GCC TOOLCHAIN’S C COMPILER WRAPPER FOR THE AMIR CPU ASSEMBLY LANGUAGE

ELINE EE BEE LING

A thesis submitted in partial fulfilment of the
requirements for the award of the degree of
Master of Engineering (Computer and Microelectronic Systems)

School of Electrical Engineering
Faculty of Engineering
Universiti Teknologi Malaysia

JULY 2020

DEDICATION

To my beloved parents, who gave me endless love, trust, constant encouragement over the years.

To my family, for the patience and support and for enduring the ups and downs during the completion of this thesis.

To Prof Madya Dr Muhammad Nasir bin Ibrahim for the support, care and encouragement during the completion of this report.

This report is dedicated to them.

ACKNOWLEDGEMENT

First and foremost, I would like to thank Universiti Teknologi Malaysia (UTM) for providing me the opportunity in completing my Master Project and my postgraduate studies. In particular, I wish to express my sincere appreciation to my main thesis supervisor, Prof Madya Dr Muhammad Nasir bin Ibrahim for encouragement, guidance, critics and support all the way in my Master Project. Without his continued support and interest, this thesis would not have been the same as presented here.

Using this opportunity, I would like to express my very great appreciation to all who have helped me in every way and making my project completion an enjoyable experience.

ABSTRACT

The microprocessor is an icon of the information age today, which evolved from the inventions of the transistor and the integrated circuit (IC). The extensiveness of the microprocessor in this age goes far beyond the wildest imagination at the time of the first microprocessor. The increased use and the importance of microprocessors have led to the appearance of microcontroller chips. Today, unique and more powerful microcontroller, AMIR 32-bit softcore processor was created in order to embrace the challenges in this era. However, currently AMIR 32-bit softcore processor have yet to develop a GCC compiler which is able to compile and run C language application program.

A GCC compiler will convert higher level language such as C programming language into low level language (assembly language), which according to instruction set architecture (ISA) of AMIR 32-bit softcore processor. In fact, C language has always been a preferred language for everyone including students since it is reliable and powerful programming language. From operating system (OS) perspectives, Linux as the well-known open source OS, allow us to implement the GCC compiler in a secure, free as well as highly accessible operating system. In addition, it is also a capable OS and commonly-used platform for all sorts of applications, especially for embedded applications. Hence, GCC Toolchain's C Compiler has been developed in Linux OS to solve the problem stated. For this project, the C compiler developed will be only focus on embedded applications. The main objectives of this research are to develop a working compiler according to ISA of AMIR CPU assembly language as well as to implement the C compiler using GCC toolchain in Linux environment.

This project will be designed based on a language processing system to convert C language to assembly language. Firstly, the user will write an application program using C programming language. The GCC C compiler will compile the program and translate it into assembly language according to ISA of AMIR CPU assembly language. The compiler will read the whole C program at once and go through few analysis before convert into assembler language. The analysis includes lexical analysis, syntax analysis and code generation. Each analysis represents different phases in the compilation process, each phase takes the input from the previous stage and feeds its output to the next phase of the compiler.

Furthermore, in the development of the compiler, Linux (Ubuntu) computer OS will be used for implementation and trouble-shooting. The assembly code generated based on AMIR CPU assembly language will be displayed through the code output in Linux environment.

ABSTRAK

Dalam zaman kini, kepelbagaian mikropempor telah melampaui imaginasi liar pada zaman mikropemproses pertama. Hari ini, mikropengawal yang unik dan lebih berkuasa, dilengkapkan dengan pemproses softcore AMIR 32-bit dicipta untuk merangkul cabaran-cabaran di era ini. Pada masa yang sama, bahasa pengaturcaraan yang boleh diproses dan dilaksanakan oleh pemproses menjadi bahagian penting untuk meningkatkan fleksibiliti. Bahasa C sentiasa menjadi bahasa pilihan untuk semua orang termasuk pelajar kerana ia adalah bahasa pengaturcaraan yang boleh dipercayai dan berkuasa. Walau bagaimanapun, pemproses softcore AMIR 32-bit pada masa kini belum mempunyai pengkompil GCC yang dapat mengkompilasi dan menjalankan program aplikasi bahasa C. Dalam projek ini, skrip pengkompil direka mengikut arkitektur set arahan (ISA) AMIR 32-bit softcore processor dan skrip pengkompil akan dikompilasi menggunakan alat GCC. Dari perspektif sistem operasi (OS), Linux sebagai OS sumber terbuka yang terkenal, membolehkan pelaksanaan pengkompil GCC dalam sistem operasi yang selamat, bebas dan juga boleh diakses. Oleh itu, GCC Toolchain's Compiler C telah dibangunkan dalam OS Linux untuk menyelesaikan masalah yang dinyatakan. Untuk projek ini, pengkompil C yang dibangunkan hanya akan memberi tumpuan kepada aplikasi terbenam. Objektif kajian ini termasuk untuk membangunkan pengkompil yang berkesan mengikut ISA dari bahasa pemasangan AMIR CPU dan melaksanakan pengkompil C menggunakan toolchain GCC dalam persekitaran Linux. Pertama, pengguna akan menulis program aplikasi am menggunakan bahasa pengaturcaraan C. Pengkompil GCC C akan menyusun program dan menterjemahkannya ke dalam bahasa pemasangan mengikut ISA dari bahasa pemasangan AMIR CPU. Pengkompil akan membaca keseluruhan program C sekaligus dan melalui beberapa analisis sebelum menukar ke bahasa pemasang termasuk analisis leksikal, analisis sintaks dan penjanaan kod. Dalam pembangunan pengompilasi, OS komputer Linux (Ubuntu) akan digunakan untuk pelaksanaan dan menimbulkan masalah. Kod pemasangan yang dijana berdasarkan bahasa pemasangan AMIR CPU akan dipaparkan melalui output kod dalam persekitaran Linux.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ISA	-	Instruction Set Architecture
OS	-	Operating System
GCC	-	GNU Compiler Collection
CPU	-	Central Processing Unit
LED	-	Light-emitting diode
IR	-	Intermediate Representation
AST	-	Abstract Syntax Tree
ARM	-	Advanced RISC Machine
ANSI	-	American National Standards Institute
K&R	-	Kidnap & Ransom
UNIX	-	UNiplexed Information and Computing System
GUI	-	Graphical User Interface
MIPS	-	Microprocessor without Interlocked Pipelined Stages
RAM	-	Random-access memory
MS	-	Microsoft Studio

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Problem Background

In this technology-driven era, microprocessors and microcontrollers are playing an extremely important roles in a wide range of engineering applications. Nearly all intelligent electronic devices nowadays use microprocessor or microcontroller chips. The proliferation of microprocessor and microcontrollers as well as their development systems has led to arising of AMIR A1420A 32-bit softcore processor with its own instruction set architecture (ISA). To embrace upcoming challenges, the microcontroller must be able to process not only low-level programming language (assembly language), but also high-level programming language. High-level programming language can be very different from the machine code that the microcontroller can execute, where they are easier to read, write and maintain. In other words, some gap bridging process is needed, and this is where the compiler comes in. From the perspectives of operating system (OS) that the program will be run on, a secure and reliable OS must be chosen. Linux, which recognized as a great platform for programming, is preferred over Microsoft Windows.

1.2 Problem Statement

Currently, AMIR A1420A 32-bit softcore processor only able to process assembly language, which are not user-friendly to most of the users, since it is very close to machine language. It is difficult to use because there are many technical or hardware details which must be memorized and understood by the developers before they can actually use the language. Low-level programming languages are machine oriented and always require the extensive knowledge of the computer architecture (computer hardware & computer configuration). In addition, low-level language programs are not portable, which means particular program cannot be run on another microprocessor. In contrast, a high-level programming language such as C enable a programmer to write program that are independent of a particular type of microprocessors or microcontrollers. From OS perspectives, Microsoft Windows brought some disadvantages compared to Linux including it supports only certain defined platforms and less secure as viruses, hackers and malware can affect the Windows quickly. Linux OS is chosen as the platform to implement the GCC toolchain's C compiler.

1.3 Research Objectives

This Master Project is conducted to fulfil the following objectives:

(a) To develop a working compiler according to ISA of AMIR CPU assembly language

(b) To implement the C compiler using GCC toolchain in Linux environment

(c) To fulfil the requirement as a part of assessment for the Master of Engineering (Computer and Microelectronic Systems)

1.4 Scope of Work

The goal of the project is to develop a working compiler according to the instruction set of AMIR A1420A 32-bit softcore processor. This project mainly focused on the development of the script of a C compiler using high-level C programming language. The script must be able to compile the input C program into assembly language of AMIR A1420A 32-bit softcore processor.

From operating system perspectives, the compiler script developed need to be able to compile and process using GCC toolchain in Linux OS environment. Since the C programming language can be applied on a very wide scope of applications, therefore the C compiler developed in this project will emphasize on data movement instruction as well as logical and arithmetic operations instruction of assembly language of AMIR A1420A 32-bit softcore processor.

REFERENCES

A. A. Maliavko, "The Lexical and Syntactic Analyzers of the Translator for the EI Language," *2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, Novosibirsk, 2018, pp. 360-364.

A. Barve and B. K. Joshi, "A parallel lexical analyzer for multi-core machines," *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, Indore, 2012, pp. 1-3.

A. Barve and B. K. Joshi, "Parallel syntax analysis on multi-core machines," *2014 International Conference on Parallel, Distributed and Grid Computing*, Solan, 2014, pp. 209-213.

A. Benso, S. Chiusano, P. Prinetto and L. Tagliaferri, "A C/C++ source-to-source compiler for dependable applications," *Proceeding International Conference on Dependable Systems and Networks. DSN 2000*, New York, NY, USA, 2000, pp. 71-78.

Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffery D. Ullman Compilers : Principles, Technique and Tools, 2nd ed. PEARSON Education 2009.

Andrew W. Appel. (1998) *Modern Compiler Implementation in C. E-book library [online]. Available at:* https://books.google.com.my/books?hl=en&lr=&id=obI0AAAAQBAJ&oi=fnd&pg=PT4&dq=compiler&ots=-odpE3Q3CX&sig=Ad82hG_39oPrDggqZQ1B6hgW3BU&redir_esc=y#v=onepage&q=compiler&f=false

A. Piotrowski, "Automatic installation of software-based fault tolerance algorithms in programs generated by GCC compiler," *Proceedings of the 17th International Conference Mixed Design of Integrated Circuits and Systems - MIXDES 2010*, Warsaw, 2010, pp. 101-105.

A. Rani, K. Mehla and A. Jangra, "Parsers and parsing approaches: Classification and state of the art," *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, Noida,

A. T. Rose, "Syntax error analysis as a problem solving technique," *30th Annual Frontiers in Education Conference. Building on A Century of Progress in Engineering Education. Conference Proceedings (IEEE Cat. No.00CH37135),* Kansas City, MO, USA, 2000, pp. F4B/8 vol.2-.

A. Woss, M. Loberbauer and H. Mossenbock, "Compiler generation tools for C#," in *IEE Proceedings - Software*, vol. 150, no. 5, pp. 323-327, 27 Oct. 2003.

B. Al Housani, B. Mutrib and H. Jaradi, "The Linux review - Ubuntu desktop edition - version 8.10," *2009 International Conference on the Current Trends in Information Technology (CTIT),* Dubai, 2009, pp. 1-6.

Baojiang Cui, Jiansong Li, Tao Guo, Jianxin Wang and Ding Ma, "Code Comparison System based on Abstract Syntax Tree," *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT),* Beijing, 2010, pp. 668-673.

B. Su, J. Wang, E. W. Hu and J. Manzano, "Assembly code conversion through pattern mapping between two VLIW DSP processors: a case study," *6th International Conference on Signal Processing, 2002.*, Beijing, China, 2002, pp. 406-409 vol.1.

C. D. Newman, R. S. AlSuhaibani, M. L. Collard and J. I. Maletic, "Lexical categories for source code identifiers," *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER),* Klagenfurt, 2017, pp. 228-239.

C. Park, M. Han, H. Lee and S. W. Kim, "Performance comparison of GCC and LLVM on the EISC processor," *2014 International Conference on Electronics, Information and Communications (ICEIC),* Kota Kinabalu, 2014, pp. 1-2.

D.A. Padua and M.J. Wolfe. Advanced compiler optimization for super computers. Cornmun. ACM, 29(12):1184–1201, 1986.

D. Bokan, M. Đukić, M. Popović and N. Četić, "Adjustment of GCC compiler frontend for embedded processors," *2014 22nd Telecommunications Forum Telfor (TELFOR),* Belgrade, 2014, pp. 983-986.

D. Owens and M. Anderson, "A generic framework for automated Quality Assurance of software models - Application of an Abstract Syntax Tree," *2013 Science and Information Conference,* London, 2013, pp. 207-211.

D. Wenjian, "ARM7TDMI Optimization Based on GCC," *2010 Second International Conference on Computer Research and Development*, Kuala Lumpur, 2010, pp. 639-642.

Dr. Matt Poole. (2002). Compilers. Department of Computer, Science University of Wales Swansea.

Dr. Muhammad Nasir Ibrahim et al 2018 J. Phys.: Conf. Ser. 1090 012003

D. Svoboda, "Beyond errno: Error Handling in "C"," *2016 IEEE Cybersecurity Development (SecDev)*, Boston, MA, 2016, pp. 161-161.

E. A. Santos, J. C. Campbell, D. Patel, A. Hindle and J. N. Amaral, "Syntax and sensibility: Using language models to detect and correct syntax errors," *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Campobasso, 2018, pp. 311-322.

Ellis, J.R. *Bulldog: a compiler for VLIW architectures*. United States: N. p., 1985. Web.

E. Moorits and G. Jervan, "Profiling in deeply embedded systems," *2012 13th Biennial Baltic Electronics Conference*, Tallinn, 2012, pp. 127-130.

F. Belli, "Regular Expressions for Fault Handling in Sequential Circuits," *ARCS 2015 - The 28th International Conference on Architecture of Computing Systems. Proceedings*, Porto, Portugal, 2015, pp. 1-5.

F. Mulla, S. Nair and A. Chhabria, "Cross Platform C Compiler," *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, Pune, 2016, pp. 1-4.

Farhanaaz and V. Sanju, "An exploration on lexical analysis," *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, Chennai, 2016, pp. 253-258.

G. Costagliola and S. -. Chang, "DR parsers: a generalization of LR parsers," *Proceedings of the 1990 IEEE Workshop on Visual Languages*, Skokie, IL, USA, 1990, pp. 174-180.

G. Fischer, J. Lusiardi and J. Wolff von Gudenberg, "Abstract Syntax Trees - and their Role in Model Driven Software Development," *International Conference on Software Engineering Advances (ICSEA 2007)*, Cap Esterel, 2007, pp. 38-38.

G. He, Y. Wang and X. Wu, "A regular expression grouping algorithm based on partitioning method," *2012 3rd IEEE International Conference on Network Infrastructure and Digital Content*, Beijing, 2012, pp. 271-274.

G. R. Gao, "Bridging the gap between ISA compilers and silicon compilers: a challenge for future SoC design," *International Symposium on System Synthesis (IEEE Cat. No.01EX526),* Montreal, Quebec, Canada, 2001, pp. 93-.

G. Zhai and Y. Li, "Analysis and Study of Security Mechanisms inside Linux Kernel," *2008 International Conference on Security Technology*, Hainan Island, 2008, pp. 58-61.

H. N. Saha, S. Jasu, S. Biswas and D. Das, "A mixed reality platform based on Linux X -Windowing system," *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, 2018, pp. 1354-1358.

I. Budiselic, S. Srbljic and M. Popovic, "RegExpert: A Tool for Visualization of Regular Expressions," *EUROCON 2007 - The International Conference on "Computer as a Tool"*, Warsaw, 2007, pp. 2387-2389.

I. C. Bertolotti, "RTOS support in C-language toolchains," *2017 IEEE International Conference on Industrial Technology (ICIT),* Toronto, ON, 2017, pp. 1328-1333.

I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna and L. Bier, "Clone detection using abstract syntax trees," *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*, Bethesda, MD, USA, 1998, pp. 368-377.

IEEE Standard for Microprocessor Assembly Language," in *IEEE Std 694-1985* , vol., no., pp.0_1-, 1985

I. Jaziri, L. Chaarabi and K. Jelassi, "A remote DC motor control using Embedded Linux and FPGA," *2015 7th International Conference on Modelling, Identification and Control (ICMIC)*, Sousse, 2015, pp. 1-5.

J. Alves, M. Held and M. Glesner, "A code generator for an application specific pipelined processor," *Proceedings of MELECON '94. Mediterranean Electrotechnical Conference*, Antalya, Turkey, 1994, pp. 306-308 vol.1.

J. Merrill: Generic and gimple: A new tree representation for entire functions, In Proceedings of the 2003 GCC Developers' Summit, pp. 171-179. 2003.

Johnson, Stephen. (2001). Yacc: Yet Another Compiler-Compiler. Unix Programmer's Manual. 2. Tanaka and King-Sun Fu, "Error-Correcting Parsers for Formal Languages," in *IEEE Transactions on Computers,* vol. C-27, no. 7, pp. 605-616, July 1978.

J. Vankeirsbilck, J. Van Waes, H. Hallez and J. Boydens, "Automated Regression Testing of a GCC Toolchain used on Embedded CPU Programs," *2019 IEEE XXVIII International Scientific Conference Electronics (ET),* Sozopol, Bulgaria, 2019, pp. 1-4.

J. Zhang, X. Wang, H. Zhang, H. Sun, K. Wang and X. Liu, "A Novel Neural Source Code Representation Based on Abstract Syntax Tree," *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE),* Montreal, QC, Canada, 2019, pp. 783-794.

Kenneth C. Louden. (2011). Compiler Construction: Principles and Practice, San Jose State University, USA.

K. H. Yu and Y. H. Hu, "Optimized code generation for programmable digital signal processors," *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing,* Minneapolis, MN, USA, 1993, pp. 461-464 vol.1.

K. Nakayama and E. Sakai, "Source code pattern as anchored abstract syntax tree," *2014 IEEE 5th International Conference on Software Engineering and Service Science,* Beijing, 2014, pp. 170-173.

L. Bogdanov, "Statement-level energy simulation in embedded systems using GCC," *2016 XXV International Scientific Conference Electronics (ET),* Sozopol, 2016, pp. 1-4.

Lei Wang, Boying Lu and Li Zhang, "The study and implementation of architecture-dependent optimization in GCC," *Proceedings Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region,* Beijing, China, 2000, pp. 253-255 vol.1.

L. J. Dyadkin, "Multibox parsers: no more handwritten lexical parsers," in *IEEE Software*, vol. 12, no. 5, pp. 61-67, Sept. 1995.

L. Johnson, D. C. Pheanis and I. A. Fulton, "Hard-to-detect errors due to the assembly-language environment," *2007 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports,* Milwaukee, WI, 2007, pp. T1E-14-T1E-19.

L. Sharmila, U. Sakthi, A. Geethanjali and S. Sagadevan, "Regular Expression Based Pattern Matching for Gene Expression Data to Identify the Abnormality Gnome," *2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM),* Tindivanam, 2017, pp. 301-305.

L. Simon, D. Chisnall and R. Anderson, "What You Get is What You C: Controlling Side Effects in Mainstream C Compilers," *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, London, 2018, pp. 1-15.

L. Shan, "Exploration of education reform based on 32-bit assembly language programming," *2011 6th International Conference on Computer Science & Education (ICCSE)*, Singapore, 2011, pp. 595-599.

M. Kong, J. Li and Wang Fengming, "Study on educational mode of Linux majors in colleges," *2010 International Conference on Artificial Intelligence and Education (ICAIE)*, Hangzhou, 2010, pp. 623-626.

M. Mernik and V. Zumer, "An educational tool for teaching compiler construction," in *IEEE Transactions on Education*, vol. 46, no. 1, pp. 61-68, Feb. 2003.

M. Sičák, "Higher order regular expressions," *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, Oradea, 2015, pp. 1-4.

M. Tabassum and K. Mathew, "Software evolution analysis of linux (Ubuntu) OS," *2014 International Conference on Computational Science and Technology (ICCST)*, Kota Kinabalu, 2014, pp. 1-7.

M. Zaki and S. Tahar, "Syntax code analysis and generation for Verilog," *CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No.03CH37436)*, Montreal, Quebec, Canada, 2003, pp. 235-240 vol.1.

N. Bellas, I. N. Hajj, C. D. Polychronopoulos and G. Stamoulis, "Architectural and compiler techniques for energy reduction in high-performance microprocessors," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 317-326, June 2000.

N. B. Jensen, P. Schleuniger, A. Hindborg, M. Walter and S. Karlsson, "Experiences with Compiler Support for Processors with Exposed Pipelines," *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*, Hyderabad, 2015, pp. 137-143.

N. Hasabnis, R. Qiao and R. Sekar, "Checking correctness of code generator architecture specifications," *2015 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, San Francisco, CA, 2015, pp. 167-178.

N. Vun, H. F. Hor and J. W. Chao, "Real-Time Enhancements for Embedded Linux," *2008 14th IEEE International Conference on Parallel and Distributed Systems*, Melbourne, VIC, 2008, pp. 737-740.

P. Wang, G. R. Bai and K. T. Stolee, "Exploring Regular Expression Evolution," *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Hangzhou, China, 2019, pp. 502-513.

R. Barua, W. Lee, S. Arnarasinghe and A. Agarwal, "Compiler support for scalable and efficient memory systems," in *IEEE Transactions on Computers*, vol. 50, no. 11, pp. 1234-1247, Nov. 2001.

R. Szabó and A. Gontean, "Image acquistion with Linux on FPGA," *2014 22nd Telecommunications Forum Telfor (TELFOR)*, Belgrade, 2014, pp. 1007-1010.I

S. D. Sharma, D. N. Sonawane, T. Chakravorty and T. Patil, "GNU/Linux shell access through a web-browser for an embedded Linux e-learning system," *2011 3rd International Conference on Electronics Computer Technology*, Kanyakumari, 2011, pp. 335-338.

Shuai, Wang & Heng, Zhang & Han-qing, Tan & Lin-ying, Jiang. (2012). Implementation of Step Motor Control under Embedded Linux Based on S3C2440. Energy Procedia. 16. 1541–1546. 10.1016/j.egypro.2012.01.241.

S. Levinson, A. Rosenberg and J. Flanagan, "Evaluation of a word recognition system using syntax analysis," *ICASSP '77. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Hartford, CT, USA, 1977, pp. 483-486.

S. Loglo, Sarula and HuaShabao, "Research on Mongolian lexical analyzer based on NFA," *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, Xiamen, 2010, pp. 240-245.

S. MacKenzie, "A structured approach to assembly language programming," in *IEEE Transactions on Education*, vol. 31, no. 2, pp. 123-128, May 1988.

Steven Muchnick, Muchnick and Associates. (1997). *Advanced Compiler Design Implementation.* Available at: https://books.google.com.my/books?hl=en&lr=&id=Pq7pHwG1_OkC&oi=fnd&pg=PR31&dq=compiler&ots=4Za_Hna8nT&sig=BdcBJ-hrc1bBn5a-eXkJ9SQGL2g&redir_esc=y#v=onepage&q=compiler&f=false

T. G. Rose and L. J. Evett, "Semantic analysis for large vocabulary cursive script recognition," *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR '93),* Tsukuba Science City, Japan, 1993, pp. 236-239.

T. Hasu, "Concrete error handling mechanisms should be configurable," *2012 5th International Workshop on Exception Handling (WEH),* Zurich, 2012, pp. 46-48.

Torben Ægidius Mogensen. (2010). Basics of Compiler Design. DEPARTMENT OF COMPUTER SCIENCE UNIVERSITY OF COPENHAGEN.

Tzer-Shyong Chen, Feipei Lai and Rung-Ji Shang, "A simple tree pattern matching algorithm for code generator," *Proceedings Nineteenth Annual International Computer Software and Applications Conference (COMPSAC'95),* Dallas, TX, USA, 1995, pp. 162-167.

Waite, William & Goos, Gerhard. (1984). Compiler Construction. 10.1007/978-1-4612-5192-7.

W. Kreuzer, M. Gotschlich and B. Wess, "A retargetable optimizing code generator for digital signal processors," *1996 IEEE International Symposium on Circuits and Systems (ISCAS),* Atlanta, GA, USA, 1996, pp. 257-260 vol.2.

W. W. Hwu *et al.*, "Compiler technology for future microprocessors," in *Proceedings of the IEEE*, vol. 83, no. 12, pp. 1625-1640, Dec. 1995.

X. Zhou, "First-Level Bottom-Up Parser," *2009 International Forum on Computer Science-Technology and Applications*, Chongqing, 2009, pp. 192-195.

Yan Zhang, Xinyu Gao, Ce Bian, Ding Ma and Baojiang Cui, "Homologous detection based on text, Token and abstract syntax tree comparison," *2010 IEEE International Conference on Information Theory and Information Security*, Beijing, 2010, pp. 70-75.

Y. Chen and A. Zhu, "Implementation of Linux centralized user authentication and cloud storage in teaching," *2014 International Conference on Information Science, Electronics and Electrical Engineering*, Sapporo, 2014, pp. 626-628.

Y. Dun-fan, Z. Fei-fan and M. Liang-liang, "Design and Implementation of High-Precision Timer in Linux," *2009 WRI World Congress on Computer Science and Information Engineering*, Los Angeles, CA, 2009, pp. 341-345.

Y.N. Srikant. (2012). NPTEL Course on Principles of Compiler Design, Department of Computer Science and Automation Indian Institute of Science Bangalore.

Y. Zhou and Q. Zhou, "The embeded real-time Linux operation system based on the Xenomai," *2011 International Conference on Electrical and Control Engineering*, Yichang, 2011, pp. 3286-3290.

Z. Fu and J. Li, "Spectral clustering based regular expression grouping," *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, Marina del Rey, CA, 2014, pp. 243-244.

Z. Yanyan and R. Xiangjin, "Analysis of Linux Kernel's Real-Time Performance," *2018 International Conference on Smart Grid and Electrical Automation (ICSGEA)*, Changsha, 2018, pp. 191-194.

Z. Zhang, S. Qin, X. Wang and D. Zhan, "Implementation of embedded Linux based on PC/104 platform," *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, 2015, pp. 1073-1077.