INSTRUCTION SET ARCHITECTURE DESIGN FOR VIDEO PLAYBACK DEVICE

Kudamage Nishadi Neranja

108411B

Degree of Master of Science

Department of Electronics and Telecommunication Engineering

University of Moratuwa

Sri Lanka

October 2017

INSTRUCTION SET ARCHITECTURE DESIGN FOR VIDEO PLAYBACK DEVICE

Kudamage Nishadi Neranja

108411B

Dissertation submitted in partial fulfillment of the requirement for the degree Master of Science in Electronics and Automation

Department of Electronics and Telecommunication Engineering

University of Moratuwa

Sri Lanka

October 2017

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The above candidate has carried out research for the Master's thesis under my supervision.

Signature of the supervisor: Date:

DEDICATION

To my family members and teachers

ACKNOWLEDGEMENT

The research project provides a great opportunity to work on my desire research area under supervision of Dr. Ajith Pasqual, Senior Lecturer, Department of Electronics and Telecommunication Engineering. My sincere thanks go to him for guiding and supporting me for carrying out the research in proper direction and successfully completion of the project. I would like to thank the course coordinator Prof. Rohan Munasinghe for the numerous support and advices given to me throughout the course.

As a student of Master's Degree, I gained a vast amount of knowledge and practice from the taught course, mini research project and the design project during the allocated period of the course. Therefore, I would like to thank all the academic staff who taught us and non–academic staff for giving me their friendly support. My sincere thanks go to Prof. (Mrs.) Dileeka Dias for supervising and guiding me for completion of my design project and Mr.Lanka Wijesinghe, Mr.Hasla and the staff who are working at the Dialog Research Laboratory for giving their numerous supports. I would like to convey my sincere gratitude to Mr.Jayantha, Chief Technical Officer for open the Postgraduate Lab without any interruption during the study period and also staff of Digital Laboratory and workshop engineer for giving me their friendly support.

I would like to thank Mr. D K Withanage, Dean, Faculty of Information Technology who was guiding me and encouraging me for carrying out my higher studies. And also my thanks go to Mr. B H Sudantha, Mr. Harshana Gunasekara, Mr. Isuru Senarth, and Mr. Anushaka – Academic Staff for covering my duties during my absence and also for Hardware Laboratory staff for their numerous support given to me in various ways. And finally, I would like to thank my parents and husband for giving me tremendous support during the period of study. This journey would not have been possible without their support.

ABSTRACT

Instruction Set Architecture Design for Video Playback Device

Keywords: Low Power, Processors, FFmpeg, decoders

Application specific processors are being considered for many applications which are used to run on general purpose processors. The primary reason for this is the enhanced energy efficiency while meeting the required performance targets. This thesis explores the design of Instruction Set Architecture (ISA) for a video playback device.

Video is ubiquitous today due to camera being a standard accessory in mobile phones. Video, at the same time, is a powerful learning tool for any age group particularly for younger children. The primary objective of the work is to develop a minimalist ISA for a single function video playback device which would allow longer run time on battery (enhanced energy efficiency) and low silicon footprint to minimize cost. This would allow video playback device to function without an operating system.

An extensive survey of low power processors was followed by a thorough investigation of essential assembly instructions for video playback using the industry standard video playback tool-ffmpeg. The minimal ISA developed was then validated by using Intel Software Development Emulator through dynamic run-time analysis of ffmpeg trace. Here most frequently used assembly instructions were found to be present in the minimal instruction set.

TABLE OF CONTENTS

Page

DECLARATION i

DEDICATION ii

ACKNOWLEDGEMENT iii

ABSTRACT iv

TABLE OF CONTENTS v

LIST OF FIGURES vii

LIST OF TABLES viii

LIST OF ABBREVIATIONS ix

LIST OF APPENDICES x

INTRODUCTION 1

1.1 Objectives 2

1.2 Functionalities 2

1.3 Overview of the Thesis 3

LITREATURE REVIEW 4

2.1 Available Laptops in the Market for Kids 4

2.1.1 Barbie B-Smart Laptop 4

2.1.2 VTech Double Vision Notebook 4

2.1.3 Hot Wheels Accelerator Laptop JW88 4

2.1.4 Meep 5

2.1.5 PeeWee Pivot 2.0 Laptop 5

2.2 Comparison with other Researches 5

2.2.1 XO laptop used as an educational tool for kids 5

2.2.2 Raspberry Pi Processor for playing videos 10

2.2.3 Playing video on Nokia color LCD using an Atmega32 16

METHODOLOGY FOR THE DESIGN OF ISA FOR VIDEO PLAYBACK 17
3.1 The x86 Architecture 17
3.2 Analysis of FFmpeg decoder and testing 19
3.2.1 FFmpeg decoder 21
3.3 Optimized Instruction Set for Video playback 23
DESIGN OF ISA FOR VIDEO PLAYBACK 27
4.1 Instruction Set Architecture (ISA) 27
4.1.1 Instruction Format Design 28
4.1.2 Register File 31
4.1.3 Addressing Modes 31
VALIDATION OF MINIMAL INSTRUCTION SET 32
5.1 Dynamic Analysis of video playback at runtime 32
CONCLUSIONS AND FUTURE WORK 40
REFERENCES 41

LIST OF FIGURES

Page

Figure 2.1: Director OLPC Europe with introduced Laptop Project. 5

Figure 2.2 : XO-1.75 Block Diagram..... 8

Figure 2.3: ARM1176JZF-S Block Diagram 11

Figure 2.4: ARM instruction Set..... 15

Figure 2.5: Thumb Instruction Set 15

Figure 2.6: Playing video on nokia colour LCD using an 8-bit AVR [ATtmega 32]16

Figure 3.1: Basic Execution Environment of x86 architecture..... 19

Figure 3.2: Compilation process of FFmpeg Decoder 21

Figure 3.3: Part of C codes converted to Assembly codes. 22

Figure 3.4: Removed repeated instructions and count of repeated instructions 23

Figure 5.1: Play videos using Intel Software Development Basic Emulator 34

LIST OF TABLES

Page

Table 2 1: Major differences of Raspberry pi models..... 12

Table 3.1: Optimized Instruction set for video playback on x86 platforms..... 23

Table 3 2: Optimized Instruction Set to design ISA for Video Playback 26

Table 5 1: Most frequently used Runtime Instructions in different formats when playing videos 35

Table 5 2: Common Instructions used in three formats and its functions obtained after Analysis 36

Table 5 3: Other Instructions found within mostly used instructions 37

Table 5 4: Minimal Instructions found among the runtime Instructions..... 39

Table 5 5: Minimal Instructions have not found among the runtime Instructions.... 39

LIST OF ABBREVIATIONS

Abbreviation	Description
TFT	Thin Film Transistor
OLPC	One Laptop Per Children
WMMX2	Wireless MMX2
LPDDR	Low Power Double Data rate memory
MIPI	Mobile Industry Processor Interface
DSI	Display Serial Interface
HSI	Horizontal Situation Indicator
SLIM	Simple Login Manager
MMC	Microsoft Management Console
HDMI	High Definition Multimedia Interface
PHY	Marvell Fast Ethernet Physical Layer
HD	High Definition
ISP	Image Signal Processor

LIST OF APPENDICES

Page

Appendix Description

Appendix A C files used in FFmpeg 3.2 decoder 43

CHAPTER 1

INTRODUCTION

Visual observation is a powerful means of obtaining information content of a given source. Video has become an important part of education. Several meta-analysis have shown that technology can enhance learning and multiple studies have shown that video, specifically, can be a highly effective educational tool. One of the strengths of video is the ability to communicate with viewers on an emotional, as well as a cognitive, level. Because of this ability to reach viewers' emotions, video can have a strong positive effect on both motivation and affective learning. Not only are these important learning components on their own, but they can also play an important role in creating the conditions through which greater cognitive learning can take place. Videos can be a very effective learning style for children that make them curious in their early childhood. As an example, children become enthusiastic of using Laptops and Mobile phones as they see their parents using them frequently and waits impatiently until they get them. This practice may lead to unnecessary complications such as loss of their valuable data. Notebooks, tablets, and laptops for children are invading primary schools as it may become the part and parcel of children in the near future. By far most computers used by children contain general purpose processors for video playback. So, providing a separate processor in a general-purpose computer for video playback would be always advantageous. As DVD players and general-purpose computers currently available for children are so expensive, the proposed video playback device can be an affordable one for many people at a reasonable price not more than $100 with some functionalities similar to an electronic digital photo album.

Researchers moved to carry out researches for developing Application Specific Instruction Set Processors (ASIP) for long time as customized processors can gain power efficiency and performance. Demanding applications like audio, security, networking, baseband, control and industrial automation are ideal for ASIPs.

Embedded vision is another key domain, with applications such as advanced driver assistance systems, gesture control and augmented reality calling for performance and power optimized processors. This thesis explores the method of minimizing the instruction set for playing video and design of particular Instruction Set Architecture (ISA) to design the customized processor for video playback. Therefore, anyone can use this ISA for designing video playback device.

Video can play in many formats, such as avi, mpeg2/4, ogg, etc. FFmpeg is a free software project that produces libraries and programs for handling multimedia data. It can record, convert and stream digital audio and video in numerous formats. FFmpeg can be used to convert many multimedia formats to one another. It is a command line tool that is composed of a collection of free software / open source libraries. The name of the project comes from the MPEG video standards group, together with "FF" for "fast forward". [14] Therefore, FFmpeg decoder is considered for designing an Instruction Set Architecture (ISA) for playing videos in this thesis.

1.1 Objectives

- Identify instructions that are essential for video playback and associated activities through a systematic profiling of video playback software in x86.
- To design and develop a minimalistic Instruction Set Architecture (ISA) customized for video playback**.**

1.2 Functionalities

Although some functionalities of the proposed video playback device are similar to that of electronic digital photo album; it has been designed especially for nursery school children. Instead of using keyboard, this device uses a button to play and TFT LCD display to watch videos. SD card is used to store videos and the proposed customized processor processes and play videos. This device can be sold at an affordable price below $100.

1.3 Overview of the Thesis

The remainder of this thesis is outlined as follows.

- Chapter 2 contains the Literature survey of customized processors for video playback.
- Chapter 3 describes the methodology for the design of ISA for a video playback device. The method of minimizing the instructions by using FFmpeg decoder is described in this chapter.
- Chapter 4 contains the Instruction Set Architecture (ISA) design of a video playback device.
- Chapter 5 contains the validation of minimal Instruction Set which is used for designing the ISA for video playback device.
- Chapter 6 concludes the thesis with a brief description of future work.

CHAPTER 2

LITREATURE REVIEW

2.1 Available Laptops in the Market for Kids

Laptops are available for children with different processors in the market. Described below are some of the latest laptops available for kids. [1]

2.1.1 Barbie B-Smart Laptop

This is a laptop which is specifically meant to be used by young girls who are Barbie fans. Oregon Scientific has come up with this model which sports 60 activities in English and 10 in Spanish. It even includes a mouse, mouse pad, QWERTY keyboard, with the function to adjust color contrast and volume. This kids' laptop is available for around $50.

2.1.2 VTech Double Vision Notebook

The main feature of this product is that it allows the user to combine color photos, text, and audio to make presentations and view them on the television. It also has programs that can help in developing the child's language, mathematics, and logical skills. VTech Double Vision Notebook is available at a price just around $50.

2.1.3 Hot Wheels Accelerator Laptop JW88

This is more of a gaming laptop computer offered by Oregon Scientific, with a 'hot wheels' theme. It is primarily intended to be used by boys. It comprises 50 games in English and 10 games in Spanish. The games include word puzzles, memory games, and number games. We can buy by this kid’s laptop around $50.

2.1.4 Meep

Oregon Scientific offers the Meep X2 as its newest Android based tablet for Kids. They also offer the Meep, the MeepStar Blue Meep, and the MeepFrog Green Meep and more.

2.1.5 PeeWee Pivot 2.0 Laptop

The PeeWee Pivot is considered to be one of the most advanced laptops available in stores. It has a rotating touch screen which can easily be converted into a tablet. It is embedded with a 1.66 GHz Intel Atom Processor N450, 160 GB hard disk capacity, 1 GB of RAM, and a 10.1-inch screen. It is available for under $600 on the manufacturer's website. It uses Windows.

2.2 Comparison with other Researches

2.2.1 XO laptop used as an educational tool for kids

After completion of many researches, One Laptop per child (OLPC) programme has been introduced to children in rural areas of several countries to explore their own potential, to be exposed to innovative ideas, and to contribute to a more productive world community.

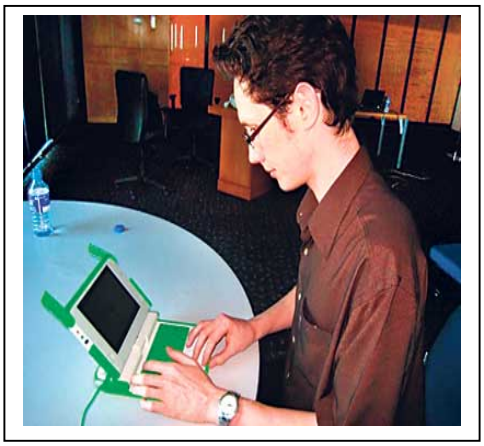


Figure 2.1: Director OLPC Europe with introduced Laptop Project.

This is being launched by One Laptop Per Child (OLPC), a US based organization in collaboration with the Education Department and several local and foreign financial, technological and academic institutions. Director OLPC Europe, Middle East and Asia Matt Keller, in an interview with The Sunday Times FT, said the World Bank has stepped into fund a pilot project to introduce laptops as an educational tool in nine provinces in the island. (Source: Article from Sunday Times on February 10, 2008)

OLPC Lanka Foundation has been set up to implement this massive education project aimed at supplying this learning tool into rural children's hands.

A small machine with a big mission. The XO is a potent learning tool designed and built especially for children in developing countries, living in some of the most remote environments. It's about the size of a small textbook. It has built-in wireless and a unique screen that is readable under direct sunlight for children who go to school outdoors. It's extremely durable, brilliantly functional, energy-efficient, and fun. [2]

Hardware specification of the XO (CL2) laptop

System architecture and hardware design requirements for the XO Children's Laptop version 1.75 (CL2). [2]

Functionality of the CL2 is similar to an ordinary notebook PC, it was designed by One Laptop per Child as a laptop for elementary school children. The foremost goal of the hardware design is low-power operation. To achieve this goal, the CL2 utilizes a very low power ARM processor and incorporates novel technologies such as low power dual mode TFT LCD display and low power networking interfaces. The CL2 is designed for outdoor use with a 19 cm (diagonal) 4:3 aspect ratio color/monochrome dual mode TF LCD panel and a dust and moisture resistant keyboard and case. For ruggedness and low power, it uses NAND Flash as storage device in the system. To allow many hours of operation it supports a 20 WH battery.

The CL2 is a system/motherboard upgrade to the existing CL1C design. No changes to the overall industrial design or tooling will be made. The addition of seven raised dots to the hinge cover allows easy differentiation of CL1, CL1B and CL2 machines in the field.

Environmental

The XO is the most energy-efficient and environmentally friendly laptop ever made, based on independent evaluations and data. XO consumes the least power, minimizes toxic materials, is extraordinarily rugged, has a long lifetime, works with renewable power sources, and is itself recyclable. XO has earned the highest environmental certifications: it is in full compliance with the European Union's rigorous Reduction of Harmful Substances (RoHS) standards; it is designed for Energy Star Version 4.0 Category A Tier 2 performance, the most stringent level.

According to ENERGY STAR®, an average idle desktop computer uses 70 watts of power and an average idling laptop computer consumes 20 watts of power. When idle, the XO laptop uses one watt of electricity. Among the XO's other environmentally friendly attributes and innovations. XO is more rugged — it will last longer, thus staying out of landfills longer. The XO has been designed for a five-year lifetime even in extreme environments like the outdoors, the jungle, and the desert. The average laptop has a two-year lifetime when used in an office and far less when brought outside or to the desert. Doubling the lifetime of the laptop halves its environmental impact. XO is about half the size and weight of typical laptops. Less material halves the environmental impact. XO is designed for use with renewable energy sources. It's the first laptop made with renewable energy accessories: a hand crank, a small solar panel, a foot pedal, or a lawnmower style rope pull will recharge the laptop.

XO uses a new battery using LiFePO4 (Lithium Ferro Phosphate) chemistry that lasts four times longer than standard laptop batteries, and is vastly safer than the current dominant technology of Lithium Ion.

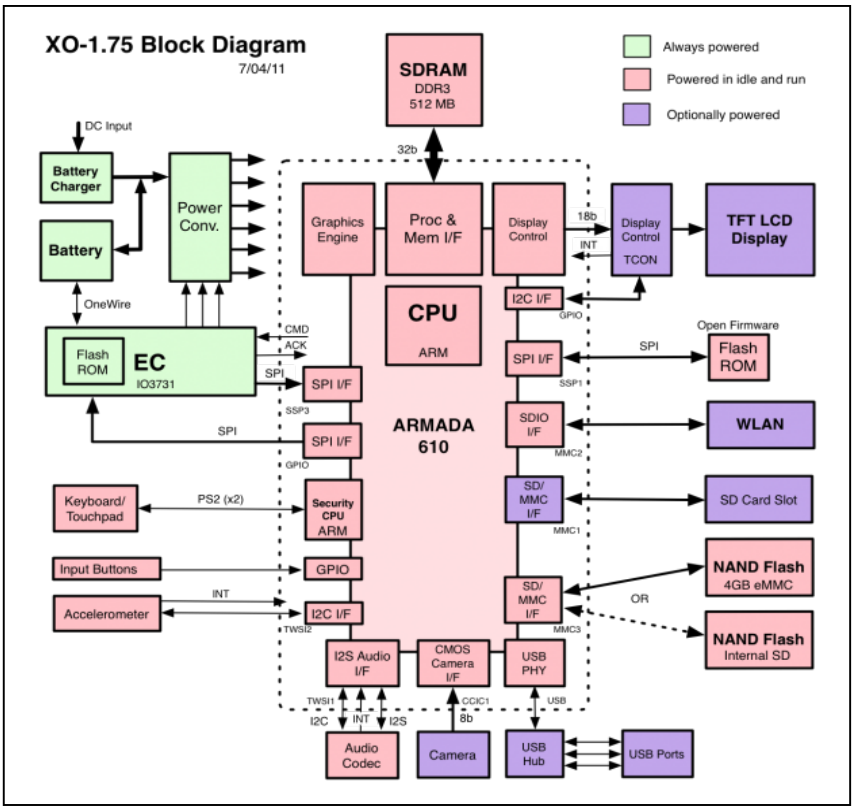


Figure 2.2: XO-1.75 Block Diagram

General Specifications

Processor & core system:

- Marvell Armada 610 Application Processor
- Integrating an Marvell Sheeva ARM CPU (800 MHz)
- 32KB/32KB L1 caches, 256KB unified L2 cache
- 512MiByte or 1024 MiByte DDR3 SDRAM system memory, running at 800MHz
- Embedded controller for system monitoring, battery charging, and solar power input
- ISA Compatibility: ARM v6 and v7, with Thumb and WMMX2* instruction set extensions

Marvell Armada 610 Application Processor

The ARMADA™ 610 processor is Marvell®'s next generation application processor that is designed for mainstream. Mobile Internet Devices (MIDs), connected consumer products, eReaders, eBooks, tablets, media players and new personal information appliances. Featuring a gigahertz-class CPU, integrated full HD 1080p encode and decode, an integrated ISP capable of 16MP image captures, an integrated audio processing engine for extremely low power audio playback and exceptional high quality sound, an integrated EPD display controller and advanced 3D graphics, the ARMADA 610 delivers the best combination of fast, PC-caliber processing**,** an uncompromised Internet experience all in the lightweight form factors with extended battery life. The ARMADA 610 is based on a 1GHz Marvell-designed ARM v7-compatible CPU offering best-in-class performance for the most demanding software applications. An integrated 3D engine renders 45M triangles-per-second via a complete floating-point pipeline and unified vertex and fragment/pixel shading for an immersive game play experience with the ability to drive the latest in 3D enabled user interfaces. The ARMADA 610 supports industry standard APIs – ensuring complete compatibility with the most hotly anticipated mobile game titles and easy porting of 3D enabled applications and user interfaces. The ARMADA 610 features Marvell's award-winning Qdeo™ technology with an integrated video accelerator that can seamlessly encode and decode 1080p video at 30fps. In addition, the ARMADA 610 incorporates a complete Image Signal Processor which can capture high resolution color pictures up to 16MP as well as stream 1080p video at 30fps. The ARMADA 610 integrates a high performance, low power EPD display controller. By integrating the controller, the ARMADA 610 can drive EPD displays at up to 5x the speed of software based or external EPD display controllers. This enables applications, such as HD IP cameras, full HD camcorders and HD video playback, and high performance eReaders which do not suffer from page turn lag, that were previously impossible for this class of device.

The ARMADA 610 offers the flexibility to use any standard memory (LPDDR and standard DDR), a highly flexible display controller capable of five simultaneous

displays up to 2k x 2k resolution and a robust security subsystem that includes a secure execution processor. The ARMADA 610 also features support for the next generation of peripheral interfaces, through support for MIPI DSI display, MIPI CSI camera, MIPI HSI and MIPI SLIM bus. Additional peripheral interfaces supported include USB 2.0 HSIC, SD/SDIO/MMC, eMMC, HDMI v1.3a w/PHY and a standard set of lower bandwidth peripherals. Legacy peripherals such as Parallel LCD and Parallel Camera interfaces with integrated laser scanner support are also included. The ARMADA 610 offers optimized OS support for Linux, Android™, Windows® Mobile 6.5, Windows 7 and Flash® 10. The ARMADA 610 comes in a 12x12mm POP package, 12x12mm Discrete and a cost saving 21x21mm 0.65mm ball pitch consumer package. ARMADA 610 customers will have one of the broadest, most flexible choices of platform in the industry to create truly innovative and marketable products. [3]

2.2.2 Raspberry Pi Processor for playing videos

The Raspberry Pi is a credit-card-sized single-board computer developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. [4]

The Raspberry Pi is manufactured in two board configurations through licensed manufacturing deals with Newark element14 (Premier Farnell), RS Components and Egoman. These companies sell the Raspberry Pi online. Egoman produces a version for distribution solely in China and Taiwan, which can be distinguished from other Pis by their red coloring and lack of FCC/CE marks. The hardware is the same across all manufacturers.

The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor, VideoCore IV GPU,and was originally shipped with 256 megabytes of RAM, later upgraded to 512 MB. It does not include a built-in hard disk or solid-state drive, but it uses an SD card for booting and persistent storage.

The Foundation provides Debian and Arch Linux ARM distributions for download. Tools are available for Python as the main programming language, with support for BBC BASIC (via the RISC OS image or the Brandy Basic clone for Linux), C, Java and Perl.

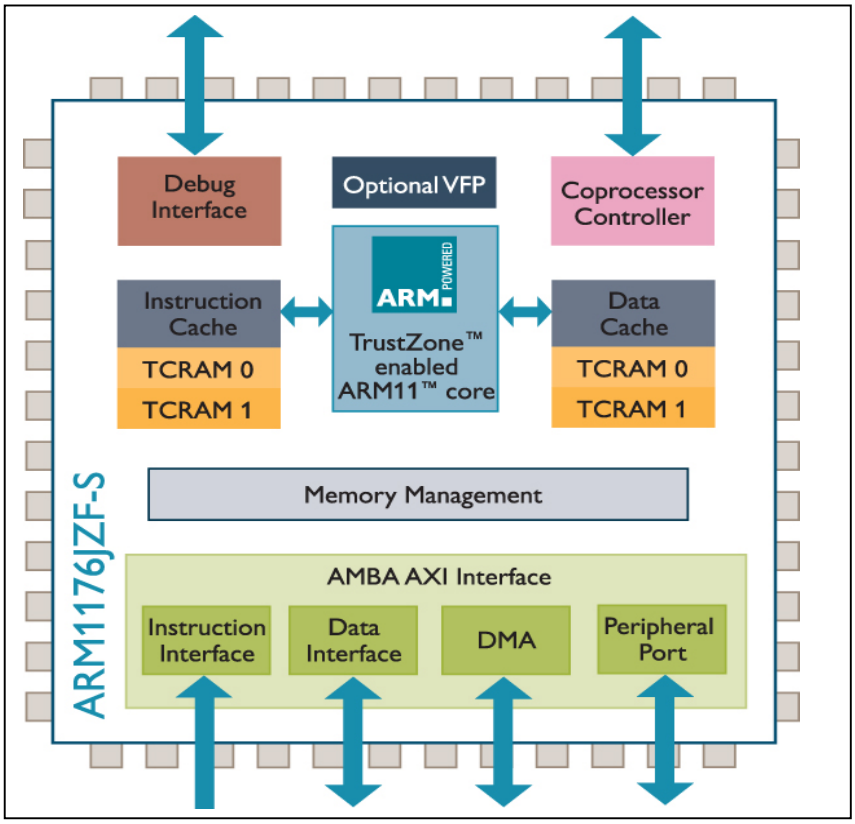


Figure 2.3: ARM1176JZF-S Block Diagram

Table 2 1: Major differences of Raspberry pi models

	Model A	**Model B**
Target price:	US$ 25	US$ 35
SoC:	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM, and single USB port)[3]	
CPU	700 MHz ARM1176JZF-S core (ARM11 family, ARMv6 instruction set)	
GPU	Broadcom Video Core IV @ 250 MHz OpenGL ES 2.0 (24 GFLOPS) MPEG-2 and VC-1 (with license), 1080p30 h.264/MPEG-4 AVC high-profile decoder and encoder.	
Memory (SDRAM):	256 MB (shared with GPU)	512 MB (shared with GPU) as of 15 October 2012
USB 2.0 ports	1 (direct from BCM2835 chip)	2 (via the built in integrated 3-port USB hub)
Video Input	A CSI input connector allows for the connection of a RPF designed camera module	
Video outputs	Composite RCA (PAL and NTSC), HDMI (rev 1.3 & 1.4), raw LCD Panels via DSI	
Audio outputs	3.5 mm jack, HDMI, and, as of revision 2 boards, I²S audio[91] (also potentially for audio input)	
Onboard storage:	SD / MMC / SDIO card slot (3.3 V card power support only)	
Onboard network:	None	10/100 Mbit/s Ethernet (8P8C) USB adapter on the third port of the USB hub
Low-level peripherals:	8 × GPIO, UART, I²C bus, SPI bus with two chip selects, I²S audio[93] +3.3 V, +5 V, ground	
Power ratings:	300 mA (1.5 W)	700 mA (3.5 W)
Power source:	5 V via Micro USB or GPIO header	
Size:	85.60 mm × 56 mm (3.370 in × 2.205 in)	
Weight:	45 g (1.6 oz)	
Operating systems:	Arch Linux ARM, Debian GNU/Linux, Gentoo, Fedora, FreeBSD, NetBSD, Plan 9, Inferno, Raspbian OS, RISC OS, Slackware Linux	

ARM1176JZF-S processor

The ARM1176JZF-S processor incorporates an integer core that implements the ARM11 ARM architecture v6. It supports the ARM and Thumb™ instruction sets, Jazelle technology to enable direct execution of Java byte codes, and a range of SIMD DSP instructions that operate on 16-bit or 8-bit data values in 32-bit registers. [4]

ARM1176JZF-S architecture with Jazelle technology

The ARM1176JZF-S processor has three instruction sets:

- the 32-bit ARM instruction set used in ARM state, with media instructions
- the 16-bit Thumb instruction set used in Thumb state
- the 8-bit Java byte codes used in Jazelle state.

Instruction compression

A typical 32-bit architecture can manipulate 32-bit integers with single instructions and address a large address space much more efficiently than a 16-bit architecture. When processing 32-bit data, a 16-bit architecture takes at least two instructions to perform the same task as a single 32-bit instruction.

When a 16-bit architecture has only 16-bit instructions, and a 32-bit architecture has only 32-bit instructions, overall the 16-bit architecture has higher code density, and greater than half the performance of the 32-bit architecture.

Thumb implements a 16-bit instruction set on a 32-bit architecture, giving higher performance than on a 16-bit architecture, with higher code density than a 32-bit architecture. The ARM1176JZ-S processor can easily switch between running in ARM state and running in Thumb state. This enables you to optimize both code density and performance to best suit your application requirements.

The Thumb instruction set

The Thumb instruction set is a subset of the most commonly used 32-bit ARM instructions. Thumb instructions are 16 bits long and have a corresponding 32-bit

ARM instruction that has the same effect on the processor model. Thumb instructions operate with the standard ARM register configuration, enabling excellent interoperability between ARM and Thumb states.

Thumb has all the advantages of a 32-bit core:
• 32-bit address space
• 32-bit registers
• 32-bit shifter and *Arithmetic Logic Unit* (ALU)
• 32-bit memory transfer.

Thumb therefore offers a long branch range, powerful arithmetic operations, and a large address space. The availability of both 16-bit Thumb and 32-bit ARM instruction sets, gives you the flexibility to emphasize performance or code size on a subroutine level, according to the requirements of their applications. For example, you can code critical loops for applications such as fast interrupts and DSP algorithms using the full ARM instruction set, and linked with Thumb code.

Java byte codes
ARM architecture v6 with Janelle technology executes variable length Java byte codes. Java byte codes fall into two classes:

Hardware execution
Byte codes that perform stack-based operations.

Software execution
Byte codes that are too complex to execute directly in hardware are executed in software. An ARM register is used to access a table of exception handlers to handle these particular byte codes.

XO Laptop uses Marvel Sheeva ARM processor and the Raspberry Pi uses ARM1176JZF-S as a core which compatible with ARM v6 architecture. When

comparing video capabilities, XO laptops uses OGG video format and Raspberry Pi uses MPEG-2 and MPEG-4 video format for playing videos.

The ARM Instruction Set

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
RSB	Reverse Subtract	RSC	Reverse Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	TEQ	Test Equivalence
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
MUL	Multiply	MLA	Multiply Accumulate
SMULL	Sign Long Multiply	SMLAL	Signed Long Multiply Accumulate
UMULL	Unsigned Long Multiply	UMLAL	Unsigned Long Multiply Accumulate
CLZ	Count Leading Zeroes	BKPT	Breakpoint
MRS	Move From Status Register	MSR	Move to Status Register
B	Branch		
BL	Branch and Link	BLX	Branch and Link and Exchange
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Halfword	STRH	Store Halfword
LDRB	Load Byte	STRB	Store Byte
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
SWP	Swap Word	SWPB	Swap Byte
CDP	Coprocessor Data Processing		
MRC	Move From Coprocessor	MCR	Move to Coprocessor
LDC	Load To Coprocessor	STC	Store From Coprocessor

Figure 2.4: ARM Instruction Set

The Thumb Instruction Set

Mnemonic	Operation	Mnemonic	Operation
MOV	Move	MVN	Move Not
ADD	Add	ADC	Add with Carry
SUB	Subtract	SBC	Subtract with Carry
RSB	Reverse Subtract	RSC	Reverse Subtract with Carry
CMP	Compare	CMN	Compare Negated
TST	Test	NEG	Negate
AND	Logical AND	BIC	Bit Clear
EOR	Logical Exclusive OR	ORR	Logical (inclusive) OR
LSL	Logical Shift Left	LSR	Logical Shift Right
ASR	Arithmetic Shift Right	ROR	Rotate Right
MUL	Multiply	BKPT	Breakpoint
B	Unconditional Branch	Bcc	Conditional Branch
BL	Branch and Link	BLX	Branch and Link and Exchange
BX	Branch and Exchange	SWI	Software Interrupt
LDR	Load Word	STR	Store Word
LDRH	Load Halfword	STRH	Store Halfword
LDRB	Load Byte	STRB	Store Byte
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
PUSH	Push Registers to stack	POP	Pop Registers from stack

Figure 2.5: Thumb Instruction Set

2.2.3 Playing video on Nokia color LCD using an Atmega32

The video player has been designed and developed by electronic hobbyists Vinod from Kerala, India. Following items have been used to implement the video player. [6]

- Microcontroller: AVR Atmega32
- LCD: 65K color LCD from old nokia 6030 mobile phone.
- LCD controller: Philips PCF8833

FFmpeg was used to convert a sample video to frames at 15 frames/second & 132x65 resolution. Used python script to convert each still image to pixel information at 16 bit/pixel. Collected all the picture information of adjacent frames into a single file and named as my_video.lcd. Then copied that file to memory card and modified the AVR program to display it on the LCD. It accesses the FAT16 file system.

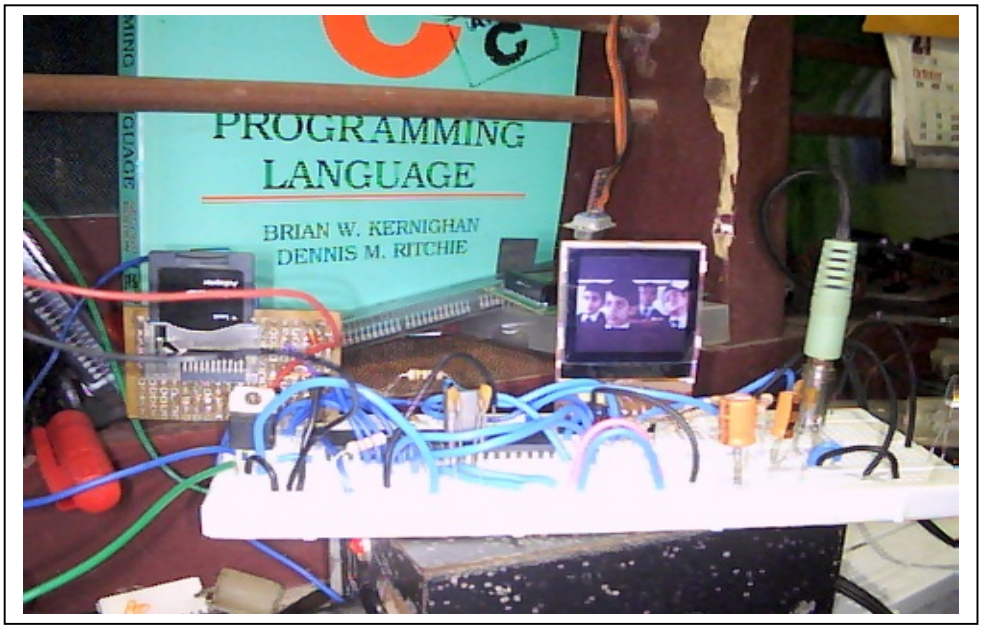


Figure 2.6: Playing video on Nokia colour LCD using an 8-bit AVR [ATtmega 32]

CHAPTER 3

METHODOLOGY FOR THE DESIGN OF ISA FOR VIDEO PLAYBACK

FFmpeg 3.2 decoder is used by many systems for video playback as it can be used to process many video formats. Therefore, current research is moved to carry out the survey of FFmpeg decoder to find out the minimal instruction set for video playback. FFmpeg 3.2 decoder was compiled by using Visual Studio 2015 which runs on windows 10 Operating System on Intel x86 platform.

3.1 The x86 Architecture

The Intel x86 platform used for analysis and testing of FFmpeg decoder. X86 is the 32 -bit version of the x86 instruction set. The Intel x86 processor uses complex instruction set computer (CISC) architecture, which means there is a modest number of special-purpose registers instead of large quantities of general-purpose registers.

The x86 architecture consists of the following unprivileged integer registers. Those are general purposes registers

- eax - Accumulator
- ebx - Base register
- ecx - Count register
- edx - Double – precision register
- esi - Source index register
- edi - Destination index register
- ebp - Base pointer registers
- esp - Stack pointer register

All integer registers are 32 bits. However, many of them have 16-bits or 8-bits sub registers.

16 bits registers

ax, bx, cx, dx, si, di, bp, sp

8 bits registers

al, ah, bl, bh, cl, ch, dl, dh

Special purpose registers

Segmented registers [16 bit -6]

CS - Code

DS - Data

SS - Stack

ES – Data

FS – Data

GS -Data

Two other registers are important for the processor's current state.

eip – Instruction pointer register

eflags – flags register

Addressing Modes

- Immediate
- Register Operand
- Displacement
- Base
- Base with Displacement
- Scaled Index with Displacement
- Base with Index and Displacement
- Base with Scaled Index and Displacement
- Relative

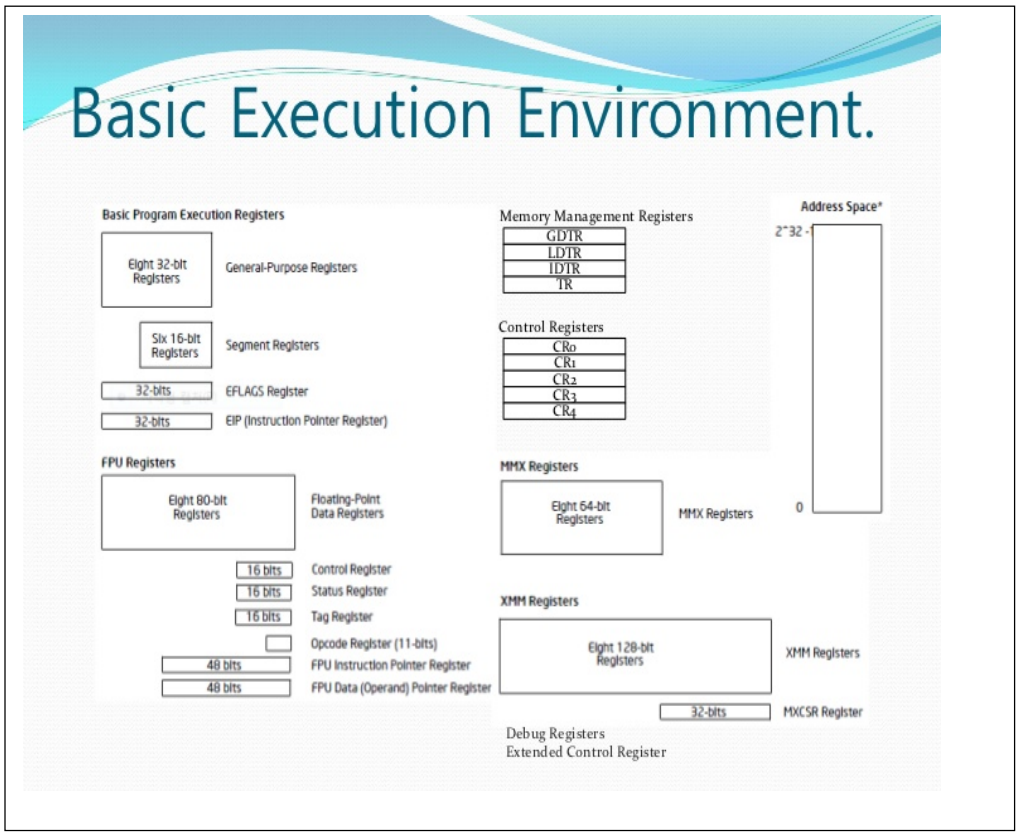


Figure 3.1: Basic Execution Environment of x86 architecture

3.2 Analysis of FFmpeg decoder and testing

FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. No matter if they were designed by some standards committee, the community or a corporation. It is also highly portable: FFmpeg compiles, runs, and passes our testing infrastructure FATE across Linux, Mac OS X, Microsoft Windows, the BSDs, Solaris, etc. under a wide variety of build environments, machine architectures, and configurations. [9]

It contains libavcodec, libavutil, libavformat, libavfilter, libavdevice, libswscale and libswresample which can be used by applications. As well as ffmpeg, ffserver, ffplay and ffprobe which can be used by end users for transcoding, streaming and playing.

FFmpeg Libraries for developers

Libavutil - is a library containing functions for simplifying programming, including random number generators, data structures, mathematics routines, core multimedia utilities, and much more.

Libavcodec - is a library containing decoders and encoders for audio/video codecs.
Libavformat- is a library containing demuxers and muxers for multimedia container formats.

Libavdevice - is a library containing input and output devices for grabbing from and rendering to many common multimedia input/output software frameworks, including Video4Linux, Video4Linux2, VfW, and ALSA.

Libavfilter - is a library containing media filters.

Libswscale - is a library performing highly optimized image scaling and color space/pixel format conversion operations.

Libswresample -is a library performing highly optimized audio resampling, rematrixing and sample format conversion operations.

3.2.1 FFmpeg decoder

FFmpeg 3.2 decoder has been compiled on Visual Studio 2015 which runs on Windows 10 and generated the object files.

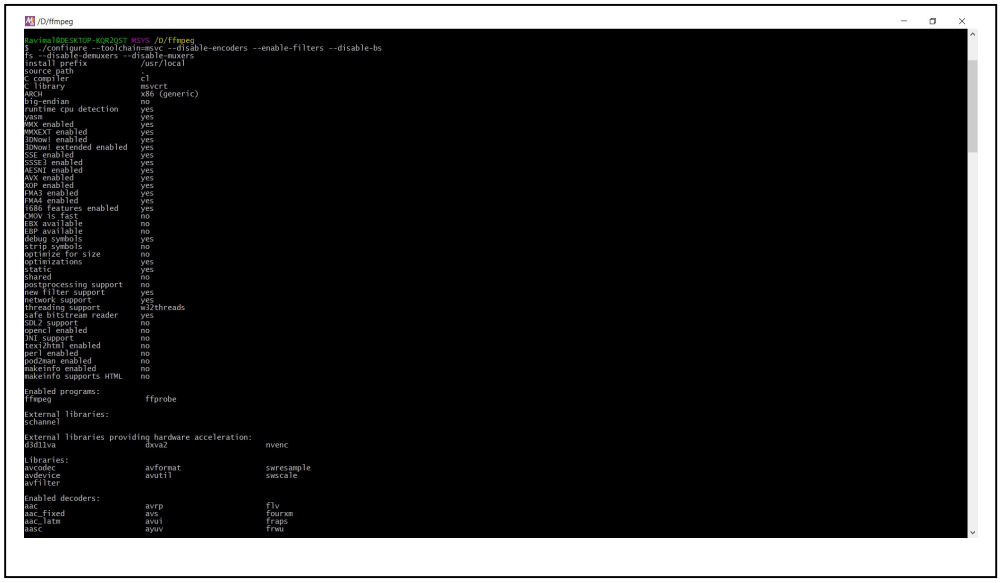


Figure 3.2: Compilation process of FFmpeg Decoder

The assembly code has been generated by using Microsoft (R) Optimizing Compiler Version 19.00.24215.1.

Following commands were added to the make file to generate Assembly code.

override CFLAGS += -FA

```
/*
 * SIMD optimized MPEG-4 Parametric Stereo decoding functions
 *
 * This file is part of FFmpeg.
 *
 * FFmpeg is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * FFmpeg is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with FFmpeg; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-
1301 USA
 */

#include "config.h"

#include "libavutil/x86/cpu.h"
#include "libavutil/attributes.h"
#include "libavcodec/aacpsdsp.h"

void ff_ps_add_squares_sse  (float *dst, const float (*src)[2], int n);
void ff_ps_add_squares_sse3 (float *dst, const float (*src)[2], int n);
void ff_ps_mul_pair_single_sse (float (*dst)[2], float (*src0)[2],
```

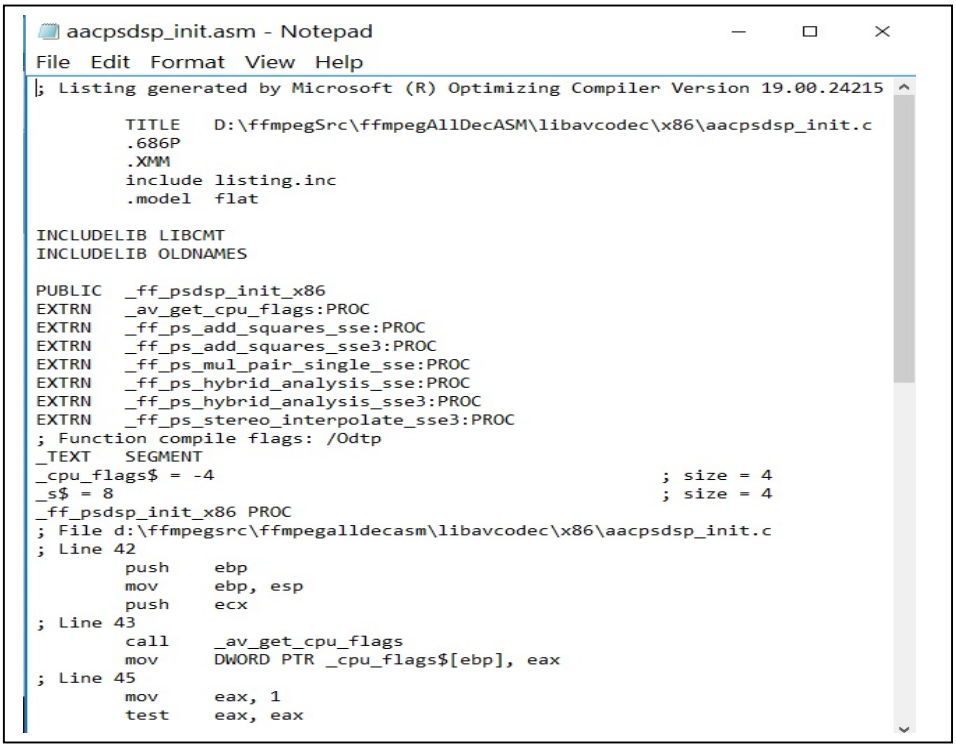
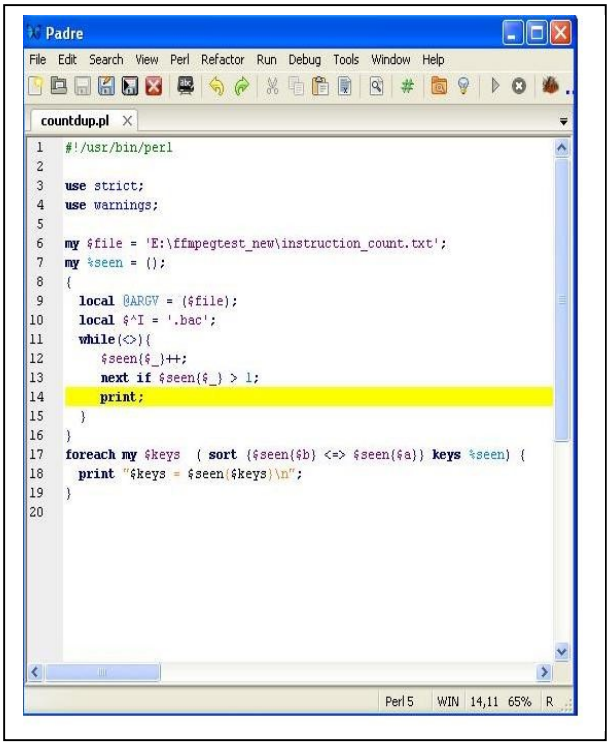


Figure 3.3: Part of C codes converted to Assembly codes.

After generation of the assembly file, duplicate instruction set was removed by using perl and analyzed the count of repeated instructions.



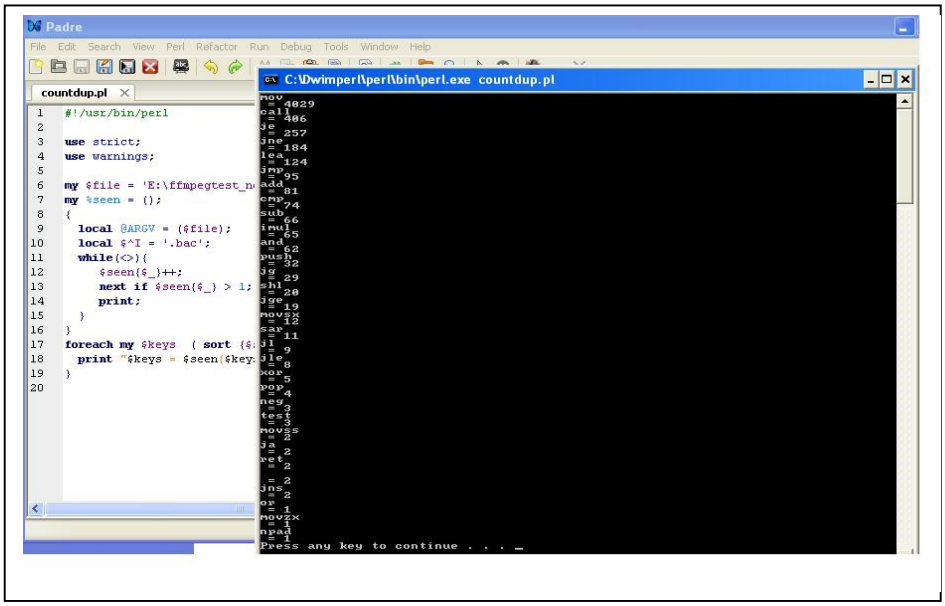


Figure 3.4: Removed repeated instructions and count of repeated instructions

3.3 Optimized Instruction Set for Video playback

Based on the instruction set of x86 platform, Analysis of Instruction Set has been carried out carefully and optimized the Instruction Set for playing video.

Table 3.1: Optimized Instruction set for video playback on x86 platforms

Format	Opcode	Operand	Description
Arithmetic and Logical Instruction			
add reg, const	add	eax, 1	add constant to register
add reg, reg	add	edi, eax	add register value to register
add reg, mem	add	eax, DWORD PTR _c$[ebp]	add value from memory to register
sub reg,const	add	eax, 1	subtract constant from registers
sub mem,const	sub	esp, 1048	subtract constant from memory value
sub reg,reg	sub	ecx, eax	subtract register value from register
sub reg, mem	sub	eax, DWORD PTR	subtract memory value from register

		_block_h$[ebp]	
imul reg, reg, const	imul	eax,DWORDPTR_block_type$[ebx], 640	Imul constant by memory value and store results in register
imul reg, mem, const	imul	eax,DWORDPTR_block_type$[ebx], 640	Imul constant by register value and store results in register
imul reg, mem	imul	eax,DWORDPTR_src_stride$[ebp]	Imul memory value with register value
neg reg	neg		perform two's complement negation of register value
and	and	eax, 1	perform and operation with their operands
and	and	eax, ecx	perform and operation with their operands
xor	xor	eax, eax	xor register value with register value
xor	xor	eax, ebp	xor memory value with register
or	or	eax, edx	or register value with register value
Shift and Rotate			
sar reg, const	sar	eax, 1	shift right register value according to the source constant
shl reg, const	shl	eax, 0	shift left register value according to the source constant
test reg,reg	test	eax, eax	check register constant is 0
Control Transfer			
cmp reg, const	cmp	DWORD PTR [eax], 16	compare constant with register value
cmp reg, mem	cmp	eax, DWORD PTR_align_end$[ebp]	compare memory value with register
cmp mem, reg	cmp	DWORD PTR _src_x$[ebp], ecx	compare register value with memory value
call mem	call	DWORD PTR _v_extend_var$[ebp]	call memory value

call reg	call	edx	call register value
call label	call	@__security_check_cookie@4	
ret	ret		
Data Movement			
mov reg, const	mov	eax, 1	move constant to register
mov reg, reg	mov	esp, ebx	move register value to register
mov mem, reg	mov	DWORD PTR _b$[ebp], eax	move register value to memory
mov mem const	mov	DWORD PTR _w$[ebp], 3	move constant to memory
pop mem	pop	ebp	pop data from memory location
push const	push	-64	push constant
push reg	push	eax	push data into register
push mem	push	ebp	push data into memory
lea reg, mem	lea	eax,DWORD PTR _half$[ebp]	compute load effective address of memory value to register value
lea reg, reg	lea	eax, DWORD PTR _ff_filters_16bpp[ecx+edx]	compute load effective address of register to register value
Ja label	ja	SHORT $LN2@ff_spatial	
Je label	je	$LN1@emulated_e	
Jg label	jg	SHORT $LN1@ff_h264chr	
Jge label	jge	$LN1@imdct36_bl	
Jl label	jl	SHORT $LN1@deblock_v_	
Jle label	jle	SHORT $LN1@ff_h264chr	
Jmp label	jmp	$LN1@avg_rv40_q	

Jne label	jne	$LN1@ff_h264_pr	
Jns label	jns	SHORT $LN19@emulated_e	
Miscellaneous			
npad			

Table 3 2: Optimized Instruction Set to design ISA for Video Playback

Binary Arithmetic	**Logical**	**Control Transfer**	**Shift and Rotate**	**Data Transfer**	**Miscellaneous**
ADD	AND	CMP	SAR	MOV	NPAD
SUB	XOR	CALL	SHL	POP	
IMUL	OR	TEST		PUSH	
NEG		RET		JA, JE, JG, JGE, JL, JLE, JMP	
				JNE, JNS	
				LEA	

CHAPTER 4

DESIGN OF ISA FOR VIDEO PLAYBACK

In this chapter, new ISA design of customized processor for video playback is presented. After completion of minimizing the instruction set by using FFmpeg decoder (minimized instruction set is listed in table 3.2 in chapter 3), designing of Instruction set format for proposed customized processor was commenced.

After conversion of FFmpeg decoder source code written in C to Assembly, analyzed that all the instructions are designed according to the CISC architecture. Therefore, considering the code simplicity and the Application specialty; it was decided to design of Instruction Set by using RISC architecture.

4.1 Instruction Set Architecture (ISA)

Instruction Set Architecture (ISA) defines the microprocessor from a machine-language programming perspective, including the following:

- Instruction set
- Structure of the register file
- Addressing modes
- Data types and data representation

This definition of the ISA makes the hardware structure and implementation details transparent to the programmer.

The design of an instruction set or rather a *good* instruction set is a challenging task since there is not any systematic way for achieving this goal. It is usually an iterative process that involves balancing different contradicting factors in order to meet the microprocessor requirements in an *optimum* way.

4.1.1 Instruction Format Design

Since the author was interested only in the architecture of a very simple RISC processor, the proposed Custom Processor was made to follow Load/Store Architecture.

Six core instruction formats (L/S/I/Y/R/C/J) have been introduced and described in this chapter. All are fixed 32 bits in length and must be aligned on a four-byte boundary in memory.

There are 26 instructions identified after the analysis of minimized instruction set. Two new instructions were added to avoid direct memory accessing. Therefore, Total instructions in ISA amounted to 28. Considering the total instructions, Opcode length in instruction format was defined as 5 bits. As 13 General Purpose Registers were used in ISA, 4 bits were defined for selecting Base, Source and Destination Registers. Addressing Modes (AMOD) were used within the instruction format and for selecting different type of Addressing Modes.

L and S type – Instruction Format

As the customized processor supports Load/Store architecture, it does not support direct memory accessing.

Minimized instructions follow Memory to Register (MOV mem, reg) and Register to Memory (MOV reg, mem) operations when transferring the data and Arithmetic Operations (ADD reg, mem), new Load/Store instructions were introduced to avoid direct memory accessing.

L type - Load

Opcode	AMOD	Destination Register	Base Register	Immediate
5bits	1bit	4 bits	4bits	
32 bits				

S type - Store

Opcode	AMOD	Base Register	Source Register	Immediate
5bits	1bit	4 bits	4bits	
32bits				

Two addressing modes used in Load and Store instruction format.

1] Base Register Addressing Mode
2] Immediate offset Addressing Mode

1] Base Register Addressing Mode
Ex:
Load R2, [R0] - load value of R0 which is located in memory to R2

STR [R0], R1 - Store R1 value to the memory location which is located by R0 location pointed by R0 is base register.

2] Immediate offset Addressing Mode
To specify memory address, Processor needs to specify two things.

1] A register which contains a pointer to memory.
2] A numerical offset (in bytes)

A desired memory address is the sum of these two values.

Ex: [R0, 28]
Specifies the memory address pointed by the value in R0, add 28. Offset is generally used in accessing elements of array or structure. Base register points to beginning of array or structure.

Ex:

LDR R2, [R0, 28]
STR [R0, 28], R1

When computing the arithmetic operation, processor has to use LDR and STR instructions in addition to **add** instruction.

Following instructions show how to compute **addl $8, -28 (% ebp)** instructions in RISC architecture instruction format.

LDR [R0, 28], R1
Add R1, 8
STR, [R0, 28], R1

I type Instruction format

Following instruction format was proposed for ADD/SUB/CMP/IMUL/
/SAR/SHL/Logical AND, OR, XOR, MOV, LEA, TEST instructions.

Opcode	AMOD	Destination register	Source register	Immediate
5bits	1bit	4 bits	4bits	
32 bits				

Addressing Modes are encoded as follows
1] Immediate -1
2] Register -0

Y type Instruction format

Following instruction format was proposed for NEG/POP/PUSH/CALL

Opcode	Register	immediate
5bits	4bits	
32bits		

R type Instruction format

Following instruction format was proposed for NPAD/RET

Opcode (5 bits)
32 bits

C type Instruction format

Following instruction format was proposed for IMUL

Opcode	Destination Register	Source register	Immediate
5 bits	4 bits	4 bits	
32 bits			

J type Instruction format

Following instruction format was proposed for JA/JE/JG/JGE/JL/JLE/JMP/JNE/JNS

Opcode	offset
5 bits	27 bits [0:26]
32 bits	

4.1.2 Register File

- **General Purposes Registers**

  32 bits registers (8)

  - R0 – R7

  16 bits registers (2)
  - R8 - R9

  8 bits registers (3)
  - R10 – R12

- **General Purposes Registers are encoded as follows**

  4 bits needed to represent the registers

  R0 - 0001
  R1 - 0010
  R2 - 0011
  R3 - 0100
  R4 - 0101
  R5 - 0110
  R6 - 0111
  R7 - 1000
  R8 - 1001
  R9 - 1010
  R10 - 1011
  R11 - 1100
  R12 - 1101

- **Special Purpose Registers**

  - FLAGS (Status & Control) register [32 bits -1]
  - IP (Instruction Pointer) register [32 bits -1]
  - MAR – Memory Address Registers
  - MDR – Memory Data Registers
  - PC – Program Counter
  - IR – Instruction Register

4.1.3 Addressing Modes

4 addressing modes used for designing the custom processor

- Immediate
- Base Register
- Immediate offset (Base + Displacement)
- Register

CHAPTER 5

VALIDATION OF MINIMAL INSTRUCTION SET

5.1 Dynamic Analysis of video playback at runtime

Intel Software Development Emulator was used to dynamically trace video application at runtime to capture runtime instructions of a video in different formats.

Intel Software Development Emulator (SDE)

Intel SDE is built upon the Pin dynamic binary instrumentation system and the XED encoder decoder. Pin controls the execution of an application. Pin examines each static instruction in the application approximately once, as it builds traces for execution. During this process, which is called instrumentation, for each instruction encountered Pin asks Intel SDE if this instruction should be emulated or not. If the instruction is to be emulated, then Intel SDE tells Pin to skip over that instruction and instead branch to the appropriate emulation routine. It also tells Pin how to invoke that emulation function, what arguments to pass, etc. [10]

Intel SDE queries CPUID to figure out what features to emulate. It also modifies the output of CPUID so that compiled applications that check for the emulated features are told that those features exist.

Intel SDE comes with several useful emulator-enabled Pin tools and the XED disassembler:

- The basic emulator
- The mix histogramming tool: This Pin tool can compute histograms by any of: dynamic instructions executed, instruction length, instruction category, and ISA extension grouping. This tool can also display the top N most frequently executed basic blocks and disassemble them.

- The debugtrace ASCII tracing tool: This versatile tool is useful for observing the dynamic behavior of your code. It prints the instructions executed, and also the registers written, memory read and written, etc.
- The footprint tool: This simple tool counts how many unique 64-byte chunks of data were referenced during the execution of the program.
- The XED command line tool which can disassemble PECOFF or ELF binary executables.

Running the Histogram Tool

To generate the instruction, mix histograms by opcode (XED iclass, the default) or instruction form (iform). As of version 4.29, the instruction length and instruction category histograms are always included.

```
path-to-kit/sde -mix -- user-application [args]
path-to-kit/sde -mix -iform -- user-application [args]
```

Notes: The ISA extension histogram is also always computed and printed as star-prefixed rows in the histograms. ISA extensions are things like (BASE, X87, MMX, SSE, SSE2, SSE3, etc.). This is useful to see which instruction set extensions are used in your application.

- The dynamic statistics are recorded and emitted several ways: (1) per-thread, (2) per function per thread, and (3) summed for the entire run. Instruction counts by function are also emitted if symbols are found for the application.
- The output is written to a sde-mix-out.txt file in the current directory. The output file name can be changed using the -omix option:

  ```
  path-to-kit/sde -mix -omix foo.out -- user-application [args]
  ```

- The top 20 basic blocks are always printed in the output with their execution weights.
- "-top_blocks N" will allow you to change 20 to N that you specify.

- **Iforms**: "Iform" is the XED term for variants of instructions. In a simple world they would be things like reg/reg or reg/mem, but things are more complicated in general. The iform names come from XED. Consider them experimental and subject to change. To see histograms by the more detailed iforms, use the "-iform" command line option.

This mix histogram tool was used for dynamic analysis and the following commands were used for analyzing the dynamic instructions when playing video.

H:\Intel SDE\ sde_h\-mix -- ffplay LTW.avi

H:\Intel SDE\ sde_h\-mix -- ffplay LTW.wmv

H:\Intel SDE\ sde_h\-mix -- ffplay LTW.mp4

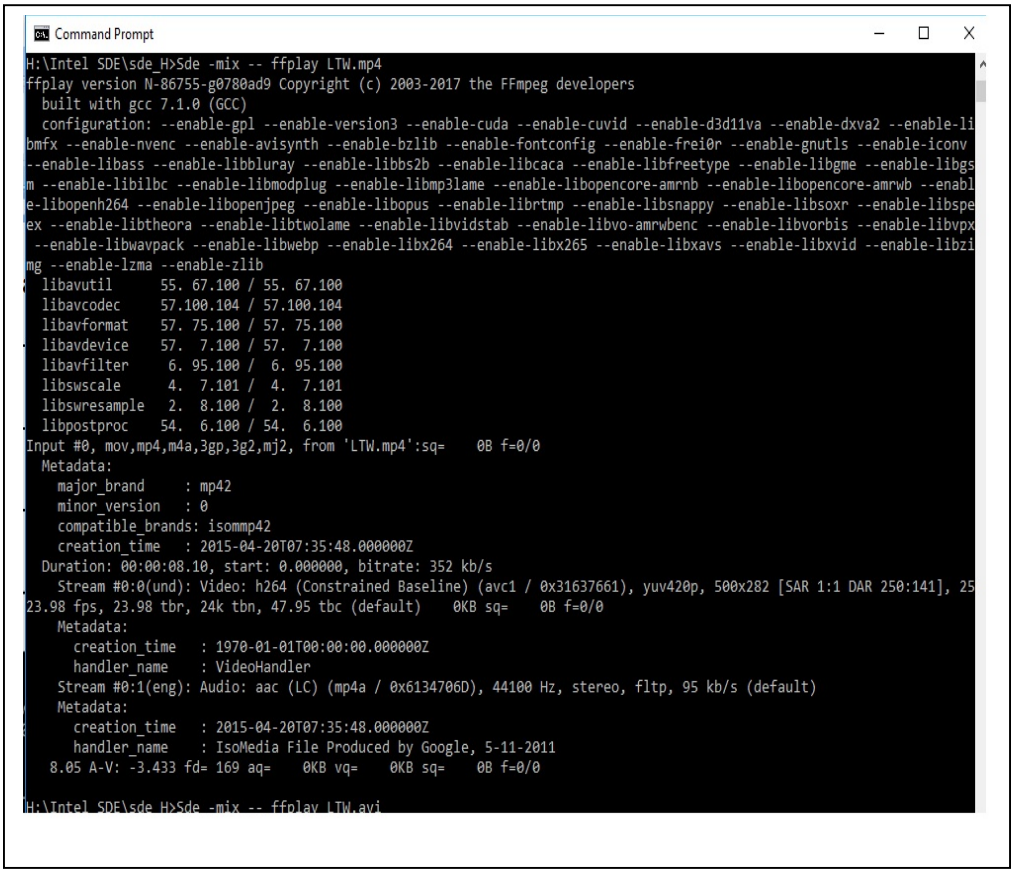


Figure 5.1: Play videos using Intel Software Development Basic Emulator

Table 5 1: Most frequently used Runtime Instructions in different formats when playing videos

AVI	MP4	WMV
MOV	MOV	MOV
ADD	ADD	ADD
SUB	MOVQ	SUB
IMUL	LEA	SAR
LEA	TEST	IMUL
TEST	CMP	LEA
CMP	JZ	MOVSX
SBB	PUSH	TEST
JNZ	POP	CMP
JZ	MOVZX	MOVZX
MOVQ	JNZ	JZ
MOVDQA	PADDW	JNZ
MOVZX	AND	SHL
SHL	MOVDQA	PUSH
AND	SUB	POP
PUSH	REP_STOSD	MOVQ
ADC	VPSUBUSB	JMP
POP	SHL	SHR
MOVSX	MOVSX	NEG
SHR	PUNPCKLBW	CMOVZ
JMP	VMOVDQA	VMOVAPS
REP_STOSD	CALL_NEAR	JLE
SAR	SAR	AND
CALL_NEAR	RET_NEAR	MOVAPS
PADDSW	PALIGNR	XOR
PMADDWD	PSUBW	CALL_NEAR
PADDD	MOVD	FMUL
XOR	IMUL	CMOVNBE
JLE	JMP	RET_NEAR
RET_NEAR	JNLE	PXOR
MOVAPS	DEC	VMULPS
PXOR	VPAVGB	JNS
NEG	SHR	JS

BSWAP	VMOVD	FLD
PSUBSW	VPAND	MOVDQA
JNS	VPXOR	FSTP
REP_MOVSD	PSRAW	BSWAP
JS	XOR	REP_STOSD
MOVDQU	PUNPCKHDQ	MOVDQU
SHRD	PACKUSWB	VSUBPS
PADDUSW	VPOR	VADDPS
PSHUFD	PSLLW	JNLE
PSRAD	PUNPCKLWD	PADDW
PSRAW	JL	PANDN
OR	OR	PCMPEQW
PUNPCKLBW	PUNPCKHWD	PCMPGTW
JNLE	VMOVAPS	PMULLW
PACKSSDW	PUNPCKHBW	PADDUSW
PACKUSWB	PREFETCHT0	VSHUFPS
PMULHW	MOVDQU	JL

Table 5 2: Common Instructions used in three formats and its functions obtained after Analysis

Instruction	Operation
ADD	Integer add
AND	Perform bitwise logical AND
CALL_NEAR	Call Procedure
CMP	Compare
IMUL	Signed Multiply
JMP	Jump
JNLE	Jump if not or equal
LEA	Load effective address
MOV	Move data between general purpose and segment
MOVDQA	Move aligned double quadword
MOVDQU	Move unaligned double quadword
MOVQ	Move quadword

MOVSX	Move with sign-extension
MOVZX	Move with zero -extend
POP	Pop a value from stack
PUNPCKLBW	Unpack high-order bytes
PUSH	Push onto stack
RET_NEAR	Return from procedure
SAR	Shift arithmetic right
SHL	Shift logical left
SHR	Shift logical right
SUB	Subtract
TEST	Logical Compare
XOR	Perform bitwise logical NOT

Table 5 3: Other Instructions found within mostly used instructions

ADC	ADD with carry
BSWAP	Byte Swap
CMOVNBE	Conditional move if not below or equal
CMOVZ	Conditional move if zero
DEC	Decrement
FLD	Load Floating point value
FMUL	Fraction Multiply Unsigned
FSTP	Store Floating Point Value
JL	Jump if less
JLE	Jump if less or equal
JNS	Jump if not sign (not negative)
JNZ	Jump if not Zero
JS	Jump if Sign
JZ	Jump if Zero
MOVAPS	Move four aligned packed single -precision floating point
MOVD	Move doubleword
NEG	Negate
OR	Perform bitwise logical AND
PACKSSDW	Pack doublewords into words with signed saturation
PACKUSWB	Pack words into bytes with unsigned saturation
PADDD	Add packed doubleword integers
PADDSW	Add packed signed word integers with signed saturation
PADDUSW	Add packed unsigned word integers with unsigned saturation

PADDW	Add packed word integers
PALIGNR	Packed align right
PANDN	Logical AND NOT
PCMPEQW	Compare packed data for equal
PCMPGTW	Compare packed signed integers for greater than
PMADDWD	Multiply and add packed with Integers
PMULHW	Multiply Packed signed integers and store high results
PREFETCHT0	Prefetch data into caches
PSHUFD	Shuffle packed doublewords
PSLLW	Shift packed data left logical
PSRAD	Shift packed data with arithmetic
PSRAW	Shift packed data right arithmetic
PSUBSW	Subtract packed signed integers with signed saturation
PSUBW	Subtract packed word integers
PUNPCKHDQ	Unpacked high data
PUNPCKHWD	Unpacked high data
PUNPCKLBW	Unpack low data
PUNPCKLWD	Unpack low data
PXOR	Logical exclusive low
REP_MOVSD	Move data from string to string
SBB	Subtract with borrow
SHRD	Shift right double

After completion of the Dynamic analysis, the mostly used runtime 50 instructions were compared with the minimal instruction set which was used for designing the Instruction Set Architecture for video playback device. Minimal Instruction Set had a total of 26 instructions after the analysis. All most 21 instructions have found in most frequently used runtime instructions and their functionalities are also the same. Among those runtime instructions, 5 instructions were not found out of 26 instructions. Therefore, 5 instructions must be removed from designed ISA for video playback.

Table 5 4: Minimal Instructions found among the runtime Instructions

ADD
AND
CALL
CMP
IMUL
JMP
LEA
MOV
POP
PUSH
RET
SAR
SHL
SUB
TEST
XOR

Table 5 5*:* Minimal Instructions have not found among the runtime Instructions

JA
JE
JGE
JNE
NPAD

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The main focus on this thesis was to design an Instruction Set Architecture of customized processor for video playback. During this research, the author faced many challenges particularly in finding the path to minimize the Instruction set. Many low-cost processor architectures were referred and finally open source multimedia decoder was used to for the task. After minimizing the instruction set, designing of the ISA was another challenge as many open source ISAs were already available in the market. When designing this ISA most popular open source ISAs were analyzed and extracted necessary features that matches with the requirement. The significant advantages of this newly created ISA are Application Specific (Used only for Video Playback), Simplicity and User friendliness.

To publish this ISA yet much testing works remain. The author is determined in making a broadly applicable user-level ISA and thereby to develop customized processor in the future by incorporating this ISA. Notably, the privileged customized processor architecture is incomplete. The reference hardware platform and the various binary interfaces remain to be specified.

By far most computers used by children contain general purpose processors for video playback. Therefore, by introducing a customized processor with minimal input devices, this video playback device will be cost effective.

REFERENCES

[1] Marcus Wellington. (2015, Nov) “Laptop Computers for Children” [Online article] Available: http://wiki.mobileread.com/wiki/Laptop Computers for Children.html.

[2] OLPC team (2014, Mar)” XO – 1.75” [Online article] Available: http://wiki.laptop.org/go/XO-1.75-OLPC.html.

[3] Marvell International Ltd., (2010, Feb) “Marvel ARMADA 610 Application Processor” [Online Article] Available: www.marvell.com/application-processors/armada-600/assets/armada610_pb.pdf

[4] Wikipedia (2014, Jan) “Raspberry Pi” [Online article] Available: http://en.wikipedia.org/wiki/Raspberry_Pi - Wikipedia.html

[5] Arm Limited (2004-2009) ARM1176JZF [Online] http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0301h/index.html

[6] Vinod S. (June 2012) “Electronics the King of Hobbies” [Online article] Available: http://blog.vinu.co.in/2012/06/avr-video-player-on-nokia-color-lcd.html

[7] Krste Asanović and David A. Patterson [2014, Aug] “Instruction Sets Should Be Free: The Case For RISC-V”[Online] Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.html

[8] Andrew Waterman, Yunsup Lee, David A. Patterson, Krste Asanovic [2014, May] "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.0" Electrical Engineering and Computer Sciences, University of California at Berkeley [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.pdf

[9] FFmpeg Developers (2014, July] "FFmpeg Documentation" [Online Article] Available: https://ffmpeg.org/ffmpeg.html.

[10] Ady Tal [2012,June] Intel Software Development Emulator, Intel Corporation [Online] Available: https://software.intel.com/en-us/articles/intel-software-development-emulator.

[11] Intel Group [1997-2018] "The Intel® 64 and IA-32 Architectures Software Developer's Manual" Intel Corporation [Online] Available: https://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-1-manual.html

APPENDIX A

C files used in FFmpeg 3.2 decoder

(1) aacpsdsp_init.c
(2) ac3dsp_init.c
(3) alacdsp_init.c
(4) audiodsp_init.c
(5) blockdsp_init.c
(6) bswapdsp_init.c
(7) cavsdsp_init.c
(8) constants.c
(9) dcadsp_init.c
(10) dct_init.c
(11) divac_dwt_init.c
(12) dirac_dwt_init.c
(13) fft_init.c
(14) flacdsp.c
(15) fmtconvert_init.c
(16) g722dsp_init.c
(17) h263dsp_init.c
(18) h264_intrapred_init.c
(19) h264_qpel.c
(20) h264chroma_init.c
(21) h264dsp_init.c
(22) hpeldsp_int.c
(23) hutyuvdsp_init.c
(24) idctdsp_init.c
(25) jpeg2000dsp_init.c
(26) lossless_audiodsp_init.c
(27) lossless_videodsp_init.c
(28) me_cmp_init.c
(29) mldsp_int.c
(30) mpegaudiodsp.c
(31) mpegvideo.c
(32) mpegvideodsp.c
(33) pixblockdsp_init.c
(34) proresdsp_init.c
(35) qpeldsp_init.c
(36) rv34dsp_init.c
(37) rv40dsp_init.c
(38) sbrdsp_init.c
(39) simple_idct.c
(40) snowdsp.c
(41) synth_filter_init.c
(42) taksp_init.c
(43) ttadsp_init.c
(44) v210_init.c
(45) vc1dsp_init.c
(46) vc1dsp_mmx.c
(47) videodsp_init.c
(48) vorbisdsp_init.c
(49) vp3dsp_init.c
(50) vp6dsp_init.c
(51) vp6dsp_init.c
(52) vp8dsp_init.c
(53) vp9dsp_init.c
(54) vp9dsp_init10bpp.c
(55) vp9dsp_init_12bpp.c
(56) xvidict_init.c
(57) mpegvideoenc.c
(58) mpegvideoencdsp_init.c