28/02/2023 – Aula 1

Overflow

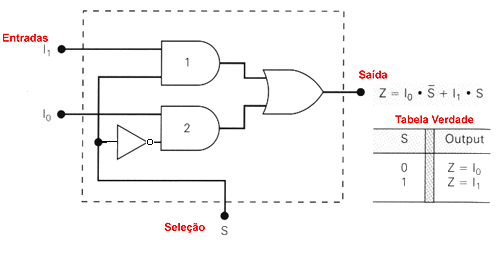
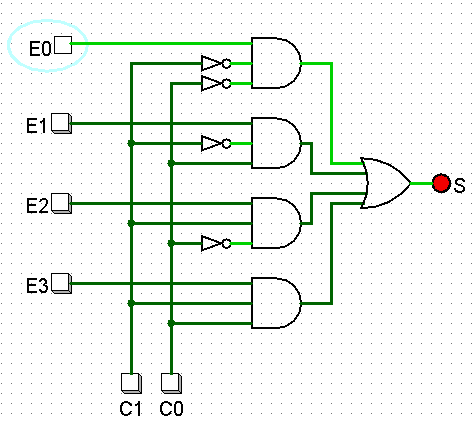
* Toda máquina precisa de um modelo matemático (representação e operação), pode-se considerar uma arvore binária como um modelo matemático
* Na multiplicação em binário, existe o problema do armazenamento do resultado, pois n1 de M bits vezes n2 de Z bits pode ter um resultado de M+Z bits, assim devemos ter um algoritmo para determinar se esse problema irá ocorrer
* XXXX.XXXX
  + Notação de ponto fixo --> ponto não move, não representamos decimal
  + Move pra esquerda --> ganha magnitude do número – melhora underflow
  + Move pra direita --> ganha precisão - melhora overflow
  + Quando o que queremos medir é muito menor que a unidade de medida (underflow)
  + Quando o que queremos medir é muito maior que a unidade de medida (overflow)
  + 5.3 --> 0101,0100 = 5.25
  + Erro absoluto (total – achado) --> 5.3 - 5.25 = 0.05
  + Erro relativo (erro absoluto /total) --> 0.94%
* Número negativo:
  + Notação sinal-magnitude ( coloca 1 ou 0 no inicio )
    - Problemas do modelo :
      * Dupla representação do 0
      * Algoritmo
  + Complemento de 2 --> existe para não termos que somar e subtrair, assim agilizando o processo--> inverte 0 com 1 e adiciona 1
* OBS: maior defeito da máquina é a quantidade de bit
* Flag de overflow --> avisa que conta está errada, CIN XOR COUT, ou seja, toda vez que o número que vc quer representar é maior que a escala

28/02/2023 – Aula 2

ULA

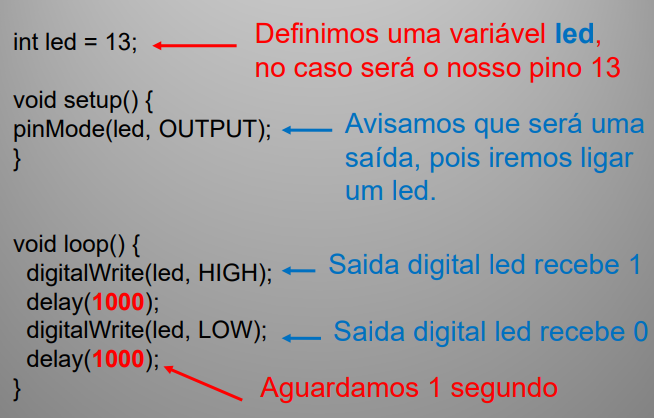
* Bitwise à bit a bit

ULA:

* MUX:
  + S = a.~b + b.c
  + 2 canais:
  + 4 canais:
* 

Aula dia 07/03 - Arduino

* Bootloader – BIOS do arduino , ou seja , conversar via porta USB
* Shields:
  + Escudos -> placas adicionais com conexões ao Arduino e que permitem interagir com tecnologias diversas e com facilidade
  + Sensor --> traduz grandeza física para elétrica
  + Atuadores --> grandeza elétrica para física
* Eletricidade --> V = R x I – (V = OHM x AMPERE)
* Programa arudino:



* + PinMode --> Esta define um pino com entrada ou saída. O arduino possui 20 pinos disponíveis, 14 digitais e 6 analógicos (0 a 13 e 14 a 19). Esse comando deverá estar preferencialmente na função setup ()
  + digitalWrite ( pino, valor ) -->Liga ou desliga uma saída digital, apenas caso o pino seja definido como OUTPUT. Os valores podem ser HIGH ou LOW (1 ou 0)
  + delay( ms) Aguarda o tempo passado como argumento em ms.
  + Serial.begin ( velocidade ) Essa função abre uma comunicação serial na velocidade passada como argumento. A função usa os pinos 0 e 1 para recepçãp e transmissão.
  + Serial.println ( valor ) Essa função transmite os caracteres ascii do arduino para o computador
  + Serial.available ( ) Essa função retorna o número de bytes disponíveis para leitura na porta serial.

Aula 07/03 – pt2 - ULA

* Formula genérica: FOTO

Aula 14/03/2023

* Multiplicador paralelo à combinacional(FOTO):
  + Problema: demora muito, explosão exponencial
* Lembrando elementos de memória cache:
  + Flip-flop(D) – entrada permanece a entrada
* Unidades básicas de construção da máquina
  + Mux
  + Ula
  + Somador
  + Registrador à sem clock na representação
* Registrador:
  + Pode empurrar coisas pra direita com shift right
  + Pode empurrar coisas para esquerda com shift left
* Para resolver o problema da multiplicação vamos criar:
  + Algoritmo foto
* Versão 2:
  + Mais barata (ULA de 32 bits)
  + Mais rápido (ULA de 32 bits)
* Versão 3: economia de registradores

21/03/2023 – Máquina para ponto flutuante

* Como fazer um ponto decimal flutuar:
  + Deixar o num representado por dois pedaços -- Notação cientifica
  + 1530 =
    - 154,3 x 10 ^1
    - 15,43 X 10^2 ...
  + Parte inteira = mantissa
  + 10 xn = expoente
  + Regra : 1<=M<10 – IEEE754 --> norma criada
  + Regra binária: 1<=M<2
* Exercicio:

5,3(10) = ? tosca5M3E

101,01001

1,0101001 \* 2^2

0(SINAL MANTISSA)

1010(MANTISSA)

010(EXPOENTE)

Pega 4 em 4:

0101 = 5

0010 = 2

5,3(10) = 52 tosca5M3E

52toscaM3E = ?(10)

0101 0010

01010(MANTISA) 010(EXPOENTE)

1,010 \* 2^2 = 101,0= 5

Erro absoluto = 5.3 - 5 = 0,3

Erro relativo = 0,3/5,3 = 5,6%

* 1 --> leading bit , pressupomos que existe, assim não precisamos representar
  + Consequência: diminuição de erro
* Notação IEEE754
  + 1 bit = sinal
  + **Sinal + expoente + mantissa**
  + **1<=M<2 e leading bit**
  + **Valor real = valor representado – BIAS (2^nbitexp-1 - 1)**
* Para fazermos contas complexas com número decimal de binário:

1,101 \* 2^-2 = 0,01101

1101 \* 2^-5 = 1101/2^5 = 1101/32 = 13/32

* Exemplo de transformação:

D2 ieee754 4E3M = ?(10)

1101 0010

1 1010(e) 010(m) (Aplica máscara)

BIAS = 2^3 – 1 = 7

e = 1010 -- 10 - BIAS (7) OLHAR SE E n é normalizado(0000 ou 1111)

e = 3

-1,010(m com leading bit) \* 2^3 --> -1010 --> -10(10)

* Exemplo 2:

11 ieee754 4E3M = ?(10)

0001 0001

0 0010(e) 001(m)

e = 0010 = 2 - 7 = -5

1,001 \* 2^-5

1001 \* 10^-3 \* 2^-5 = 1001 / 2^8 -- 9 /256

* Exemplo 3:

5,3(10) = ? ieee4E3M

101,01001

1,0101001 \* 2^2(APLICA REGRA MATISSA)

expReal = representado - BIAS

2 = representado - 7

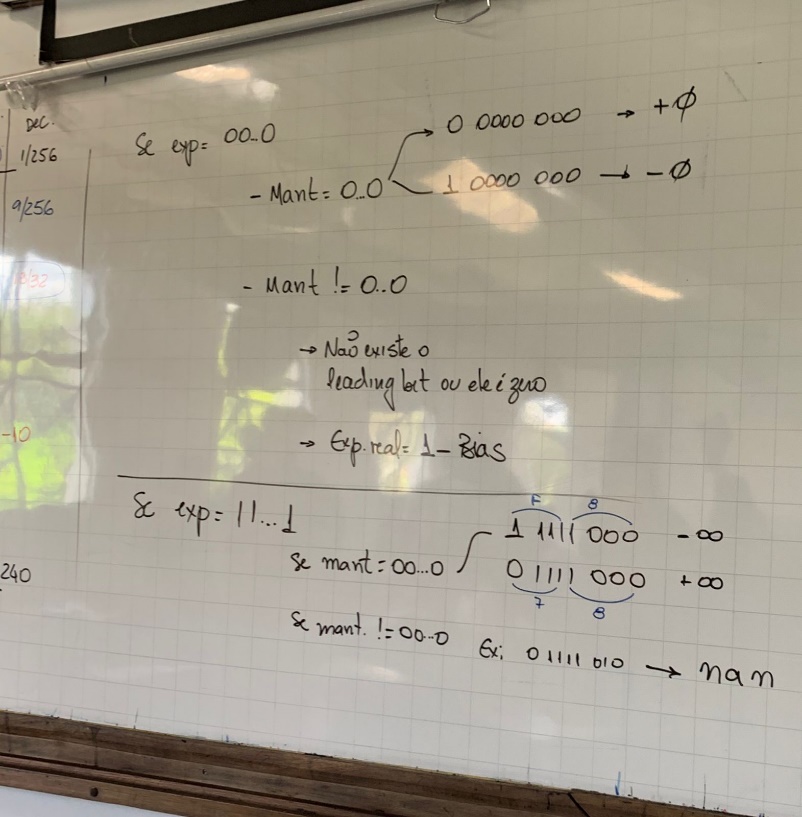
representado = 9 --> 1001

MASCARA

0 1001 010

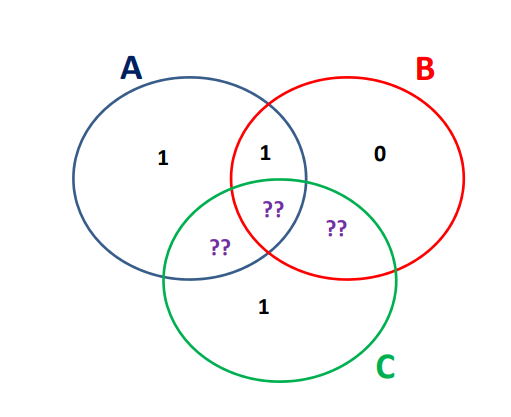
0100 1010

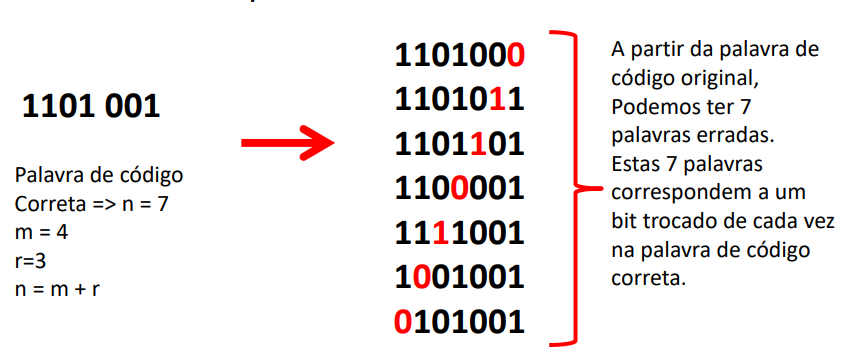
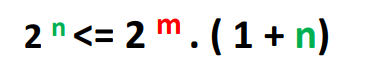
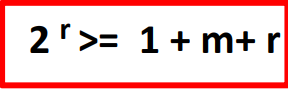
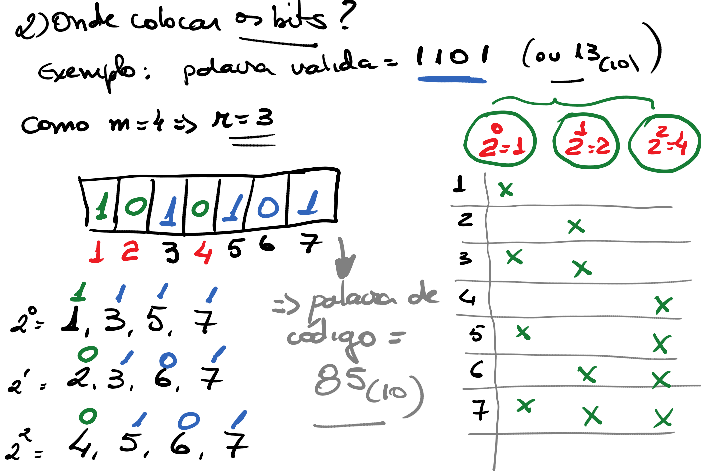
4A à não é uma representação exata, pois tem erro

* Qual o menor numero positivo normalizado?

28/03/2023 - Código de Hamming

* Detecta e corrige erro – 1bit
* Bit paridade --> Bit extra para quantidade de bits ficar par
* Bit de verificação --> Certeza de que número transmitido está correto
* Distancia Hamming:
  + Distância de erro simples que uma palavra está da outra
  + 111 – 000 = 111
  + Diferença dos números
  + Bit de paridade dectecta erro de 1 bit



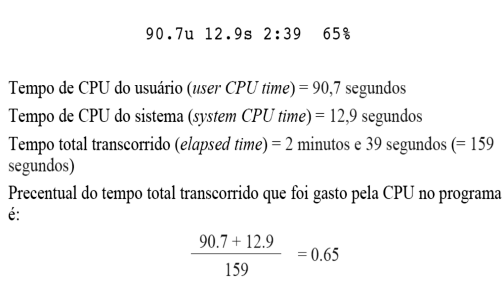
* ?? = 0 0 1
* O receptor procurará preservar a paridade e a única forma será ajustar o bit que foi trocado durante a transmissão, esse bit, mesmo desconhecido, será encontrado e corrigido.
* 1101 001 --> M e R – M+R = N
  + N = palavra de código (7)
  + M = palavra válida (4) -- Única que tem dado relevante
  + R = bit de validação (3)
  + 
  + Para n bits temos 2 ^n palavras
  + 
  + 
  + Quanto maior a palavra menor será o impacto para corrigir a transmissão
  + 

Aula 28/03/2023 pt 02

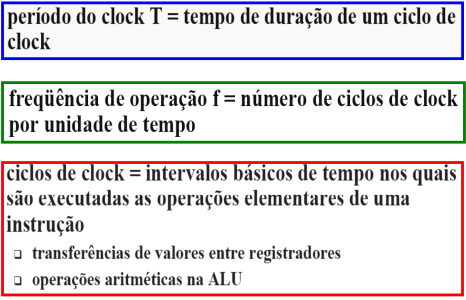
* Multiplicando em IEEE754:
  + SP = XOR(S1,S2)
  + MP = M1 x M2
  + EP = E1 + E2
  + Se mantissa fora da Regra : Shift para direita e incrementa 1 no expoente à perda no número , ou seja , perda na precisão
* Somando em IEEE754:
  + N de shifts : E1 – E2 à E1 = maior expoente
  + Se mantissa fora da Regra : Shift para direita e incrementa 1 no expoente à perda no número , ou seja , perda na precisão

Aula 04/04 -- Medidas de desempenho

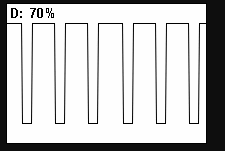
* Para medir não é justo levarmos em consideração alguns tempos : como cpu, como input ...
  + Ou seja tem que ter uma distinção de tempos
* **User CPU time** --> tempo gasto executando instruções do programa de usuario
* **System CPU time** --> tempo gasto com tarefas de SO necessária para execução do programa



* Definições:



Precisamos de uma métrica que depende da frequência, ou seja, precisa-se de uma medida única e constante --> ciclos de clock

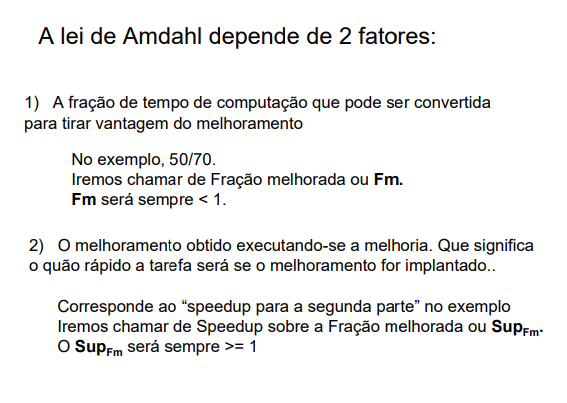
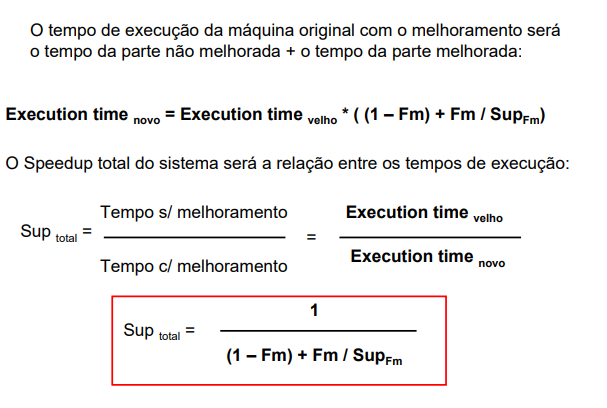
* + Duty cycle --> fator do tempo que circuito ta on e off – quanto, quanto maior tempo ligado maior será o brilho 
  + 

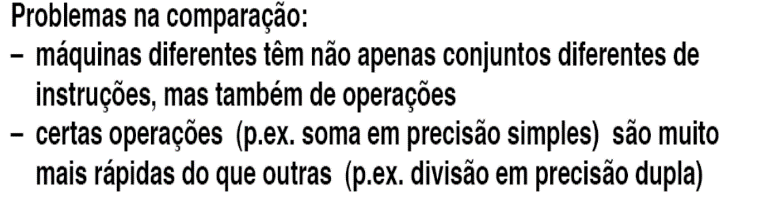
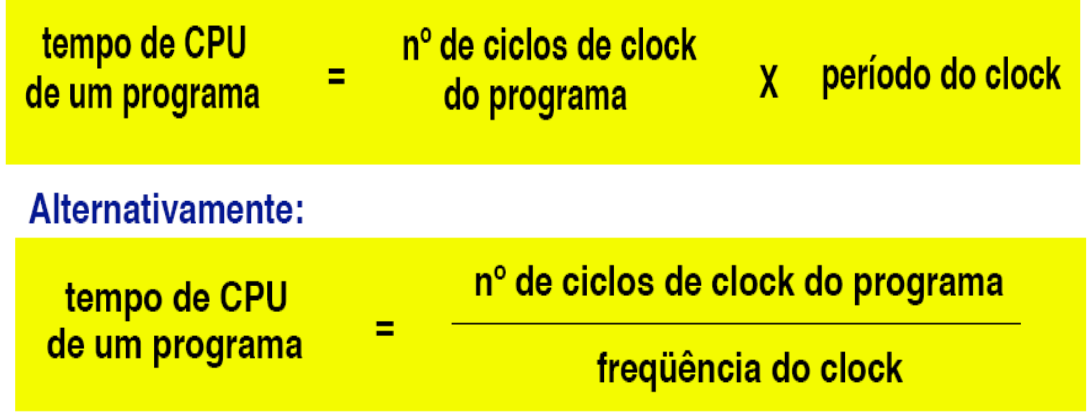
Aula 04/04/2023

* Se dissermos que um computador é mais rápido que outro:
  + Usuário à um determinado programa roda mais rápido do que outro
  + CRC à um sistema executa mais tarefas por hora
* Tabela

  Descrição gerada automaticamentePerformace = 1/Execution Time
* Lei de Amdahl:
* Texto, Carta

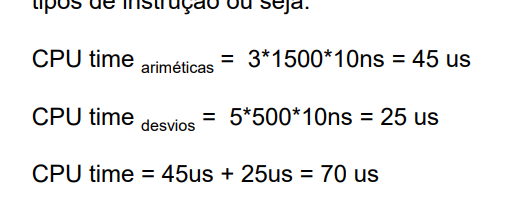
  Descrição gerada automaticamente

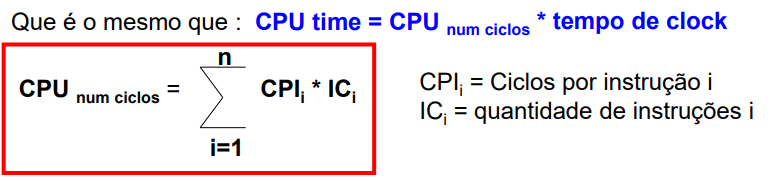
Aula dia 11/04/2023

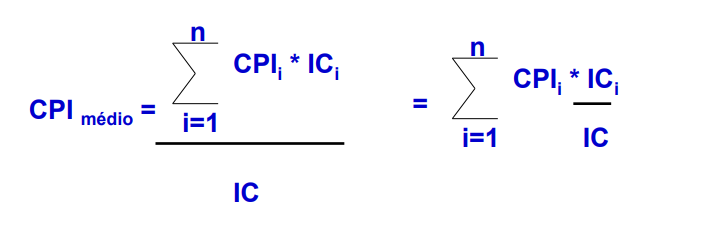
* MIPS:
  + Significa um milhão de instruções por segundo
  + Normalmente é aplicada a instrução mais rápida, porém não é justo comparar soma com ponto flutuante e soma normal , assim esse sistema parou de ser usada
* MFLOPS:
  + N de op de ponto flutuante/ tempo cpu \* 10^6
  + Existe porem não é o mais usado
* Proposta à usar uma métrica única , mas uma que dependa de mais fatores(**Equação da CPU**):
  + Tempo necessário para cada instrução
  + Quantidade de instruções de um programa
  + Características físicas da máquina
* Tempo de CPU de um programa:
* CPU time = IC \* CPI \* tempo de clock

CPI médio:

* O que acontece é que normalmente um programa possui diferentes tipos de instruções. Dessa forma podemos calcular CPI médio --> média ponderada:





* 

Benchmarks:

Conjunto de programas executado pelo usuário

09/05/2023 - MIPS

* ADD a,b,c = a = b+c;
* Variaveis:
  + Não temos variáveis , porem temos registradores, maquina GPR não possui operações em memoria
* MIPS tem 32 registradores , palavra de 32 bits , 32 bits de endereçamento
* Registadores – iniciam c/ $:
  + Tipo S à programador define
  + Tipos Tà temporário
* Comentário à #
* Texto, Carta

  Descrição gerada automaticamenteSintaxe de instruções:

Sempre teremos que associar aos registradores

Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente com confiança médiaEX:

* Como fazer i = 0:
  + SUB $s0,$s0,$t0
* Imediatos ADDI:
  + São constantes numéricas
  + Texto

    Descrição gerada automaticamenteÚltimo termo vai ser uma constante

13/06/2023

* Busca de instrução --> pc aponta para memória de instrução e acrescenta 4 unidades em pc