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

05 de maio de 2020

Gabriel R. Zsigmond

COMPUTADORES CLÁSSICOS E QUÂNTICOS:

ESTUDOS, IMPLEMENTAÇÕES E IMPACTOS SOCIAIS

Projeto de Iniciação Científica para a Escola Superior de Propaganda e Marketing, sob a orientação do Professor Doutor Humberto Sandmann.

Orientador: Prof. Dr. Humberto Sandmann

Brasil

05 de maio de 2020

ESTUDOS, IMPLEMENTAÇÕES E IMPACTOS SOCIAIS:

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 toda uma arquitetura na forma de 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. Também, esse projeto 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

1 Introdução

1.1 Definição justificada 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 leva hoje, sendo utilizada, até a década de 1940, como nome da profissão de alguém que calcula, segundo o dicionário Michaelis: “Aquele ou aquilo que calcula baseado em valores digitais; calculador, calculista”. [4]

Tendo em vista o antigo significado da palavra “computador”, pode-se questionar sobre como passamos a utilizar de uma palavra usada para se referir à pessoas, para mera maquinas. A fim de responder essa pergunta, recuperaremos a origem dos computadores. Pode parecer um questionamento simplista que não precisa ser respondido, porém, é uma pergunta da qual muitas pessoas não sabem a verdadeira resposta. Computadores existem há muito mais tempo que o transistor – dispositivo semicondutor usado para amplificar ou alternar sinais eletrônicos e eletricidade – na forma mecânica e teórica. A definição real de um computador foi elaborada por Alan Turing (Reino Unido, 1912-1954), um matemático, lógico, criptógrafo e herói de guerra que se preocupava exatamente com a questão relacionada ao que era computável e o que não era. Ele foi responsável por elaborar a definição do computador, descrevendo a Maquina de Turing, trabalho publicado em 1937 que deu origem aos computadores e celulares que você, leitor, pode estar usando para ler o presente trabalho.

A Maquina de Turing, considerada o modelo mais poderoso de computador, é similar a um automato finito [1], porém com uma mémoria ilimitada e irrestrita, constituindo um modelo mais exato de um computador de forma geral. Esta é composta por três principais componentes: fita infinita; processador; máquina de estado finito.

A fita infinita é dividida em células, cada uma contendo um simbolo de um alfabeto

[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.

finito. O processador é responsável por se deslocar para a direita ou para a esquerda e efetuar a leitura ou escrita em uma célula. Assim, o Prof. Dr. Fabio Gagliardi Cozman [3], ilustra seu funcionamento da seguinte forma:

1. Inicialmente a fita contém somente a cadeia de entrada, disposta no “meio” [2] da fita , com o processador posicionado no início da cadeia (o resto está em branco);

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

3. O processador 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;

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

O primeiro computador digital eletrônico de grande escala, foi criado em fevereiro de 1946 por cientistas norte-americanos, John Presper Eckert e John W. Mauchly, da Electronic Control Company . No final de sua operação em 1956, o ENIAC (Electrical Numerical Integrator and Calculator), 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 de tamanho, ocupava 167 m2 e consumia 150 kW de eletricidade. [13]

A partir do ENIAC, as possibiliades tecnologicas tomaram uma nova proporção. Em 1969, apenas 13 anos após o desligamento do primeiro computador digital eletrônico, o computador de bordo da Apollo 11 – missão que levou o homem à Lua – tinha 32.768 bits [3] de RAM, o suficiente para armazenar apenas um texto não formatado, com

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

[3] Para uma explicação elaborada sobre bits refira-se ao cápitilo 2 pagina 8

cerca de 2.000 palavras. O que contrasta com o Iphone XS com 4gb de RAM, que em 2018, tinha cerca de 1 milhão de vezes mais de memória que o Apollo Guinche Computer. [9]

(falar da Spacex e falcon9 e CRS-12 Dragon 2020)

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.

Levando em consideração o rápido avanço e desenvolvimento computacional, mencionados anteriormente, entende-se que os computadores agregam à sociedade, seja facilitando a comunicação e o compartilhamento de conhecimentos, como em outros aspectos. No entanto, a agilidade pela qual se deu tais **transformações da tecnologia da computação**, também gera grandes expectativas e incertezas sobre o que ainda está por vir, tanto nas questões de mudanças tecnológicas quanto nos impactos na sociedade.

Tendo em vista a incerteza que acompanha o futuro da computação e de seus próximos avanços, a presente pesquisa se propôs 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 um ilustração clara do funcionamento. Junto a isso será desenvolvida uma aplicação web que ilustre 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.

[4] A tecnologia em questão não somente pode ser usada como uma peça de roupa ou um acessório, 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.

1.2 Metodologia a ser empregada

Para a realização da pesquisa de iniciação científica, é indispensável o uso de pesquisa bibliográfica, afim de recuperar e se aprofundar no conhecimento cientifico. 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 [5]. 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 entendimento da obra e a construção do projeto 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 O 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]. Assim, entende-se que esta é a melhor metodologia para desenvolver um protótipo físico de um computador e programar um site.

2 Computação Clássica

Entende-se a importância de se compreender a origem e o desenvolvimento da computação clássica para o desenrolar da pesquisa, o que será apresentado em meio a este capítulo.

A computação clássica consiste em computadores que dependem da física clássica para operar. Estes são os computadores tradicionais que usamos em nosso dia-a-dia – seja 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 simbolo 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 simbolo “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. [7] É com base nos bits [6] 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:

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

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

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

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

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

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 mantem verdadeira para qualquer base númerica.

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

Ao decorrer do texto serão referidos números em base 2, 10 e 16.

2.1 A computação clássica e sua evolução

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” [12] em 12 de novembro de 1937, esse formaria a teoria básica da computabilidade por várias décadas.

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. O cientista forneceu a formalização dos conceitos de “algoritmo” e “computação” na infância da ciência da computação. Apesar de sua simplicidade, uma Máquina de Turing pode ser adaptada para simular a lógica de qualquer algoritmo de computador e é útil para explicar as funções de uma CPU.

Apesar dos computadores clássicos serem atualmente descritos como tendo a chamada arquitetura de Von Neumann, acredita-se que o idealizador dessa arquitetura partiu dos modelos matemáticos desenvolvidos por Alan Turing, que incluia o conceito de programa armazenado, originado a partir da construção da Maquina de Turing. Conceito também encontrado no projeto EDVAC de Von Neumann [6], que possibilita o armazenamento de instruções e dados na mesma memória, permitindo a manipulação de programas como dados, que se constitui como característica definidora do computador clássico. Assim, pode-se alegar que Turing é o pai do computador já que suas publicações antecedem a arquitetura de Von Neumann, além de existirem razões para se acreditar que este último conhecia os resultados do trabalho de Alan Turing [1], que podem ter inspirado as suas criações. Outro fator que corrobora para este entendimento, é 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 ele-

tromecânico 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." [11]

Por fim, conclui-se que baseado na teoria da máquina de Turing, o físico e matemático John von Neumann desenvolveu uma arquitetura que era capaz de executar as seguintes tarefas.

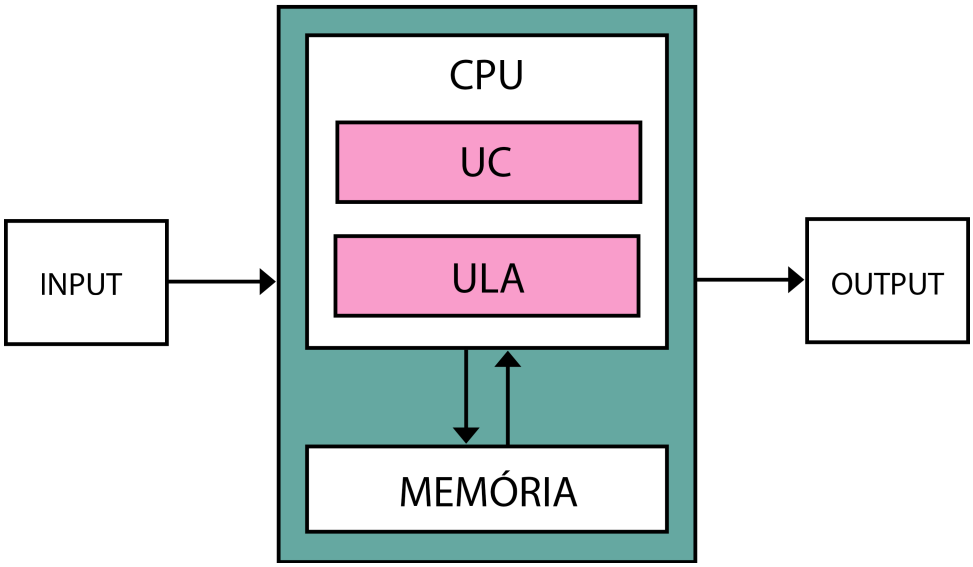

Figura 1: Diagrama arquitetura de Von Neumann

Nessa arquitetura, o cabeçalho passa a ser uma CPU (Central Processing Unit), a

fita se transforma em memória RAM e as operações são construidas e executadas em circuitos formados por portas lógicas chamadas de ALU (Arithmetic/Logic Unit). [8]

Atualmente, a maioria dos computadores modernos são construídos sobre a praxis da arquitetura von Neumann. A fim de simular um processador moderno de forma didática, foi elaborado o ASM 24bits que apresenta as principais características de um processador moderno e permite a sua programação utilizando um Assembly [7] de 20 instruções, sua Documentação. Similar à arquitetura de von Neumann esse emulador é composto por memória e CPU, essa CPU simples possui 2 registradores, eles são :

Registradores	
Nome	Descrição
Accumulator (ACC)	Este é o registro mais usado para armazenar dados extraídos da memória. Está em diferentes números em diferentes microprocessadores.
Instruction Register (IR)	É o registro que contém a instrução que está sendo executada atualmente.

Uma unidade lógica aritmética (ALU) é um circuito digital usado para realizar operações aritméticas e lógicas. Ele representa o bloco de construção fundamental da unidade central de processamento CPU. A ULA do ASM 24bits possui 20 instruções, sendo elas:

ULA

[7] 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.

Mnemônico	Descrição
nop	Slot de memória vazio
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 executará 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 executará 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	Carregue uma constante diretamente no acumulador. Recebe uma constante em notação hexadecimal 0x00F2 por exemplo.
add	Adicione 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	Subtraia 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	Multiplique 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	Armazene 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.

Nesse momento é importante adentrar o tema de o que é a Linguagem Assembly e como funciona. Como brevemente mencionado acima, esta é uma linguagem de computação altamente dependente do hardware que esta sendo utilizado. A linguagem assembly, frequentemente abreviada como asm, é qualquer linguagem de programação de baixo nível em que há uma correspondência muito forte entre as instruções na linguagem e as instruções do código de máquina [8] da arquitetura. Como

[8]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

a montagem depende das instruções do código de máquina, cada “assembler” possui sua própria linguagem de maquina, projetada para unicamente para uma arquitetura de computador específica. Nesse projeto a arquitetura usada é a Intel x86.

Ao ligar qualquer computador clássico da atualidade, ele passara por algumas etapas importantes antes de chagar ao seu esta ‘usável’ para o seu usuário final.

Quando reinicializamos nosso computador, ele deve iniciar novamente, inicialmente sem qualquer noção de um sistema operacional. De alguma forma, ele deve carregar o sistema operacional – qualquer variante que possa ser – a partir de algum dispositivo de armazenamento permanente que está atualmente conectado ao computador (um cd, um disco rígido, um USB, etc.). O ambiente pré-SO oferece poucos serviços ricos: neste estágio, mesmo um sistema de arquivos simples seria um luxo (ler e gravar arquivos lógicos em um disco), mas não temos nada disso. Felizmente, o que temos é 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 o BIOS conclui alguns testes de baixo nível do hardware, principalmente se a memória instalada está funcionando corretamente ou não, ele deve inicializar o sistema operacional armazenado em um de seus dispositivos. Aqui, somos lembrados, entretanto, que o BIOS não pode simplesmente carregar um arquivo que representa seu sistema operacional a partir de um disco, já que o BIOS não tem noção de um sistema de arquivos. O BIOS deve ler setores específicos de dados (geralmente 512 bytes de tamanho) de localizações físicas específicas dos dispositivos de disco.

Já vimos alguns exemplos de hexadecimal, então é importante entender por que hexadecimal é frequentemente usado na programação de nível inferior.

Em primeiro lugar, pode ser útil considerar por que contar em dez parece tão natural para nós, porque quando vemos hexadecimal pela primeira vez sempre nos perguntamos: por que não simplesmente contar até dez? Não sendo um especialista

ou memória da CPU.

no assunto, assumirei que contar até dez tem algo a ver com a maioria das pessoas ter um total de dez dedos em suas mãos, o que levou às idéias de números sendo representados como 10 símbolos distintos: 0,1, 2, ... 8,9

O decimal tem uma base de dez (ou seja, tem dez símbolos de dígitos distintos), mas o hexadecimal tem uma base de 16, então temos que inventar alguns novos símbolos numéricos; e a maneira mais preguiçosa é apenas usar algumas letras, nos dando: 0,1,2, ... 8,9, a, b, c, d, e, f, onde o único dígito, por exemplo, representa uma contagem de 13.

Para distinguir entre hexadecimal e outros sistemas numéricos, costumamos usar o prefixo0x, ou às vezes o sufixo, que é especialmente importante para dígitos hexadecimais que podem não conter nenhum dos dígitos da letra, por exemplo: 0x50não é igual (decimal) 50 — 0x50é realmente 80 em decimal.

A questão é que um computador representa um número como uma sequência de bits (dígitos binários), uma vez que fundamentalmente seu circuito pode distinguir apenas entre dois estados elétricos: 0 e 1 — 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 contra-mais de 9 tendo dois ou mais dígitos (por exemplo, 456, 23, etc.).

Nomes foram adotados para séries de bits de certos comprimentos para facilitar a conversa e concordar sobre o tamanho dos números com os quais estamos lidando. As instruções da maioria dos computadores lidam com um mínimo de valores de 8 bits, que são denominados bytes. Outros agrupamentos são short, int elong, que geralmente representam valores de 16 bits, 32 bits e 64 bits, respectivamente. Também vemos o termo palavra, 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 palavra se refere a um valor de 16 bits; no modo protegido de 32 bits, uma palavra se refere a um valor de 32 bits e assim por diante.

Portanto, voltando ao benefício do hexadecimal: as sequências de bits são bastante demoradas para serem escritas, mas são muito mais fáceis de converter de e

para a notação hexadecimal abreviada do que de e para nosso sistema decimal natural, essencialmente porque podemos quebrar a conversão em menores , Segmentos de 4 bits do número binário, em vez de tentar somar todos os bits componentes em um total geral, o que fica muito mais difícil para cadeias de bits maiores (por exemplo, 16, 32, 64, etc.).

Para o melhor intendimento de computadores clássicos, foi desenvolvido um sistema operacional simples que passa por todas as etapas essências do processo de “boot” do state of art de computadores clássicos, liga em ‘16 bit mode’, transfere para ‘32 bit protected mode’ e ‘gerencia interrupções’

3 Computação Quântica

3.1 A computação quântica e sua evolução

4 Computador

Ressalva: *Esse cápitulo foi reservado para descrever o passo a passo de como desenvolver um computador clássico apenas com portas logicas. Devido ao fato que as compras solicitadas para o desenvolvimento dessa parte do projeto atrasou um tempo bastante significante e devido a pandemia gerada pelo COVID-19 eu não estou morando em São Paulo. Seguindo o cronograma a montagem seria feita no decorrer dos primeiros meses do projeto, tendo em vista que o protótipo ainda não foi desenvolvido e a segundo metade do cronograma é inteiramente voltada a computação quântica devida a sua relevância ao projeto e complexidade, é proposto que essa parte seja adiada para um possível futuro trabalho.*

Construir um computador parece uma tarefa complicada e ousada. Porém, uma CPU [9] é bastante simples em operação depois que os fundamentos por trás de todos os seus processos são compreendidos. Desta maneira, este capítulo destina-se em explicar o passo a passo para que qualquer pessoa interessada, seja capaz de construir seu próprio computador e obter o conhecimento que acompanha o processo.

4.1 Módulos

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

[9] CPU é a sigla para Central Process Unit, ou Unidade Central de Processamento. É o principal item de hardware do computador, que também é conhecido como processador, essa é a parte responsável por calcular e realizar tarefas determinadas pelo usuário.

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 qual instrução 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.

4.1.8 Materiais Necessários

5 Simulador

6 Criptografia

Conforme definido por Bruce Schneier *"The art and science of keeping messages secure is cryptography [...]."* [10] Embora a criptografia é considerada fundamental em nossas vidas digitais, não está especificamente relacionada à computação. Ele existe em diversas formas há milênios.

Na segurança cibernética, há uma série de coisas com as quais nos preocupamos quando se trata de dados. Isso inclui confidencialidade, integridade, disponibilidade e não repúdio.

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

Integridade significa que nossos dados chegam a nós 100% intactos e não foram modificados, seja por um ator malicioso, perda de dados ou outros.

Disponibilidade significa que nossos dados estão acessíveis quando necessário.

Não-repúdio significa que, se um sujeito enviar alguns dados para outro, ele não poderá alegar mais tarde que não era, de fato, o remetente dessa informação. Em outras palavras, existe uma maneira de determinar que ninguém além de ele próprio poderia ter enviado os dados.

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 a um dos seguintes:

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

Esses conceitos serão explicados e exemplificados na próxima secção.

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

Esses sistemas de criptografia nunca foram inquebráveis, 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 especificamente para que a decodificação demore tanto tempo que são praticamente inquebráveis.

Mas os computadores quânticos mudam esse pensamento. Essas máquinas são muito mais poderosas que os computadores clássicos e devem poder quebrar esses códigos com facilidade.

6.1 Conceitos básicos de criptografia

Antes de mergulharmos nisso: o que exatamente queremos dizer com ”criptografia”? Criptografar e descriptografar são normalmente usadas para significar criptografia e decifração, respectivamente; Para simplificar, 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 mais uma vez.

6.1.1 Criptografia simétrica

Para criptografar e descriptografar corretamente nossos dados, precisamos dos dados e de uma chave (que determina a saída da nossa cifra). Com a criptografia simétrica, a chave usada para criptografar e descriptografar dados é a mesma.

6.1.2 Criptografia assimétrica

O problema da criptografia simétrica é o seguinte: E se eu precisar enviar dados com segurança em um ambiente hostil, como a Internet? Se a mesma chave for usada para criptografar e descriptografar dados, primeiro eu precisaria enviar a chave de descriptografia para estabelecer uma conexão segura. Mas isso significa que estou enviando a chave por uma conexão insegura, o que significa que a chave pode ser interceptada e usada por terceiros! Como contornar isso?

Para exemplificar, usaremos um cadeado que possui três estados: A (bloqueado), B (desbloqueado) e C (bloqueado). E tem duas chaves distintas. A primeiro 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).

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

A segunda chave, sua chave “pública”: Pode ser distribuida 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 pratica, imagine que você queira enviar um documento privado para o usuario. 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, diferente 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. Como tal, não é uma maneira útil de armazenar dados, mas é uma maneira útil de verificar a integridade de alguns dados. 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 Rivest Shamir Adleman – RSA Criando chave publica e privada

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	$(11, 14)$

RSA – Cryptosystem	
Descrição	matemática
Criptografar Chave publica: $(5, 14)$ mensagem: “B” $A \to 1$ $B \to 2$ $C \to 3$ $D \to 4$	$2^5 (mod 14)$ $32 (mod 14)$ $4 (mod 14)$ Mensagem criptografada: 4 $\to$ D
Descriptografar Chave privada: $(11, 14)$ mensagem: “D” $A \to 1$ $B \to 2$ $C \to 3$ $D \to 4$	$4^{11} (mod 14)$ $4194304 (mod 14)$ $2 (mod 14)$ Mensagem original: 2 $\to$ B

6.3 Criptografia aplicada computação clássica

6.4 Criptografia aplicada computação quântica

7 Próximos passos

7.1 Plano de Redação

7.2 Cronograma

7.3 Problemas

Com o acontecimento de alguns imprevistos com a pandemia e atraso das compras o prototipo que estava no cronograma de 2020.1 ainda não foi desenvolvido, assim é proposto que essa parte da pesquisa seja adiada para um possível projeto no futuro.

8 Impactos sociais

Computadores quânticos podem facilmente fazer o que parece ser uma tarefa impossível para computadores clássicos. O que os computadores quânticos podem fazer? Esses computadores podem resolver e quebrar algoritmos de criptografia que protegem os dados de um usuário e a infraestrutura da Internet. Em outubro de 2019, o Google revelou sua bem-sucedida experiência “Quantum Supremacy” alegando que um problema particularmente difícil foi resolvido em 200 segundos.

Embora a maior parte da criptografia atual seja matemática e decodificação, é uma tarefa bastante complicada para computadores clássicos, computadores quânticos executam essa tarefa com grande facilidade. O algoritmo de fatoração de Shor [10]

[10] 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

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.

9 Conclusões

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

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

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

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

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

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

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

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

[11] 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.

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

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