COMPUTADORES CLÁSSICOS E QUÂNTICOS:

ESTUDOS, IMPLEMENTAÇÕES E IMPACTOS SOCIAIS

Gabriel R. Zsigmond

INICIAÇÃO CIENTÍFICA

ESCOLA SUPERIOR DE PROPAGANDA E MARKETING

Sistemas de Informação em Comunicação e Gestão

Brasil

17 de agosto de 2020

Gabriel R. Zsigmond

Relatório Parcial

COMPUTADORES CLÁSSICOS E QUÂNTICOS:

ESTUDOS, IMPLEMENTAÇÕES E IMPACTOS SOCIAIS

Projeto de Iniciação Científica para a Escola Superior de Propaganda e Marketing.

Orientador: Prof. Dr. Humberto Sandmann

Brasil

17 de agosto de 2020

COMPUTADORES CLÁSSICOS E QUÂNTICOS:

ESTUDOS, IMPLEMENTAÇÕES E IMPACTOS SOCIAIS

Gabriel R. Zsigmond

Resumo

O presente projeto, “Computadores Clássicos e Quânticos: Estudos, Implementação, Simuladores e Impactos Sociais”, se propõe a estudar a história do computador e sua evolução. A mais recente inovação na área é a computação quântica, que certamente, inaugura a próxima geração de computadores. A computação quântica muda a arquitetura da computação, permitindo que os novos computadores sejam exponencialmente mais eficientes quando comparados aos mais modernos da atualidade. O projeto se propõe a entender e prototipar um computador tradicional de 8 bits em hardware, usando apenas portas lógicas simples, a fim de ilustrar, de forma clara, o funcionamento de um computador tradicional. E, busca entender e estimar as consequências sociais que os avanços da tecnologia e o desenvolvimento da computação quântica pode gerar. Entende-se que para essa análise, se faz necessário, inicialmente, uma ampla revisão bibliográfica, a fim de comparar esses dois tipos de computadores. Para ilustrar esta, será desenvolvida uma aplicação web que simula um computador tradicional e um computador quântico executando o mesmo algoritmo. É esperado que ao final do projeto, esse estudo traga um amplo e aprofundado conhecimento da área, além de, uma contribuição em relação ao impacto social do uso computação da quântica.

Sumário

Lista de Tabelas

Lista de Figuras

1 Introdução

1.1 Definição dos objetivos e da sua relevância

A palavra “computador” é usada desde o século XVII, tendo a sua primeira referência escrita datada de 1613. No entanto, por muito tempo “computador” não tinha o mesmo significado que tem hoje, sendo utilizado, até a década de 1940, para se referir à profissão de alguém que calcula, segundo o dicionário Michaelis: “Aquele ou aquilo que calcula baseado em valores digitais; calculador, calculista” [7].

Tendo em vista o antigo significado atribuído à palavra “computador”, pode-se questionar sobre como uma palavra antes atribuída às pessoas, passa a se referir a máquinas. Isso tem a ver, em ordem cronológica, com o estudo de Alan Turing (Reino Unido, 1912-1954) – um matemático, lógico, criptógrafo e herói de guerra – que tinha como objetivo compreender aquilo que podia ou não ser calculado, resultando no modelo mais poderoso de computador – a Máquina da Turing, – também conhecida como “Maquina Universal”. Ao lermos o termo “Máquina de Turing” logo vem à nossa mente a figura de um computador, mas engana-se quem pensa isso – a Máquina de Turing nada mais é que uma fórmula matemática, ou um conceito abrangente, que fundamenta o pensamento computacional. Esse trabalho, publicado em 1937, trouxe muitas contribuições para o desenvolvimento da computação, e observamos isso na influência dele em computadores e celulares que você, leitor, pode estar usando para ler esse trabalho.

A Máquina de Turning será o ponto de início desse trabalho, dada a sua importância e contribuição para a computação – tanto em sua constituição como em seu funcionamento.

O modelo matemático da Máquina de Turing foi desenvolvido a partir de três principais componentes: fita infinita; cabeçote; e a máquina de estado finito; – sendo similar a um autômato finito[1], porém com uma memória ilimitada e irrestrita,

[1] Um sub-tópico da Ciência da computação teórica, também chamado máquina de estados finita determinística — é uma máquina de estados finita que aceita ou rejeita cadeias de símbolos gerando um único ramo de computação para cada cadeia de entrada.

ilustrada na figura 1 abaixo.

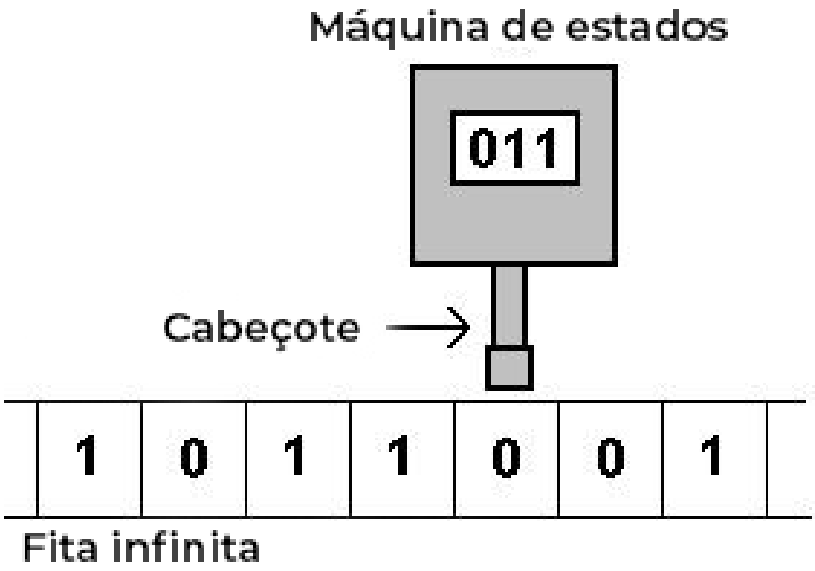

Figura 1: Desenho esquemático Máquina de Turing

Neste modelo, a fita infinita é dividida em células, cada uma contendo um símbolo de um alfabeto finito. O cabeçote é responsável por se deslocar para a direita ou para a esquerda e efetuar a leitura ou escrita em uma célula. Este processo é ilustrado pelo Prof. Dr. Fabio Gagliardi Cozman [4] da seguinte forma:

1. Inicialmente a fita contém somente a cadeia de entrada (dados originários do programa), disposta no "meio"[2] da fita, com o cabeçote posicionado no início da cadeia, tendo o resto das células da fita vazias;

2. Para armazenar algo, a máquina escreve na fita;

3. O cabeçote pode ser movido livremente para a esquerda ou direita, afim de ler ou escrever valores em qualquer célula;

4. As saídas "aceita" e "rejeita" são obtidas ao entrar nos estados de aceitação e rejeição;

[2]Meio é algo abstrato nesse sentido, pois não existe meio de um valor infinito

5. Se não entrar em um estado de aceitação ou rejeição, continuará sua computação para sempre, em “loop infinito”.

Assim, a partir desse funcionamento, o modelo computacional de Turing possibilitou três operações fundamentais básicas: a leitura, a escrita e a movimentação.

Após aproximadamente uma década, em 1946, esse conceito foi implementado pela primeira vez, no primeiro computador digital eletrônico de grande escala, criado pelos cientistas norte-americanos, John Presper Eckert e John W. Mauchly, da Electronic Control Company. Chamado de ENIAC (Electrical Numerical Integrator and Calculator), essa máquina foi mais um marco da evolução da tecnologia. Em 1956, ao final dos seus 10 anos de operação, e com constantes aprimoramentos, o ENIAC continha 20.000 tubos de vácuo, 7.200 diodos de cristal, 1.500 relés, 70.000 resistores, 10.000 capacitores e aproximadamente 5.000.000 juntas soldadas à mão. Ele pesava mais de 27 toneladas, tinha aproximadamente 2,4m × 0,9m × 30m, consumia 150 kW de eletricidade e ocupava 167 m^2 [19], tendo a sua disposição ilustrada na figura 2 abaixo.

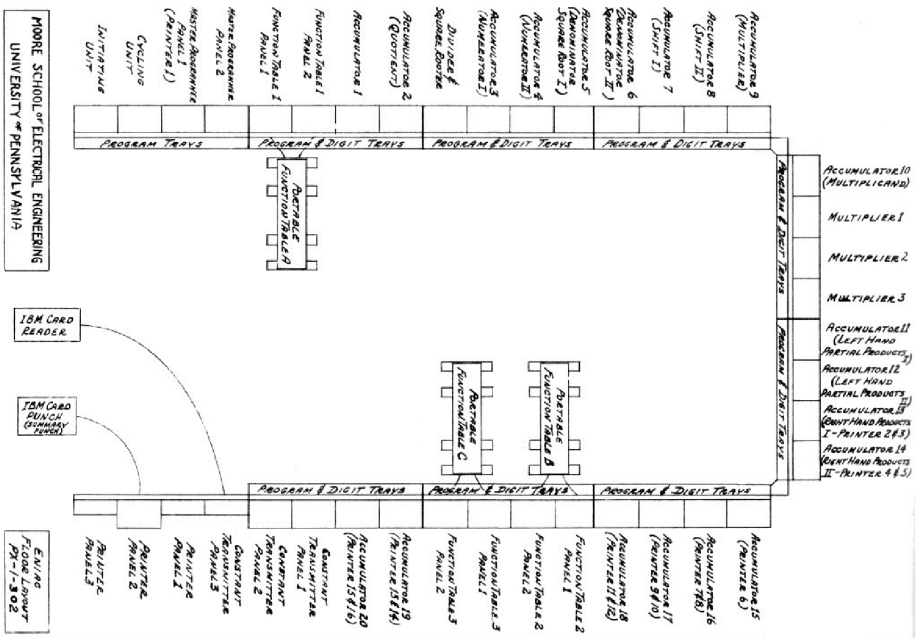

Figura 2: Desenho esquemático do ENIAC/US Center for Military History — Adele Goldstine

A partir do ENIAC, as possibilidades tecnológicas tomaram uma nova proporção e em 1969, apenas 13 anos após o desligamento do primeiro computador digital eletrônico, surge o computador de bordo da Apollo 11 — na missão que levou o homem à Lua. O Apollo Guidance Computer, uma revolução no mundo computacional, era bem menor do que o seu antecessor e tinha 32.768 bits[3] de RAM, o suficiente para armazenar apenas um texto não formatado, com cerca de 2.000 palavras.

Ponto de interesse: O que contrasta o Iphone XS com 4gb de RAM lançado em 2018 ao Apollo Guidance Computer de 1969 é que o equipamento mais novo tem cerca de 1 milhão de vezes mais memória do que aquele que estava no Apollo 11 [14].

Durante o século XX, além do aumento do poder computacional dos dispositivos, outro fator impactante foi a refatoração de seus tamanhos. Assim, com tecnologias wearables[4], os computadores se tornam ativamente presentes no cotidiano.

[3]Para uma explicação elaborada sobre bits refira-se ao capítulo 2 pagina 18

[4]A tecnologia em questão não somente pode ser usada como uma peça de roupa ou um acessó-

Levando em consideração o rápido avanço e desenvolvimento computacional, entende-se que os computadores trazem benefícios à sociedade, seja facilitando a comunicação e o compartilhamento de conhecimentos, assim como em vários outros aspectos. No entanto, a agilidade dessas **transformações da tecnologia da computação**, gera grandes expectativas e incertezas sobre o que ainda está por vir, tanto nas questões de mudanças tecnológicas quanto no impacto que trará para a sociedade, principalmente no que se refere à segurança das informações.

Tendo em vista a incerteza que acompanha o futuro da computação e de seus próximos avanços, a presente pesquisa se propõe a estudar os conceitos da física clássica e da física quântica aplicados à computação, além do estudo dos princípios da criptografia[5]. Com base nesses conceitos será prototipado um computador clássico de 8 bits em hardware usando apenas portas lógicas simples, de forma a possibilitar uma ilustração clara do funcionamento. Junto a isso será desenvolvida uma aplicação web que ilustra o funcionamento de um processador quântico. Ao final, conceitos de criptografia serão utilizados para exemplificar possíveis mudanças sociais que os próximos avanços tecnológicos podem gerar.

1.2 Metodologia empregada

Para a realização da presente pesquisa de iniciação científica, foi indispensável a realização de uma revisão bibliográfica, afim de recuperar e aprofundar o conhecimento cientifico já produzido. Segundo Telma Cristiane Sasso de Lima, o conhecimento da realidade não é apenas a simples transposição dessa realidade para o pensamento, mas sim a reflexão crítica, que se dá a partir de um conhecimento acumulado que irá gerar uma síntese, o concreto pensado [8]. E também a utilização do processo científico para a elaboração e efetivação do projeto em si. Ambas metodologias citadas acima são cruciais para o desenvolvimento do relatório final na área de pesquisa em computação, já que em grande parte dos estudos, a utilização do processo científico é frequentemente utilizada para um maior

rio, como também tem que possuir características que a conectem a outros aparelhos ou à internet.

[5]Criptografia é um sistema de algoritmos matemáticos que codificam dados para que só o destinatário possa ler.

entendimento da obra e para que a construção do projeto possa se tornar mais facilmente executável.

Assim, para o desenvolvimento da entrega do protótipo — computador de 8-bits — e para o simulador do computador quântico web, a principal metodologia utilizada será Project Based Learnig (PBL). De acordo com David Van Andel, o PBL envolve os alunos em um processo rigoroso de investigação, onde eles fazem perguntas, encontram recursos e aplicam informações para resolver problemas do mundo real [2]. Entende-se que esta é a melhor metodologia para desenvolver um protótipo físico de um computador e programar um site.

Nos dois capítulos a seguir, 2 e 3, respectivamente Computação Clássica e Computação Quântica é apresentada a revisão bibliográfica

2 Computação Clássica

Como já mencionado anteriormente, Alan Turing – matemático inglês, cientista da computação, lógico, criptoanalista, filósofo e biólogo teórico – publicou o artigo “On Computable Numbers with an Application to the Entscheidungs-problem” [17] em 12 de novembro de 1937, artigo que viria a constituir a teoria básica da computabilidade, que se faz presente até hoje.

O mecanismo abstrato descrito no artigo de Turing fornece os conceitos fundamentais de computadores que outros engenheiros conceberam posteriormente. Na sua essência, uma Máquina de Turing é um dispositivo que manipula símbolos em uma tira de fita de acordo com uma tabela de regras, funcionamento que possibilitou a formalização dos conceitos de “algoritmo” e “computação”. Ainda, apesar da simplicidade da Máquina de Turing, ela pode ser adaptada para simular a lógica de qualquer algoritmo de computador.

Apesar de hoje se ter a compreensão das contribuições de Alan Turing para a base da computação, os computadores clássicos são atualmente descritos como tendo a chamada arquitetura de von Neumann. No entanto acredita-se que o idealizador dessa arquitetura partiu dos modelos matemáticos desenvolvidos por Turing[1], que incluía o conceito de programa armazenado, originado a partir da construção da Máquina de Turing. Conceito, também encontrado no projeto EDVAC de von Neumann [9], que possibilita o armazenamento de instruções e dados na mesma memória, permitindo a manipulação de programas como dados, que se constitui como característica do computador clássico.

Assim, pode-se considerar Turing como pai do computador, já que suas publicações antecedem a arquitetura de von Neumann. Ademais, tem-se que Turing foi o primeiro a explorar a idéia de uma máquina de uso geral por meio de sua noção de máquina universal – que constitui a base do computador clássico ao possibilitar o funcionamento da CPU em conjunto com a memória RAM. Ainda, tendo em vista as contribuições do matemático para a construção de uma classe importante de dispositivos de computação: o Bombe – um dispositivo eletromecânico usado pelos criptologistas britânicos para ajudar a decifrar as mensagens secretas criptografadas pela máquina alemã Enigma durante a Segunda Guerra Mundial – e

posteriormente o seu design do ACE (Automatic Computing Engine), fica evidente as contribuições de Alan Turing para a invenção do computador moderno. A partir da fala de Turing abaixo, pode-se identificar o ACE como um tipo de realização física da máquina universal.

"Some years ago I was researching on what might now be described as an investigation of the theoretical possibilities and limitations of digital computing machines. [. . .] Machines such as the ACE may be regarded as practical versions of this same type of machine." [16]

Por fim, entende-se que baseado na teoria da máquina de Turing, o físico e matemático John von Neumann desenvolveu uma arquitetura que consiste em uma única memória compartilhada para programas e dados; um único acesso à memória; uma unidade aritmética e uma unidade de controle de programa - ilustrado no diagrama abaixo.

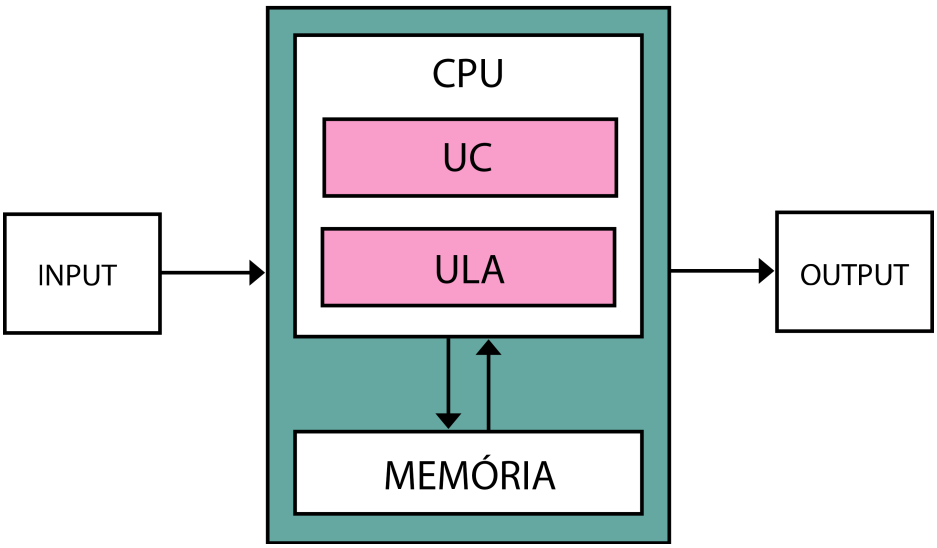

Figura 3: Diagrama arquitetura de Von Neumann

Nessa arquitetura, o cabeçote passa a ser uma CPU (Central Processing Unit), a fita se transforma em memória RAM e as operações são construídas e executadas em circuitos formados por portas lógicas chamadas de ULA (unidade lógica e aritmética) [13].

Atualmente, a maioria dos computadores modernos são construídos sobre a práxis da arquitetura von Neumann. A fim de simular um processador moderno de forma didática, foi elaborado o emulador ASM 24bits que permite a sua programação utilizando um Assembly[6] de 20 instruções. Similar à arquitetura de von Neumann esse emulador é composto por memória e CPU, na qual a última possui dois registradores descritos abaixo:

Registradores	
Nome	Descrição
Accumulator (ACC)	Registro mais usado para armazenar dados extraídos da memória. Está em diferentes números em diferentes microprocessadores.
Instruction Register (IR)	Registro que contém a instrução que está sendo executada no momento.

Tabela 1: Registradores ASM 24 bit

Dentro da CPU, está a unidade lógica aritmética (ULA), um circuito digital usado para realizar operações aritméticas e lógicas. Ela representa o bloco de construção fundamental da unidade central de processamento (CPU). No caso do ASM 24bits, a sua ULA possui 20 instruções, sendo elas:

ULA	
Mnemônico	Descrição
nop	Slot de memória vazio

[6]Uma linguagem assembly é uma linguagem de programação de baixo nível projetada para um tipo específico de processador. Ele pode ser produzido compilando o código-fonte de uma linguagem de programação de alto nível (como C / C ++), mas também pode ser escrito do zero.

jmp	Salto incondicional. Recebe uma variável como parâmetro e executará o código a partir da linha abaixo.
jz	Salta se o valor do acumulador (AC) é 0. Recebe uma variável como parâmetro e executa o código a partir da linha abaixo se o valor do AC for zero. Se o valor de AC NÃO for zero, a próxima linha será executada.
jnz	Pula se o valor do acumulador (AC) NÃO é 0. Recebe uma variável como parâmetro e executa o código a partir da linha abaixo se o valor do AC não for zero. Se o valor de AC for zero, a próxima linha será executada.
lv	Carrega uma constante diretamente no acumulador. Recebe uma constante em notação hexadecimal 0x00F2 por exemplo.
add	Adiciona uma constante ao valor do acumulador. Recebe uma constante em notação hexadecimal 0x00FA por exemplo.
addm	Recebe uma variável como parâmetro e adiciona o valor da variável ao valor do acumulador.
sub	Subtrai uma constante do valor do acumulador. Recebe uma constante em notação hexadecimal 0x00FA por exemplo.
subm	Recebe uma variável como parâmetro e subtrai o valor da variável do valor do acumulador.
mul	Multiplica uma constante do valor do acumulador. Recebe uma constante em notação hexadecimal 0x00FA por exemplo.
mulm	Recebe uma variável como parâmetro e multiplica o valor da variável pelo valor do acumulador.

div	Divide o valor do acumulador por uma constante. Recebe uma constante em notação hexadecimal como parâmetro, 0x00FA por exemplo.
divm	Recebe uma variável como parâmetro e divide o valor do acumulador pelo valor da variável.
load	Recebe uma variável como parâmetro e carrega o valor da variável para o acumulador.
stor	Armazena o valor atual do acumulador em uma variável.
sc	Chamada de função.
rc	Retorno de função, irá pular para a linha abaixo da chamada de função mantendo o valor atual no acumulador.
end	Irá parar a execução do programa.
in	Solicita ao usuário uma entrada (o número inserido deve ser em hexadecimal) e carrega a entrada no acumulador.
out	Alerta o usuário com o valor atual do acumulador.

Tabela 2: Instruções ASM 24 bit

É a partir das instruções pertencentes à ULA que se compõe uma Linguagem Assembly, essencial para o funcionamento de qualquer computador clássico e utilizada na criação do ASM 24bits (mencionado acima). A linguagem Assembly, frequentemente abreviada como “asm”, é uma linguagem de programação de baixo nível em que há uma correspondência muito forte entre as instruções da linguagem e as instruções do código de máquina[7], sendo altamente dependente do hardware que está sendo utilizado. Dessa forma, a asm é indispensável durante a primeira etapa do ‘boot’, já que o hardware ainda não está configurado para o sistema ope-

[7]Na programação de computadores, o código de máquina, que consiste em instruções em linguagem de máquina, é uma linguagem de programação de baixo nível usada para controlar diretamente uma CPU. Cada instrução faz com que a CPU execute uma tarefa muito específica, como uma carga, um armazenamento, um salto ou uma operação ALU em uma ou mais unidades de dados nos registros ou memória da CPU.

racional.

Assim, ao ligar qualquer computador clássico da atualidade, ele passará por algumas etapas importantes antes de poder ser utilizado por seu usuário final. Ao reiniciarmos um computador ele deverá carregar o sistema operacional de qualquer variante (Mac, Windows, Linux, etc.), a partir de algum dispositivo de armazenamento permanente que está atualmente conectado ao computador: seja um cd, um disco rígido, um USB, etc. O ambiente pré-SO oferece poucos serviços de alto nível. Neste estágio, mesmo um sistema de arquivos simples, que possa ler e gravar arquivos lógicos em um disco, seria um luxo. Felizmente, temos disponível o Basic Input/Output Software (BIOS), uma coleção de rotinas de software que são carregadas inicialmente de um chip para a memória e inicializadas quando o computador é ligado. O BIOS fornece detecção automática e controle básico dos dispositivos essenciais do seu computador, como tela, teclado e discos rígidos. Depois que este conclui alguns testes de baixo nível do hardware, principalmente se a memória instalada está funcionando corretamente ou não, ele deve iniciar o sistema operacional já armazenado em um de seus dispositivos. No entanto, precisamos lembrar que o BIOS não pode simplesmente carregar um arquivo que represente seu sistema operacional a partir de um disco, pois o ele não tem noção de um sistema de arquivos. Este deve ler setores específicos de dados (geralmente 512 bytes de tamanho) de localizações físicas específicas dos dispositivos de disco, para carregar do sistema operacional.

É a partir desses conceitos abordados que se constitui a estrutura básica da computação clássica, que dependem da física clássica para operar. Estes são os computadores tradicionais que usamos em nosso dia-a-dia – sejam eles Apple, Samsung, Dell ou qualquer outro –, também classificados como computadores binários, pois processam as instruções a partir de números binários, compostos apenas pelos símbolos “1” e “0”, ligado e desligado, respectivamente. Assim, julga-se importante e de larga relevância ao tema compreender essa representação numérica.

Números binários ou números em base 2 são compostos por apenas dois dígitos, {0, 1}. Dessa forma, seu funcionamento é similar ao sistema decimal, ou

base 10, que são compostos por dez dígitos, {0; 9}. No sistema decimal, é simples contar até nove, porém não existe um símbolo ou dígito para representar o número dez, sendo então representado por dois dígitos, “10”, sendo uma simples lógica de posicionamento. Mais uma vez, após o número “99”, é necessário utilizar da mesma regra para representar o número cem, “100”. Já em base 2, o número zero é representado pelo símbolo “0”, e o número um por “1”. O mesmo dilema é enfrentado ao chegar no próximo valor: dois. E então é usada a mesma lógica de posicionamento, em base dois. Desta forma, o número dois é representado por “10”, o três por “11”, quatro por “100” e assim por diante. Portanto, números binários podem se tornar longos e compostos por muitos dígitos que levam o nome de bits [11]. É com base nos bits[8] ligados e desligados que o computador baseia a sua linguagem. Para transforma-lo em base dez é preciso avaliar o valor de cada bit de acordo com a sua posição.

Exemplo: número binário 1011:

$$1011(b) = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$$

$$= 8 + 0 + 2 + 1 = 11(decimal)$$

O peso de cada bit de um número binário depende da sua posição relativa ao número completo, sempre partindo da direita para a esquerda.

- O peso do primeiro bit é $bit * 2^0$
- O peso do segundo bit é $bit * 2^1$
- O peso do terceiro bit é $bit * 2^2$
- O peso do quarto bit é $bit * 2^3$

A fórmula ilustrada acima, pode ser exemplificada em uma fórmula genérica:

$$= nth\ bit * 2^{n-1}$$

[8] A menor unidade de informação que pode ser armazenada ou transmitida na comunicação de dados.

É possível notar que a regra para números binários, se repete para números em base 10.

Exemplo: número decimal 4392:

- O peso do primeiro bit é $2 * 10^0$
- O peso do segundo bit é $9 * 10^1$
- O peso do terceiro bit é $3 * 10^2$
- O peso do quarto bit é $4 * 10^3$

$$4392 = 4 * 10^3 + 3 * 10^2 + 9 * 10^1 + 2 * 10^0$$

$$= nth\ bit * 10^{n-1}$$

Essa regra se mantém verdadeira para qualquer base numérica.

$$= nth\ bit * (base)^{n-1}$$

Apesar dos computadores funcionarem a partir de números binários, ao se referir a eles é comum se usar notação hexadecimal. No entanto, pode-se questionar o porquê desse uso, já que é mais natural que optemos por usar os números em base 10, uma vez que culturalmente estamos mais acostumados a eles. Georges Ifrah em seu livro, "The Universal History of Numbers" [6] aponta que:

"Traços da origem antropomórfica dos sistemas de contagem podem ser encontrados em muitas línguas. Na língua Ali (África Central), por exemplo, "cinco" e "dez" são respectivamente moro e mbouna: moro é na verdade a palavra para "mão" e mbouna é uma contração de moro ("cinco") e bouna, que significa "dois" (portanto, "dez" = "duas mãos").

Portanto, é muito provável que as palavras indo-européias, semíticas e mongóis para os dez primeiros números derivem de expressões relacionadas à contagem dos dedos."

Nesse sentido, Ifrah explica que:

"[...] a mão torna os dois aspectos complementares dos inteiros inteiramente intuitivos. Ele serve como um instrumento que permite o movimento natural entre

a numeração cardinal e ordinal. Se você precisa mostrar que um conjunto contém três, quatro, sete ou dez elementos, você levanta ou dobra simultaneamente três, quatro, sete ou dez dedos, usando sua mão como mapeamento cardinal. Se quiser contar as mesmas coisas, dobre ou levante três, quatro, sete ou dez dedos em sucessão, usando a mão como ferramenta de contagem ordinal."

Partindo dessa explicação, surge a ideia de números sendo representados a partir de 10 símbolos distintos: {0; 9}. Assim, o número decimal tem uma base de dez (ou seja, dez símbolos de dígitos distintos), já o número hexadecimal tem uma base de 16, por tanto, sendo necessário criar seis novos símbolos numéricos para possibilitar a contagem de 0 a 15 com apenas um símbolo. A maneira mais simples é usando algumas letras, como: 0,1,2, ... 8,9, a, b, c, d, e, f, em que, por exemplo, o símbolo 'd', representaria uma contagem de 13.

Para distinguir entre hexadecimal e outros sistemas numéricos, costumamos usar o prefixo "$0x$", que é especialmente importante para indicar dígitos hexadecimais que podem não conter nenhum dos dígitos da letra, por exemplo: $0x50$ não é igual $(decimal)50$ – $0x50$ é realmente $(decimal)80$.

A questão é que um computador representa um número como uma sequência de bits, uma vez que fundamentalmente seu circuito pode distinguir apenas entre dois estados elétricos: 0 e 1 – fazendo referência ao motivo pelo qual usamos base 10, é como se o computador tivesse um total de apenas dois dedos. Portanto, para representar um número maior que 1, o computador pode agrupar uma série de bits, assim como podemos contar para além de 9 agrupando dois ou mais símbolos, por exemplo, 456, 23, etc.

Para facilitar a conversa sobre o tamanho dos números com os quais estamos lidando foram adotadas nomenclaturas para séries de bits de certos comprimentos. As instruções na maioria dos computadores lidam com um mínimo de valores de 8 bits, que são denominados bytes. Outros agrupamentos são "short", "int" e "long", que geralmente representam valores de 16 bits, 32 bits e 64 bits, respectivamente. Também tem-se o termo "word", que é usado para descrever o tamanho da unidade máxima de processamento do modo atual da CPU: portanto, no modo real de 16 bits, a "word" se refere a um valor de 16 bits, já no modo protegido de 32 bits, uma

“word” se refere a um valor de 32 bits e assim por diante.

Desse modo, pelo fato de as sequências de bits serem compridas é mais fácil convertê-las para notação hexadecimal do que para o sistema decimal natural, ilustrado na imagem abaixo. Assim, é possível quebrar a conversão em segmentos menores de 4 bits do número binário, em vez de tentar somar todos os bits componentes em um total geral, o que seria muito mais difícil para cadeias de bits maiores, como para: 16, 32, 64, etc.

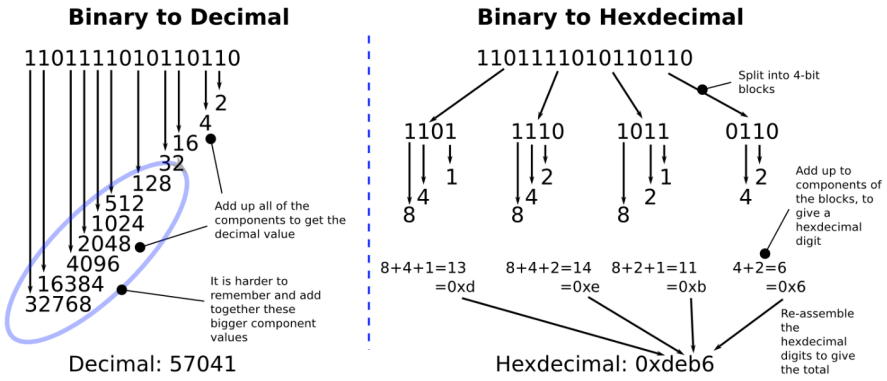

Figura 4: Conversão de 1101111010110110 para decimal e hexadecimal

Esse capítulo constituiu-se em apresentar conteúdos básicos e essenciais para possibilitar a compreensão do funcionamento de computadores clássicos. Conteúdos que foram compreendidos a partir do desenvolvimento de um sistema operacional simples que passa por todas as etapas essenciais do processo de “boot” – ilustrado na figura a baixo – *the-state-of-art* de computadores clássicos: ligando em ‘modo real de 16 bits’, carregando o kernel[9], transferindo-se para ‘modo protegido de 32 bits’ e ‘gerenciando interrupções’.

[9]Uma parte integrante de qualquer sistema operacional. Que se constitui em um programa com controle completo sobre tudo no sistema.

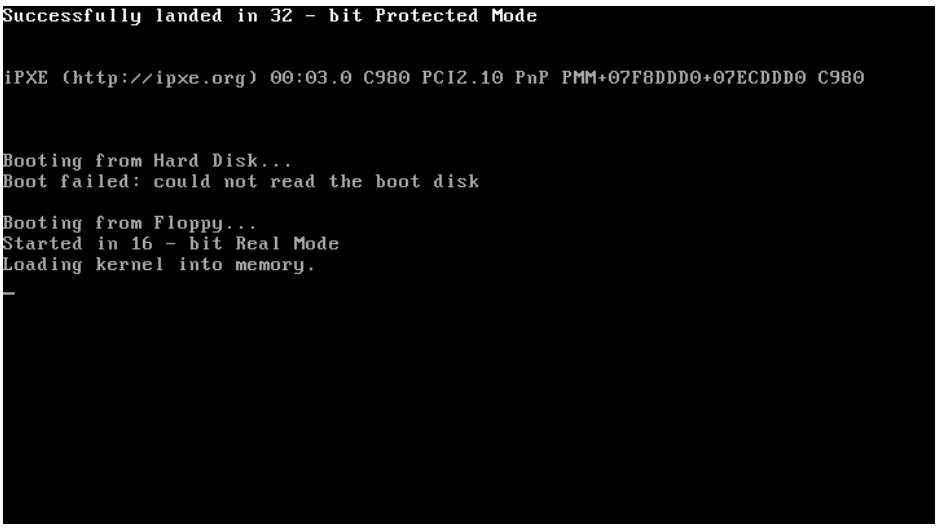

Figura 5: Processo de boot

Uma vez que o conteúdo sobre a computação clássica foi compreendido, podemos avançar para a próxima geração: computadores quânticos.

3 Computação Quântica

A computação quântica é uma abordagem inovadora quando se trata de processamento de dados na computação. Neste capítulo é apresentado uma breve recuperação da história da área, assim bem como seus conceitos basilares e até mesmo os previstos impactos que acompanharão a funcionalidade desta inovação.

3.1 História

Como proposto pelo cronograma desse estudo[10], o presente capítulo, a ser desenvolvido no decorrer do segundo semestre, adentrará o desenvolvimento da Computação Quântica assim como a descrição de sua arquitetura – tema de extrema atualidade. O capítulo partirá do estudo de Alexander Holevo, publicado em 1973, que obteve como resultado o conhecido "teorema de Holevo"[11] – marco da computação quântica –, perpassando pela sua historia até os dias atuais. Assim, poderemos compreender as implicações dos avanços em relação aos computadores quânticos, que como mencionado pelo artigo “Google claims its quantum computer can do the impossible in 200 seconds” [12] já consegue resolver um problema, que levaria 10.000 anos para ser solucionado pelo supercomputador (clássico) mais rápido do mundo, em 200 segundos.

3.2 Qubits

Na mecânica quântica, uma partícula pode assumir dois estados ao mesmo tempo, o que é chamado de superposição. Ao aplicar este conceito à computação, temos uma partícula como um bit [12] quântico, qubit (um bit quântico, unidade básica de informação em um computador quântico) que poderia retornar três posições: ligado (1), desligado (0), ou uma superposição de ambos, 1 e 0. Assim, um qubit pode comportar dois valores de uma vez; dois qubits quatro valores; e assim por

[10] página 39 tabela 6

[11] um importante teorema limitativo na computação quântica, um campo interdisciplinar da física e da ciência da computação. Às vezes chamado de limite de Holevo, uma vez que estabelece um limite superior para a quantidade de informação que pode ser conhecida sobre um estado quântico

[12] melhor explicado na página 18

diante. Isso significa que uma quantidade pequena de átomos pode comportar valores muito maiores do que os transístores de um computador binário, sendo mais eficiente. Um exemplo disto, seria o computador quântico da Google, que segundo WHYTE(2019), realizou um cálculo de um problema em 200 segundos, que poderia demorar cerca de 10.000 anos para ser resolvido pelo melhor supercomputador existente, alcançando assim, a supremacia quântica.

3.3 Implementções

Desde a decada de 2000 as grandes empresas da computação vem se enfrentando na corrida por atingir a melhor arquitetura quântica funcional. Entre elas, destaca-se as duas grandes referencias no mercado da tecnologia: Google e IBM. Em 2019 a Google anunciou que havia atingido supremacia quântica como visto através do artigo: “Quantum supremacy using a programmable superconducting processor”, [3] coloca toda a sociedade às portas de um novo mundo, pois com esse poder computacional, problemas de otimização, que até então jamais seriam solucionados em tempo hábil, agora, estariam resolvidos em períodos mínimos de tempo.

Já a IBM, vem traçando a sua trajetória quântica desde xxx, lançando em maio de 2016 o IBM Q Experience, que continha um processador quântico de cinco qubit e um simulador de correspondência. Essa arquitetura, possibilitou que os usuários só interagissem por meio do compositor quântico, com um conjunto limitado de interações de dois qubit e um guia do usuário que assumiu experiência em álgebra linear. Em 2018 essa experiencia teve significantes avanços, como exposto por Yuanhao Wang em seu artigo “16-qubit IBM universal quantum computer can be fully entangled”, “Os dispositivos backend atuais, da IBM incluem dois processadores com 5 qubits supercondutores sendo eles o ibmqx2 e o ibmqx4, um processador de 16 qubit, ibmqx5 e um processador de 20 qubit QS1_1.” [18]. Atualmente, a empresa já vislumbra horizontes que pareciam muito distantes, anunciando que em 2023 terá um computador e David Nield – jornalista que escreve sobre ciência e tecnologia há mais de 20 anos tendo artigos publicados na Wired, Popular Science, The Guardian e Gizmodo – acredita que “Com o IBM Quantum Condor

planejado para 2023 - executando 1121 qubits, para ser exato - devemos começar a ver os computadores quânticos começarem a lidar com um número substancial de cálculos genuínos do mundo real, em vez de ficarem restritos a experimentos de laboratório.” [10]

3.4 Dificuldades

Apesar do avanços trazidos anteriormente, existem quatro principais dificuldades que impossibilitam o desenvolvimento e a escala de computadores quânticos no cenário atual: a qualidade do Qubit; implementação de algoritmos de correção de erros; o controle dos Qubits; e a grande quantidade de fios presentes na arquitetura. Em relação à qualidade dos Qubits, a comunidade de estudiosos da computação quântica ainda não conseguiu chegar a um Qubit capaz de gerar instruções úteis em grande escala. Atualmente os poucos qubits nos computadores quânticos baseados em nuvem não são bons o suficiente para sistemas de grande escala,gerando erros ao executar operações entre dois qubits a uma taxa muito maior do que o necessário para se calcular com eficácia. Assim, após um certo número de instruções ou operações, os qubits atuaisproduzem uma resposta errada quando executamos cálculos, tendo a possibilidade do resultado obtido ser indistinguível do ruído. Uma vez que os qubits não são bons o suficiente para a escala em que precisamos que eles operem, precisamos implementar algoritmos de correção de erros que verifiquem e corrijam os erros de qubit aleatórios à medida que ocorrem. Esses são conjuntos de instruções complexos que utilizam muitos qubits físicos para estender efetivamente o tempo de vida das informações no sistema. A correção de erros ainda não foi comprovada em escala para a computação quântica, mas é uma área prioritária de pesquisa, a qual é considerada pré-requisito para um sistema quântico comercial em escala real. Controle de Qubit: Para implementar algoritmos complexos, incluindo esquemas de correção de erros, é necessário provar a possibilidade controlar vários qubits. Esse controle deve ter baixa latência - da ordem de 10 de nanossegundos e deve vir de circuitos de controle de

feedback adaptativo baseados em CMOS[13]. Este é um argumento semelhante ao apresentado no artigo do IEEE Spectrum , “The Case Against Quantum Computing”. No entanto, embora seja assustador, existem motivos para acreditar que não é impossível. Por fim, precisamos abordar o “fan-out” - ou como aumentar o número de qubits em um chip quântico. Hoje, exigimos vários fios de controle, ou vários lasers, para criar cada qubit. É difícil acreditar a viabilidade de construir um chip de um milhão de qubit com muitos milhões de fios conectando-se à placa de circuito ou saindo da câmara de medição criogênica. Na verdade, a indústria de semicondutores reconheceu esse problema em meados da década de 1960 e o chamou de Regra de Rent.

Além das dificuldades de implementações e apesar das diversas iniciativas, o maior problema repousa em como se provar que a computação quântica foi alcançada, pois para isso, um problema que poderia ser resolvido em 50 bilhoes de anos de fato precisa ser solucionado. Complexidade computacional N e NP

3.5 Impactos

Avançando alguns anos para o futuro, nos situando em um tempo na qual os problemas apresentados anteriormente já foram e resolvidos, tendo um computador quântico funcional, podemos imaginar os impactos que acompanharão essa nova tecnologia. Com a capacidade de “multitarefas” da computação quântica, diversos problemas computacionais cuja complexidade é NP poderão ser resolvidos em tempo hábil, o quê poderá se tornar um problema para áreas que fazem uso desta característica como artifício de trabalho, como a criptografia, temática abordada no capítulo 6. Além da criptografia, há diversas outras frentes de trabalho que serão impactadas pela computação quântica e que poderão impactar outras soluções que são empregadas em problemas do cotidiano, dos quais a sociedade faz uso e que tal inovação poderá pôr por terra toda a engenhosidade da solução, aprofundado pelo cápitulo X, que trata sobre a temática de forma mais ampla.

[13]abreviação de "Complementary Metal Oxide Semiconductor". O CMOS é uma pequena área de memória volátil, alimentada por uma bateria, que é usada para gravar as configurações do Setup da placa mãe.

4 Computador

Tendo em vista a metodologia de Project based learning – em que se incentiva a aprendizagem a partir da aplicação de conhecimentos e de habilidades através da experiência – esta pesquisa se propõe a construir um computador clássico (ilustrado a baixo), de forma que se possa por em prática a teoria estudada, possibilitando uma maior compreensão sobre a temática. Assim, o presente capítulo se debruça em detalhar o processo prático da construção do computador, sendo possível implementar as teorias, anteriormente vistas, na prática. Desta maneira, qualquer pessoa interessada, será capaz de construir seu próprio dispositivo, vivenciando o processo de Project based learning.

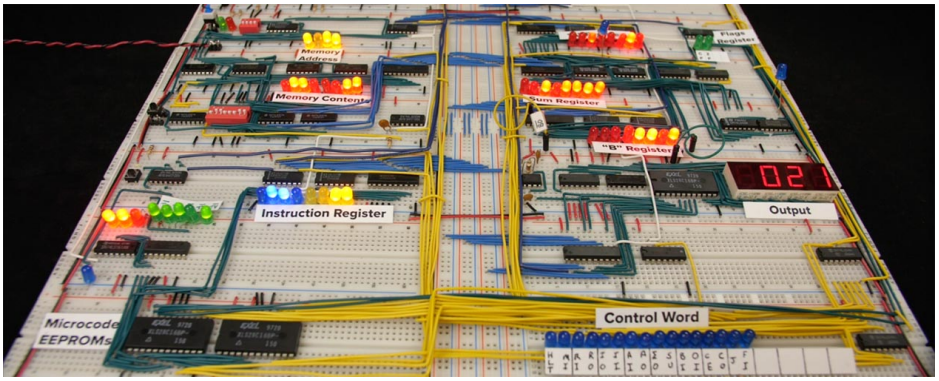

Figura 6: Protótipo a ser implementado, imagem do site eater.net

Ainda, tem-se que a implementação física dos modelos contribuirá para academia, uma vez que os professores poderão usufruir dos mesmos como ferramentas didática, agregando para a compreensão da teoria ao aproximá-la da prática.

No entanto, apesar desse capítulo ter sido reservado para descrever o passo a passo de como desenvolver um computador clássico apenas com portas lógicas simples, devido a alguns problemas esclarecidos na seção 7.2 não foi possível concluí-lo. Porém, de forma a manter a metodologia de Project based Learning, foi desenvolvido um sistema operacional que permitiu colocar em prática o conteúdo estudado ao implementar um processo de boot, usando o assembly intelx86. Além

disso pode-se produzir um emulador de assembly, de forma didática e publicado na web, a fim de contribuir com material para academia. Ademais, sugere-se que a proposta original seja adiada e implementada em um trabalho futuro.

Apesar desta alteração no projeto, optou-se por compartilhar o conteúdo já produzido no início do processo de construção.

4.1 Módulos

Para facilitar a compreensão e também o desenvolvimento do computador, este capítulo será dividido em alguns subcapítulos, cada qual abordando uma parte do computador.

4.1.1 Clock

O clock do computador é uma parte essencial para o seu funcionamento. Este tem a função de sincronizar todas as operações. A ação mais rápida que o computador consegue executar é equivalente a uma vibração do seu clock.

4.1.2 Registers

A maioria das CPUs possuem vários registradores que armazenam pequenas quantidades de dados processados pela CPU. Em nossa CPU de breadboard, criaremos três registradores de 8 bits: A, B e IR. Os registradores A e B são para uso geral. Já o IR (instruction register), apesar de funcionar da mesma forma, é usado para armazenar a instrução atual que está sendo executada.

4.1.3 Arithmetic logic unit (ALU)

A parte da unidade lógica aritmética (ALU) de uma CPU geralmente é capaz de executar várias operações aritméticas, bit a bit e de comparação em números binários. Em nossa CPU de breadboard, a ALU pode apenas adicionar e subtrair. Ele está conectado aos registradores A e B e gera a soma de A + B ou a diferença de A-B.

4.1.4 Random access memory (RAM)

A memória de acesso aleatório (RAM) armazena o programa que o computador está executando, bem como todos os dados que o programa precisa. Nosso computador de breadboard utiliza endereços de 4 bits, o que significa que ele terá apenas 16 bytes de RAM, limitando o tamanho e a complexidade dos programas que poderá executar.

4.1.5 Program counter

O contador do programa (Program counter) conta em binário para acompanhar a instrução que o computador está executando no momento.

4.1.6 Output register

O registrador de saída é semelhante a qualquer outro registrador (como os registradores A e B), exceto que, em vez de exibir seu conteúdo em binário em 8 LEDs, ele exibe seu conteúdo de forma decimal em um display de 7 segmentos, o que requer uma lógica complexa.

4.1.7 CPU control logic

A lógica de controle é o coração da CPU, é o que define os códigos de operação (opcode) que o processador reconhece e o que acontece quando ele executa cada instrução.

5 Simulador

A partir dos conceitos estudados sobre computação quântica, o atual capítulo irá acompanhar a elaboração de uma aplicação web que simula os principais conceitos da física quântica aplicada à computação. Ao final espera-se que o simulador seja publicado na internet, de forma permanente e gratuita, com a intenção de agregar na área do estudo, possibilitando uma melhor compreensão por ser uma ferramenta visual e intuitiva.

6 Criptografia na computação

Dentre as mudanças e impactos possíveis para a computação, a computação quântica possivelmente irá impactar diretamente a área de criptografia. Isso porque, os métodos criptográficos até então existentes podem ser quebrados através de estratégias de força bruta (que são algoritmos que buscam solução por tentativa e erro). Portanto, se o computador quântico pode executar diversas tarefas de forma exponencial, então, tais métodos criptográficos atuais se tornam obsoletos, o que gera um impacto direto na sociedade digital.

Todas as transações bancárias, correspondência civil e militar, diagnósticos médicos, votos em democracias, todas as formas de interação social que demandam alguma política de privacidade (por menor que sejam) serão afetadas. Portanto a computação quântica pode acirrar mais ainda o paradigma das questões privadas e sociais.

Esse capítulo apresenta uma introdução à área de criptografia, a fim de exemplificar em que premissas toda área repousa, por isso, será apresentada a estratégia de criptografia utilizando chaves pública-privada.

Conforme definido por Bruce Schneier *"The art and science of keeping messages secure is cryptography [...]."* [15] Embora a criptografia seja considerada fundamental em nossas vidas digitais, ela não está especificamente relacionada à computação. A criptografia existe há milênios.

Na segurança cibernética, há uma série de preocupações quando se trata de dados. Isso inclui confidencialidade, integridade, disponibilidade e não repúdio.

A confidencialidade significa que nossos dados não podem ser acessados / lidos por usuários não autorizados.

A integridade do dado diz respeito à originalidade com a qual os dados chegam a nós, estando 100% intactos, sem terem sido modificados – seja por um ato malicioso, perda de dados ou por algum outro fator.

A disponibilidade se refere à acessibilidade dos dados quando necessário.

Examinaremos as várias formas de criptografia digital e como elas podem nos ajudar a alcançar os três objetivos listados acima. Quando falamos de criptografia digital, geralmente nos referimos aos conceitos abaixo, que serão especificados.

1. Criptografia simétrica
2. Criptografia assimétrica
3. Funções de hash

Temos receio que os computadores quânticos sejam capazes de decifrar certos códigos usados para enviar mensagens seguras. Esses códigos criptografam dados usando funções matemáticas de “trapdoor” que funcionam facilmente em uma direção, mas não em outra. Isso facilita a criptografia de dados, mas a decodificação é extremamente difícil sem a ajuda de uma chave especial.

Esses sistemas de criptografia não são inquebráveis e sua segurança se baseia na enorme quantidade de tempo que um computador clássico levaria para fazer o trabalho. Os métodos modernos de criptografia são projetados para que a decodificação demore tanto tempo que se tornem praticamente inquebráveis.

A chegada dos computadores quânticos mudou esse pensamento. Essas máquinas são muito mais poderosas que os computadores clássicos, tornando possível o rompimento desses códigos com facilidade, já que realizam contas matemáticas de forma mais eficiente e significativamente mais rápida que os computadores clássicos.

6.1 Conceitos básicos de criptografia

Antes de mergulharmos no conceito de criptografia, precisamos saber o que exatamente ela significa. Criptografar uma mensagem significa torná-la ilegível para partes não autorizadas usando uma cifra (o método específico para fazer isso). Descriptografar a mensagem significa reverter o processo e tornar os dados legíveis, decifrando-os mais uma vez.

6.1.1 Criptografia simétrica

Para criptografar e descriptografar corretamente nossos dados, precisamos de uma chave (que determina a saída da nossa cifra). Na criptografia simétrica, a chave usada para criptografar e decifrar os dados é a mesma, o que pode constituir

em um problema de segurança. De forma a compreender este problema, pode-se refletir sobre como enviar dados com segurança em um ambiente hostil como a Internet. Nesse caso, se a mesma chave for usada para criptografar e decifrar dados, primeiro precisaríamos enviar a chave de descriptografia para estabelecer uma conexão segura. Porém, isso significa que estaríamos enviando a chave por uma conexão insegura, e assim ela poderia ser interceptada e usada por terceiros, se constituindo em um grande problema de segurança.

6.1.2 Criptografia assimétrica

A criptografia assimétrica surge para resolver o problema da simétrica, possibilitando uma forma segura de compartilhamento de dados.

Para exemplificar esse novo formato, usaremos um cadeado que possui três estados – A (bloqueado), B (desbloqueado) e C (bloqueado) – E duas chaves distintas – A primeira pode girar apenas no sentido horário (de A a B a C) e a segunda pode girar apenas no sentido anti-horário (de C a B a A).

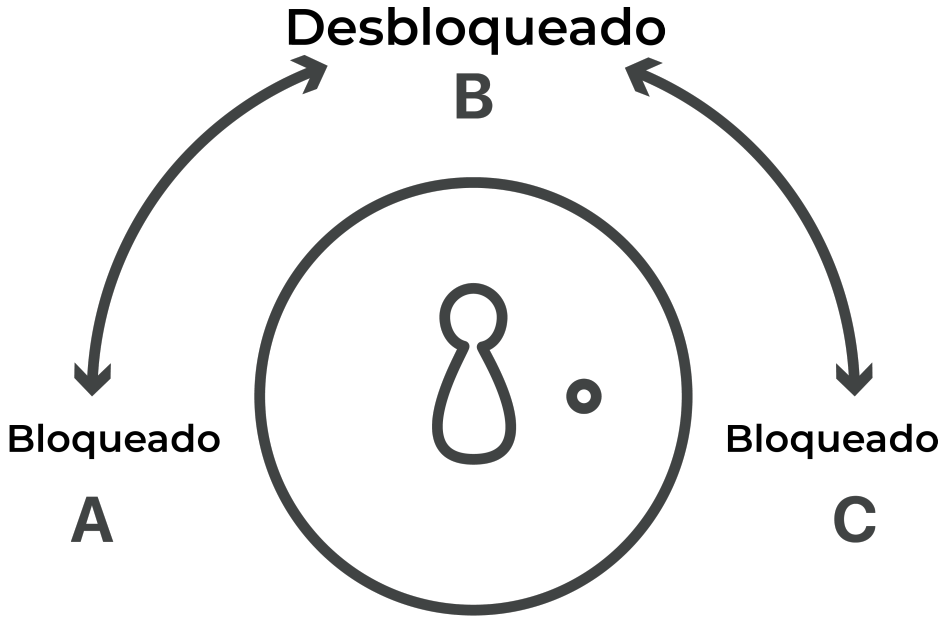

Figura 7: Ilustração da posição das chaves

Ao criptografar uma mensagem, o usuário pega a primeira chave e guarda para si mesmo. Essa chave é sua chave “privada” - porque apenas ele a possui.

A segunda chave é sua chave “pública”: Pode ser distribuída para qualquer pessoa. Assim, o usuário tem sua chave privada que pode mudar de A para B para C. E todo os outros tem sua chave pública que pode mudar de C para B para A.

Colocando isso em prática, imagine que você queira enviar um documento privado para o usuário. Você coloca o documento na caixa e usa uma cópia da chave pública dele para bloqueá-lo. Lembre-se de que a chave pública dele gira apenas no sentido anti-horário, e você a coloca na posição A. Agora a caixa está bloqueada. A única chave que pode passar de A para B é a chave privada, a que ele guardou para si.

6.1.3 Funções de hash

Uma função de hash, diferentemente da criptografia simétrica / assimétrica, é uma função unidirecional. Você pode criar um hash a partir de alguns dados, mas não há como reverter o processo. Portanto não sendo considerado uma maneira útil de armazenar dados, mas sim para verificar a integridade dos mesmos. Uma função de hash recebe alguns dados como entrada e gera uma string aparentemente aleatória (mas nem tanto) que sempre terá o mesmo comprimento. Uma função de hash ideal cria valores exclusivos para diferentes entradas. A mesma entrada exata sempre produzirá exatamente o mesmo hash - e é por isso que podemos usá-la para verificar a integridade dos dados.

6.2 Criando chave publica e privada RSA – Rivest Shamir Adleman

De forma a aprofundar o conceito de criptografia assimétrica, foi criado um programa em linguagem C que utiliza o algoritmo RSA[14]. O programa tem como objetivo apresentar de forma didática e visual o processo da execução do algoritmo

[14] Principal algoritmo de criptografia usado até os dias atuais, criado por Ron Rivest, Adi Shamire Leonard Adleman em 1977.

RSA, gerando um par de chaves – pública e privada – e realizando o processo de criptografar e decifrar uma mensagem. A tabela 3 a baixo ilustra esse processo.

RSA – Cryptosystem	
Descrição	matemática
Escolher dois números primos	$p = 2$ $q = 7$
Produto dos números escolhidos	$n = 14$
função Phi $\Phi(n) = (p-1)(q-1)$	$\Phi(14) = (7-1)(2-1)$ $\Phi(14) = (6)(1)$ $\Phi(14) = 6$ $Coprimos\ de\ 14:\ 1, 3, 5, 9, 11, 13$
Escolher o numero e 'encryption' • $1 < e < \Phi(n)$ • $coprimo\ de: n\ e\ \Phi(n)$	$Coprimos\ de\ 14:\ 1, 3, 5, 9, 11, 13$ $Coprimos\ de\ 6:\ 1, 5$ $Menor\ que\ 6\ e\ coprimo\ de\ 14\ e\ de\ 6: 5$ $e = 5$
Chave publica	$(5, 14)$

Escolher o numero d 'decryption' • $d * e(mod\Phi(n)) = 1$	$d * 5(mod6) = 1$ $5 * d = 5, 10, 15, 20, 25 \ldots 55$ $5 * 1(mod6) = 5$ $5 * 2(mod6) = 4$ $5 * 3(mod6) = 3$ $5 * 4(mod6) = 2$ $5 * 5(mod6) = 1$ $5 * 6(mod6) = 0$ $\ldots$ $5 * 11(mod6) = 1$
Chave privada	$(11, 14)$

Tabela 3: Passo a passo para gerar par de chaves

RSA – Cryptosystem	
Descrição	matemática

Criptografar Chave publica: $(5, 14)$ mensagem: “B” $A \rightarrow 1$ $B \rightarrow 2$ $C \rightarrow 3$ $D \rightarrow 4$	$2^5(mod14)$ $32(mod14)$ $4(mod14)$ Mensagem criptografada: 4 $\rightarrow$ D
Decifrar Chave privada: $(11, 14)$ mensagem: “D” $A \rightarrow 1$ $B \rightarrow 2$ $C \rightarrow 3$ $D \rightarrow 4$	$4^{11}(mod14)$ $4194304(mod14)$ $2(mod14)$ Mensagem original: 2 $\rightarrow$ B

Tabela 4: Como criptografar e decifrar

6.3 Criptografia aplicada computação clássica

6.4 Criptografia aplicada computação quântica

7 Próximos passos

7.1 Cronograma

Etapas	2020											2021	
	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez	jan	fev
Clássica													
Quântico													
Protótipo													
Entrega Parcial													
Comparação													
Web App													
Impacto Social													
Entrega Final													

Tabela 5: Cronograma original

Acima está uma réplica do cronograma originalmente proposto para este projeto. Na tabela a seguir está mesmo cronograma porém com algumas alterações pontuais:

Justificativa de cada alteração:

1. **O projeto não começou em fevereiro de 2020** pois foi aprovado com ressalvas pela banca, assim esse mês foi dedicado ao aprimoramento da proposta.

2. **O projeto foi separado em duas grandes partes** 2020.1 foi voltado a computação clássica, e o plano é que 2020.2 será focada em computação quântica. Essa mudança relevante se dá pelo entendimento de que uma pesquisa de apenas dois meses no ramo da computação se tornaria algo raso e os

principais conceitos não seriam abordados com a devida intensidade.

3. **O não desenvolvimento do protótipo** ocorreu pelo atraso na compra dos materiais.

4. **A inclusão do Sistema Operacional e Simulador 24bits** foi uma forma de manter a metodologia *"Project Based Learning"* para o aprendizado na área de computação clássica sem ter os materiais desejados em mãos.

Etapas	2020											2021	
	fev	mar	abr	mai	jun	jul	ago	set	out	nov	dez	jan	fev
Clássica													
Quântico													
Protótipo													
Entrega Parcial													
Comparação													
Códigos													
Sistema Operacional													
Simulador 24bits													
Criptografia didática													
Impacto Social													
Entrega Final													

Tabela 6: Cronograma atualizado

7.2 Problemas

Devido a pandemia e principalmente ao atraso não previsto de quatro meses da compra dos materiais para o desenvolvimento do protótipo, este, que estava no cronograma de 2020.1 ainda não foi desenvolvido. Em seu lugar outros projetos como um sistema operacional e um simulador de processador moderno foram feitos – dessa forma é proposto que essa parte da pesquisa seja adiada para um possível projeto no futuro. Com isso, deixarei o segundo semestre de 2020 integralmente voltado à computação quântica, pois, entende-se que é um tema de alta complexidade e relevância nos dias atuais.

8 Impactos sociais

O que os computadores quânticos podem fazer? Eles podem facilmente fazer o que parece ser uma tarefa impossível para computadores clássicos. Eles conseguem resolver e quebrar algoritmos de criptografia que protegem os dados de um usuário e a infraestrutura da Internet.

EX: Em outubro de 2019, o Google revelou sua bem-sucedida experiência "Quantum Supremacy" alegando que um problema particularmente difícil foi resolvido em 200 segundos [5].

A criptografia atual é baseada em matemática e decodificação e essa é uma tarefa bastante complicada para computadores clássicos, já os computadores quânticos a executam com grande facilidade.

EX: O algoritmo de fatoração de Shor[15] é de particular importância porque esse algoritmo quântico significa que a criptografia de chave pública pode ser facilmente quebrada caso o computador quântico seja suficientemente grande.

[15] Um algoritmo de computador quântico de tempo polinomial para fatoração de número inteiro. Informalmente, resolve o seguinte problema: Dado um número inteiro, encontre seus fatores primos. Foi inventado em 1994 pelo matemático americano Peter Shor

9 Conclusões

Nesse capítulo serão colocadas as conclusões ao final do projeto.

9.1 Perspectivas

Referências

[1] *A Half-Century Survey on The Universal Turing Machine*, USA, 1988. Oxford University Press, Inc.

[2] Grand Rapids Business Journal – David Van Andel. Project-based learning is the future of education, out. 2019.

[3] Frank Arute. Quantum supremacy using a programmable superconducting processor. *Nature*, 2019.

[4] Fabio Gagliardi Cozman. Turing e complexidade. University Lecture, 2000.

[5] Elizabeth Gibney. Hello quantum world! google publishes landmark quantum supremacy claim. *Nature*, 2019.

[6] G. Ifrah. *The Universal History of Numbers: From Prehistory to the Invention of the Computer*. Wiley, 2000.

[7] Melhoramentos ltda. – Michaels. Computador, out. 2019.

[8] Regina Célia Tamaso Mioto. Procedimentos metodológicos na construção do conhecimento científico: a pesquisa bibliográfica. *Revista Katálysis*, 10(SPE):37–45, 2007.

[9] John von Neumann. First draft of a report on the edvac. Technical report, 1945.

[10] David Nield. Ibm just committed to having a functioning 1,000 qubit quantum computer by 2023. *Science Alert*, 2020.

[11] B. Ram. *Computer Fundamentals: Architecture and Organization*. New Age International, 2000.

[12] Charles Riley. Google claims its quantum computer can do the impossible in 200 seconds., out. 2019.

[13] Humberto Rodrigo Sandmann. Ambiente de produção. personal website, 2019.

[14] Bruno Santana. Iphone 6 é 120 milhões de vezes mais poderoso que o computador de bordo da apollo 11, jul. 2019.

[15] Bruce Schneier and Phil Sutherland. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, Inc., USA, 2nd edition, 1995.

[16] A. M. Turing, Michael Woodger, B. E. Carpenter, and R. W. Doran. *A. M. Turing's ACE Report of 1946 and Other Papers*. The MIT Press, Cambridge, Mass. : Los Angeles, April 1986.

[17] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2:230–265, 1936.

[18] Yuanhao Wang. 16-qubit ibm universal quantum computer can be fully entangled. *Nature*, 2018.

[19] Paul Wazlawick. *História da computação*. Elsevier, Rio de Janeiro, RJ, 2016.