradores:
    - Controle de Interrupcao
    - Controle de I/O
    - Temporização e conrole
    - Unidade Lógico-Aritmética
    - Acumulador
    - TEMP
    - Flip-flops dos flags
  + Endereçamento direto:
    - O endereço da mémoria tem que ser dito no cógido
    - EX:
  + Endereçamento implicito:
    - CMA,RAL,RAR,DAA
  + Conjuntos de intrucoes:

Aula dia 16/11/2022

* Avaliacao 2 dia 23/11 : Circuitos combinatorios e sequencias
* Modelos de arquitetura 🡪 categorias de maquinas:
  + Acumulador(8085)
  + Pilha
  + Memoria-memoria
  + Registrador-memoria
  + Registrador-Registrador(MIPS)
* É necessario saber a organanizacao da memoria para possiblitar operacoes – saber enderecos
* Acumulador:
  + Existem varios modos de endereçamento(OLHAR EM CIMA )
  + 1 endereço
  + Busca e voltar para memória 🡪 processo lento
* Pilha:
  + Item de memoria que para ser manuseada precisa do endereço.
  + 0 endereço
  + Gasta tempo para ir na memoria , porém é mais eficiente que o acumulador
* Memoria memoria:
  + Operacoes são feitas diretamente na memoria , sem ter que passar por umas unidade aritmedica
  + 2 enderecos + 3 enderecos
  + Não precisa mover os dados por meio dos barramentos
* Registrador-memoria
  + 2 enderecos
  + Consiguimos ter mais liberdade quanto ao lugar de armazenamento
* Registrador – registrador:
  + MIPS
  + Mais moderno
  + 3 endereços
* Dependendo do conjunto de instrucoes, o processador pode divir em varias fases como visto no pipeline 🡪 **modelo de execução**
* Hirarquia de Memoria:
  + Quanto menor mais tecnologico e rapido será , além disso, maior custo
  + Registradores 🡪 bem pequeno porém de alta velocidade(UIA)
* Modo de endereçamento:
* Implícito ( HALT, \_\_, \_\_, \_\_ ) (operando dispensável)
* Imediato ( + , R0, R1, valor ) com valor constante
* Registrador ( + , R0, R1, R2 ) com dado em registrador
* Deslocamento ( + , R0, R1, M [base+R2] ) com variável local
* Registrador indireto ( + , R0, R1, M [R2] ) com acesso a dado via apontador
* Indexado ( + , R0, R1, M [R2+R3] ) com endereçamento de arranjo
* Direto ou absoluto ( + , R0, R1, M8080 ) com acesso à  memória estática
* Indireto em relação à memória ( + , R0, R1, M [M [R2]] ) com combinações de apontadores
* Pilha:
  + Prioridade de expressão 🡪 duas pilhas (operando , operadores).
  + Reverse Polish Notation 🡪 2 \* 3 + 4 = 2 3 \*4 +