Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Um sistema computacional completo sobre uma máquina de instrução única implementada em FPGA

Alexandre Silva Dantas
Matheus Costa de Sousa Carvalho Pimenta

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcus Vinicius Lamar

Coorientador
Prof. Dr. Diego de Freitas Aranha

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Coordenador

Banca examinadora composta por:

Prof. Dr. Marcus Vinicius Lamar (Orientador) — CIC/UnB
Prof. Dr. Professor I — CIC/UnB
Prof. Dr. Professor II — CIC/UnB

CIP — Catalogação Internacional na Publicação

Dantas, Alexandre Silva.

Um sistema computacional completo sobre uma máquina de instrução única implementada em FPGA / Alexandre Silva Dantas, Matheus Costa de Sousa Carvalho Pimenta. Brasília : UnB, 2014.
35 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2014.

1. palvrachave1, 2. palvrachave2, 3. palvrachave3

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Um sistema computacional completo sobre uma máquina de instrução única implementada em FPGA

Alexandre Silva Dantas
Matheus Costa de Sousa Carvalho Pimenta

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcus Vinicius Lamar (Orientador)
CIC/UnB

Prof. Dr. Professor I
CIC/UnB

Prof. Dr. Professor II
CIC/UnB

Prof. Dr. Coordenador
Coordenador do Bacharelado em Ciência da Computação

Brasília, 30 de março de 2014

Dedicatória

Dedico a....**mamãe**

Agradecimentos

Agradeço a....*papai*

Abstract

A ciência...

Palavras-chave: palvrachave1, palvrachave2, palvrachave3

Abstract

The science...

Keywords: keyword1, keyword2, keyword3

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com a necessidade humana de se comunicar à distância, a engenharia deu luz às Telecomunicações (uma ref. aqui). Com esta novidade, é possível tanto que pais e filhos se comuniquem estando em cidades distintas, quanto estratégias de guerra sejam elaboradas em conjunto por países de continentes diferentes. Comum em ambas as situações, é o fato de que as duas pontas da comunicação desejam privacidade. Isto é, pais e filhos não querem que seus vizinhos tomem conhecimento das mensagens que trocam. Tampouco, países aliados pretendem que suas estratégias falhem por vazamento de informação.

Para tornar possível o sigilo na troca de mensagens à distância, estudos são realizados na área que hoje chamamos de Segurança da Informação (uma ref. aqui). Diversas técnicas são desenvolvidas nesta área até hoje, para tentar garantir que um par de comunicação possa trocar informações sem que estas chegem ao conhecimento de adversários. Entre estas técnicas, as mais conhecidas e utilizadas nasceram da Criptografia (uma ref. aqui).

A Criptografia estuda maneiras de criar uma versão ilegível de uma determinada mensagem, de modo que adversários com acesso ao canal inseguro pelo qual a mensagem será transmitida, por exemplo a Internet (uma ref. aqui), não tenham acesso à informação contida na mensagem, e de modo que somente o destinatário seja capaz de reverter este processo, que chamamos de cifragem. A Criptografia estuda também maneiras de autenticar uma fonte, isto é, um destinatário que recebe uma mensagem deve poder estar seguro de que esta foi de fato enviada pelo remetente do qual este destinatário espera receber esta mensagem.

Atualmente, os sistemas criptográficos mais empregados são os sistemas assimétricos (uma ref. aqui). Nestes sistemas, cada ponta da comunicação possui um par do que chamamos de chaves criptográficas. Uma chave criptográfica pode ser, por exemplo, uma frase. Os pares de chaves criptográficas são utilizados para cifrar e decifrar mensagens através de algoritmos criptográficos. Um algoritmo de criptografia assimétrica é uma sequência de passos que utiliza uma mensagem e uma chave de um par de chaves criptográficas para produzir algo que chamamos de criptograma, uma versão ilegível da mensagem original. Para reconstruir a mensagem original, utiliza-se uma sequência de passos de volta do algoritmo criptográfico, que utiliza o criptograma gerado anteriormente e a outra chave do par de chaves criptográficas. Sistemas criptográficos assimétricos utilizam pares de chaves, para que uma das chaves de alguém que se comunica seja pública, ou seja, conhecida por todos os que se comunicam, enquanto a outra chave do par deve ser privada, ou seja, somente este alguém que se comunica conhece sua chave privada.

Deste modo, é possível trocar mensagens de maneira segura e simultaneamente autêntica, seguindo por exemplo a convenção de "assinar e colocar em um envelope"(cria-se um criptograma com a chave privada do remetente, une-se este criptograma com a mensagem original em uma única mensagem e transmite-se um criptograma da mensagem total, criado com a chave pública do destinatário. Deste modo, só o destinatário é capaz de abrir a mensagem total. Além disso, para verificar a autenticidade, basta verificar se a decifragem do criptograma interno utilizando a chave pública do remetente bate com a mensagem original).

É claro que entre os adversários interessados em obter informações sigilosas existem os mais astutos, praticantes de Criptanálise (uma ref. aqui). Diversas maneiras de se quebrar uma segurança são descobertas todos os dias. Uma maneira que vem sendo utilizada mais recentemente, devido ao aumento do poder computacional disponível, é a busca exaustiva por chaves (uma ref. aqui). É normal determinar que um sistema criptográfico é seguro se o melhor ataque conhecido não é mais eficiente do que a busca exaustiva no espaço de chaves.

Dos tipos de ataque existentes, o que é abordado neste trabalho chamamos de ataque de canal lateral (uma ref. aqui). Um ataque de canal lateral se baseia nas informações fornecidas pela parte física do sistema computacional utilizado para executar um algoritmo criptográfico, como por exemplo o consumo de energia em função do tempo.

Um computador funciona através de instruções. Uma instrução é um código que contém a informação de qual operação deve ser realiza pela máquina e quais dados devem ser utilizados como operandos. Historicamente, os primeiros computadores desenvolvidos são hoje chamados de computadores *CISC - Complex Instruction Set Computer*, ou Computador de Conjunto de Instruções Complexo (uma ref. aqui). O nome vem do fato de que os computadores oferenciam uma grande variedade de instruções, com diversas funcionalides complexas e por isso a estrutura interna da unidade central de processamento - *CPU* - era bastante irregular, ou desorganizada.

Passado um certo tempo após a invenção dos processadores digitais, um novo modelo de arquitetura foi proposto. O modelo *RISC - Reduced Instruction Set Computer*, ou Computador de Conjunto de Instruções Reduzido (uma ref. aqui) - prega que o conjunto de instruções de um computador deve ser regular, de modo que é possível otimizar as operações mais frequentes na implementação da *CPU*.

Sabe-se que a intensidade do consumo de energia de um processador digital, em um determinado instante do tempo, depende diretamente da instrução que está sendo executada (uma ref. aqui). Em um computador *CISC* isto é mais evidente, dado que a irregularidade do conjunto de instruções se reflete na implementação física do processador. Em contrapartida, é de se esperar que computadores *RISC* reflitam consumos de energia por instrução mais ininteligíveis. No entanto, os consumos de energia por instrução em computadores *RISC* não são indiferenciáveis ao ponto de que um atacante experiente seja impedido de identificar um algoritmo criptográfico que está sendo executado em uma máquina deste tipo.

Mais recentemente, surgiu o modelo de computador *OISC - One Instruction Set Computer*, ou Computador de Instrução Única (uma ref. aqui). Computadores *OISC* possuem a vantagem de que, independente do consumo de energia em função do tempo, não é possível diferenciar quais instruções estão sendo executadas em um intervalo de tempo, porque só existe uma única instrução! A tendência do consumo de energia de uma *CPU OISC* em

função do tempo é ser uma função periódica, isto é, uma função cujo valor em qualquer ponto inicial é exatamente o mesmo que o avaliado em qualquer ponto cuja distância ao ponto inicial é um valor múltiplo de um determinado período (neste caso, um período de tempo). No entanto, por mais que hajam pequenas oscilações no consumo de energia, a dificuldade de se não poder identificar qual operação está sendo de fato executada em um determinado instante cria uma grande dificuldade para ataques de canal lateral.

Apesar de ser um modelo de computador mais seguro, computadores *OISC* não são muito atraentes, por conta do fato de que quanto mais reduzido é o conjunto de instruções de um computador, mais trabalho é colocado sobre os ombros dos programadores. O objetivo deste trabalho, no entanto, é mostrar que é possível construir um sistema computacional completo, de propósito geral, sobre uma máquina de instrução única. O sistema foi construído em um *FPGA - Field-programmable Gate Array*, ou Arranjo de Portas Programável em Campo (uma ref. aqui) - utilizando a instrução Turing-completa *subleq - Subtract and branch if less or equal to zero*, ou subtrair e pular para outra instrução se o resultado for menor ou igual a zero (uma ref. aqui).

O sistema computacional aqui proposto contempla todos os níveis de abstração de um sistema computacional. Indo do nível mais baixo ao mais alto, implementamos o *hardware* (incluindo *CPU* e controladores de dispositivos externos), o *software* básico (incluindo compilador, montador, ligador e sistema operacional) e *softwares* de aplicação (incluindo aplicações com algoritmos criptográficos).

Capítulo 2

Referencial Teórico

Computadores são sistemas incrivelmente complexos. Inúmeros componentes com papeis específicos necessitam de se intercomunicar para executar a mais simples das tarefas. Dessa forma, para compreender seu funcionamento, se faz o uso de camadas de abstrações.

Essas camadas exercem funções diferentes e são visíveis de acordo com seu uso — um usuário final não precisa saber programar para usar um processador de texto; da mesma forma, um programador não necessita saber da estrutura dos circuitos internos. Cada camada possui seu domínio, sendo as mais próximas do usuário final denominadas de "alto-nível" e as mais próximas dos transistores e fios, "baixo-nível". Observe a figura 2.1, especificada de acordo com Murdocca [5].

2.1 Arquitetura de Processadores

CPU

2.2 Controladores de Dispositivos Externos

CPU

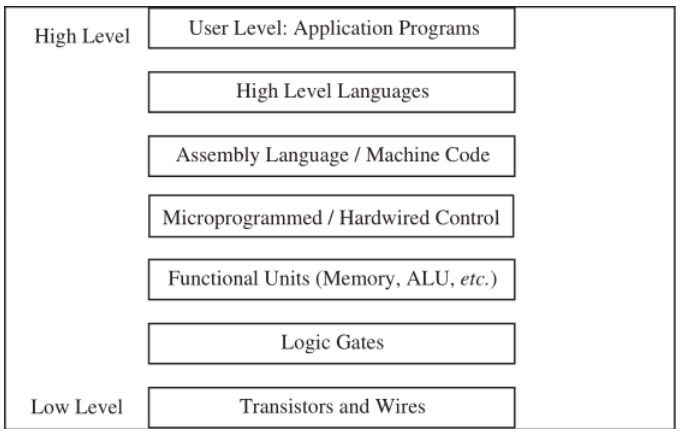

Figura 2.1: Camadas de abstração de um computador.

2.3 Sistemas Operacionais

CPU

2.4 Carregadores

CPU

2.5 Ligadores

2.6 Montadores

Montadores são programas que traduzem um módulo de compilação em linguagem *assembly* para uma versão em linguagem de máquina. O armazenamento desta versão é feito em um arquivo objeto. Como veremos em uma seção adiante, um arquivo objeto é uma versão binária de um módulo de compilação, que geralmente fica a um passo de poder ser carregada para execução.

2.6.1 Linguagem de máquina

Anteriormente vimos que uma linguagem de montagem, ou linguagem *assembly*, é formada por símbolos mnemônicos, ou palavras-chave, que identificam as instruções que um processador é capaz de executar.

A etapa seguinte à compilação é a montagem. Nesta etapa, é necessário traduzir os mnemônicos *assembly* para sequências de *bits*. Um *bit* é um dígito que pode assumir apenas o valor 0 (zero) ou o valor 1 (um). Durante a execução de um programa, estas sequências de *bits* serão diretamente interpretadas por uma *CPU*.

Costuma-se dizer que a etapa de montagem é onde está localizada a interface *software-hardware*. Uma *ISA - Instruction Set Architecture*, ou Arquitetura de Conjunto de Instruções - é uma especificação das instruções que uma implementação de processador digital deve fornecer. Esta especificação estabelece os formatos que as sequências de *bits* geradas por montadores devem seguir.

2.6.2 Algoritmos de montagem

AE

2.7 Compiladores

CPU

2.8 Criptografia

CPU

Capítulo 3

Revisão Bibliográfica

Capítulo 4

Metodologia

Capítulo 5

Protótipo do Quinto Capítulo

5.1 Conceitos Básicos

(Aqui estão coisas que não consegui colocar em outros lugares)

Uma definição muito importante para o programador de sistema é a Arquitetura do Conjunto de Instruções (*Instruction Set Arquitecture*) — de agora em diante referida apenas como arquitetura, ou *ISA*. Hennessy a define como "o limite entre *software* e *hardware*" [2].

A *ISA* descreve vários componentes essenciais para a criação de programas de sistema. Seu *design* define a memória interna do processador, o endereçamento de memória interna e externa, quais instruções/operações são suportadas, tipos e tamanhos de operandos, dentre muitos outros [6].

Existem tipos diferentes de *ISA*, sendo comuns as arquiteturas *RISC* e *CISC*.

Arquiteturas *RISC* (*Reduced Instruction Set Computer*) possuem uma quantidade reduzida de instruções. Em geral são instruções simples e rápidas que têm de ser combinadas para ações mais complexas. Já arquiteturas *CISC* (*Complex Instruction Set Computer*) provêem uma quantidade maior de instruções, que nativamente executam ações mais complicadas e abrangentes.

Instruções da arquitetura *RISC* são mais simples; elas partem da filosofia de otimizar os casos frequentes, visando tornar o comportamento geral mais rápido. Murdocca argumenta que um conjunto de instruções mais simples resulta numa central de processamento simples e menor, liberando espaço no processador para outros componentes, como registradores [5].

Porém Mostafa argumenta que isso traz a desvantagem de que uma grande quantidade de instruções são necessárias para executar uma função simples [4], possivelmente reduzindo o desempenho geral. Por fim, isso tambem causa um problema cognitivo, já que programas *RISC* tendem a ser mais verbosos e depositarem a complexidade do programa nos ombros do programador.

Além de ambas as *ISA*s citadas acima, existe a arquitetura *OISC* (*One Instruction Set Computer*). Ela define computadores com apenas uma única instrução. De acordo com Gilreath, *OISC* é como um *CISC* em um nível mais alto de abstração, já que precisa-se combinar essa única instrução de diversas formas para sintetizar o que seriam as instruções mais complexas [1].

Pela própria definição, arquiteturas *OISC* possuem as desvantagens de *RISC* em escala muito maior. Qualquer função simples necessitará de várias combinações da única instrução, dificultando tanto a velocidade quanto compreensão do programa final.

Entretanto, existem vantagens na previsibilidade de máquinas *OISC*.

Em máquinas não-*OISC*, existe um conjunto bem-definido de instruções que podem ser executadas. Cada uma exige demandas específicas do processador, resultando em gastos de energia possivelmente diferentes.

Dessa forma, ao se monitorar o gasto de energia por um período suficiente de tempo, pode-se observar os padrões de gasto de energia do processador. Então, um observador externo poderá deduzir quais instruções foram executadas na máquina sem necessariamente ter acesso à mesma.

Considerando que arquiteturas *OISC* possuem apenas uma instrução, cuja demanda ao processador é única, assume-se que o gasto de energia será constante. Logo, não seria possível determinar quais ações essa máquina executou independentemente da quantidade de tempo de monitoramento.

O ponto abordado nesse trabalho é exatamente esse — determinar se o gasto de energia em função do tempo é constante numa máquina *OISC*. Se for o caso, pode-se determinar aplicações interessantes para esse tipo de computador na área de segurança de informação e criptografia.

Primeiramente, devemos determinar que instrução será usada na nossa máquina *OISC*.

5.2 Linguagem de Montagem Subleq

Existem várias máquinas de arquitetura *OISC*. Uma delas é a máquina que possui apenas a instrução *SUBLEQ* (*Subtract and Branch on Less or Equal* [3].

Referências

[1] Laplante Phillip A. Gilreath William F. *Computer Architecture: A Minimalist Perspective: Dynamics and Sustainability*. Springer, 2003. 8

[2] Patterson David Hennessy John L. *Computer Architecture: A Quantitative Approach*. Morgan Kaufman Publishers, 1989. 8

[3] O. Mazonka and A. Kolodin. A Simple Multi-Processor Computer Based on Subleq. *ArXiv e-prints*, June 2011. 9

[4] Hesham El-Rewini Mostafa Abd-El-Barr. *Fundamentals of Computer Organization and Architecture*. Wiley, December 2004. 8

[5] Heuring Vincent P. Murdocca Miles. *Principles of Computer Architecture*. Prentice Hall, 1999. 4, 8

[6] Hennessy John L. Patterson David. *Computer Organization and Design: the Hardware/Software Interface*. The Morgan Kaufmann Series in Computer Architecture and Design. Morgan Kaufmann Publishers, 2011. 8